МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №5**

**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2023

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

**Теоретическая часть:**

В 1972 г. Национальное бюро стандартов США (ныне – Национальный институт стандартов и технологии, National Institute of Standarts & Technology – NIST) инициировало программу защиты каналов связи и компьютерных данных. Одна из целей – разработка единого стандарта криптографического шифрования. Основными критериями оценки алгоритма являлись следующие:

• алгоритм должен обеспечить высокий уровень защиты;

• алгоритм должен быть понятен и детально описан;

• криптостойкость алгоритма должна зависеть только от ключа;

• алгоритм должен допускать адаптацию к различным применениям;

• алгоритм должен быть разрешен для экспорта.

В качестве начального варианта нового алгоритма рассматривался Lucifer – разработка компании IBM начала семидесятых годов. В основе указанного алгоритма использовались два запатентованных в 1971 г. Хорстом Фейстелем (Horst Feistel) устройства, реализующие различные алгоритмы шифрования, позже получившие название шифр (сеть) Фейстеля (Feistel Cipher, Feistel Network). В первой версии проекта Lucifer сеть Фейстеля не использовалась.

После многочисленных согласований, специальных конференций, где рассматривались в основном вопросы криптостойкости алгоритма, подлежащего утверждению в качестве федерального стандарта, в ноябре 1976 г. был утвержден стандарт DES (Data Encryption Standard – стандарт шифрования данных). Предполагалось, что стандарт будет реализовываться только аппаратно.

В 1981 г. ANSI одобрил DES в качестве стандарта для публичного использования (стандарт ANSI Х3.92), назвав его алгоритмом шифрования данных (Data Encryption Algorithm – DEA).

В 1987 г. были разработаны алгоритмы FEAL и RC2. Сети Фейстеля получили широкое распространение в 1990-е гг. – в годы появления таких алгоритмов, как Blowfish (1993), TEA (1994), RC5 (1994), CAST-128 (1996), XTEA (1997), XXTEA (1998), RC6 (1998) и др. На основе сети Фейстеля в 1990 г. в СССР был принят в качестве ГОСТ 28147–89 стандарт шифрования.

Предполагалось, что DES будет сертифицироваться каждые 5 лет. Срок действия последнего сертификата на территории США истек практически к концу ХХ в. К тому времени DES был вскрыт «лобовой атакой».

В 1998 г. NIST объявил конкурс на новый стандарт, который завершился в 2001 г. принятием AES (Advanced Encryption Standard).

Все перечисленные 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.

Основные требования к шифрам рассматриваемого класса можно сформулировать следующим образом:

• даже незначительное изменение исходного сообщения должно приводить к существенному изменению зашифрованного сообщения;

• устойчивость к атакам по выбранному тексту;

• алгоритмы зашифрования/расшифрования должны быть реализуемыми на различных платформах;

• алгоритмы должны базироваться на простых операциях;

• алгоритмы должны быть простыми для написания кода, вероятность появления программных ошибок должна быть низкой;

• алгоритмы должны допускать их модификацию при переходе на иные требования по уровню криптостойкости.

**Сеть Фейстеля**

Сеть Фейстеля является одним из методов построения блочных шифров.

Само название конструкции Фейстеля (сеть) означает ее ячеистую топологию. Формально одна ячейка сети соответствует одному раунду зашифрования или расшифрования сообщения.

При зашифровании сообщение разбивается на блоки одинаковой (фиксированной) длины (как правило – 64 или 128 битов).

Полученные блоки называются входными. В случае если длина входного блока меньше, чем выбранный размер, блок удлиняется установленным способом.

Каждый входной блок шифруемого сообщения изначально делится на два подблока одинакового размера: левый (L0) и правый (R0). Далее в каждом i-м раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля:

Изображение выглядит как текст

Автоматически созданное описание

По какому-либо математическому правилу вычисляется раундовый ключ *Ki*. В приведенном выражение знак «+» соответствует поразрядному суммированию на основе «XOR». 8

Расшифрование происходит так же, как и зашифрование, с той лишь разницей, что раундовые ключи будут использоваться в обратном порядке по отношению к зашифрованию.

**Практическая часть**

На рисунке 1 представлены варианты заданий. По заданию пятого варианта необходимо использовать алгоритм DES-EDE2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Варианты заданий

При использовании DES-EDE2, каждый блок данных шифруется дважды, первый раз с помощью одного ключа, затем дешифруется с помощью другого ключа и, наконец, шифруется еще раз с использованием первого ключа.

На лабораторной работе было решено применить библиотеку Crypto.Cipher для шифрования исходного текста. В приложение А представлен код приложения.

Вначале исходная строка разбивается на блоки фиксированной длины, при этом последний блок дополняется до заданной длины. Если используется режим шифрования ECB, то входной поток разбивается на блоки по 8 байт (64 бит) по умолчанию. В данном случае режим дополнения является Zeros, то есть последний блок дополняется нулями. Также происходит хеширование ключевой информации с использованием алгоритма MD5.

Данное программное средство реализует алгоритм DES-EDE2, который включает в себя операции шифрования, расшифрования и еще одно шифрование, при этом на первом и третьем шагах используется один и тот же ключ. Реализация данного алгоритма представлена на рисунке 3

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2– Реализация алгоритма DES-EDE2

. Для процедуры зашифрования и расшифрования в данном программном средстве используются ключи key1 и key2. При расшифровании ключи используются в обратном порядке по сравнению с процедурой зашифрования.

Алгоритм шифрования исходного текста основан на маршрутной перестановке. Сначала исходная строка разбивается на подстроки длиной k (ключ), после чего символы каждой подстроки последовательно считываются: сначала первый символ каждой подстроки, затем второй и т.д. по возрастанию. Реализация данного алгоритма представлена на рисунке 2.

Результат выполнения данного консольного приложения можно увидеть на рисунке 4.

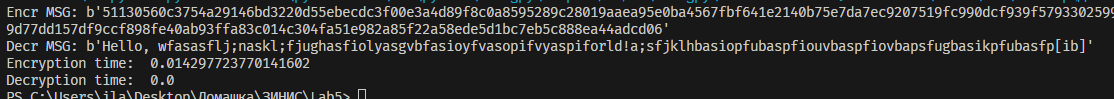


Рис. 3 - Результат работы приложения

Из проведенных оценок скорости выполнения операций зашифрования и расшифрования алгоритма DES-EDE2 можно сделать вывод о том, что расшифрование происходит значительно быстрее зашифрования из-за использования уже кэшированных данных. Кроме того, высокая скорость выполнения алгоритма обусловлена малой длиной ключей key1 и key2.

При проведении шифрования алгоритмом DES-EDE2 было обнаружено, что на каждом из трех этапов шифрования количество символов в шифротексте вырастает примерно в два раза. Это является результатом постоянно возрастающего "лавинного эффекта", который вызывает рост размера файла с шифротекстом.

Так же был проверен "лавинный эффект", путем изменения ключа. Результат представлен на рисунке 4.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 4 –Лавинный эффект

Полный листинг программного обеспечения изложен в приложение А.

**Вывод:**

В процессе выполнения лабораторной работы мы получили практические навыки создания и использования программ для работы с блочными шифрами. Мы также провели анализ криптостойкости этих шифров, оценили скорость выполнения операций зашифрования/расшифрования и сделали выводы на основе полученных результатов.

**Контрольные вопросы**

1. Какие простейшие операции применяются в блочных алгоритмах шифрования?

Побитовое сложение XOR (исключающее ИЛИ) — это операция, которая применяется для комбинирования двух битов. Результатом операции XOR является "1" только в том случае, если один из двух битов равен "1", но не оба. Используется для выполнения простой замены битов в открытом тексте.

Перестановка – это операция, которая переставляет порядок битов или байтов в блоке открытого текста. Перестановка может быть фиксированной, то есть использоваться всегда в одном и том же порядке, или зависеть от ключа.

Замена – это операция, которая заменяет один блок данных на другой, используя некоторую таблицу замены (S-блок). S-блоки используются для сильной замены битов в открытом тексте и для усложнения анализа криптоаналитиками.

Циклический сдвиг – это операция, которая сдвигает блок битов или байтов на заданное количество позиций влево или вправо. Эта операция используется для усиления лавинного эффекта и усложнения анализа криптоаналитиками.

Обратная перестановка – это операция, которая обратна перестановке и используется для восстановления исходного порядка битов или байтов после прохождения блока через несколько раундов шифрования.

2. В чем отличие блочных алгоритмов шифрования от потоковых?

Блочные алгоритмы шифрования обрабатывают данные порциями фиксированного размера (обычно блоки состоят из 64 или 128 битов). Перед тем, как начать шифрование, данные разбиваются на блоки, которые затем последовательно обрабатываются алгоритмом. Каждый блок зашифровывается по отдельности, используя один и тот же ключ шифрования. Примерами блочных алгоритмов шифрования являются DES, AES, Blowfish.

Потоковые алгоритмы шифрования, напротив, обрабатывают данные бит за битом, то есть каждый бит входного потока обрабатывается отдельно. Они обычно используются для шифрования потоковых данных, таких как аудио или видео, а также для шифрования интернет-трафика. Ключ шифрования используется для генерации псевдослучайного потока битов, который затем складывается с исходными данными для их зашифрования. Примерами потоковых алгоритмов шифрования являются RC4, ChaCha20.

3. Что понимается под «раундом» алгоритма шифрования?

В криптографии термин "раунд" (англ. round) относится к одному циклу шифрования или дешифрования в блочных алгоритмах шифрования. Во время каждого раунда входной блок данных проходит через определенную последовательность операций (преобразований), направленных на изменение его состояния, что приводит к созданию шифротекста.

4. Охарактеризовать и привести формальное описание сети Фейстеля.

Сеть Фейстеля – это тип блочных алгоритмов шифрования, который был предложен Хорстом Фейстелем в 1973 году. Она используется в таких известных алгоритмах, как DES, Blowfish, и Twofish.

Основная идея сети Фейстеля заключается в том, что исходный блок данных разбивается на две половины, и затем происходит ряд раундов, каждый из которых преобразует правую половину блока на основе левой половины и некоторого ключа. После всех раундов происходит обмен местами левой и правой половин блока и повторение процедуры до тех пор, пока не будет достигнуто нужное количество раундов или пока не будет достигнут конечный результат.

Формально, сеть Фейстеля может быть описана следующим образом:

Разделение блока данных на две половины: *L0* и *R0*.

Повторение *N* раундов:

1) Вычисление функции *f(Ri, Ki)*, где *Ri* - правая половина блока, *Ki* - ключ для *i*-го раунда, *f* - некоторая нелинейная функция, которая обычно является перестановкой битов.

2) Вычисление *Li+1 = Ri*

3) Вычисление *Ri+1 = Li XOR f(Ri, Ki)*

Обмен местами левой и правой половин блока: *Ln+1 = Rn* и *Rn+1 = Ln*

Ключи для каждого раунда могут быть получены из основного ключа шифрования с помощью алгоритма расширения ключа, который преобразует основной ключ в набор ключей, каждый из которых используется для шифрования соответствующего раунда.

Преимуществом сети Фейстеля является её простота, скорость работы и высокий уровень безопасности, который обеспечивается использованием нелинейной функции *f*.

5. Какие стандартные операции используются в блочных алгоритмах шифрования?

Побитовые операции (XOR, AND, OR) – используются для комбинирования блоков данных и ключей шифрования.

Перестановки битов – используются для изменения порядка битов в блоках данных и ключах шифрования.

Замены – используются для замены блоков битов на другие блоки битов в блоках данных и ключах шифрования.

Функции замешивания – используются для смешивания блоков данных и ключей шифрования с целью усложнения атак на шифр.

Циклы раундов – используются для повторения стандартных операций над блоками данных и ключами шифрования, увеличивая тем самым уровень безопасности шифра.

6. Дать пояснение принципам реализации «лавинного» эффекта.

Принцип реализации "лавинного" эффекта заключается в том, что малейшее изменение входных данных должно приводить к значительным изменениям в выходных данных. Это достигается за счет того, что любая изменение входных данных должна "распространяться" на все последующие операции алгоритма и влиять на результаты этих операций.

Чтобы достичь лавинного эффекта, блочные алгоритмы шифрования обычно используют операции, которые максимально "размазывают" входные данные по всему блоку и включают в себя нелинейные компоненты. Например, операции подстановки и перестановки в алгоритмах DES, а также операции нелинейного преобразования в алгоритме AES.

7. Выбрать два произвольных блочных алгоритма. В чем состоят отличия между ними?

AES (Advanced Encryption Standard) – это симметричный блочный алгоритм шифрования, используемый для защиты данных. Он использует ключи длиной 128, 192 или 256 битов и работает с блоками длиной 128 битов. Основные отличия AES от DES:

• AES использует длинные ключи (128, 192 или 256 битов), в то время как DES использует ключи длиной только 56 битов. Благодаря этому AES обеспечивает более высокий уровень безопасности.

• AES использует 10, 12 или 14 раундов шифрования в зависимости от длины ключа, тогда как DES использует только 16 раундов.

• AES использует более сложные операции замены и перестановки, чем DES, что делает его более устойчивым к атакам.

DES (Data Encryption Standard) – это симметричный блочный алгоритм шифрования, разработанный в 1970-х годах. Он использует ключи длиной 56 битов и работает с блоками длиной 64 бита. Основные отличия DES от AES:

• DES использует короткие ключи длиной 56 битов, что делает его менее безопасным по сравнению с AES.

• DES использует 16 раундов шифрования, в то время как AES использует только 10-14 раундов в зависимости от длины ключа.

• DES использует более простые операции замены и перестановки, чем AES.

8. Представить графически и пояснить функционал одного раунда блочного алгоритма DES (АES, ГОСТ 28147–89, Blowfish).

1) Начальная перестановка (IP): биты блока входных данных переставляются в соответствии с определенной таблицей перестановки.

2) Разделение блока на левую и правую части: входной блок размером 64 бита разбивается на две части по 32 бита каждая.

3) Функция расширения (E): правая часть блока размером 32 бита расширяется до 48 бит с помощью определенной таблицы расширения.

4) Ключевая функция (K): 48-битный ключ, полученный из основного ключа DES в соответствии с определенным алгоритмом, складывается по модулю 2 с расширенной правой частью блока.

5) Функция подстановки (S): результат функции K разбивается на 8 блоков по 6 бит, каждый из которых заменяется на 4-битное значение, выбираемое из соответствующей таблицы замены.

6) Функция перестановки (P): результат функции S подвергается перестановке с помощью определенной таблицы перестановки.

7) XOR с левой частью: результат функции P складывается по модулю 2 с левой частью блока.

8) Обмен правой и левой частями блока: левая часть блока становится новой правой частью, а правая - новой левой.

Изображение выглядит как диаграмма

Автоматически созданное описание

9. Сколько можно реализовать (теоретически) разновидностей алгоритма 3DES?

Теоретически, можно реализовать несколько разновидностей алгоритма 3DES, в зависимости от использования различных ключей и режимов работы.

3DES (Triple DES) – это блочный шифр, который использует блоки размером 64 бита и работает с тремя ключами, каждый из которых имеет длину 56 бит. Существуют два режима работы 3DES: EDE (Encrypt-Decrypt-Encrypt) и EEE (Encrypt-Encrypt-Encrypt).

Количество возможных вариантов 3DES можно оценить, учитывая следующее:

• Существует три ключа, каждый из которых имеет длину 56 бит. Это означает, что можно создать 256 различных ключей для каждой из трех позиций.

• Существует два режима работы: EDE и EEE.

• Кроме того, можно изменить порядок ключей в режиме EDE, что дает еще один вариант.

Таким образом, общее число возможных вариантов 3DES можно оценить как: 256 × 256 × 256 × 2 × 2 = 2168 × 4.

Такое большое число означает, что количество возможных вариантов 3DES настолько велико, что практически невозможно перебрать их все.

10. Какие факторы влияют на стойкость блочного алгоритма шифрования?

Длина ключа: чем длиннее ключ, тем сложнее его подобрать методом перебора.

Алгоритм перестановки: чем более сложный алгоритм перестановки используется в блочном алгоритме шифрования, тем сложнее подобрать ключ методом перебора.

Алгоритм замены: чем более сложный алгоритм замены используется в блочном алгоритме шифрования, тем сложнее подобрать ключ методом перебора.

Количество раундов: чем больше раундов используется в блочном алгоритме шифрования, тем сложнее его взломать методом перебора.

Лавинный эффект: блочный алгоритм шифрования должен проявлять лавинный эффект при изменении одного бита входного сообщения. Это означает, что изменение даже одного бита входного сообщения должно приводить к значительному изменению выходного сообщения.

Вектор инициализации (IV): блочный алгоритм шифрования должен использовать случайный и уникальный IV для каждого зашифрованного сообщения, чтобы избежать повторения ключевых потоков.

Противодействие криптоанализу: блочный алгоритм шифрования должен быть устойчив к различным методам криптоанализа, таким как атаки по времени, по памяти, дифференциальный криптоанализ, линейный криптоанализ и другим подобным методам.

11. В чем состоит сущность дифференциального криптоанализа?

Сущность дифференциального криптоанализа заключается в анализе различий (дифференциалов) между парами открытого текста и соответствующих им шифротекстов, полученных с использованием одного и того же ключа.

Дифференциальный криптоанализ заключается в поиске таких пар открытого текста, для которых существует большая вероятность получения определенного различия в соответствующих им шифротекстах. Затем на основе этой информации атакующий может использовать статистические методы для вычисления ключа шифрования.

При проведении дифференциального криптоанализа используется понятие дифференциальной характеристики, которая описывает вероятность получения определенного различия в шифротекстах для заданных пар открытого текста. Хороший блочный шифр должен иметь диффузионные и конфузионные свойства, которые затрудняют проведение дифференциального криптоанализа.

Дифференциальный криптоанализ является одним из наиболее эффективных методов взлома блочных шифров, и поэтому он широко используется для оценки стойкости блочных шифров и разработки новых криптографических алгоритмов.

12. В чем состоит сущность линейного криптоанализа?

Сущность линейного криптоанализа заключается в анализе линейных зависимостей между битами ключа, открытым текстом и соответствующим шифротекстом. Линейный криптоанализ ищет линейную функцию (или комбинацию функций), которая наилучшим образом аппроксимирует зависимость между входными и выходными данными алгоритма шифрования.

Для проведения линейного криптоанализа атакующий использует линейные приближения, которые описывают зависимость между битами ключа и битами шифротекста с некоторой вероятностью. Далее, атакующий строит линейное уравнение, которое приближает эту зависимость, и на основе этого уравнения вычисляет биты ключа.

Линейный криптоанализ особенно эффективен для блочных шифров с небольшой длиной ключа, где линейные приближения могут описывать значительную часть зависимостей между ключом и шифротекстом. Однако, для более современных блочных шифров, которые имеют более длинный ключ, линейный криптоанализ обычно не является таким эффективным методом взлома, как дифференциальный криптоанализ.

Линейный криптоанализ также используется для оценки стойкости блочных шифров и разработки новых криптографических алгоритмов.

13. Какие ключевые комбинации относятся к слабым (к полуслабым) и почему?

Из-за того, что первоначальный ключ изменяется при получении подключа для каждого раунда алгоритма, определенные первоначальные ключи являются слабыми. Вспомним, что первоначальное значение разделяется на две половины, каждая из которых сдвигается независимо. 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.

Изображение выглядит как стол

Автоматически созданное описание

Кроме того, некоторые пары ключей при зашифровании переводят открытый текст в идентичный шифртекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары. Это происходит из-за метода, используемого DES для генерации подключей: вместо 16 различных подключей эти ключи генерируют только два различных подключа. Исследование блочных шифров 79 В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называются полуслабыми.



14. Где применяются блочные криптоалгоритмы?

Компьютерная безопасность: блочные криптоалгоритмы используются для шифрования конфиденциальной информации в компьютерных сетях, интернет-передаче данных, защиты файлов, дисков и т.д.

Финансы: блочные криптоалгоритмы используются для защиты банковских транзакций и финансовых операций, включая электронные платежи, кредитные карты, банковские переводы и т.д.

Телекоммуникации: блочные криптоалгоритмы используются для защиты конфиденциальности телефонных звонков и сообщений, а также в сотовых сетях.

Государственная безопасность: блочные криптоалгоритмы широко используются для защиты государственной тайны и другой конфиденциальной информации.

Медицина: блочные криптоалгоритмы могут использоваться для защиты конфиденциальных медицинских данных пациентов, таких как медицинские записи, результаты анализов, истории болезни и т.д.

Промышленность: блочные криптоалгоритмы используются для защиты конфиденциальной промышленной информации, такой как патенты, проекты, технологии, изобретения и т.д.

Армия и военная безопасность: блочные криптоалгоритмы широко используются в военных целях для защиты конфиденциальных данных, таких как планы операций, геополитические данные, военные сообщения и т.д.

Личная безопасность: блочные криптоалгоритмы могут использоваться для защиты личных данных, таких как пароли, пин-коды, личная информация и т.д.

**Приложение А**

import binascii

import binascii

from Crypto.Cipher import DES

import time

import hashlib

def pad(data):

    # дополнение данных до кратности 8 байтам

    padding\_len = 8 - (len(data) % 8)

    data += bytes([padding\_len] \* padding\_len)

    return data

def unpad(data):

    # удаление дополнения данных

    padding\_len = data[-1]

    return data[:-padding\_len]

def des\_encrypt(key, data):

    # шифрование блока данных

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

    return cipher.encrypt(data)

def des\_decrypt(key, data):

    # дешифрование блока данных

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

    return cipher.decrypt(data)

def triple\_des\_encrypt(key, data):

    # шифрование данных с использованием Triple DES (DES-EDE2)

    key1, key2 = key[:8], key[8:16]

    data = pad(data)

    data = des\_encrypt(key1, data)

    data = des\_decrypt(key2, data)

    data = des\_encrypt(key1, data)

    return data

def triple\_des\_decrypt(key, data):

    # дешифрование данных с использованием Triple DES (DES-EDE2)

    key1, key2 = key[:8], key[8:16]

    data = des\_decrypt(key1, data)

    data = des\_encrypt(key2, data)

    data = des\_decrypt(key1, data)

    data = unpad(data)

    return data

# пример использования

key = "my secret key"

key\_bytes = key.encode("utf-8") # переводим строку в байты

key\_hash = hashlib.md5(key\_bytes).hexdigest() # хэшируем байты и переводим хэш в шестнадцатеричное представление

key = binascii.unhexlify(key\_hash)

data = b'Hello, from univer in my python code'

start\_time = time.time()

encrypted\_data = triple\_des\_encrypt(key, data)

end\_time = time.time()

encryption\_time = end\_time - start\_time

print("MSG",data)

print("-" \* 20)

print("Encr MSG:",binascii.hexlify(encrypted\_data))

start\_time = time.time()

decrypted\_data = triple\_des\_decrypt(key, encrypted\_data)

end\_time = time.time()

decryption\_time = end\_time - start\_time

print("Decr MSG:",decrypted\_data)

print("-" \* 20)

print("Encryption time: ", encryption\_time)

print("Decryption time: ", decryption\_time)

print("-" \* 20)

print("Лавиный эффект")

print("-" \* 20)

d=0

keyka = "my secret key"

for d in range(0,len(key)):

    d=d+1

    keyka = keyka[:-1]

    key = keyka

    key\_bytes = key.encode("utf-8") # переводим строку в байты

    key\_hash = hashlib.md5(key\_bytes).hexdigest() # хэшируем байты и переводим хэш в шестнадцатеричное представление

    key = binascii.unhexlify(key\_hash)

    encrypted\_data = triple\_des\_encrypt(key, data)

    print("Key:",keyka)

    print("Encr MSG:",binascii.hexlify(encrypted\_data))

    print("-" \* 20)