МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование блочных шифров

Студент: Евсеенко В. П.

ФИТ 3 курс 4 группа

Преподаватель: Савельева М. Г.

Минск 2025

Содержание

[Введение 3](#_Toc195185975)

[1 Практическое задание 4](#_Toc195185976)

[Задание 1 4](#_Toc195185977)

[Задание 2 7](#_Toc195185978)

[Задание 3 8](#_Toc195185979)

[2 Оценка времени 9](#_Toc195185980)

[Вывод 10](#_Toc195185981)

# **Введение**

Цель лабораторной работы:

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

Задачи:

* закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости блочных шифров;
* разработать приложение для реализации указанных преподавателем методов блочного зашифрования/расшифрования.
* выполнить анализ криптостойкости блочных шифров;
* оценить скорость зашифрования/расшифрования реализованных шифров.

# **1 Практическое задание**

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.

# **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. 1.1.

Таблица 1.1 – Варианты задания

|  |  |  |
| --- | --- | --- |
| Вариант | Алгоритм | Ключ |
| 1 | DES | Первые 8 символов собственных фамилииимени |
| 2 | DES-EEE3 | а) Информационнаябезопаснос\*  б) лабораторная--работа\_\_№5\*  \*каждый из трех ключей выделен шрифтом |
| 3 | DES-EDE3 | По указанию преподавателя |
| 4 | DES-EEE2 | По указанию преподавателя |
| 5 | DES-EDE2 | По указанию преподавателя |

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

Функция encrypt принимает на вход шифруемый текст, ключ и выполняет зашифрование. Код функции представлен в листинге 1.1.

|  |
| --- |
| export const encrypt = (msg: string, key: Bytes) => {      const cipher = forge.cipher.createCipher('DES-ECB', key);      cipher.start({iv: iv});      cipher.update(forge.util.createBuffer(msg));      cipher.finish();      return cipher.output;  }; |

Листинг 1.1 – Код функции шифрования

Для дешифрования была разработана функция decrypt. Код функции представлен на рисунке 1.2.

|  |
| --- |
| export const decrypt = (encrypted: util.ByteStringBuffer, key: string) => {      const decipher = forge.cipher.createDecipher('DES-ECB', key);      decipher.start({iv: iv});      decipher.update(encrypted);      decipher.finish();      return hex2a(decipher.output.toHex());  }; |

Листинг 1.2 – Код функции расшифрования

Результат работы приложения с исходным текстом «EvseenkoViktoriyaPavlovna» и ключом «evseenko» приведен на рисунке 1.1.

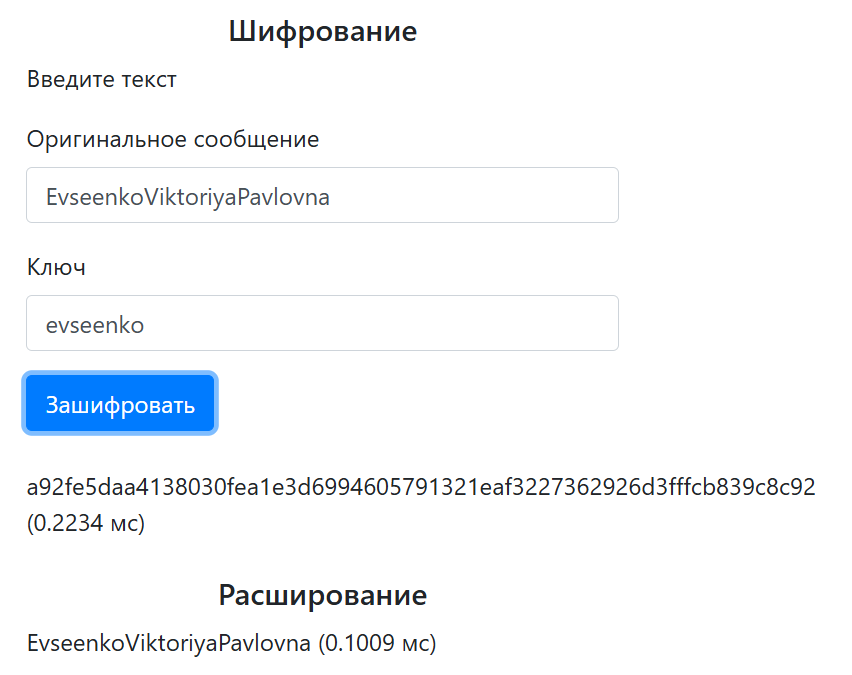


Рисунок 1.1 – Результат шифрования и дешифрования текста

Функция avalancheEffect принимает исходный текст и шифрует его. Затем изменяет один бит исходного текста, снова шифрует его и подсчитывает количество измененных бит и процентах. Код функции представлен в листинге 1.3.

|  |
| --- |
| export const avalancheEffect = (originalText: string, key: Bytes) => {let changedBits: number = 0;   const encryptedText1: string = hexToBinary(encrypt(originalText, key).toHex());      let stringWithOneBitChanged = convertBinaryToString(invertLastBit(convertStringToBinary(originalText)));      const encryptedText2: string = hexToBinary(encrypt(stringWithOneBitChanged, key).toHex());      for (let i = 0; i < encryptedText1.length; i++) {          if (encryptedText1[i] !== encryptedText2[i]) {              changedBits++;}}      const avalancheEffect = changedBits / encryptedText1.length \* 100;      return {          avalancheEffect: avalancheEffect,          stringChanged: stringWithOneBitChanged};} |

Листинг 1.4 – Код функции, рассчитывающей лавинный эффект

Результат анализа лавинного эффекта представлен на рисунке 1.2.

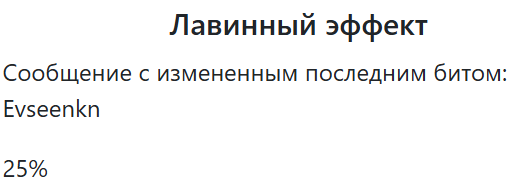


Рисунок 1.2 – Эффект лавины для исходного текста

# **Задание 2**

Проанализировать влияние слабых ключей (пример в таблице 1.2)

Таблица 1.2 – Слабые ключи DES

|  |  |  |  |
| --- | --- | --- | --- |
| 0101 | 0101 | 0101 | 0101 |
| 1F1F | 1F1F | 0E0E | 0E0E |
| E0E0 | E0E0 | F1F1 | F1F1 |
| FEFE | FEFE | FEFE | FEFE |

Также полуслабых ключей (пример в таблице 1.3) на конечный результат зашифрования и на лавинный эффект.

Таблица 1.3 – Полуслабые ключи DES

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 01FE | 01FE | 01FE | 01FE | FE01 | FE01 | FE01 | FE01 |
| 1FE0 | 1FE0 | 0EF1 | 0EF1 | E01F | E01F | F10E | F10E |
| 01E0 | 01E0 | 01F1 | 01F1 | E001 | E001 | F101 | F101 |
| 1FFE | 1EEE | 0EFE | 0EFE | FE1F | FE1F | FE0E | FE0E |
| 011F | 011F | 010E | 010E | 1F01 | 1F01 | 0E01 | 0E01 |
| E0FE | E0FE | F1FE | F1FE | FEE0 | FEE0 | FEE1 | FEE1 |

Пример работы приложения со слабым ключом 1F1F приведен на рисунке 1.3.

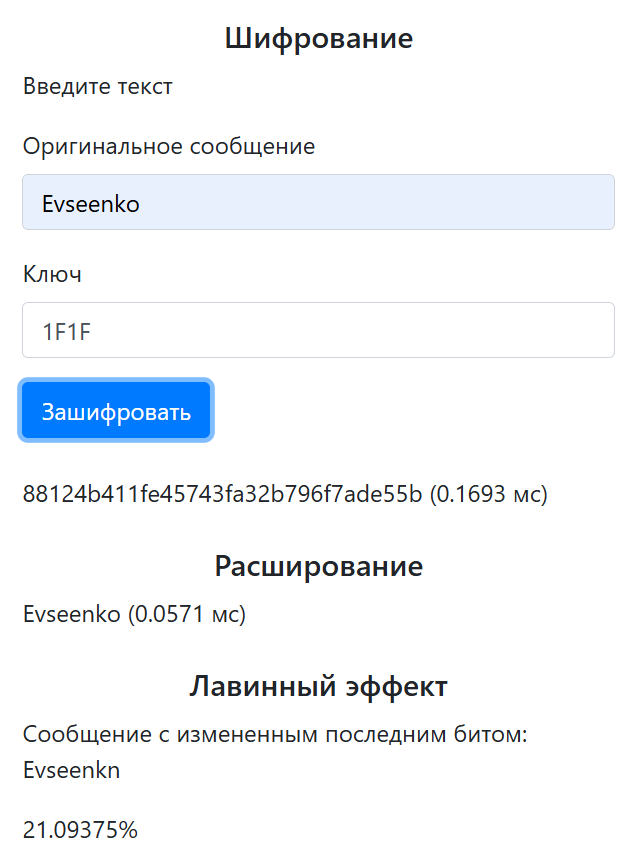


Рисунок 1.3 – Эффект лавины для исходного текста при шифровании со слабым ключом

Пример работы приложения со полуслабым ключом 01FE приведен на рисунке 1.4.

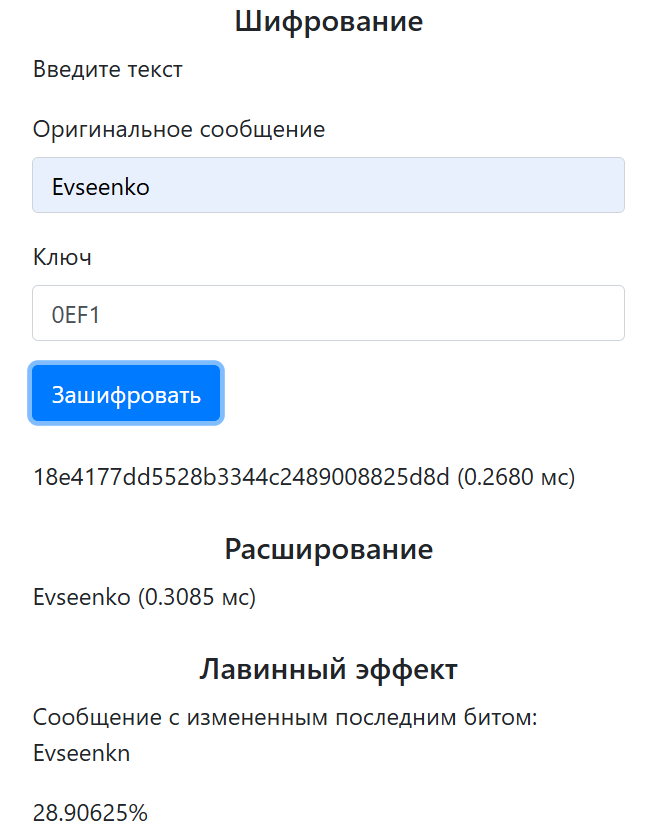


Рисунок 1.4 – Эффект лавины для исходного текста при шифровании с полуслабым ключом

Можно заметить, что эффект лавины значительно уменьшается, поскольку при разделении первоначального значения ключа на две половины, получаются части, в которые все биты равны 0 или 1, которые в последующем не изменяются в каждом раунде алгоритма. В случае полуслабых ключей, каждый из подключей используется 8 раз, что также снижает действие лавинного эффекта, а чем ниже процент, тем меньше вероятность успешной атаки.

# **Задание 3**

Оценить степень сжатия (используя любой доступный архиватор) открытого текста и соответствующего зашифрованного текста. Дать пояснения к полученному результату. Результат приведен на рисунке 1.5.

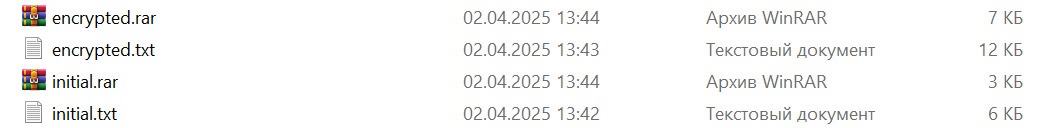


Рисунок 1.5 – Степень сжатия текста

Можно заметить, что исходный текст сжался на 50%, в то время как зашифрованный только на 40%. Это происходит потому, что шифрование преобразует информацию таким образом, что создается псевдослучайная последовательность битов, к которым практически невозможно применить существующие алгоритмы сжатия.

# **2 Оценка времени**

Оценить время выполнения операций зашифрования/расшифрования (напоминание: во многих языках программирования есть встроенные методы для замеров времени; при отсутствии такового в используемом языке можно воспользоваться разностью двух дат (например, в миллисекундах: время после выполнения программы – время до начала выполнения преобразования)).

Оценка времени выполнения операций шифрования и расшифрования в блочном шифре представлена на рисунке 2.1.

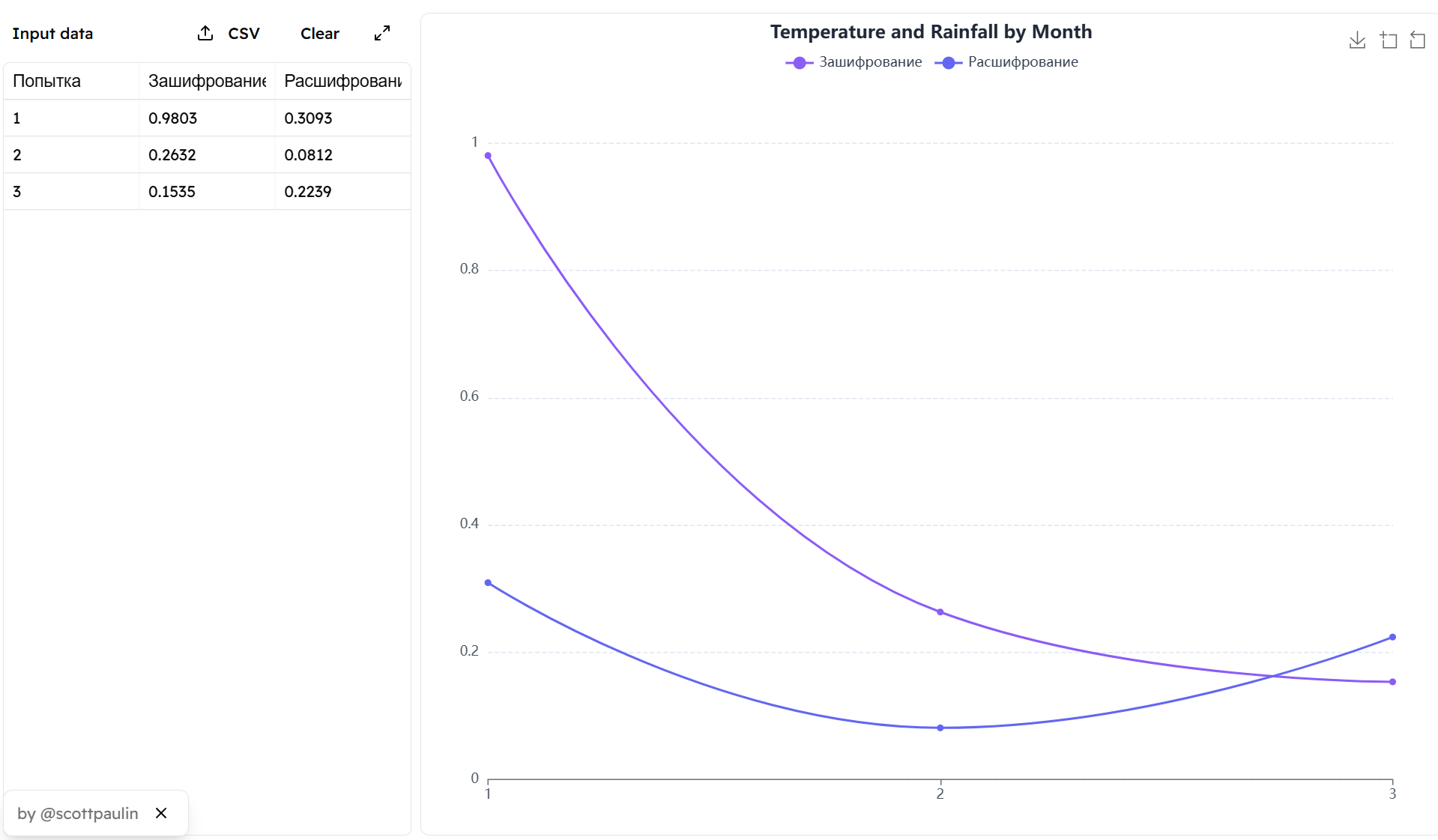


Рисунок 2.1 – Оценка времени операций зашифрования/расшифрования

# **Вывод**

В ходе выполнения лабораторной работы были закреплены теоретические знания о принципах построения блочных шифров, их алгебраическом описании, а также алгоритмах зашифрования и расшифрования. Разработано программное приложение, реализующее заданные преподавателем методы блочного шифрования и расшифрования. Проведен анализ криптостойкости реализованных шифров с учетом их устойчивости к различным видам криптоатак. Также произведена оценка скорости выполнения операций шифрования и расшифрования, что позволило сравнить эффективность различных алгоритмов. Полученные результаты подтверждают как практическую применимость изученных шифров, так и важность выбора алгоритма в зависимости от требований к безопасности и производительности.