**Автономная некоммерческая общеобразовательная организация "Физтех-лицей"
(АНОО «Физтех-лицей» им. П.Л. Капицы)**

XX научно-практическая конференция

«Старт в инновации»

Исследовательская работа
«Изучение принципов работы алгоритмов шифрования»

Выполнили:
Горбачёв Вадим Максимович, 9 «И» класс

Руководитель:
Горбачев Максим Иванович

Московская область, г. Долгопрудный

2021 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ *3*

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ *4*

Основные термины **4**

Структура алгоритмов симметричного шифрования **6**

Структура алгоритмов ассиметричного шифрования **6**

Взлом алгоритмов шифрования **7**

ПРАКТИЧЕСКАЯ ЧАСТЬ *8*

ЗАКЛЮЧЕНИЕ *9*

ВВЕДЕНИЕ

Каждый из нас хоть раз пользовался мессенджером в телефоне. Чтобы злоумышленники не могли перехватить и использовать переписку в корыстных целях, каждый мессенджер шифрует сообщения. Шифровать сообщения начали задолго до появления телефонов, ещё до нашей эры. И всегда существовали любопытные люди, которым было важно обойти простые способы шифрования и прочитать чужое сообщение. Как следствие способы шифрования стали усложняться, но их тоже научились расшифровывать. В итоге появлялись все более совершенные и защищенные способы шифрования, многими из которых мы пользуемся сегодня. Мне стало интересно, какие существуют на сегодняшний момент системы шифрования, насколько они устойчивы и какую из них лучше выбрать.

Цель работы: изучение принципа работы разных алгоритмов шифрования и создание программы, шифрующее сообщения

Предмет исследования: алгоритмы шифрования

Задачи исследования:

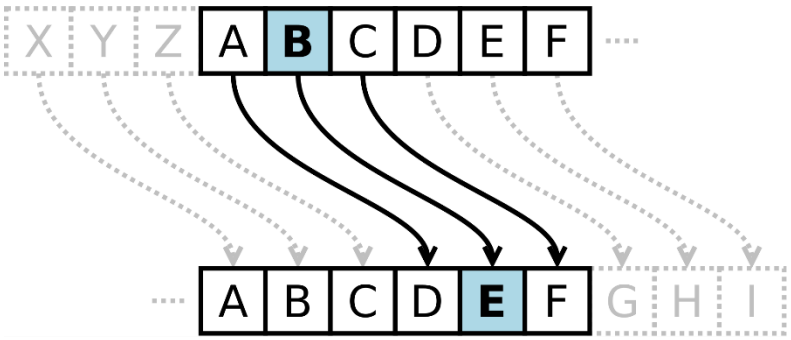
1) Узнать историю развития алгоритмов шифрования
2) Разобраться в основных принципах работы современных алгоритмов шифрования
3) Написать программу для шифровки и расшифровки сообщений
4) Сформулировать выводы

Данную работу я выбрал по причине актуальности – сегодня весь мир использует алгоритмы шифрования. Нам важно быть уверенным, что наши данные не попадут в чужие руки. Также важно знать, насколько разные способы шифрования уязвимы, чтобы предотвратить кражу данных. Для определения самого надёжного способа шифрования я планирую написать программу, которая будет шифровать текст разными алгоритмами шифрования. После этого я зашифрованный текст буду пытаться «взломать», вычисляя время, требуемое для этого. Составив графики зависимостей «длина текста – время для расшифровки» можно легко узнать самый лучшей способ шифрования.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Основные термины

Несмотря на то, что алгоритмы шифрования повсеместно начали использоваться 50 лет назад, наипростейшие алгоритмы придумали и начали использовать ещё до нашей эры. И в то время их тоже использовали для того, чтобы сообщения могли прочитать только те люди, которые знали, как работает алгоритм. Самый первый алгоритм – шифр Цезаря, появился ещё до нашей эры. Другие шифры использовались и ранее, но Цезарь был первым зафиксированным человеком, использующим эту схему. Шифр представляет собой простой сдвиг алфавита на n символов влево или вправо. Шифрование и дешифрование можно выразить следующими формулами:

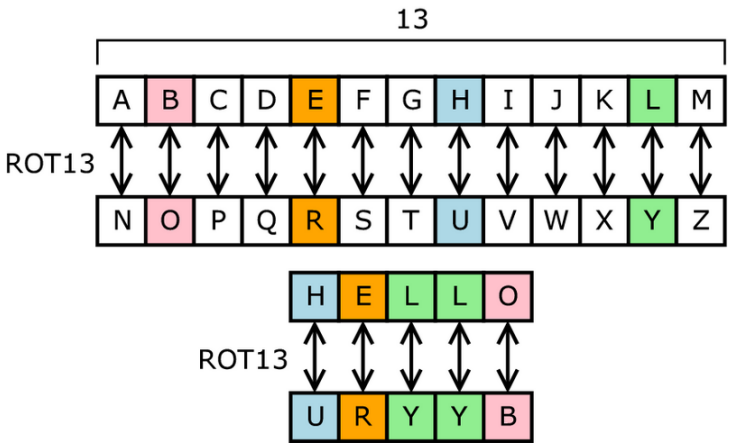


$y = (x + k) \bmod n$

$x = (y - k) \bmod n$

где x – исходный символ, y – зашифрованный символ, k – ключ (сдвиг) и n – мощность (кол-во символов) алфавита

Взломать такой шифр не составляет труда – достаточно просто перебрать все варианты перестановки символов (25 вариантов для английского языка и 32 для русского), что возможно сделать вручную за несколько минут. Несмотря на свою ненадёжность, он всё ещё имеет современное приложение в системе ROT13, хотя не имеет почти никакого применения на практике на сегодняшний день.



Более «безопасный» вариант шифрования – шифр простой замены, когда мы каждую букву заменяем на другую, без привязки к порядку в алфавите. В этом случае мы уже получаем 26! способов (33! способов для русского языка), что уже невозможно перебрать вручную, но компьютер переберёт все эти способы за несколько минут, поэтому эти способы шифрования редко используют, чаще всего для скрытия спойлеров.

Существует ещё много алгоритмов шифрования, но я не буду писать обо всех из них, потому что это не цель нашей работы. Упомяну лишь те, которые я буду использовать в работе. Но для того, чтобы перейти к более сложным алгоритмам, необходимо разобраться с некоторыми терминами.

Как мы уже знаем, **ключ** – это некая последовательность, с помощью которой зашифровывали/можно расшифровать сообщение. Алгоритмы шифрования можно разделить на следующие виды:

- **Бесключевые** – алгоритмы, которые не используют какие-либо ключи в процессе шифрования
    - Хеш-функции – выполняют «свёртку» данных в последовательность определённого размера. Фактически – это суммирование данных, чаще всего используется для проверки целостности файлов или для цифровой подписи, когда подписываются не все данные, а только хеш.

- Генераторы случайных чисел – чаще всего нужны для генерации случайных ключей, которые, в идеале, должны быть *абсолютно* случайными.

- **Одноключевые** – алгоритмы, в которых используется некий ключ, делятся на
  - **Алгоритмы симметричного шифрования** – алгоритмы шифрования, в которых для зашифровывания и расшифровывания используется один и тот же ключ, или ключ расшифровывания легко вычисляется из ключа зашифровывания и наоборот.
    - Блочное шифрование – в этом случае информация разделяется на равные блоки фиксированной длины (например, 64 или 128 байт), после чего эти блоки поочерёдно шифруются. Могут шифроваться «со сцеплением» - когда результат зашифровывания текущего блока зависит от значения предыдущего блока.
    - Потоковое шифрование – необходим, когда невозможно разделить данные на определённые блоки, скажем, когда нужно зашифровать символ, не дожидаясь остальных данных. Поэтому данные шифруются побитно (можно сказать, что потоковое шифрование – это блочное шифрование с блоками единичной длины).
  - Хеш-функции также могут выполнятся с использованием некого ключа.
  - Генераторы псевдослучайных чисел – из-за того, что практически невозможно получить *абсолютно* случайное число, их в основном генерируют на основе предыдущего числа и некого ключа (который может изменятся, например, текущее время).
  - Алгоритмы аутентификации – позволяют проверить, что пользователь (или компьютер) действительно является тем, за кого себя выдаёт. Простейшая схема аутентификации – парольная.
- **Двухключевые** – алгоритмы, в которых на различных этапах вычислений применяется два вида ключей: секретные и открытые.
  - **Алгоритмы ассиметричного шифрования** – применяют 2 вида ключей: открытый ключ для зашифровывания информации и секретный для расшифровывания. Секретный и открытые ключи связаны между собой достаточно сложным отношением, главное в котором – легкость вычисления открытого ключа из секретного и невозможность вычисления секретного ключа из открытого. Любая информация, зашифрованная открытым ключом, может быть расшифрована только секретным ключом.
  - Алгоритмы аутентификации, о которых мы уже говорили

Шифрование информации – преобразование открытых данных в зашифрованные и наоборот. Любое шифрование и расшифровывание можно представить с помощью формул:

$C = E_{k1}(M)$

$M = D_{k2}(C)$

где C (cipher text) – зашифрованный текст, E (encryption) – функция зашифровывания, M (message) – исходное сообщение, D (decryption) – функция расшифровывания, k_1 и k_2 (key) – ключи, с помощью которых происходило зашифровывание и расшифровывание информации. Чтобы эти функции работали корректно, необходимо выполнение двух условий:

- Функция расшифровывания D должна соответствовать функции шифрования E
- Ключ расшифровывания k_2 должен соответствовать ключу шифрования k_1

При этом в алгоритмах симметричного шифрования выполняется следующее правило: $k_1 = k_2$, то есть ключи для зашифровывания и расшифровывания одинаковы. В алгоритмах

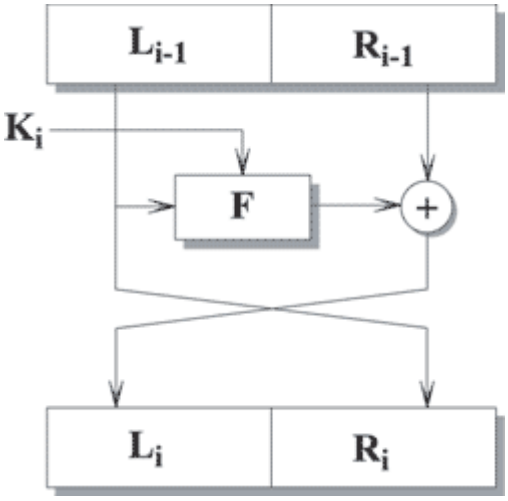
ассиметричного шифрования, где ключ зашифровывания k_1 можно вычислить из ключа расшифровывания k_2, но не наоборот, например, по следующей формуле:

$k_1 = a^{k2} \bmod p$

где a и p – параметры используемого алгоритма шифрования.

Структура алгоритмов симметричного шифрования

Большинство современных алгоритмов шифрования работают весьма схожим образом: над шифруемым текстом выполняется некое преобразование с участием ключа, которое может повторяться некоторое количество раз (раундов). Я приведу в пример один из простых алгоритмов симметричного шифрования: алгоритмы на основе сети Фейстеля. Такие алгоритмы подразумевают разбиение обрабатываемого блока данных на несколько субблоков (чаще всего на 2), т. е. это блочное шифрование, один из которых обрабатывается некой функцией *F* с ключом k_i (его ещё называют ключом раунда) и накладывается на один из оставшихся субблоков. Ключ раунда является результатом обработки ключа шифрования под *i-й* раунд (в настоящее время для симметричного шифрования считается размер 128 бит). Наложение обработанного субблока на необработанные чаще всего происходит с помощью логической операции *XOR* (исключающее или), которое применяется к каждому биту по-отдельности. Также складывание может происходить к нескольким субблокам и некоторое количество раундов. Структура получила такое название по имени Хорса Фейселя – одного из разработчиков алгоритма шифрования Lucifer. На сети Фейстеля основано большинство современных алгоритмов шифрования.



Например, шифрование DES целиком основано на сети Фейстеля.

Главное преимущество, и одновременно недостаток симметричных алгоритмов шифрования – для зашифровывания и расшифровки используется один и тот же ключ. Это упрощает его передачу, но в тоже время хакеру будет проще его найти и взломать.

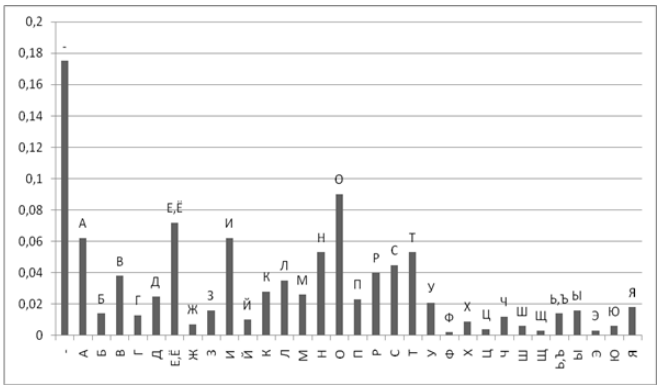
Структура алгоритмов ассиметричного шифрования

Наряду с алгоритмами симметричного шифрования есть алгоритмы ассиметричного шифрования. В них используются 2 типа ключей – открытый и закрытый. Пользователь задаёт закрытый ключ, а открытый генерируется на основе закрытого. Главная особенность – лёгкость вычисления открытого ключа из закрытого и невозможность вычисления закрытого ключа из открытого. Таким образом повышается безопасность, ведь тот, кто знает открытый ключ, который он использовал для зашифровки, не сможет узнать закрытый ключ для расшифровки. Например, алгоритм шифрования RSA использует произведение простых чисел, пропущенное через несколько функций как открытый ключ. На сегодняшний день нет эффективного алгоритма разложения числа на простые множители, поэтому почти невозможно выделить закрытый ключ из открытого. В тоже время открытый ключ доступен всем, поэтому кто угодно может зашифровать и отправить сообщение. Поэтому алгоритмы ассиметричного шифрования намного эффективнее и надёжнее, ведь пропадает риск, что кто-то из шифровальщиков опубликует ключ зашифровки, который совпадает с ключом расшифровки.

Взлом алгоритмов шифрования

Алгоритмы шифрования не совершенны и любой из них можно взломать. В алгоритме может найтись слабое место, с помощью которого можно легко узнать ключ. Я не ставил себе цели научится взламывать алгоритмы, так как это ещё более сложная и интересная тема, на которую просто не хватит времени. Но рассказывая о шифровании, просто невозможно поверхностно рассказать о алгоритмах взлома, которые используются злоумышленниками.

Для расшифровки любого алгоритма нужно знать каким именно алгоритмом он был зашифрован и ключ расшифровки. Для простоты не будем вдаваться в подробности, как узнать конкретный алгоритм шифрования, так как это очень непростая тема. Будем считать, что мы уже заранее знаем алгоритм, при котором зашифровывали сообщение, а соответственно и алгоритм, как сообщение надо расшифровать. Самый простой способ, который приходит в голову – это просто взять и перебрать все возможные ключи. Этот метод называется методом грубой силы или **brute force**. Думаю, не стоит говорить, насколько он неэффективен и медленный. К примеру, если сообщение зашифровывалось 64-х битным ключом, то на компьютере, который может за секунду перебрать 1 миллиард ключей в секунду (домашний компьютер не сможет подбирать ключи с такой скоростью, только мощный сервер) понадобится 213503 часа или 584 дня. Сейчас же стандартом шифрования является 256-ти ключ, то есть для расшифровки потребуется в 4 раза больше времени. То есть с помощью всего лишь 32-х символов можно обеспечить почти полную безопасность сообщения.

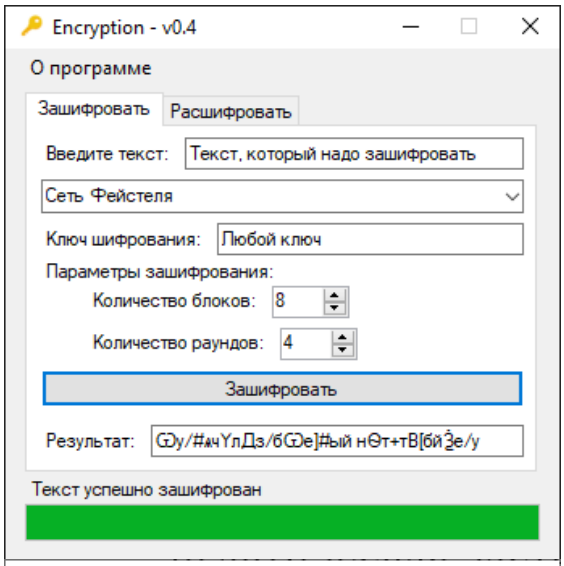


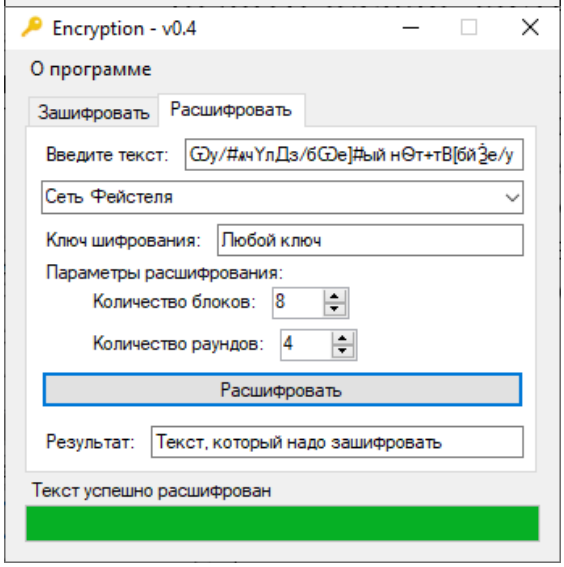
Но у каждого метода шифрования, каким бы сложным он ни был, всегда будет какая-то лазейка или недочёт, с помощью которого можно взломать сообщение. Иногда на поиск этих слабых мест уходят десятки лет. Например, шифр Цезаря и шифр простой замены можно взломать с помощью частотного анализа. Для каждой буквы в алфавите известно, сколько примерно раз она встречается. Но если просто поменять местами все буквы, частота их появления поменяется соответственно, а значит, сопоставив табличные значения частотного анализа и зашифрованного, можно узнать, какую букву заменили на какую. Такие уязвимости можно найти в каждом алгоритме, но для простоты не буду расписывать уязвимости каждого алгоритма, так как это очень сложная тема и это не цель нашей работы.

Самый же «эффективный» способ взлома сообщений – социальная инженерия. Во всей цепочке от адресата до адресанта самым слабым звеном является человек, который и держит у себя в голове пароль. И узнать пароль от человека проще, чем пытаться взломать чьё-то сообщение. Есть масса вариантов, как узнать пароль от человека, начиная «тёплой душевной» беседой, когда человек через некоторое время сам называет свой пароль, и заканчивая банальной загрузкой вируса на компьютер, который считывает нажатые клавиши на клавиатуре. Для этого, конечно, нужны знания психологии, но злоумышленнику гораздо дешевле и выгоднее «приобрести» услуги такого психолога, чем покупать дорогой сервер, который будет долго подбирать пароли, чтобы потом выяснить, что человек этот пароль уже поменял.

ПРАКТИЧЕСКАЯ ЧАСТЬ

В качестве практической части я решил написать программу, которая сможет зашифровывать и расшифровывать сообщения алгоритмами симметричного шифрования.





Возможно я позже добавлю поддержку алгоритмов ассиметричного шифрования, но они более сложные по структуре и для понимания, поэтому я ограничусь только алгоритмами асимметричного шифрования. Программу можно найти и протестировать по этой ссылке: https://github.com/LiWinDom/Encryption/releases. Я решил пока не делать суперсложные и надёжные алгоритмы, ведь вряд ли кто-то будет пользоваться моей программой для реальной передачи зашифрованных сообщений, и для этого у меня недостаточно времени. Также я не стал искать готовые алгоритмы шифрования в интернете, а делал их сам, чтобы самостоятельно понять, как эти алгоритмы работают. На данный момент в программе не очень много алгоритмов шифрования, но в будущем я буду дальше добавлять новые алгоритмы.

За основу я взял стандартный конструктор приложений в visual studio – windows forms. С помощью него можно легко создавать приложения с пользовательским интерфейсом, всего лишь перетаскивая нужные элементы на область и программируя их поведение. Таким образом можно немного упростить написание программы. Основной язык в конструкторе windows forms – C#. Он похож на язык C++, который я учу, так что мне не пришлось изучать кардинально новый язык программирования.

В начале я определился с необходимыми элементами интерфейса – это ввод текста для шифрования, ключ шифрования, выбор количества блоков и раундов шифрования, большая кнопка для запуска операции и окошечко с конечным результатом. После чего я запрограммировал каждую кнопку на определённую функцию и делаю определённые условия, чтобы предотвратить некорректный ввод пользователя и возможные из-за этого ошибки программы. После чего я создал отдельные файлы для каждого алгоритма шифрования, где есть 2 функции – зашифровать и расшифровать. В итоге получилась «модульная» программа, где я и любой другой человек с лёгкостью может добавить или исправить алгоритм шифрования, без необходимости искать в большом коде нужную функцию. После чего я сделал маленькие красивости – иконку приложения, пункт «о программе». Я опубликовал эту программу и любой желающий может её скачать, опробовать, написать идеи, пожелания, баги.

ЗАКЛЮЧЕНИЕ

В процессе данной работе я узнал историю алгоритмов шифрования и их основные принципы. Также была создана программа для шифрования и расшифровки сообщений, которую может скачать кто угодно и протестировать работу моей программы.

Таким образом поставленная задача моей проектной работы была выполнена.

В процессе работы у меня возникли некоторые трудности, которые я успешно смог преодолеть. Также стоит заметить, что некоторые вещи не получились идеально и их нужно будет переделать или исправить.

Так, например, в моей программе нет многих современных алгоритмов шифрования, из-за того, что я их не успел реализовать и добавить. Также неплохо было бы перевести программу на английский язык, чтобы ей могли пользоваться и англоговорящие люди. Ещё есть идея добавить функцию «взлома» сообщений путём перебора ключей и различных параметров шифрования, но пока у меня нет идей, как это можно грамотно реализовать.

Другим вопросом становится значение реальной необходимости в моей программе, ведь если человеку нужно зашифровать какое-то сообщение, он скорее всего воспользуется специализированным приложением, которое будет удобнее, ведь зачастую там нет выбора способа шифрования (что обычному пользователю и не особо нужно). Но зато у моей программы открытый исходный код, и любой человек может доработать её и подать запрос на обновление. Также благодаря открытому коду с комментариями легче понимать, как конкретно реализовать алгоритм шифрования, ведь на Википедии можно найти только принцип работы, а код шифрования нужно додумывать самому.

Считаю данный проект успешным.

ЛИТЕРАТУРА

Книга «Алгоритмы Шифрования» - https://litportal.ru/avtory/sergey-panasenko/kniga-algoritmy-shifrovaniya-specialnyy-spravochnik-716722.html

Принцип алгоритма шифрования DES - https://ru.wikipedia.org/wiki/DES