Открытый конкурс проектно-исследовательских работ школьников «Паруса науки»

Информационный проект «Развитие методов шифрования»

Работу выполнил:

Гордиенков Захар Юрьевич

ученик 9 “А” класса

телефон +7-963-242-03-05

ГБОУ СОШ №257

Руководитель: Мухина Ольга Петровна

Санкт-Петербург  
2022

Оглавление

[Глоссарий 4](#__RefHeading___Toc1183_2900574084)

[Вступление 5](#__RefHeading___Toc1185_2900574084)

[Первые шифры 5](#__RefHeading___Toc1187_2900574084)

[Шифр Цезаря 5](#__RefHeading___Toc1189_2900574084)

[Пример 6](#__RefHeading___Toc1191_2900574084)

[Шифр простой замены 7](#__RefHeading___Toc1193_2900574084)

[Пример 7](#__RefHeading___Toc1195_2900574084)

[Атбаш 8](#__RefHeading___Toc1197_2900574084)

[Недостатки одноалфавитных шифров 9](#__RefHeading___Toc3973_2598738905)

[Многоалфавитная замена 12](#__RefHeading___Toc1201_2900574084)

[Пример 12](#__RefHeading___Toc1203_2900574084)

[AES (Rijndael) 15](#__RefHeading___Toc1205_2900574084)

[Процедуры шифрования 16](#__RefHeading___Toc1207_2900574084)

[Вторая операция - ShiftRows 16](#__RefHeading___Toc1209_2900574084)

[Третья операция - MixColumns 17](#__RefHeading___Toc1211_2900574084)

[Генерация ключей.KeySchedule 17](#__RefHeading___Toc1213_2900574084)

[Продукт 20](#__RefHeading___Toc1215_2900574084)

[Вывод 20](#__RefHeading___Toc1217_2900574084)

[Список литературы 21](#__RefHeading___Toc1219_2900574084)

[Приложение 1 22](#__RefHeading___Toc3975_2598738905)

[Приложение 2 24](#__RefHeading___Toc3977_2598738905)

[Приложение 3 27](#__RefHeading___Toc3979_2598738905)

[Приложение 4 29](#__RefHeading___Toc3981_2598738905)

[Приложение 5 31](#__RefHeading___Toc3983_2598738905)

[Приложение 6 32](#__RefHeading___Toc3985_2598738905)

[Приложение 7 33](#__RefHeading___Toc3987_2598738905)

[Приложение 8 34](#__RefHeading___Toc3989_2598738905)

*Цель:*

Разобраться в разных способах шифрования данных и сравнить их.

*Задачи:*

1. Собрать информацию о старых, ныне неактуальных шифрах
2. Собрать информацию о современных способах шифрования
3. Понять принцип их работы
4. Сравнить их, сделать выводы
5. Реализовать шифрование на практике

# Глоссарий

* Криптоло́гия (от др.-греч. κρυπτός — скрытый и λόγος — слово) — наука, занимающаяся методами шифрования и дешифровки. Криптология состоит из двух частей — криптографии и криптоанализа.
* Криптогра́фия (от др.-греч. κρυπτός «скрытый» + γράφω «пишу») — наука о методах обеспечения конфиденциальности, обеспечения целостности данных, аутентификации и шифрования.
* Криптоана́лиз (от др.-греч. κρυπτός «скрытый» + «анализ») — наука о методах дешифровки зашифрованной информации без предназначенного для этого ключа, а также сам процесс такой дешифровки.
* Шифротекст, шифртекст, криптограмма — результат операции шифрования
* Шифр — система обратимых преобразований, зависящая от некоторого секретного параметра и предназначенная для обеспечения секретности информации.
* Опера́нд (англ. operand) в математике и в языках программирования ― аргумент операции; данные, которые обрабатываются командой; математическое выражение, задающее значение аргумента операции. Иногда операндом называют место, позицию в тексте, где должен стоять аргумент операции. В зависимости от положения знака операции различают префиксные (напр., sin x (x — операнд)), инфиксные (например, a + b (a, b — операнды)) и постфиксные.
* Modulo, mod - оператор из модульной арифметики.
* Алфавит (в информатике) - множество символов, использующихся в языке.
* Мощность алфавита - (в информатике) число символов в алфавите.
* Операция XOR (строгая дизъюнкция) – булева функция, а также логическая и побитовая функция, возвращающая положительный результат только если один операнд отрицателен, а другой положителен.

# Вступление

Нельзя недооценивать необходимость защиты информации в наше время. Примером халатного отношения к данным могут быть утечки баз данных разных онлайн-сервисов в марте этого года. Были затронуты такие компании как Яндекс, Delivery club, СДЕК, Avito, Wildberries, Билайн, ВТБ, Hgclub.ru, а также ГИБДД. В общей сложности, в открытый доступ попали персональные данные (номера телефонов, физические адреса и адреса электронных почт, информация о заказах) более 8 миллионов человек.

*Эти события подтолкнули меня к созданию этого проекта.*

# Первые шифры

Удивительно, но как только люди научились писать, то сразу же начали пытаться скрыть написанное друг от друга. Самый древний рукописный текст был найден в Месопотамии на территории современного Ирака почти 5500 лет назад и представлял собой глиняную табличку с выдавленными на ней знаками. А самым древним шифром учёные считают древнеегипетский папирус с перечислением монументов времён фараона Аменемхета II (4000 до н.э.). В нём общепринятые на тот момент иероглифы были видоизменены, что затрудняло прочтение документа. Учёные сомневаются, было ли это сделано с целью сокрытия содержания или как некий литературный приём, который должен был произвести впечатление на читателя. Рассмотрим некоторые шифры и принцип их работы.

## Шифр Цезаря

Шифр Цезаря - один из самых старых и широко известных шифров. Он был назван в честь римского полководца. Гай Юлий Цезарь использовал этот шифр для переписки со своими генералами, но впоследствии им пользовались и другие представители римской аристократии. Алгоритм оставался неизменным почти 2100 лет.

Он основывается на том, что каждый символ в исходном тексте заменяется другим, находящимся на некоторое количество позиций дальше по алфавиту.

То есть, используется формула:

(x + n) mod a,

где:

* x - алфавитный номер знака исходного сообщения;
* n - значение, на сколько позиций происходит сдвиг в алфавите (сдвиг шифрования);
* a - мощность алфавита.

Например, если число сдвига n равно 10, используется русский алфавит и необходимо зашифровать букву "ю"

(x = 32; n = 10; a = 33),

то получится

(32 + 10) mod 33 = 42 mod 33 = 9

(девятая) буква алфавита, то есть "з". Таким образом, деление по модулю обеспечивает обработку последних букв алфавита.

*Если в результате получился ноль, то берётся последняя буква алфавита. Иначе, можно нумеровать символы алфавита с 0. В таком случае этой проблемы можно избежать.*

### Пример

Допустим, надо зашифровать слово "ананас". Используется русский алфавит (абвгдеёжзиклмнопрстуфхцчшщъыьэюя), шаг шифрования пусть будет равен 7.

а = (1 + 7) mod 33 = ж;

н = (14 + 7) mod 33 = у;

с = (18 + 7) mod 33 = ч;

Получается: "жужужч". Чтобы получить исходное сообщение, необходимо произвести обратные действия. То есть, по формуле

(x - n) mod a:

ж = (8 - 7) mod 32 = 1 = а

у = (21 - 7) mod 32 = 14 = н

ч = (18 - 7) mod 32 = 11 = с,

жужужч = ананас.

Шифр Цезаря относится к *моноалфавитными* (или *одноалфавитным*, смыл одинаков) шифрам - в них каждой букве шифртекста соответствует только одна буква исходного сообщения.

Другими словами, можно составить таблицу. Для русского алфавита и сдвига 7 она будет выглядеть так же, как в приложении 1.

Концепция шага шифрования, выполняющегося в шифре Цезаря, часто используется в других, более сложных и современных схемах шифрования. Например, в шифре Вижинера, (он будет рассмотрен позже), системе ROT13 и других, рассмотрение которых выходит за рамки одного проекта. Информация о них доступна в открытом доступе.

Как и все моноалфавитные шифры, шифр Цезаря довольно легко взламывается, и в наше время не имеет почти никакого применения на практике. Теперь рассмотрим другой моноалфавитный шифр.

## Шифр простой замены

Этот шифр, по сути, есть расширенная версия шифра Цезаря.

Чтобы зашифровать сообщение, составляется алфавит со всеми нужными символами, включая знаки препинания, буквы других языков и т.д. Потом чертится таблица из двух строк. В первой строке в каждой колонке находится одна буква исходного алфавита. А во второй для каждой буквы случайным образом подбирается замена из других букв алфавита. По сути, это то же самое, что и шифр Цезаря, только шаг шифрования разнится от буквы к букве.

### Пример

Попробуем зашифровать сообщение Привет, Антон!. Здесь используются знаки препинания и большие буквы, поэтому расширим алфавит:

абвгдеёжзиклмнопрстуфхцчшщъыьэяAБВГДЕЁЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ, .!?

(71 знак, пробел - 68-й)

Составим таблицу, представленную в приложении 2. Согласно ней, получается хЖяи вЗЪюлв!лш.

## Атбаш

Шифр под названием Атбаш был впервые использован в древних экземплярах Ветхого завета, написанных на древнесемитском языке.

В нём одни буквы алфавита заменялись другими по формуле n - i + 1, где n - количество букв в алфавите, а i - порядковый номер буквы в алфавите.

То есть, использовалась таблица, подобная представленной в приложении 3.

Другие шифры моноалфавитной замены работают похожим образом.

# Недостатки одноалфавитных шифров

Несомненно, очевидным преимуществом всех упомянутых выше шифров является их простота. Оригинальное сообщение можно получить вручную, и это не займёт много времени.

Однако, шифр Цезаря (и ему подобные) очень легко взломать, если есть достаточное количество зашифрованного текста. Для этого нужно просто пробовать все ключи (то есть, значения всех возможных сдвигов) до тех пор, пока не будет получено исходное сообщение. Например, если получен шифртекст ежсфрф, зашифрованное шифром Цезаря, то необходимо будет перебрать все смещения от 1 до 32. (Смещение на 33 ничего не изменит: (21 + 33) mod 33 = 21) Уже 6-я попытка даст результат: было зашифровано слово ("яблоко"). То есть,

количество возможных ключей = количество букв в алфавите - 1.

При случайном значении сдвига шифрования , среднее количество попыток будет равно

количеству букв в алфавите // 2;

при использовании 33-буквенного алфавита

33 // 2 = 17.

Можно обезопасить шифр от такого способа взлома просто предварительно перемешав алфавит (то есть, воспользовавшись шифром простой замены). Шифр замены взломать вышеописанным способом не получится, но он всё равно не является безопасным. При условии, что есть достаточное количество шифртекста, его можно сломать с помощью частотного анализа. Впервые упоминание об этом методе было найдено в труде арабского математика аль-Кинди "Манускрипт о дешифровке криптографических сообщений», который в IX впервые описал недостатки моноалфавитных шифров.

*(Однако, в некоторых, ещё более древних источниках содержатся упоминания о некоем статистическом методе, но их авторы неизвестны, а в текстах не было описания самого алгоритма.)*

Он основан на том, что определённые символы в языке встречаются чаще остальных. Проанализировав большое количество текстов на определённом языке, можно определить частоту вхождений каждого символа.

Например, самые частыми буквами в текстах на русском языке являются о, е, а.

Средняя частота всех букв русского языка в случайных текстах указана в приложении 4.

Знаки препинания в таблицу не входят, так как встречаются реже любой буквы *(за исключением пробела. Это самый частый символ).* Также существуют таблицы для других языков; их можно найти в открытом доступе.

При расшифровке частота вхождений одного символа будет примерно равно частоте заменяющего её символа в шифртексте. То есть, при взломе шифра таким способом, необходимо подсчитать количество вхождений каждого символа в текст. На место символа, который встречается чаще всего, подставляется самый частый символ; на место второго по частоте - второй, и так далее. Если в процессе расшифровки получилось осмысленное слово, то буквы этого слова были верно угаданы, и следовательно, все шифрующие их символы можно заменить во всём тексте.

Однако, чтобы частоты вхождений символов совпадали, порой необходимо большое количество шифртекста, что может стать проблемой, если перехваченный объём слишком мал. К тому же частота вхождений букв в текст зависит от автора, тематики и так далее. Поэтому существуют специальные таблицы частот символов для стихотворений, научных текстов, литературных произведений и прочего.

Следует отметить, что много проблем при взломе могут возникнуть из-за того, что было использовано большое количество малоиспользуемых символов (например, знаков препинания или смайликов). Также очень важно с самого начала выбрать правильный язык, на котором было составлено сообщение.

Взлом шифров обычно автоматизируют. Ключи перебираются машиной, получающиеся слова сравниваются с заранее подготовленным словарём. Это значительно ускоряет процесс взлома. В наше время в большинстве случаев взлом шифра вручную невозможен из-за сложности применяемых методов.

Стоит упомянуть ещё один способ взлома шифра замены. Он называется поиск восхождением в вершине.

Вначале случайным образом выбирается ключ (т.е. строится таблица соответствия символов), и с её помощью производится расшифровка. Далее, текст подвергается анализу на так называемые биграммы и триграммы. Это двух- и трёхбуквенные сочетания, которые характерны для определённого языка. В английском это ing, the, and. В русском - СТ, НО, ЕН, ТО, НА, ОВ, НИ, РА, ВО, КО, СТО, ЕНО, НОВ, ТОВ, ОВО, ОВА и другие. *(Тема триграмм выходит за рамки одного проекта, так как, по мнению автора, довольно объёмна и требует отдельного рассмотрения.)*

После первого анализа в ключ вносятся незначительные изменения, и проводится повторный анализ. Если би- и триграмм стало больше, то вероятность того, что текст стал более похож на осмысленный язык, увеличилась. В противном случае ключ возвращают в прежнее состояние и вносятся другие изменения. Цикл повторяется до тех пор, пока текст не будет расшифрован. Этот способ расшифровки сложнее всех вышеперечисленных, но далеко не самый сложный. Существует ещё много способов взлома таких шифров. Например, алгоритм имитации отжига и другие; информация о них есть в открытом доступе.

Из всего вышесказанного можно сделать вывод, что моноалфавитные способы шифрования ненадёжны. Поэтому в настоящее время они не используются там, где по-настоящему важна секретность информации.

# Многоалфавитная замена

Все вышеприведённые техники шифрования были моноалфавитными; но это далеко не единственный способ шифрования. В некоторых шифрах применяется сразу несколько моноалфавитный шифров, что делает их гораздо более устойчивыми ко взлому. Одним из таких способов шифрования является шифр Виженера.

Он многократно изобретался, но впервые он был упомянут в итальянским учёным Леоном Баттиста Альберти в его трактате о шифрах как "шифр, достойный королей" в 1466 году. Это изобретение намного опередило своё время; подобные шифры стали использоваться в Европе лишь 400 лет спустя. Но впоследствии случилось так, что метод получил имя французского посла в Риме Блеза де Вижинера, который увлекался криптографией и сделал первое хорошо задокументированное описание полиалфавитного шифра в своём "трактате о шифрах". *(Справедливости ради стоит отметить, что он лишь объединил труды учёных-криптологов в одну книгу, не внеся туда почти ничего нового.)*

Этот метод шифрования оказался простым и понятным, но гораздо более надёжным в сравнении со своими предшественниками, благодаря чему получил широкое распространение. В частности, во время гражданской войны в США он использовался Конфедератами для пересылки секретных сообщений.

Чтобы воспользоваться этим шифром, необходимо составить таблицу Виженера для алфавита, который будет использоваться. Она представлена в приложении 5.

Затем выбирается ключ: слово или любая последовательность символов, не важно. Допустим, что длина ключа равна n. Тогда, чтобы зашифровать первый символ сообщения, необходимо соединить в таблице строку с исходным символом и столбец с первым символом ключа. На месте пересечения и будет первая буква шифртекста. Далее, берётся строка со второй буквой сообщения и столбец со второй буквой ключа. Когда зашифровано уже n символов, операция производится с n + 1 символом сообщения и первым символом ключа.

## Пример

Например, необходимо зашифровать сообщение "THE BUS IS LATE" (автобус опаздывает). Пусть ключом будет слово "tulip" (тюльпан). Тогда, по таблице, первая буква шифртекста будет M. По аналогии зашифруем все остальные буквы. Получится MBP JJL CD EPMQ. *(Обычно пробел тоже шифруют. Я оставил его для простоты и наглядности).*

Один и тот же символ ("S") оказался зашифрован по-разному внутри одного сообщения (в одном месте L, в другом D). Эта особенность делает шифр устойчивым к частотному анализу. Однако, в случае выше, символ T оказался одинаковым во всех двух местах своего появления "M". Это произошло из-за того что при зашифровке символ T оба раза "попал" на символ T ключа (tulip), и был зашифрован по одному и тому же столбцу.

В этом и заключается основной недостаток шифра Виженера. Однако, если подобрать длинный ключ, в котором будет мало повторяющихся букв, таких совпадений будет гораздо меньше. Следовательно, сложность шифра и его устойчивость ко взлому напрямую зависят от характеристик ключа.

Метод криптоанализа шифра Виженера был разработан независимо друг от друга немцем Фридрихом Касиски и англичанином Чарльзом Бэббиджем в 1863 году. Чарльз стал первым человеком на земле, взломавшим полиалфавитный шифр.

*Примечательно, что Бэббидж первым изобрёл универсальную вычислительную машину, являвшуюся прообразом современного цифрового компьютера. Машина была построена уже после его смерти, он так и не увидел вживую своё изобретение. Он также участвовал в создании тахометра и создал приспособление для сбрасывания посторонних предметов с железнодорожных путей перед идущим локомотивом.*

*Фридрих Касиски родился на территории современной Польши. В 17 лет стал военным, к концу жизни стал заниматься археологией. Всегда увлекался криптографией. Разработал способ взлома шифра Виженера и описал его в своей книге "Тайнопись и искусство дешифрования" ("Die Geheimschriften und die Dechiffrirknust"). До того момента, в течении почти 400 лет, шифр считался невзламываемым. Впоследствии этот способ взлома был назван тестом Касиски.*

Он основан на уже упомянутых мной биграммах и триграммах. Если два таких сочетания были зашифрованы одной и той же частью ключа, то в шифртексте они будут выглядеть одинаково. Расстояние между ними будет кратно длине ключа. Криптоаналитик должен определить длину ключа, а затем разбивает шифртекст на части, равные длине ключа, и выписывает их в столбик. Каждый столбец окажется зашифрован с помощью одной и той же буквы ключа. То есть, по сути, остаётся лишь решить несколько шифров простой замены (один шифр на каждый столбец), которые поддаются частотному анализу. Получается, что все такие столбцы были зашифрованы с помощью разных таблиц шифра простой замены (разными алфавитами). Поэтому шифр назван многоалфавитным.

В наше время уже никто всерьёз не взламывает этот шифр вручную. Вместо этого используются мощности электронно-вычислительных машин, что значительно ускоряет процесс. *(Стоит отметить, что это далеко не единственный способ взлома, но они все имеют один и тот же принцип. Я не буду рассматривать другие техники.)* Со временем появились более надёжные способы шифрования. Об одном из них будет рассказано ниже.

# AES (Rijndael)

AES - это симметричный алгоритм блочного шифрования. ("симметричный" - данные могут быть расшифрованы обратно; "блочный" - перед обработкой данные разбиваются на блоки определённой длинны). В настоящее время широко используется; например, в программах-архиваторах, файловых системах с поддержкой шифрования и т.д.

Этот алгоритм был принят как стандарт шифрования правительством США в результате конкурса AES (Advanced Encryption Standard, стандарт продвинутого шифрования), организованного Национальным институтом стандартов и технологий США в 1997 году. *(Принятый алгоритм назывался Rijndael (Рэйндал), но теперь носит название конкурса - AES. Однако, алгоритм был улучшен и в настоящее время отличается от варианта, принятого в 1997 году). Причиной поиска нового алгоритма стал предшественник Rijndael - DES. Он имел несколько недостатков, которые делали его уязвимым для взлома. Подробнее об этом можно прочитать в открытых источниках.*

*Следует уточнить, что только алгоритм Rijndael, в котором размер блока равен 128 бит, а размер ключа – 128, 192 или 256 бит, соответствует стандарту шифрования AES. Но в настоящее время понятия Рэйндал и AES – почти что синонимы.*

Перед началом шифрования данные разбиваются на блоки по 128 бит. Длина блока не изменяется в процессе шифрования (т.е. всегда постоянна). Длина шифрключа равна 128 бит. *(128 бит это 16 символов в кодировке UTF-8; один символ равен 8 бит).*

Весь процесс шифрования блока состоит из 10 так называемых "раундов". По завершению 10 раунда начинает шифроваться следующий блок. В конце блоки складываются, и получается шифртекст. Каждый раунд состоит из 4 процедур (заключительный, десятый - из трёх).

## Процедуры шифрования

State - текущее состояние блока во время шифрования.

Процесс шифрования схематично представлен в приложении 6.

Сначала, к каждому биту i каждого байта bприменяется следующая операция:

bi’= bi XOR b(i+4) mod 8 XOR b(i+5) mod 8 XOR b(i+6) mod 8 XOR b(i+7) mod 8 XOR ci

Где ci - константа, равняющаяся 99 в десятичной и 1100011 в двоичной системах исчисления соответственно. То есть, преобразование проводится 8 раз (в байте 8 бит, для каждого бита соответственно), в течение которых байт меняется.

Пример таблицы Sbox представлен в приложении 7.

Далее, каждый байт state заменяется с помощью постоянной (константной) таблицы Sbox по следующему принципу:

1. *Берётся строка, обозначенная первым символом байта, и столбец, обозначенный вторым символом байта.*
2. *Байт, оказавшийся на месте их пересечения, заменяет собой исходный байт.*
3. *Замена производится для каждого символа блока (state)*

*Пример таблицы Sbox представлен в приложении 7.*

### Вторая операция - ShiftRows

Представим блок в виде квадрата 4x4 из байтов (приложение 8).

Для каждой "строки" блока происходит сдвиг на (номер строки - 1) позиций влево. "Вылезшие" за границу блока байты вставляются на освободившиеся места справа.

*Если для примера взять таблицу выше, то*

* *Первая строка останется без изменений.*
* *Во второй строке произойдёт сдвиг на (2 - 1) одну позицию.*

*Строка будет выглядеть так:*

|  |  |  |  |
| --- | --- | --- | --- |
| 6байт | 7байт | 8байт | 5байт |

*(5байт сдвинулся налево и встал в конец строки)*

* *В третей строке произойдёт сдвиг на 2 позиции:*

|  |  |  |  |
| --- | --- | --- | --- |
| 12байт | 9байт | 10байт | 11байт |

### Третья операция - MixColumns

* Каждый столбец state определённым образом перемножается:

S’0, j = (2 \* S0, j) XOR (3 \* S1, j) XOR S2, j XOR S3, j

S’1, j = S0, j XOR (2 \* S1, j) XOR (3 \* S2, j) XOR S3, j

S’2, j = S0, j XOR S1, j XOR (2 \* S2, j) XOR (3 \* S3, j)

S’3, j = (3 \* S0, j) XOR S1, j XOR S2, j XOR (2 \* S3, j)

Где j - ряд байта в state.

Результат операции заменяет исходную колонку state.

* В заключительном десятом раунде эта процедура не выполняется.

Эти три операции делают статистический анализ почти невозможным.

*Для вышеперечисленных процедур есть инверсные (обратные) им - InvSubBytes, InvShiftRows и InvMixColumns.* Они используются для расшифровки.

Четвёртая операция - AddRoundKey

В ней state побитово складывается с ключом этого раунда (операция XOR).

Раундовые ключи будут подробнее рассмотрены далее.

Стоит отметить, что для этой операции нет обратного варианта.

Это объясняется тем, что выражение a XOR b XOR b в результате даёт a.

## Генерация ключей.KeySchedule

Перед началом шифрования происходит генерация раундовых ключей. *(Хотя их можно генерировать и во время шифрования.)* Это происходит посредством процедуры KeyExpansion (расширение ключа).

В результате получаются 10 раундовых ключей, или так называемый KeySchedule (расписание ключей). Рассмотрим этот процесс.

1. Вначале берётся основной ключ и представляется в виде таблицы (матрицы) 4x4. (Очень похоже на state - по сути, это таблицы одинакового размера из 16 байтов. Общий информационный вес таблицы - 128 бит (16 байт = 16 \* 8 = 128 бит)).
2. Берётся копия последней (4-ой) колонки получившейся таблицы (назовём эту колонку *текущей*). В ней происходит сдвиг на одну позицию вверх, причём "вылезший" за границу таблицы символ перемещается на освободившееся место внизу. Эта процедура имеет название RotWord (rotate - вращать, word - слово). *То есть, из массива* [a1, a2, a3, a4] *делается массив* [a2, a3, a4, a1]*.*
3. Далее, к этой же колонке применяется процедура SubWord. *Она полностью аналогична процедуре* SubBytes, которая была описана выше. В ней первый и второй символ байта определяют местоположение его замены в таблице Sbox.
4. Затем, берётся первая колонка шифрключа h и первая колонка k таблицы Rcon. Производится операция h XOR g XOR k, где g - текущая колонка.

*Заметьте - вышеописанные операции не изменяли шифрключ. Операции производились над её копией одной его колонки.*

1. Результат записывается в 5-ю колонку. Назовём получающуюся таблицу w.

Чтобы получить вторую колонку i первого раундового ключа (то есть, w[6]), необходимо взять колонки w[i - 1] и w[i — 4].

w[i] = w[i - 4] XOR w[i — 1]

1. Результат записывается в колонку w[i]. Операция повторяется ещё трижды, т.е. пока не будет заполнена 4 колонка первого раундового ключа. Таблица w[4, 9] будет первым раундовым ключом. Над последней колонкой производятся операции RorWord (см. выше) и SubBytes.

Также, выполняется операция XOR этой колонки i - 1, колонки

i - 4 и второй колонки Rcon. Результат становится 9 колонкой (w[9]). По аналогии, трижды выполняются операции

w[i] = w[i - 4 XOR w[i - 1].

Колонки с 9 по 12 включительно - это второй раундовый ключ. Операции повторяются до тех пор, пока не будут найдены все 10 ключей.

В настоящее время существует несколько разных реализаций алгоритма AES. Они отличаются степенью защиты от взлома, однако все основаны на вышеизложенном алгоритме.

# Продукт

Я сделал веб-приложение, в котором реализован алгоритм шифрования AES в режиме EAX (шифрование + аутентификация). Оно написано на веб-фреймворке flask, который мало весит, прост и идеально подходит для r написания небольших проектов. Также, используется WSGI-сервер waitress и библиотека pycryptodome.

С помощью этого приложения помощью него можно зашифровать любой файл при условии, что ключ состоит из 16 символов *(точнее, весит 256 бит)*. После зашифровки возвращается преобразованный файл и так называемый "временный код". Дело в том, что использование одинаковой таблицы Sbox делает ключ уязвимым. Временный код применяется для создания таблицы с нуля для каждой криптограммы (т.е. для каждого файла). Это повышает криптографическую стойкость шифра.

Исходный код программы лежит в репозитории на github и доступен для просмотра и клонирования всем желающим: https://github.com/mb6ockatf/aes\_n\_flask

В перспективе планируется улучшение дизайна главной страницы и публикация в Интернете. Однако, поскольку за хостинг необоходимо платить (или за электричество, если хостить самим), поддержание круглосуточной работы сайта не является приоритетной задачей. Любой желающий может скачать исходный код и запустить программу самому. Планируется создать пошаговую инструкцию в ближайшем будущем — в настоящее время это не первоочередная задача.

Более подробные сведения о продукте находятся в приложении 9

# Вывод

Таким образом, можно сделать вывод, что во время перехода к математической криптографии сформировался новый подход, построенный на строгих научных основах. Были определены основные требования к зашифрованной информации: нужно сделать так, чтобы было непонятно, откуда и куда движентся информация (неотслеживаемость), чтобы информация доставлялась или сохранялась в неизменном виде (целостность) и чтобы она не была доступна для третьих лиц (конфеденциальность).

Современные способы шифрования сложнее и изощрённее своих "предшественников", что делает их более надёжными и сложными для взлома. В наше время шифрование используется повсеместно, и для современного человека важно понимать, как оно работает.

# Список литературы

1. Жуан Гомес. “Математики, шпионы и хакеры. Кодирование и криптография” - М: Де Агостини, 2014. 142 страницы
2. Митани Масааки, Сато Синъити. “Занимательная информатика. Криптография. Манга” - М: “ДМК Пресс”, 2019. 228 страниц
3. Пол Берри. “Изучаем программирование на Python” O’REILLY - М: “Эксмо”, 2020. 617 страниц
4. Роман Душкин. “Шифры и квесты. Таинственные истории в логических загадках” - М: Издательство “АСТ”, 2017. 285 страниц

Документация FIPS, а также материалы Национального Института Стандартов и Технологий США (NIST)

Документация к языку программирования python, а к сторонней библиотеке pycryptodome и фреймворку flask, а также информация из открытых источников.

Искренне благодарен авторам вышеперечисленных книг, статей и видеоматериалов.

# Приложение 1

|  |  |
| --- | --- |
| Таблица алфавита со сдвигом 7 | |
| а | ж |
| б | з |
| в | и |
| г | й |
| д | к |
| е | л |
| ё | м |
| ж | н |
| з | о |
| и | п |
| й | р |
| к | с |
| л | т |
| м | у |
| н | ф |
| о | х |
| п | ц |
| р | ч |
| с | ш |
| т | щ |
| у | ъ |
| ф | ы |
| х | ь |
| ц | э |
| ч | ю |
| ш | я |
| щ | а |
| ъ | б |
| ы | в |
| ь | г |
| э | д |
| ю | е |
| я | ё |

# Приложение 2

|  |  |
| --- | --- |
| Таблица шифра простой замены | |
| И | е |
| Х | г |
| г | , |
| М | Р |
| я | Л |
| , | Ъ |
| ц | ё |
| м | щ |
| Ч | м |
| э | Ь |
| о | ! |
| к | ф |
| К | Т |
| р | Ж |
| у | с |
| ф | Д |
| Ъ | Б |
| н | л |
| Д | З |
| в | и |
| Я | Ю |
| ь | Ф |
| A | ю |
| ё | a |
| С | О |
| е |  |
| П | х |
| Б | Щ |
| Э | Ы |
| п | ъ |
| л | ж |
| Ц | к |
| Г | М |
| . | ц |
| В | Г |
| д | С |
| a | р |
| з | Ё |
|  | П |
| Л | Х |
| Ё | д |
| ы | И |
| Е | б |
| Щ | Я |
| б | У |
| Т | ь |
| ю | н |
| У | у |
| и | я |
| ж | . |
| Ю | A |
| щ | К |
| Ы | ш |
| т | в |
| Ь | Е |
| Ж | о |
| З | Ц |
| Р | Э |
| с | т |
| ? | п |
| Н | Ч |
| ч | э |
| Ф | з |
| х | ы |
| ъ | В |
| ! | Ш |
| ш | ч |
| О | ? |
| Ш | Н |

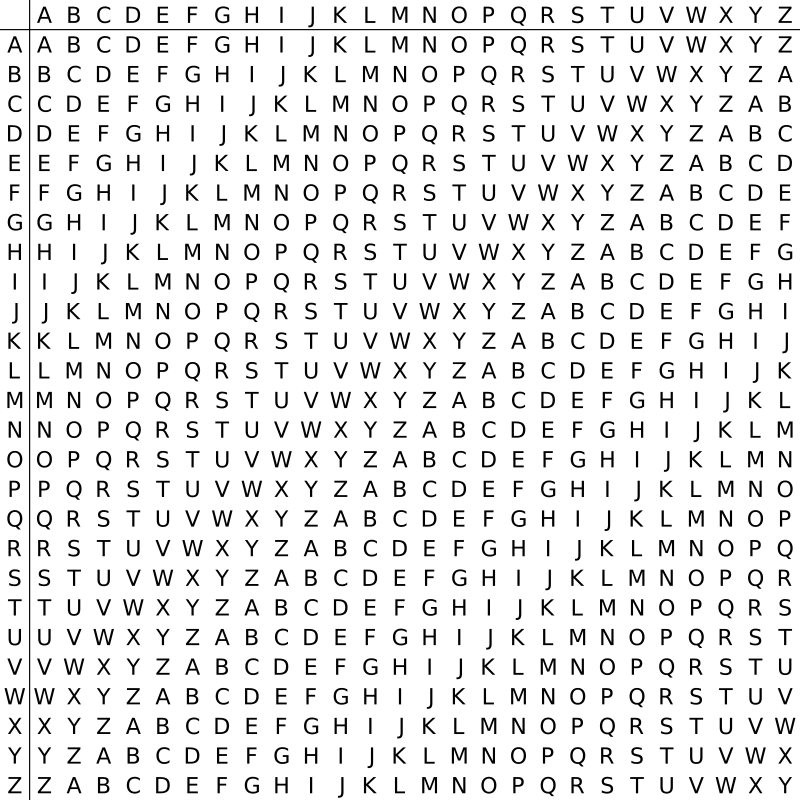
# Приложение 3

|  |  |
| --- | --- |
| Таблица, используемая в шифре Атбаш | |
| а | я |
| б | ю |
| в | э |
| г | ь |
| д | ы |
| е | ъ |
| ё | щ |
| ж | ш |
| з | ч |
| и | ц |
| й | х |
| к | ф |
| л | у |
| м | т |
| н | с |
| о | р |
| п | п |
| р | о |
| с | н |
| т | м |
| у | л |
| ф | к |
| х | й |
| ц | и |
| ч | з |
| ш | ж |
| щ | ё |
| ъ | е |
| ы | д |
| ь | г |
| э | в |
| ю | б |
| я | а |

# Приложение 4

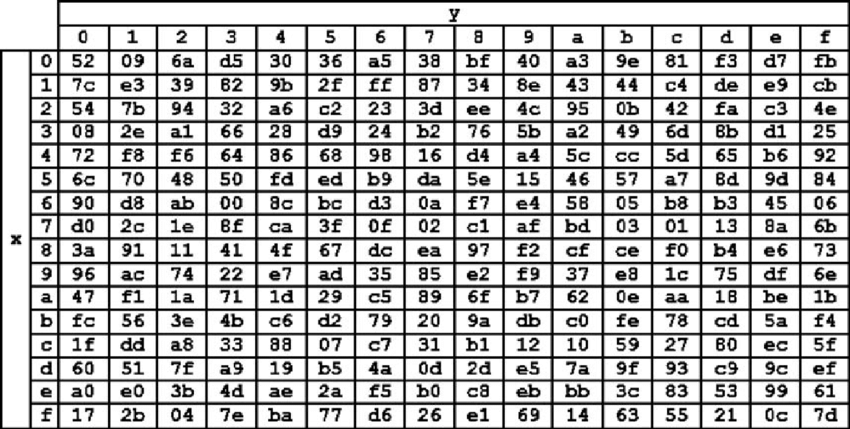
|  |  |  |
| --- | --- | --- |
| Средняя частота всех букв русского алфавита в случайных текстах | | |
| О | 1 | 10,97 |
| Е | 2 | 8,45 |
| А | 3 | 8,01 |
| И | 4 | 7,35 |
| Н | 5 | 6,70 |
| Т | 6 | 6,26 |
| C | 7 | 5,47 |
| Р | 8 | 4,73 |
| В | 9 | 4.54 |
| Г | 19 | 1.7 |
| З | 20 | 1.65 |
| Б | 21 | 1.59 |
| Ч | 22 | 1.44 |
| Й | 23 | 1.21 |
| Х | 24 | 0.97 |
| Ж | 25 | 0.94 |
| Ш | 26 | 0.73 |
| Л | 10 | 4.4 |
| К | 11 | 3.49 |
| М | 12 | 3.21 |
| Д | 13 | 2.98 |
| П | 14 | 2.81 |
| У | 15 | 2.62 |
| Я | 16 | 2.01 |
| Ы | 17 | 1.9 |
| Ь | 18 | 1.74 |
| Ю | 27 | 0.64 |
| Ц | 28 | 0.4 |
| Щ | 29 | 0.36 |
| Э | 30 | 0.32 |
| Ф | 31 | 0.26 |
| Ъ | 32 | 0.04 |
| Ё | 33 | 0.04 |

# Приложение 5



# Приложение 6

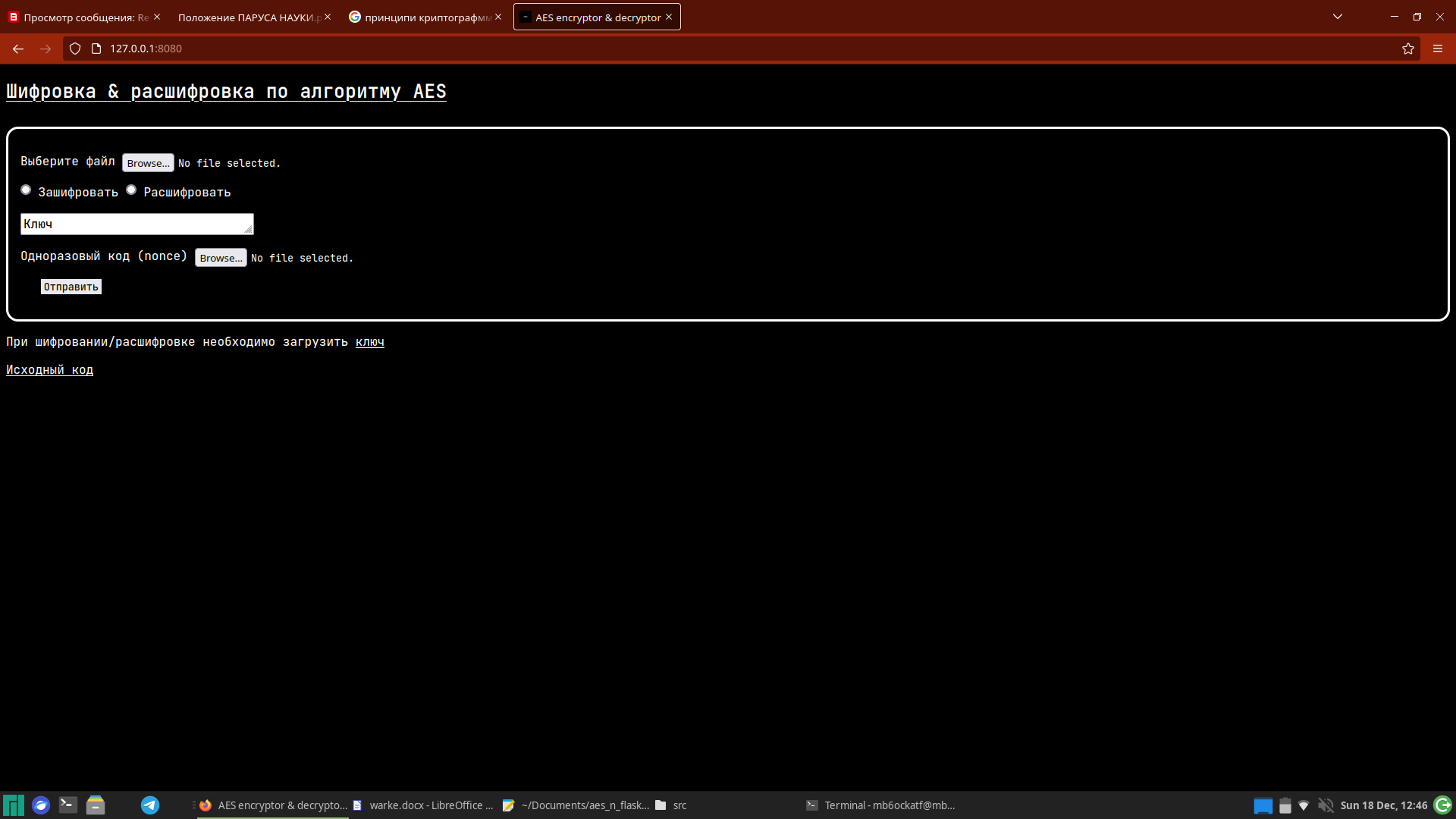
# Приложение 7



# Приложение 8

|  |  |  |  |
| --- | --- | --- | --- |
| Абстракция State для визуализации процесса ShiftRows | | | |
| 1 байт | 2 байт | 3 байт | 4 байт |
| 5 байт | 6 байт | 7 байт | 8 байт |
| 9 байт | 10 байт | 11 байт | 12 байт |
| 13 байт | 14 байт | 15 байт | 16 байт |

# Приложение 9

Главная страница локально запущенного приложения