МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7 НА ТЕМУ:**

**Исследование блочных шифров**

Выполнил студент 3 курса 6 группы

Подобед Владислав

Минск 2024

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

**Теоретические сведения**

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.

**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, о которых мы упоминали выше. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый раундом или циклом. Общая схема алгоритма DES представлена на рисунке 1.

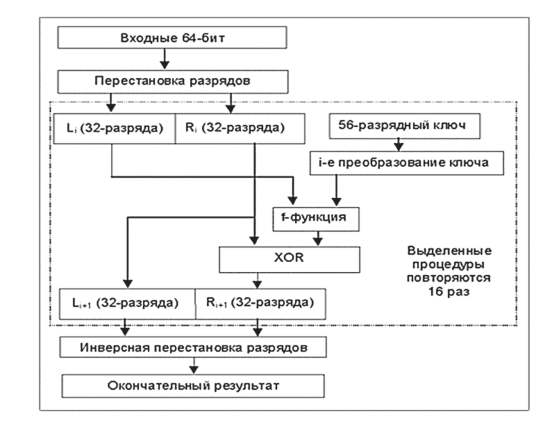


Рисунок 1 – общая схема алгоритма DES

Советском Союзе в качестве стандарта на блочные алгоритмы шифрования с закрытым ключом в 1989 г. был принят ГОСТ 28147–89 [31]. ГОСТ предусматривает три режима шифрования (простая замена, гаммирование, гаммирование с обратной связью) и один режим выработки имитовставки.

Первый из режимов шифрования предназначен для шифрования ключевой информации и не может использоваться для шифрования других данных, для этого предусмотрены два других режима. Режим выработки имитовставки (криптографической контрольной комбинации) предназначен для имитозащиты шифруемых данных, т. е. для их защиты от случайных или преднамеренных несанкционированных изменений.

Шифр ГОСТ 28147–89 построен по тем же принципам, что и американский DES, однако по сравнению с DES первый более удобен для программной реализации. В ГОСТ 28147–89 применяется более длинный ключ – 256 битов, здесь используются 32 раунда шифрования. Таким образом, основные параметры алгоритма криптографического преобразования данных ГОСТ 28147–89: размер блока составляет 64 бита, размер ключа – 256 битов, количество раундов – 32. Алгоритм представляет собой классическую сеть Фейстеля. Шифруемый блок данных разбивается на две одинаковые части, правую R и левую L. Правая часть складывается по модулю 232 с подключом раунда и посредством принятого алгоритма шифрует левую часть. Перед следующим раундом левая и правая части меняются местами. Такая структура позволяет использовать один и тот же алгоритм как для зашифрования, так и для расшифрования блока (рис. 5.7).

Таким образом, в алгоритме используются следующие операции:

• сложение слов по модулю 232: правый блок (Ri) складывается по модулю 232 с текущим подключом (Ki);

• циклический сдвиг слова влево на указанное число битов;

• побитовое сложение по модулю 2 (XOR);

• замена (подстановка в блоке S) по таблице.

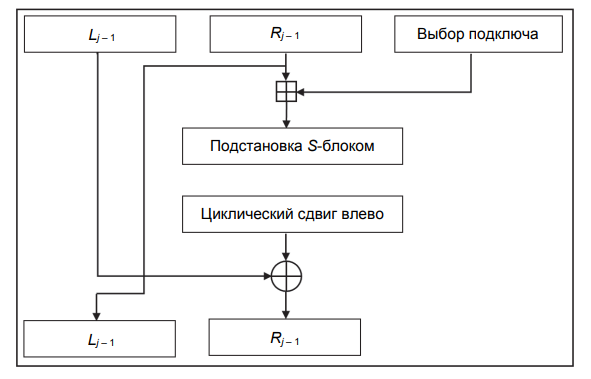


Рисунок 2 – общая схема алгоритма ГОСТ 28147–89

Основой алгоритма является сеть Фейстеля с 16 раундами.

Длина блока равна 64 битам, ключ может иметь любую длину в пределах 448 битов. Хотя перед началом любого зашифрования выполняется сложная фаза инициализации, само зашифрование данных выполняется достаточно быстро.

Алгоритм предназначен в основном для приложений, в которых ключ меняется нечасто, к тому же существует фаза начального рукопожатия, во время которой происходит аутентификация сторон. Blowfish характеризуется более высокой скоростью обработки блоков, чем DES.

Алгоритм состоит из двух частей: расширение ключа и зашифрование/расшифрование данных. Расширение ключа преобразует ключ длиной, по крайней мере, 448 битов в несколько массивов подключей общей длиной 4168 байтов.

Каждый раунд состоит из перестановки, зависящей от ключа, и подстановки, зависящей от ключа и данных.

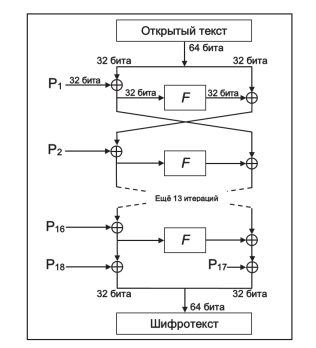


Рисунок 3 – общая схема алгоритма Blowfish

**Практическое задание**

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;

Для измерения скорости операций зашифрования/расшифрования был использован модуль *encoding\_time*. Для оценки скорости возьмем исходное сообщение длиной входящего символа, время выполнения операций представлено на рисунке 1.



Рисунок 1 – Время зашифрования/расшифрования

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

Метод шифрования и ключевая информация – в соответствии с вариантом из таблицы 1.

Таблица 1­– Задание по варианту

|  |  |  |
| --- | --- | --- |
| Вариант | Алгоритм | Ключ |
| 1 | DES | Первые 8 символов собственных фамилииимени |

Алгоритм строится на основе сети Фейстеля.

Функция encrypt принимает на вход шифруемый текст и выполняет зашифрование. Код функции представлен на рисунке 2 и в листинге 1 представлен код функции.

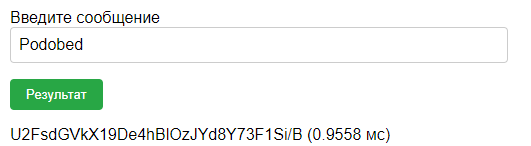


Рисунок 2 – Результат шифрования алгоритма DES

|  |
| --- |
| def encrypt(plain\_text):      cipher = DES.new(ALGORITHM\_KEY, DES.MODE\_ECB)      padded\_text = plain\_text + ' ' \* (8 - len(plain\_text) % 8)      encrypted\_text = cipher.encrypt(padded\_text.encode())      return encrypted\_text.hex() |

Листинг 1 – код шифрования алгоритма DES

Для дешифрования была разработана функция decrypt. Код функции представлен на рисунке 3 и результат выполнения алгоритма DES представлен в листинге 2.

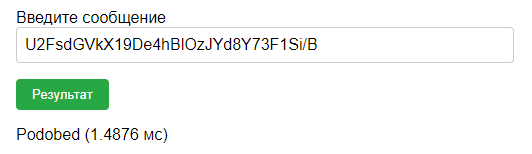


Рисунок 3 – Результат дешифрования алгоритма DES

|  |
| --- |
| def decrypt(encrypted\_text):      cipher = DES.new(ALGORITHM\_KEY, DES.MODE\_ECB)      decrypted\_text = cipher.decrypt(bytes.fromhex(encrypted\_text)).decode().strip()      return decrypted\_text |

Листинг 2 – код шифрования алгоритма DES

Эффект лавины означает, что изменение значения малого количества битов во входном тексте или ключе ведет к «лавинному» изменению значений выходных битов шифротекста. Другими словами, это зависимость всех (или хотя бы половины) выходных битов от каждого входного бита. Результат представлен на рисунке 4 и код представлен в листинге 3.

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

Листинг 3 – Замена символа на случайны

Для оценки лавинного эффекта зашифруем алгоритмом DES с использованием слабым ключом “0b0E0E”, изменим один символ в исходном тексте, и сравним количество изменённых символов.

Зашифрование изначального текста с изначальным ключом представлено на рисунке 4.

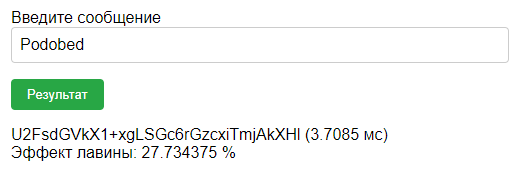


Рисунок 4 – Лавинный эффект с использованием слабым ключом

Как видно из рисунка, при одном зашифровании алгоритмом DES лавинный эффект составляет примерно 27% – то есть в шифротексте, в сравнении с открытым текстом, изменена примерно одну четвёртую бита.

Далее изменим один символ открытого текста и оценим, насколько изменился шифротекст. Данные действия представлены на рисунке 5.

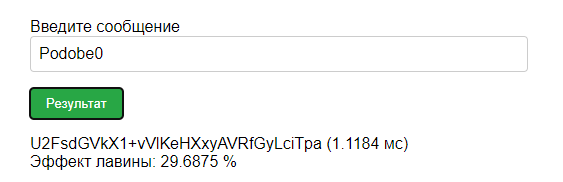


Рисунок 5 – Лавинный эффект, текст изменён на один символ

То есть, при изменении всего одного символа открытого текста, в шифротексте изменяется пару битов, то есть примерно на 2% больше прошлого всех битов текста.

Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные первоначальные ключи являются слабыми. Если все биты каждой половины равны 0 или 1, то для всех раундов алгоритма используется один и тот же ключ. Это может произойти, если ключ состоит из одних 1, из одних 0, или если одна половина ключа состоит из одних 1, а другая – из одних 0.

Оценим влияние полу слабого ключа “0b01FE”на лавинный эффект. Вывод функции подсчёта измененных битов представлен на рисунке 6.

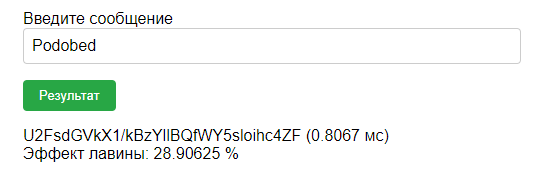


Рисунок 6 – Лавинный эффект с полу слабым ключом

При использовании полу слабого ключа лавинный эффект составляет всего 28%. Это происходит за счёт того, что при использовании полу слабого ключа раундовые ключи состоят только из единиц или нулей, что при использовании операции XOR с шифротекстом не будут изменять данный шифротекст. Поэтому изменяется меньшее количество битов, и шифрование получается более слабым.

При использовании слабого ключа, лавинный эффект примерно равен 27%, что немного выше, чем при полу слабом ключе, но так же ниже, чем с обычным ключом. Это происходит из-за того, что на некоторых раундах раундовые ключи переводят открытый текст в шифротекст, который идентичен открытому тексту. За счёт большого количества раундов, на выходе алгоритма мы не получим исходный открытый текст, однако полуслабые ключи, по сути, уменьшают количество раундов зашифрования, что негативно сказывается на криптостойкости алгоритма.

Оценить степень сжатия (используя любой доступный архиватор) открытого текста и соответствующего зашифрованного текста. Дать пояснения к полученному результату представлен на рисунке 7.

Файлы с исходным сообщением, состоящим из 5000 символов, и зашифрованным сообщением были сжаты с помощью архиватора WinRar. Размер сжатого файла с исходным сообщением стал меньше на 30%, в то время как архив с зашифрованным стал весить меньше на 25%.

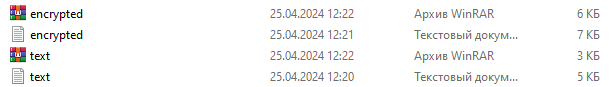


Рисунок 7 – Размеры исходных и заархивированных файлов

**Вывод:** блочные шифры широко применяются для защиты конфиденциальных данных, таких как пароли, кредитные карты, медицинские записи и другие.

Разработка и использование приложений для реализации блочных шифров требует глубоких знаний в области криптографии, математических основ криптографии и программирования. Для эффективной реализации и использования блочных шифров необходимо учитывать такие факторы, как скорость шифрования, сложность алгоритма, длина ключа и стойкость шифра.