Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет к лабораторной работе**:

«Исследование блочных шифров»

Выполнил:

студент 3 курса 8 группы

специальности ПОИБМС

Романович Н.Р.

Минск 2020

**1 Теоретические сведения**

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

* 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 1971 г. Хорстом Фейстелем (Horst Feistel) устройства, реализующие различные алгоритмы шифрования, позже получившие 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 году были разработаны алгоритмы FEAL и RC2. Сети Фейстеля получили широкое распространение в 1990-е годы — в годы появления таких алгоритмов, как Blowfish (1993), TEA (1994), RC5 (1994), CAST-128 (1996), XTEA (1997), XXTEA (1998), RC6 (1998) и других. На основе сети Фейстеля в 1990 году 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 20 века. 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, 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.

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 бит).

Полученные блоки называются входными. В случае, если длина входного блока меньше, чем выбранный размер, то 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-ом раунде выполняются преобразования в соответствии с формальным представлением ячейки сети Фейстеля.

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

Расшифрование происходит так же, как и зашифрование, с той лишь разницей, что раундовые ключи будут использоваться в обратном порядке по отношению к зашифрованию. В своей статье Х. Фейстель описывает два блока преобразований с использованием функции *f* (Ri-1, Ki):

* блок подстановок (S-блок, англ. S-box);
* блок перестановок (P-блок, англ. P-box).

Блок подстановок состоит из:

* дешифратора, преобразующего n-разрядное двоичное число в одноразрядное сигнал по основанию 2n;
* внутреннего коммутатора;
* шифратора, преобразующего сигнала из одноразрядного 2n-ричного в nразрядный двоичный.

Блок перестановок изменяет положение цифр, т. е. является линейным устройством. Этот блок может иметь очень большое количество входов-выходов, однако в силу линейность является слабым местом преобразования с точки зрения криптостойкости.

Термин «блок» в оригинальной статье используется вместо термина «функция» вследствие того, что речь идет о блочном шифре.

При большом размере блоков шифрования (128 бит и более) реализация такой конструкции Фейстеля на 32-разрядных архитектурах может вызвать затруднения, поэтому применяются модифицированные варианты этой конструкции. Обычно используются сети с четырьмя ветвями. При большом размере блоков (128 бит и более) реализация сети Фейстеля на 32-разрядных архитектурах может вызвать затруднения, поэтому применяются модифицированные варианты сети. Такие модификации предусматривают использование не 2-х (L и R на рис. 5.1), а 4-х ветвей.

В основе криптостойкости блочных шифров лежит идея К. Шеннона в представлении составного шифра таким образом, чтобы от обладал двумя важными свойствами: рассеянием и перемешиванием. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом. Рассеивание подразумевает, что каждый символ (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. Алгоритм строится на основе сети Фейстеля. Входной блок данных, состоящий из 64 бит, 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. 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.

Выполненная перестановка означает, например, что первый бит входного блока сообщения будет размещен на 40-й позиции (цифра «1» выделена жирным), а 58-й (выделено жирным с подчеркиванием) – на первой и т.д. Из беглого анализа выполненной перестановки легко понять принцип. Алгоритм перестановки разрабатывался для облегчения загрузки блока входного сообщения в специализированную микросхему. Вместе с тем, эта операция придает некоторую "хаотичность" исходному сообщению, снижая возможность использования криптоанализа статистическими методами.

Вначале правая часть блока Ri расширяется до 48 битов, используя таблицу, которая определяет перестановку плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется «лавинным эффектом»).

После выполнения перестановки с расширением для полученного 48-битного значения выполняется операция XOR с 48-битным подключом Ki. Затем полученное 48-битное значение подается на вход блока подстановки S (от англ. Substitution – подстановка), результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках подстановки или восьми Sблоках (S-boxes). При выполнении этой операции 48 битов данных делятся на восемь 6-битных подблоков, каждый из которых по соответствующей таблице замен замещается четырьмя битами. Подстановка с помощью S-блоков является одним из важнейших этапом DES. Таблицы замен для этой операции специально спроектированы так, чтобы обеспечивать максимальную криптостойкость. В результате выполнения этого этапа получаются восемь 4-битных блоков, которые вновь объединяются в единое 32-битное значение. Далее полученное 32-битное значение обрабатывается с помощью перестановки Р (от англ. Permutation – перестановка), которая не зависит от используемого ключа.

Целью перестановки является максимальное переупорядочивание битов такое, чтобы в следующем раунде шифрования каждый бит с большой вероятностью обрабатывался другим S-блоком. И, наконец, результат перестановки объединяется с помощью операции XOR с левой половиной первоначального 64-битового блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд.

После выполнения 16-раундового зашифрования 64-битного блока данных осуществляется конечная перестановка (IP−1). Она является обратной к перестановке IP.

Один избыточный бит в ключе DES формируется, как видим, в соответствии с кодом простой четности. Этот код позволяет в кодовом слове (в нашем случае – в каждом байте ключа) обнаруживать ошибки, количество которых нечетно.

При расшифровании на вход алгоритма подается зашифрованный текст. Единственное отличие состоит в обратном порядке использования частичных ключей Ki. Ключ K16 используется в первом раунде, K1 – в последнем. После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был составлен из подблоков R16 и L16. Выходом этой стадии является расшифрованный текст.

Слабые и полуслабые ключи. 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.

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. В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называемые полуслабыми.

Дифференциальный криптоанализ базируется на таблице неоднородных дифференциальных распределений S-блоков в блочном шифре. Криптоанализ шифртекстов на основе рассматриваемого стандарта «работает» с парами шифртекстов, открытые тексты которых имеют определенные разности. Метод анализирует эволюцию этих разностей в процессе прохождения открытых текстов раундов DES при шифровании одним и тем же ключом.

Для DES термин «разность» определяется с помощью операции XOR. Затем, используя разности полученных шифртекстов, присваивают различные вероятности различным ключам. В процессе дальнейшего анализа следующих пар шифртекстов один из ключей станет наиболее вероятным. Это и есть правильный ключ.

Линейный криптоанализ. Для того, чтобы найти линейное приближение для DES нужно найти «хорошие» однораундовые линейные приближения и объединить их. Обратим внимание на S-блоки. У них 6 входных битов и 4 выходных. Входные биты можно объединить с помощью операции XOR 63 способами (26 - 1), а выходные биты – 15 способами. Теперь для каждого S-блока можно оценить вероятность того, что для случайно выбранного входа входная комбинация XOR равна некоторой выходной комбинации XOR. И т.д.

DES давно характеризуется низкой криптостойкостью: в январе 1999 г. закодированное посредством DES сообщение было взломано с помощью связанных через Internet в единую сеть 100 тыс. персональных компьютеров за 24 часа. Данному алгоритму присуща проблема так называемых «слабых» и «частично слабых» ключей.

Основное достоинство DES – относительно высокая скорость (из-за малой длины ключа); бесплатное распространение по всему миру; общедоступность и отсутствие необходимости лицензионных отчислений.

Cуществуют 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.

AES (Advanced Encryption Standard) – алгоритм шифрования, действующий в качестве государственного стандарта в США с 2001 года. В основу стандарта положен шифр Rijndael. Шифр Rijndael/AES (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа.

Основу Rijndael составляют так называемые линейно-подстановочные преобразования. В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т. е. не зависят ни от ключа, ни от данных.

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 бит, здесь используются 32 раунда шифрования.

Таким образом, основные параметры алгоритма криптографического преобразования данных ГОСТ 28147-89: размер блока составляет 64 бита, размер ключа – 256 бит, количество раундов – 32.

Алгоритм Blowfish. Основой алгоритма является сеть Фейстеля с 16 раундами. Длина блока равна 64 битам, ключ может иметь любую длину в пределах 448 бит. 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 боков, чем DES.

Алгоритм состоит из двух частей: расширение ключа и зашифрование/расшифрование данных. Расширение ключа преобразует ключ длиной, по крайней мере, 448 бит в несколько массивов подключей общей длиной 4168 байт. Каждый раунд состоит из перестановки, зависящей от ключа, и подстановки, зависящей от ключа и данных. Операциями являются XOR и сложение по модулю 232.

Элементы алгоритма:

* Р-массив, состоящий из восемнадцати 32-битных подключей: Р1, Р2, …, Р18;
* S-блоки: каждый из четырех 32-битных S-блоков содержит 256 элементов;
* функция F.

Следует также обратить внимание на еще одно обстоятельство: файл, который содержит зашифрованные данные, практически нельзя сжать. Это может служить одним из критериев поиска зашифрованных файлов, которые, как и обычные файлы, представляют собой набор случайных битов.

В криптографии, Twofish является симметричным ключом блочного шифра с размером блока 128 бит и ключевыми размерами до 256 бит. Это был один из пяти финалистов конкурса Advanced Encryption Standard, но он не был выбран для стандартизации. Twofish связано с более ранним блочным шифром Blowfish.

Отличительные особенности Twofish являются использованием предварительно вычисленным ключевыми зависящих от S-блоков, и относительно сложного ключа графика. Одна половина из п-битного ключа используется в качестве фактического ключа шифрования, а другая половина ключа н-битном используется для изменения алгоритма шифрования (ключ-зависимые S-боксы). Twofish заимствует некоторые элементы из других конструкций; например, псевдо-преобразование Адамара (PHT) от SAFER семейства шифров. Twofish имеет структуру Фейстеля, как DES.

Еще в 2000 году на большинстве программных платформ Twofish был немного медленнее, чем Rijndael (выбранного алгоритма Advanced Encryption Standard) для 128-битных ключей, но несколько быстрее 256-битных ключей. Но после того, как Rijndael был выбран в качестве Advanced Encryption Standard, Twofish стал намного медленнее, чем Rijndael на процессорах, которые поддерживают набор инструкций AES.

Twofish был разработан Брюс Шнайер, Джон Келси , Дуг Уайтинг , Дэвид Вагнер , Крис Холл , и Niels Ferguson ; «Расширенный Twofish команда» , которые собрались , чтобы выполнить дальнейший криптоанализ Twofish и других участников конкурса AES включена Стефан Лакс , Тадаёсите Коы и Майк Stay .

Twofish шифр не запатентован и эталонная реализация была помещена в общественном достоянии. В результате алгоритм Twofish свободен для любого использования без каких - либо ограничений. Это один из нескольких шифров, включенных в OpenPGP стандарт ( RFC 4880 ). Тем не менее, Twofish видел менее широкое применение, чем Blowfish, который был доступен дольше.

Serpent (с латыни — «змея») — симметричный блочный алгоритм шифрования.

Разработан Россом Андерсоном, Эли Бихамом и Ларсом Кнудсеном.

Алгоритм являлся одним из финалистов 2-го этапа конкурса AES. Как и другие алгоритмы, участвовавшие в конкурсе AES, Serpent имеет размер блока 128 бит и возможные длины ключа 128, 192 или 256 бит. Алгоритм представляет собой 32-раундовую SP-сеть, работающую с блоком из четырёх 32-битных слов. Serpent был разработан так, что все операции могут быть выполнены параллельно, используя 32 1-битных «потока».

При разработке Serpent использовался более консервативный подход к безопасности, нежели у других финалистов AES, проектировщики шифра считали, что 16 раундов достаточно, чтобы противостоять известным видам криптоанализа, но увеличили число раундов до 32, чтобы алгоритм мог лучше противостоять ещё неизвестным методам криптоанализа.

Став финалистом конкурса AES, алгоритм Serpent в результате голосования занял 2 место.

Шифр Serpent не запатентован и является общественным достоянием.

Алгоритм Serpent представляет собой SP-сеть, в которой весь блок данных длиной 128 бит на каждом раунде разбивается на 4 слова длиной по 32 бита. Все значения, использующиеся при шифровании, представляются битовыми потоками. Индексы бит пробегают значения от 0 до 31 для 32-битных слов, от 0 до 127 для 128-битных блоков, от 0 до 255 для 256-битных ключей и так далее. Для внутренних вычислений все биты величин представлены в прямом порядке (little-endian).

Serpent шифрует открытый текст P длиной 128 бит в шифротекст C длиной так же 128 бит за 32 раунда с помощью 33 подключей K0,…K32 длиной 128 бит. Длина используемого ключа может принимать различные значения, но для конкретики зафиксируем их длину в 128, 192 или 256 бит. Короткие ключи длиной менее 256 бит дополняются до полной длины в 256 бит.

Шифрование состоит из следующих основных шагов:

* начальная перестановка;
* 32 раунда, каждый из которых состоит из операции смешивания с 128-битным ключом (побитовое логическое исключающее «или»), табличная замена (S-box) и линейное преобразование. В последнем раунде линейное преобразование заменяется дополнительным наложением ключа;
* конечная перестановка;

Начальная и конечная перестановки не имеют какой-либо криптографической значимости. Они используются для упрощения оптимизированной реализации алгоритма и повышения вычислительной эффективности.

Расшифрование отличается от шифрования только тем, что должны быть использованы инверсные (обратные) таблицы замен, а также обратные линейные преобразования, но только для раундовой функции R. Ключи подаются в обратном порядке. Битовые операции раундовой функции расшифрования (в т. ч. замены и перестановки) должны проходить в порядке обратном порядку битовых операций раундовой функции шифрования.

При разработке и анализе алгоритма Serpent не было выявлено каких-либо уязвимостей в полной 32-раундовой версии. Но при выборе победителя конкурса AES это было справедливо и для остальных алгоритмов-финалистов.

По мнению создателей Serpent, алгоритм может быть взломан, только если будет создана новая мощная математическая теория.

Стоит отметить, что XSL-атака, если будет доказана эффективность её проведения, ослабит криптостойкость Serpent.

**2 Практическая часть**

В данной лабораторной работе необходимо было разработать авторское приложение с шифрованием/расшифрованием текста с помощью алгоритмов DES, 3DES, AES, TwoFish, Serpent.

Работа приложения с алгоритмом DES представлена на скриншотах ниже:

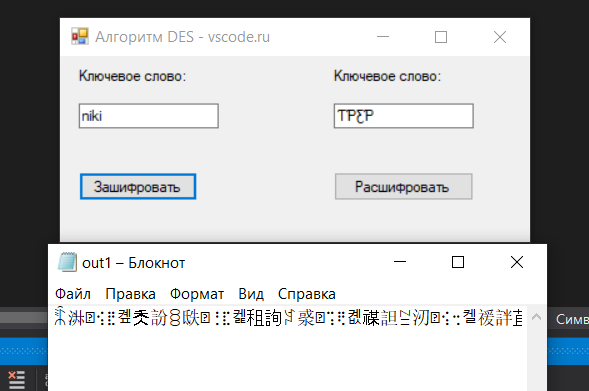


Рисунок 1. 1 Зашифровка сообщения

Расшифровка сообщения в приложении представлена на скриншоте:

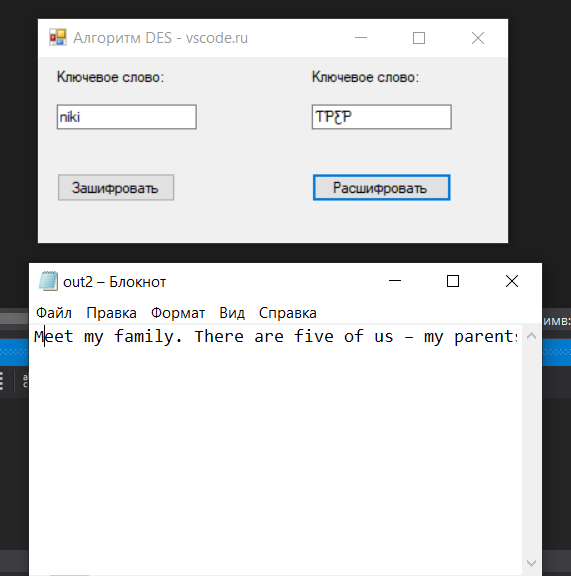


Рисунок 1. 2 Расшифровка сообщения

Работа приложения с алгоритмом 3DES представлена на скриншоте ниже:

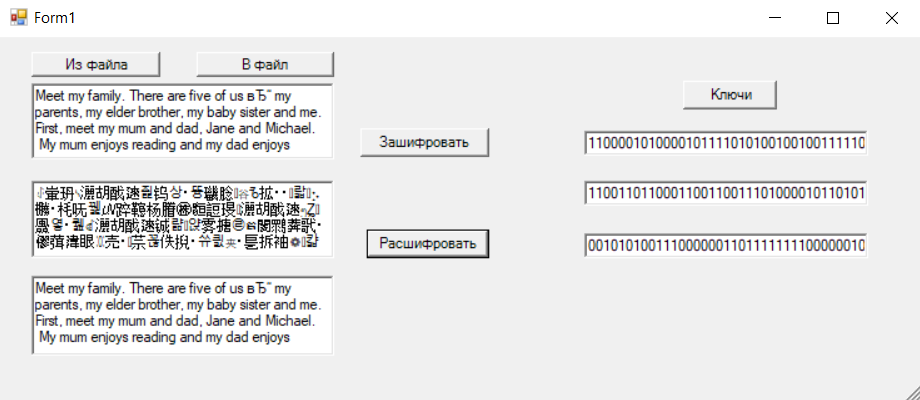


Рисунок 1. 3 Алгоритм 3DES

Работа приложения с алгоритмом AES представлена на скриншоте ниже:

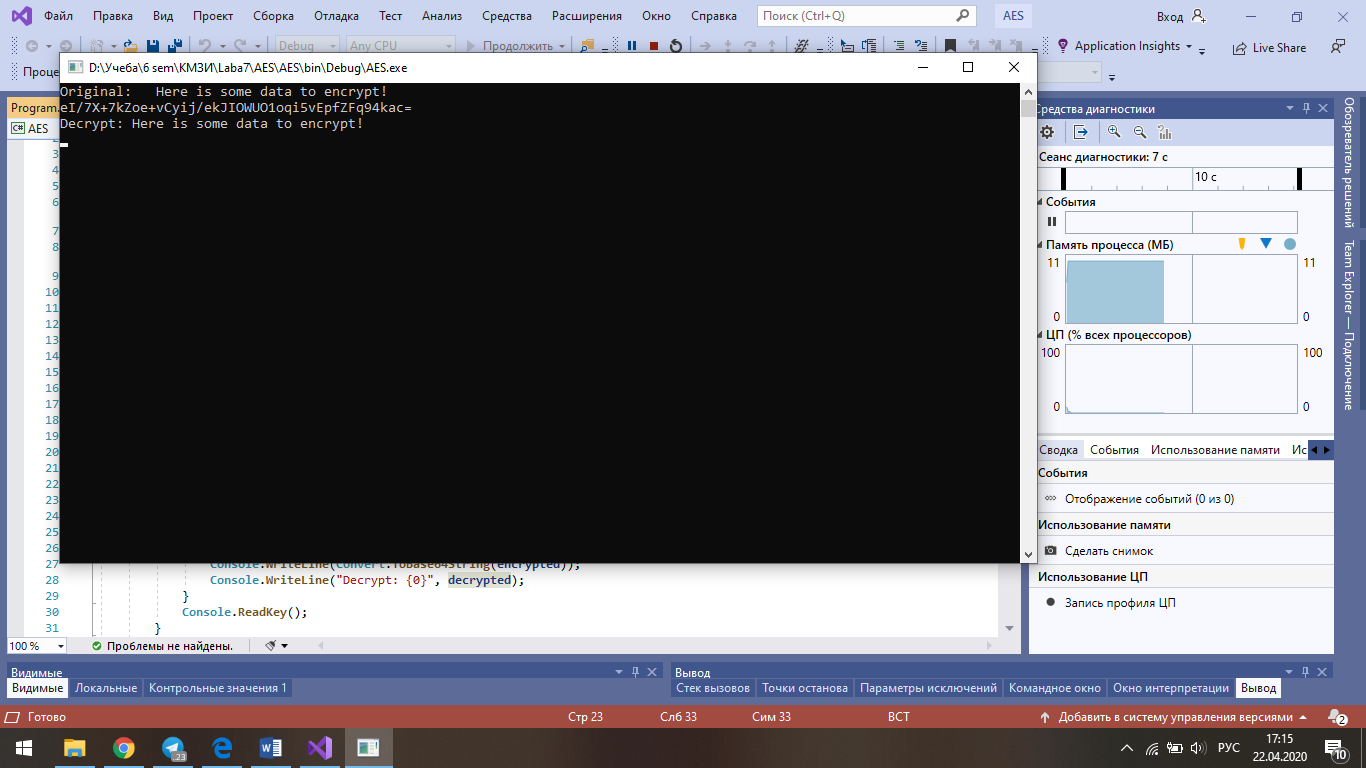


Рисунок 1. 4 Алгоритм AES

Работа приложения с алгоритмом TwoFish представлена на скриншоте ниже:

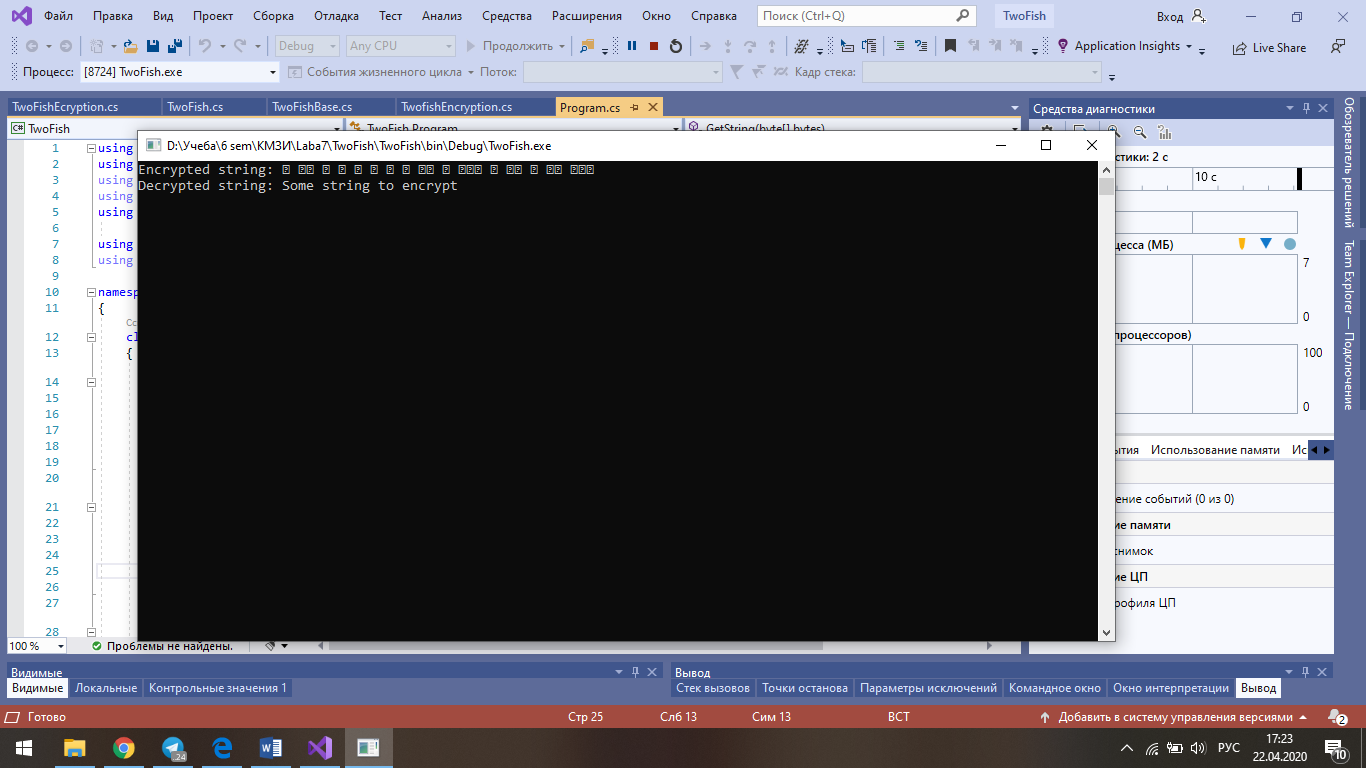


Рисунок 1. 5 Алгоритм TwoFish

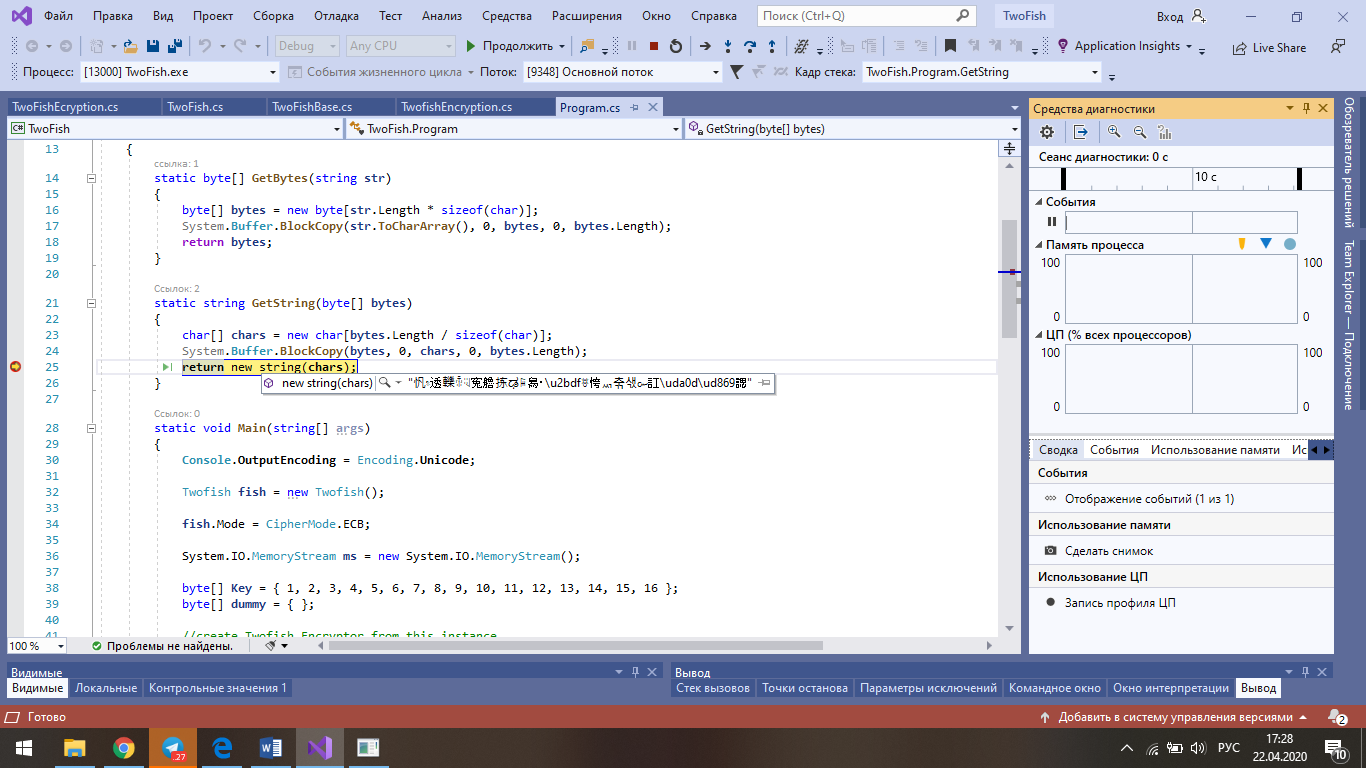


Рисунок 1. 6 Зашифрованные символы в алгоритме TwoFish

Работа приложения с алгоритмом Serpent представлена на скриншотах ниже.

На скриншоте ниже можно увидеть, как происходит зашифрование файла в приложении:

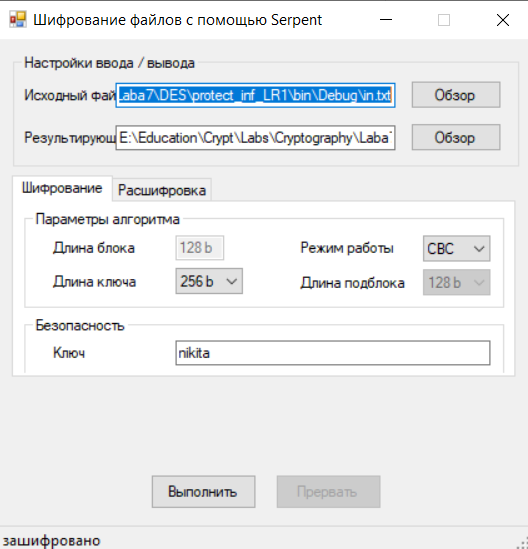


Рисунок 1. 7 Зашифрование алгоритм Serpent

На скриншоте ниже можно увидеть, как происходит расшифрование файла в приложении:

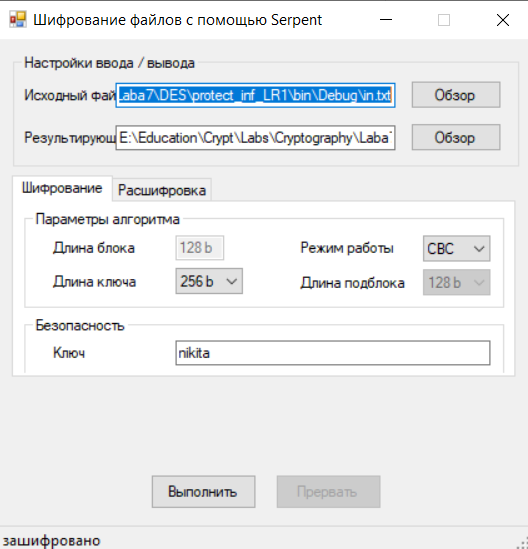


Рисунок 1. 8 Расшифрование алгоритм Serpent

В данной лабораторной работе я изучила и приобрела практические навыки разработки и использовании приложений для реализации блочных шифров.