МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждения**

**высшего образования**

**«Казанский национальный исследовательский технологический университет»**

**(ФГБОУ ВО «КНИТУ»)**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Криптографические методы защиты информации»

на тему «Программная реализация алгоритма шифрования Энигмы»

Казань, 2022 г.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 4

1 ИСТОРИЯ

1.1 Польский этап 5

1.2 Британский этап 10

2 АНАЛИЗ АЛГОРИТМА ШИФРОВАНИЯ ЭНИГМЫ

2.1 Устройство Энигмы. Алгоритм шифрования 14

2.2 Математический анализ 15

2.3 Пример использования Энигмы 16

2.4 Криптографический анализ 17

2.5 Взлом Энигмы 18

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ШИФРОВАНИЯ ЭНИГМЫ

3.1 Обзор программы 21

3.2 Описание программы 22

4 ОЦЕНКА УРОВНЯ СЛОЖНОСТИ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ АЛГОРИТМА ЭНИГМЫ

ЗАКЛЮЧЕНИЕ 25

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26

ПРИЛОЖЕНИЕ A (обязательное) Программная реализация алгоритма шифрования «Энигмы» 29

# ВВЕДЕНИЕ

В этой курсовой работе речь пойдет об Энигме – самой популярной шифровальной машине.

Она известна тем, что использовалась немцами во время второй мировой. Сегодня шифровальные машины уже в прошлом, операции шифрования и расшифрования выполняются на процессорах. Однако сами процессоры появились именно благодаря шифровальным машинам, потому что первым программируемым компьютером считается Колосс Марк 2, построенный в 1943 году для дешифровки другой немецкой шифровальной машины – «Lorenz SZ»[1].

Конечно же, мы рассмотрим не только историю, но и то, как она устроена, каким образом работает и как её смогли взломать Британцы, среди которых был и всемирно-известный математик Алан Тьюринг.

Актуальность курсовой работы обусловлена тем, что, судя по количеству просмотров видео в интернете, эта тема интересна многим, а хорошей программы, эмулирующей работу Энигмы, я найти не смог. Собственно написание такой программы и является задачей и целью этой курсовой работы. Она поможет многим разобраться в устройстве и алгоритме работы Энигмы.

# 

# 1 ИСТОРИЯ

История электрической роторной шифровальной машины «Энигма» начинается в 1917 году с патента, полученного голландцем Хьюго Кочем. В следующем году патент был перекуплен Артуром Шербиусом, начавшим коммерческую деятельность с продажи экземпляров машины как частным лицам, так и немецкой армии и флоту[2]. До середины 1920-х годов продажи шли плохо, в частности из-за высокой цены[3].

В июне 1924 года британская криптографическая служба (Room 40) заинтересовалась устройством машины. С этой целью была закуплена партия машин у германской компании Chiffrier-maschinen AG, производившей «Энигму». Одним из условий сделки была регистрация патента в британском патентном бюро, благодаря чему криптографическая служба получила доступ к описанию криптографической схемы[3].

Начиная с 1925 года, когда германские военные начали массовые закупки шифровальной машины, и до конца Второй мировой войны было произведено около 200 тысяч машин[3]. С 1926 года на использование машины переходит германский флот, с 1928 года — сухопутные войска. Они применялись также в службе безопасности и разведке[3].

# 1.1 Польский этап

Первые перехваты сообщений, зашифрованных при помощи «Энигмы», относятся к 1926 году. Однако прочитать их долгое время не могли. В январе 1929 года коробка с коммерческим вариантом «Энигмы» случайно попала на варшавскую таможню. Германия попросила вернуть коробку, после чего её содержимым заинтересовались поляки. По поручению польского «Бюро шифров» машина была изучена специалистами фирмы «AVA», в том числе её руководителем криптоаналитиком Антонием Пальтхом, после чего коробку отправили в германское посольство. Изучение машины не позволило дешифровать сообщения, к тому же германские военные использовали свой, усиленный вариант «Энигмы»[3].

В 1928—1929 годах в Польше были организованы первые математические курсы по криптографии. Слушателями были два десятка студентов-математиков со знанием немецкого языка. Трое из слушателей — Мариан Реевский, Генрих Зыгальский и Ежи Рожицкий — поступили на службу в «Бюро шифров». Впоследствии именно они получат первые результаты по вскрытию кода «Энигмы»[3].

В 1931 году сотрудник шифрбюро министерства обороны Германии Ганс-Тило Шмидт, ставший уже агентом «Аше», начал передавать французской разведке вышедшие из употребления коды, которые, согласно служебным обязанностям, ему требовалось уничтожать, а также передал инструкцию по использованию военного варианта «Энигмы». Среди причин, побудивших Ганс-Тило сделать это, были и материальное вознаграждение, и обида на родную страну, не оценившую его успехи во время Первой мировой войны, и зависть к армейской карьере брата Рудольфа Шмидта[2]. Первыми двумя документами стали «Gebrauchsanweisung für die Chiffriermaschine Enigma»[4] и «Schlüsselanleitung für die Chiffriermaschine Enigma»[5]. Французская и английская разведки, однако, не проявили интереса к полученным данным — возможно, считалось, что вскрыть шифр «Энигмы» невозможно. Полковник французской разведки Гюстав Бертран передал материалы польскому «Бюро шифров» и продолжал передавать им дальнейшую информацию от агента до осени 1939 года[2].

# Благодаря полученным документам, криптоаналитики знали, что для шифрования сообщения оператор машины должен был выбрать из кодовой книги так называемый дневной ключ, который состоял из[6]:

# - настроек коммутационной панели (нем. Steckerverbindungen);

# - порядка установки роторов (нем. Walzenlage);

# - положений колец (нем. Ringstellung);

# - начальных установок роторов (нем. Kenngruppen).

# Однако оператор не должен был использовать дневной ключ для шифрования сообщений. Вместо этого оператор придумывал новый ключ из трёх букв (нем. Spruchschlüssel) и дважды шифровал его с использованием дневного ключа. После чего настройки роторов менялись в соответствии с придуманным ключом и производилось шифрование сообщения[7].

# В декабре 1932 года Реевский получил кодовые книги с дневными ключами для сентября и октября 1932 года. Данные материалы позволили восстановить внутреннюю электропроводку роторов и построить военный вариант «Энигмы» (коммерческий вариант уже был, но он имел другую электропроводку роторов)[8]. Более ни он, ни его коллеги не имели доступа к кодовым книгам, хотя агент «Аше» их успешно поставлял. Вероятно это было вызвано тем, что начальство хотело заставить криптоаналитиков суметь найти дневной ключ без кодовых книг, чтобы подготовиться к тому моменту, когда агент больше не сможет их поставлять[7].

# Усилия Мариана сосредоточились на анализе уязвимости протокола обмена сообщениями, а именно — на повторении ключа сообщения. Из ежедневных сообщений выбирались первые шесть букв и на их основе составлялась таблица соответствия. То есть, если первые 6 букв в сообщении «KAGBEK», то Мариан делал вывод, что «K» шифруется как «B», «A» как «E», «G» как «K».

# Особенность полного варианта таблицы заключалась в том, что пока дневной ключ остаётся без изменений, содержимое таблицы также не меняется. И, с большой степени вероятности, наоборот. Можно было бы составить каталог таблиц, однако их количество равно 26!, что делает эту работу невозможной в обозримое время. Реевский стал пытаться выделить из таблиц некоторые шаблоны или найти некоторые структурные закономерности. И это ему удалось. Он стал рассматривать цепочки букв следующего вида[7]:

# A → F → W → A

# B → Q → Z → K → V → E → L → R → I → B

# C → H → G → O → Y → D → P → C

# J → M → X → S → T → N → U → J

# Следующим открытием Мариана стало то, что хотя конкретные буквы зависели от дневной настройки «Энигмы» полностью, количество цепочек и букв в них задавалось только настройками роторов. Так как количество роторов было равно 3 (но они могли стоять в любом порядке), а начальная настройка состояла из трёх букв латинского алфавита, то число вариантов было равно 3!\*26^3=105456. Это было значительно меньше, чем 4\*10^26, что позволило, используя построенные (или украденные) машины «Энигма» составить каталог, содержащий все возможные цепочки. Данная работа заняла почти год, однако результатом стала возможность читать германскую переписку[7].

# Как отмечает Сингх, именно возможность разделить задачу на две составляющих (настройки роторов и настройки коммутационной панели) позволили Реевскому справиться с данной задачей, а также помощь как математиков «Бюро шифров», так и Шмидта[7].

# После того, как настройки роторов для дневного сообщения были восстановлены, оставалось выяснить настройки коммутационной панели. С криптографической точки зрения это был простой моноалфавитный шифр, дополнительно ограниченный лишь 6 парами замен букв. Текст даже часто не требовалось подвергать частотному криптоанализу, а всего лишь присмотреться к строкам вроде «alliveinbelrin» (англ. arrive in Berlin с заменой R ↔ L) и другим, которые было легко восстановить «на глаз»[7].

# В 1934 году Германия начала менять конфигурацию положения роторов каждый месяц вместо каждого квартала. В ответ на это Мариан Реевский спроектировал устройство под названием «циклометр», позволяющий быстро воссоздать каталог циклов[9].

# 15 сентября 1938 года Германия изменила процедуру шифрования, сделав невозможным использование каталогов, составленных с использованием циклометра. Польские криптографы были вынуждены искать другие методы для расшифровки. Такими стали использование машин bombe и перфорированных листов Зыгальского[8].

# 15 декабря 1938 года Германия добавила 4-й и 5-й роторы, а 1 января 1939 года увеличила количество соединений коммутационной панели с 6 до 10. Всё это значительно затруднило криптоанализ «Энигмы»[7], [8].

# Летом 1939 года, когда неизбежность вторжения в Польшу стала очевидна, бюро передало результаты своей работы английской и французской разведкам. Хотя польские математики так и не смогли вскрыть шифр, среди результатов были и «живые» экземпляры машины, и электромеханическая «Bomba», состоявшая из двух спаренных Энигм и помогавшая в расшифровке, а также уникальные методики криптоанализа[2].

# 1.2 Британский этап

# Дальнейшая работа по взлому «Энигмы» проходила в секретном центре британской разведки «Station X», известном впоследствии как Блетчли-парк.

# Руководителем проекта был назначен ветеран военной разведки Алистер Деннистон. Работу по дешифровке возглавил коллега Деннистона по комнате № 40, известный лингвист и криптоаналитик Альфред Нокс. За общую организацию работы отвечал профессор-математик Гордон Уэлчман. Деннистон начал набирать штат криптоаналитиков по принципу умственных способностей: лингвистов, математиков, шахматистов, чемпионов по решению кроссвордов, египтологов и даже палеонтологов. В частности, одним из первых был принят известный шахматный мастер Стюарт Милнер-Бэрри. Среди математиков был и молодой профессор логики из Кембриджа — Алан Тьюринг[2].

# Перехват радиосообщений противника выполняли десятки приёмных станций, имевших кодовое название «Y-station». Ежедневно в Блетчли-парк поступали тысячи таких сообщений. Блетчли-парк имел в своем распоряжении точную копию «Энигмы», поэтому расшифровка сообщений сводилась к подбору установки дисков и, для более поздних моделей, — штекерного коммутатора. Сложность задачи усугублялась тем, что установки роторов менялись ежедневно, поэтому службы дешифровки работали круглосуточно в три смены.

# Конструкция «Энигмы» при правильном использовании обеспечивала практически полную секретность. На практике, однако, со стороны немецких пользователей «Энигмы» зачастую допускались небрежные действия, дававшие подсказки британским аналитикам. Именно на использовании и систематизации таких погрешностей и был основан метод дешифровки.

# Подсказками служили любые часто повторяющиеся тексты, такие как приветствия, цифры (кодировались по произношению: «один», «два» и т. д.). Все подсказки заносились в картотеку вместе с контекстом: почерком радиста, местом и временем передачи и т. п.

# При отсутствии необходимого количества подсказок, особенно накануне крупных операций, проводились специальные мероприятия по их получению. Этот приём получил кодовое название «садоводство». Например, перед выходом очередного полярного конвоя проводилось демонстративное минирование определённого участка моря. Если противник докладывал результаты разминирования с указанием заранее известных координат, это давало искомую подсказку.

# Одним из основных теоретиков Блетчли-парка был Алан Тьюринг. После изучения польских материалов Тьюринг пришёл к выводу, что использовать прежний подход с полным перебором сообщений уже не получится. Во-первых, это потребует создания более 30 машин польского типа, что во много раз превышало годовой бюджет «Station X», во-вторых, можно было ожидать, что Германия может исправить конструктивный недостаток, на котором основывался польский метод. Поэтому он разработал собственный метод, основанный на переборе последовательностей символов исходного текста.

# Вскоре немцы добавили в конструкцию Энигмы коммутирующее устройство, существенно расширив этим количество вариантов кода. Возникшую для англичан задачу решил Гордон Уэлчман, предложив конструкцию «диагональной доски». В результате этой работы в августе 1940 года была построена криптоаналитическая машина Bombe. Со временем в Блетчли-Парке было установлено более 200 машин[2], что позволило довести темп расшифровки до двух-трёх тысяч сообщений в день[10].

# Хотя Bombe претерпевала некоторые изменения в деталях, её общий вид оставался прежним: шкаф массой около тонны, передняя панель два на три метра и 36 групп роторов на ней, по три в каждой. Использование машины требовало специальных навыков и сильно зависело от квалификации обслуживающего персонала — девушек-добровольцев из Женской вспомогательной службы ВМС. Впоследствии, когда часть работ была перенесена в США, вместе с технологиями была направлена и часть сотрудниц[2].

# Английское правительство делало всё возможное для того, чтобы скрыть успехи в расшифровке немецких шифров как от противника, так и от руководства СССР. С этой целью все действия, основанные на данных программы «Ультра» должны были сопровождаться операциями прикрытия, маскирующими истинный источник информации. Так, для передачи сведений «Ультра» в СССР использовалась швейцарская организация Lucy, располагавшая по легенде источником в верхах немецкого руководства. Информацию, полученную от Lucy, передавал в СССР резидент советской разведки в Швейцарии Шандор Радо.

# Для маскировки «Ультры» применялись фиктивные разведывательные полёты, радиоигра и т. п. мероприятия.

# О существовании программы «Ультра» было известно строго ограниченному кругу лиц, число которых составляло порядка десяти человек. Необходимые сведения передавались по назначению сетью подразделений разведки, прикомандированных к штабам командующих армии и флота. Источник сведений при этом не раскрывался, что иногда приводило к недооценке британским командованием вполне надёжных сведений «Ультры» и крупным потерям (см. Гибель авианосца «Глориес»).

# В числе полученной Великобританией информации были и сведения о подготовке вторжения в СССР. Несмотря на риск раскрытия источника, сведения были переданы советскому правительству[11]. Однако Сталин требовал, чтобы информация приходила от трёх независимых источников[12][13].

# Несмотря на опасения о возможности Германии слушать советские радиопереговоры, 24 июля 1941 года Черчилль распорядился всё-таки делиться с СССР информацией, получаемой в результате операции «Ультра», при условии полного исключения риска компрометации источникa.

# Некоторые авторы указывают, что с современной точки зрения шифр «Энигмы» был не очень надёжным[2]. Однако в своё время его абсолютная надёжность не вызывала никаких сомнений у немецких специалистов: до самого конца войны немецкое командование искало причины утечек секретной информации где угодно, но не в раскрытии «Энигмы». Именно поэтому успех британских дешифровщиков стал очень значительным вкладом в дело победы над нацизмом.

# После окончания войны почти все дешифровальные машины были уничтожены. В начале XXI века группа британских энтузиастов воссоздала машину Colossus, на что ушло около 10 лет[10].

# 2 АНАЛИЗ АЛГОРИТМА ШИФРОВАНИЯ ЭНИГМЫ

## 2.1 Устройство Энигмы. Алгоритм шифрования

# При нажатии на клавишу Энигмы ток не только зажигает лампочку с зашифрованной буквой, но и изменяет положение шестеренок, чтобы нажатие на следующую клавишу кодировалось по-другому, таким образом одна и та же буква будет кодироваться разными символами в зависимости от положений роторов (рисунок 1). При этом буквы образуют пары, то есть кодируются друг другом, это позволяет расшифровывать сообщения Энигмы точно таким же образом как и зашифровывать, надо просто вводить текст и смотреть на лампочки[14].

# Роторов в Энигмах установлено от трех до четырех штук в зависимости от её предназначения и года выпуска[15], при этом Энигмы могли комплектоваться дополнительными сменными роторами для увеличения количества возможных комбинаций настроек[14].

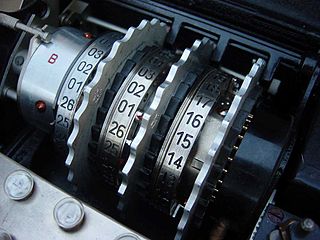


Рисунок 1 - Роторы Энигмы [16]

# Значение приходит в один контакт ротора и выходит из другого, соединения перепутаны. Сами роторы также можно менять местами. Грубо говоря за один проход по трем шестерёнкам буква изменяется три раза, а потом еще три так как сигнал пойдет через роторы в обратном направлении.

# Механизм шестеренок работает как часы. Когда первая шестеренка сделает один круг, вторая шестеренка сдвинется на одно деление. Когда вторая шестеренка сделает один круг, третья сдвинется на одно деление. По аналогии с секундами, минутами и часами соответственно.

# У военной версии Энигмы был дополнительный уровень защиты – коммутатор. Он позволяет менять провода между роторами и лампочками местами, это увеличивает количество возможных комбинаций и соответственно затрудняет подбор ключах[14].

## **2.2 Математический анализ**

# Мы знаем, что при шифровании в Энигме происходит несколько преобразований. Сперва сигнал проходит через коммутационную панель. Результат преобразования на коммутационной панели поступает в роторы. После чего сигнал попадает на рефлектор и возвращается через роторы на коммутационную панель, где выполняется последняя подстановка[15]. Все эти операции можно представить математической формулой (1):

# Ei = S-1R-1TRS, (1)

# где S-1 и S - преобразование на коммутационной панели на входе и выходе соответственно;

# R и R-1 - преобразование в роторах на входе и выходе;

# T - преобразование на рефлекторе.

## 

## **2.3 Пример использования Энигмы**

# Давайте попробуем зашифровать букву «A» с первым ротором передвинутым на одно положение назад. Все буквы взяты наугад, просто для примера. Для упрощения понимания представим, что ротор один, потому что суть от этого не поменяется, остальные роторы нужны для увеличения количества возможных комбинаций, чтобы было сложнее взломать шифр. Мы можем представить хоть 100 роторов, но смысла в этом нет.

# Сигнал от нажатия клавиши «А» придет в то место в первом роторе, в котором раньше была буква «B». Допустим, что в первом роторе контакт буквы «B» с одного конца соединен с контактом буквы «K» с другого конца. Но ток на этом контакте означал бы букву «K» если бы мы ввели букву «B» при роторе в нулевом положении, в положении 1 он придет не на контакт буквы «K», а на тот контакт, который стоит позади изначального положения буквы «K».

# После прохождения через все роторы в одном направлении электрический ток сталкивается с рефлектором. Рефлектор представляет из себя 13 проводов, которые соединяют все 26 контактов попарно. Пусть в рефлекторе соединены буквы «E» и «Q», и ток ток из буквы «K», смещенной на одно деление назад первым ротором, попадает именно на букву «E» и выходит из контакта обозначающем «Q».

# В нулевом положении контакт «Q» бы соединялся с буквой «F» с обратной стороны первого ротора, но в положении ротора -1 он будет соединяться с буквой «O», а после прохода через первый ротор в обратном направлении станет буквой «H» и на машине загорится соответствующая лампочка.

# Таким образом, при смещении ротора на одно положение назад, буква «A» зашифруется как «H», а буква «H», проходя тот же путь в обратном направлении, станет буквой «A».

## 2.4 Криптографический анализ

# Давайте посчитаем возможное количество комбинаций настроек Энигмы. Предположим, что наша модификация имеет коммутационную панель и 5 роторов, из которых 3 могут быть установлены в машину одновременно.

# Количество возможных различных вариантов расположения роторов составит 5\*4\*3=60. Для каждого из трех роторов есть 26 стартовых позиций, соответственно возможное количество комбинаций положений роторов 26\*26\*26=17.576.

# Теперь посчитаем количество возможных вариантов подключения букв в коммутаторе. В английском алфавите 26 букв, а количество комбинаций расположений этих букв равно 26!, но нам не нужны все комбинации 26 букв, у нас 10 проводов, соответственно остается 6 букв, которые не будут меняться местами и мы можем поделить 26! на 6!. Также нам не важен порядок подключения букв, мы можем подключать их в любом, следовательно делим наше значение на 10!. Пары A-B и B-A для нас равнозначны, поэтому делим результат на 2 в 10-ой степени и получаем 150.738.274.937.250 комбинаций.

# Если умножить это значение на количество комбинаций роторов и количество комбинаций их положений, то есть на 60 и 17.576, то получим 158.962.555.217.826.360.000[14]. Звучит внушающе даже по сегодняшним меркам, именно эта цифра и вводила людей в заблуждение относительно надежности шифрования Энигмы.

## 2.5 Взлом Энигмы

# Как я уже писал, из-за устройства Энигмы буквы образуют пары и не могут шифроваться сами собой. Это и было ключевой уязвимостью системы, которой Британцы успешно воспользовались. Они брали популярные слова и искали место в тексте, в котором каждая буква из этого слова не соответствует букве в тексте. Желательно взять достаточно длинное слово, чтобы снизить вероятность ошибки. Как пример, каждый день утром немцы отправляли прогноз погоды в котором было слово «Wetterbericht», которое переводится как прогноз погоды[17].

# Для расшифровки использовалась машина «Bombe» (рисунок 2). Поляки тоже называли свою машину для расшифрования немецких текстов «Bombe», однако, кроме названия, ничего общего эти два устройства не имели. Британская «Bombe» перебирала все возможные варианты настроек, в которых зашифрованные буквы будут расшифровываться как «Wetterbericht», вернее она исключала неправильные варианты, пока не оставался один верный[17].



Рисунок 2 - Машина «Bombe» [15]

# Машина работает по весьма хитроумному методу, позволяющему пренебречь настройками коммутационной панели в процессе поиска ключевой комбинации, что сводит пространство ключей Энигмы всего к 105456 комбинациям и делает весь шифр фатально уязвимым[15].

# Метод эксплуатирует наличие в паре открытый-закрытый текст так называемых «циклов». Чтобы объяснить понятие «цикл», рассмотрим следующее открытое сообщение «WETTERVORHERSAGEBISKAYA» и соответствующий ему зашифрованный текст «RWIVTYRESXBFOGKUHQBAISE»[15].

# Обратите внимание на подстановки, реализуемые энигмой в 14, 15 и 20 позициях. На 14 шаге буква A шифруется в G, а она шифруется в K на 15 шаге, затем буква K зашифровывается в A на 20 шаге, закольцовывая тем самым цепочку A-G-K-A. Такие закольцованные цепочки называются циклами. Наличие циклов позволяет разделить задачу взлома Энигмы на две простые составные части: 1) поиск стартового положения роторов и 2) поиск соединений коммутационной панели при известных установках роторов.

# После выполнения действий, показанных на рисунке 3, задача сводится к тривиальному перебору всех возможных установок ротора. Для каждой комбинации роторов необходимо проверить выполнение равенства. Если равенство выполняется для буквы S, это означает что найдена правильная конфигурация роторов и что буква A соединена на коммутационной панели с буквой S. Поиск остальных пар сводится к по буквенной расшифровке зашифрованного текста и сопоставлению результата с известным открытым текстом[15].

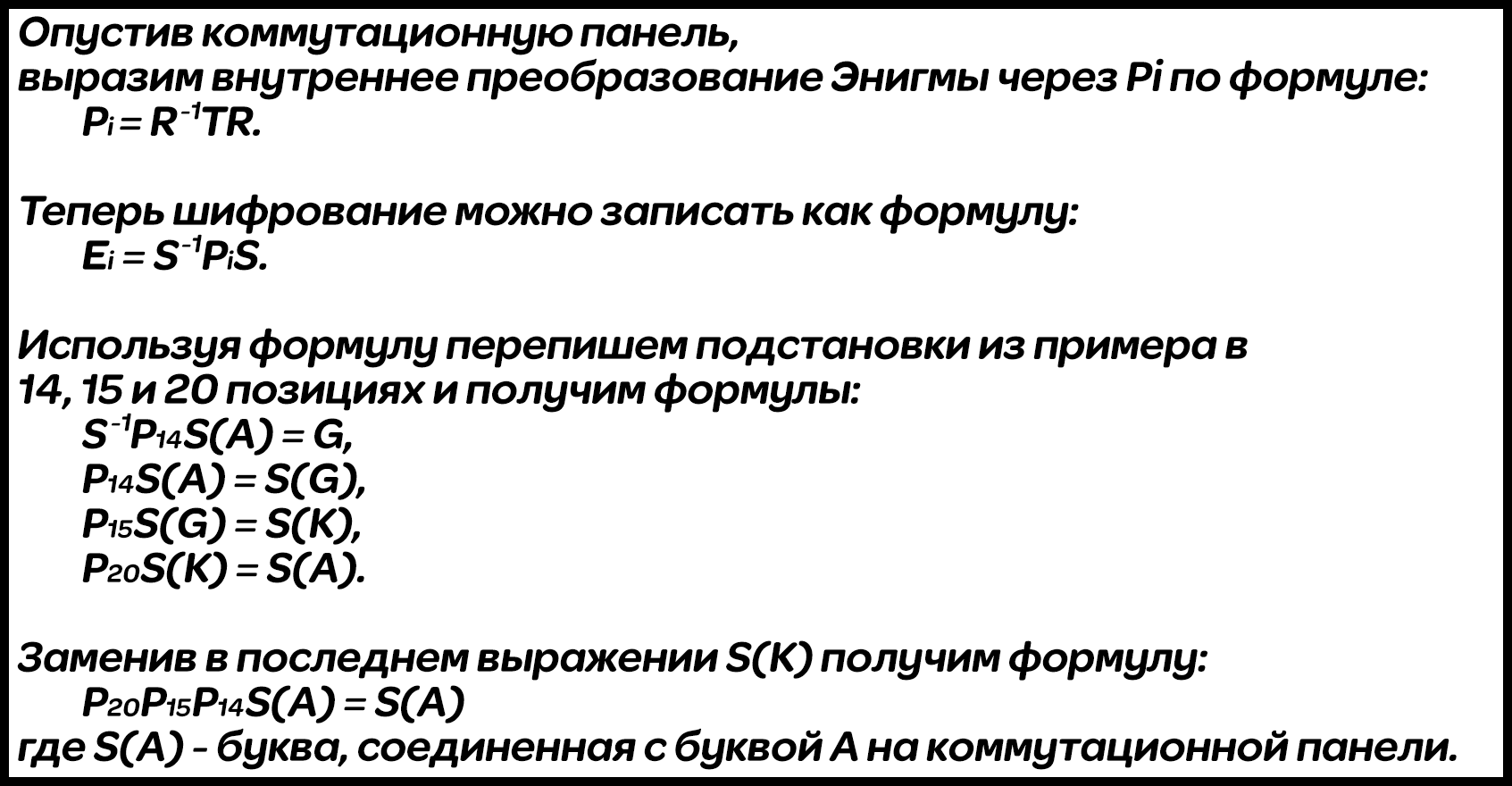


Рисунок 3 – Расчёты

# Следует отметить, что с вероятностью 1/26 равенство может выполняться и при неправильной установке роторов, поэтому для повышения надежности алгоритма желательно использовать несколько «циклов».

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ШИФРОВАНИЯ ЭНИГМЫ

# 3.1 Обзор программы

# Я написал два варианта программы-эмулятора Энигмы. Первый – простой, с одним ротором и без проверки ввода, второй – посложнее, с тремя роторами и проверкой ввода. Первый сделал для упрощения понимания людьми, не разбирающимися в программировании, но интересующимися темой. С той же целью сделано три массива со значениями роторов вместо одного. Оба варианта программы эмулируют Энигму без коммутационной панели, возможно сделаю версию с ней в будущем. Полный код программы с тремя роторами представлен в приложении А.

# Алгоритм работы версии программы с тремя роторами:

# а) получение начальных положений роторов от пользователя и проверка введенных данных на удовлетворению условию 0<input<27 функцией «rotorPosInput()»,

# б) получение символа или нескольких символов от пользователя для шифрования и перевод в верхний регистр в случае ввода в нижнем т.к. у Энигмы не было разделения на регистры,

# в) эмуляция прохождения электрического сигнала по роторам и рефлектору функциями «rotor1()», «rotor2()», «rotor3()», «reflector()». Замена буквы введенной пользователем на другую букву проводится три раза в одном направлении и три раза в другом, после прохождения через рефлектор,

# г) расчёт новых позиций роторов и смена их положений,

# д) вывод результата, одной буквы или нескольких,

# е) возврат к пункту 2.

# 3.2 Описание программы

# После запуска программа предлагает пользователю ввести положения роторов (рисунок 4).

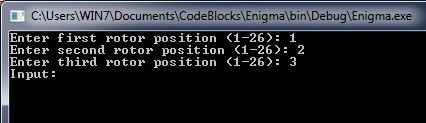


Рисунок 4 - Ввод положений роторов

# Далее она ждет ввода буквы. Для примера введём «A», «B», «C» и «D». Результат шифрования изображен на рисунке 5.

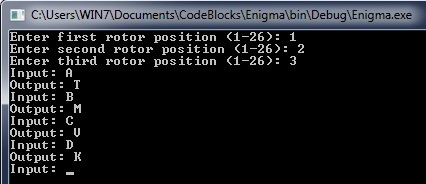


Рисунок 5 - Результат шифрования

Результатом шифрования стали буквы **«**T», «M», «V» и «K». Для расшифровки текста запускаем программу, вводим те же начальные позиции роторов и зашифрованный текст, получаем расшифрованный текст. Результат расшифрования изображен на рисунке 6.

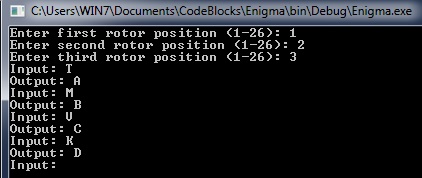


Рисунок 6 - Результат расшифрования

В таблице 1 представлены все переменные, инициализированные в программе.

Таблица 1 - Переменные в программе

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Назначение |
| reflectorValues[26] | uint8\_t array | Содержит коды замены букв рефлектором. Буква с номером элемента массива меняется буквой с номером стоящим в ячейке массива и наоборот. |
| rotor1Values[2][26],  rotor2Values[2][26],  rotor3Values[2][26] | uint8\_t array | Содержит коды замены букв роторами.  Буква с номером элемента массива меняется буквой с номером стоящим в ячейке массива и наоборот. Первая строка массива для прохода в одном направлении, вторая строка – для прохода в другом. |
| rotorPos[3] | int array | Содержит текущие положения роторов. Использовал int вместо uint8\_t т.к. у uint8\_t были проблемы с вводом, которые я не хотел решать. |
| input | uint8\_t | Используется в функциях. Означает переданный функции номер буквы. |
| direction | uint8\_t | Используется в функциях. Означает направление. 0 – Вперед, 1 – обратно. |
| result | uint8\_t | Используется в функциях для хранения промежуточного результата. |
| symbol | char | Содержит введенный символ, по сути нужна только для использования функции “toupper”. |
| asciiVal | uint8\_t | Используется для хранения ASCII-кода. |

**4 ОЦЕНКА УРОВНЯ СЛОЖНОСТИ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ АЛГОРИТМА ЭНИГМЫ**

Да, шифр Энигмы имеет целых 158.962.555.217.826.360.000 вариантов различных ключей, и в своё время это было удивительно. Однако это же создавало иллюзию защищенности шифра, который на самом деле имеет серьезные уязвимости, которые позволяют быстро взламывать его, используя устройства сравнимые по мощности с калькулятором.

Нами был рассмотрен алгоритм взлома Энигмы. Различные её версии успешно взламывали в первой половине прошлого века, поэтому про современные стандарты говорить не приходится,– до них она явно не дотягивает.

**ЗАКЛЮЧЕНИЕ**

Шифровальная машина Энигма была весьма кстати в своё время. На данный момент она, конечно, устарела и представляет скорее историческую ценность, чем практическую. Большинство уцелевших экземпляров находится у коллекционеров и в музеях.

В ходе выполнения данного курсового проекта я выполнил свою задачу по изучению Энигмы и её алгоритма шифрования. Мной были получены знания и опыт в областях истории, инженерии, программирования и криптографии. Я изучил устройство и алгоритм шифрования Энигмы, алгоритм взлома Энигмы, провел анализ алгоритмов и написал свои версии эмуляторов Энигмы на языке C++ в виде консольных приложений.

Результат работы в виде двух программ был опубликован на сайте **«**Github.com», чтобы люди, интересующиеся темой, могли с ним ознакомится. Эти программы могут быть использованы людьми при изучении истории или при программирования криптографических систем в качестве примера таковой.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Colossus (компьютер) / Текст: электронный // ru.wikipedia.org –URL: https://ru.wikipedia.org/wiki/Colossus\_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) (дата обращения: 19.05.2022).

2 Тайны проекта Ultra / Текст: электронный // osp.ru, автор Леонид Черняк – URL: https://www.osp.ru/os/2003/07-08/183294 (дата обращения: 03.05.2022).

3 Вторая мировая война в эфире: некоторые аспекты операции «Ультра» / Текст: электронный // agentura.ru, Д. А. Ларин, к. т. н., Г. П. Шанкин, д. т. н., профессор. В рамках совместного проекта с журналом "Защита информации. Инсайд" - URL: https://web.archive.org/web/20140120091311/http://www.agentura.ru/press/about/jointprojects/inside-zi/ultra1/ (дата обращения: 08.05.2022).

4 Gebrauchsanweisung für die Chiffriermaschine Enigma / Текст: электронный // ilord.com – URL: http://web.archive.org/web/20110517183020/http://www.ilord.com/enigma-manual1937-german.html (дата обращения: 08.05.2022).

5 Schlüsselanleitung für die Chiffriermaschine Enigma / Текст: электронный // ilord.com – URL: http://web.archive.org/web/20150924033652/http://www.ilord.com/enigma-manual1940-german.pdf (дата обращения: 08.05.2022).

6 Enigma Message Procedures / Текст: электронный // ciphermachinesandcryptology.com – URL: https://www.ciphermachinesandcryptology.com/en/enigmaproc.htm (дата обращения: 08.05.2022).

7 Книга шифров. Тайная история шифров и их расшифровки / Текст: электронный // vixri.ru, Саймон Сигнх – URL: http://www.vixri.ru/d/Singx%20Sajmon%20\_Kniga%20shifrov.pdf (дата обращения: 08.05.2022).

8 A conversation with Marian Rejewski / Текст: электронный // tandfonline.com – URL: https://www.tandfonline.com/doi/abs/10.1080/0161-118291856830 (дата обращения: 08.05.2022).

9 Защита компьютерной информации / Текст: электронный // https://uchebnik.biz, Борис Анин – URL: https://uchebnik.biz/book/57-zashhita-kompyuternoj-informacii/ (дата обращения: 08.05.2022).

10 Британские энтузиасты воссоздали дешифратор "Энигмы" / Текст: электронный // soft.compulenta.ru, Владимир Парамонов – URL: http://soft.compulenta.ru/285638/ (дата обращения: 08.05.2022).

11 Colossus: Breaking the German ‘Tunny’ Code at Bletchley Park. An Illustrated History / Текст: электронный // rutherfordjournal.org, B. Jack Copeland – URL: http://www.rutherfordjournal.org/article030109.html#section05 (дата обращения: 08.05.2022).

12 Операция «УЛЬТРА» / Текст: электронный // avidreaders.ru, Уинтерботэм Фред – URL: https://avidreaders.ru/book/operaciya-ultra.html (дата обращения: 08.05.2022).

13 Ultra Goes to War / Lewin, Ronald // Classic Military History, London, England: Hutchinson & Co, 2001 - ISBN 978-1-56649-231-7.

14 158,962,555,217,826,360,000 (Enigma Machine) / Видео // YouTube.com, канал Numberphile – URL: https://www.youtube.com/watch?v=G2\_Q9FoD-oQ (дата обращения: 03.05.2022).

15 Криптоанализ «Энигмы» / Текст: электронный // Habr.com, блог пользователя NeverWalkAloner – URL: https://habr.com/ru/post/269519/ (дата обращения: 03.05.2022).

16 Роторы «Энигмы» / Фото // ru.wikipedia.org – URL: https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D0%B8%D0%B3%D0%BC%D0%B0 (дата обращения: 19.05.2022).

17 Flaw in the Enigma Code / Видео // YouTube.com, канал Numberphile – URL: https://www.youtube.com/watch?v=V4V2bpZlqx8 (дата обращения: 03.05.2022).

**Приложение А**

**(обязательное)**

**Программная реализация алгоритма шифрования «Энигмы»**

1. using namespace std;
2. #include <iostream>
3. #include<limits>
4. // A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
5. const uint8\_t reflectorValues[26]={12,5,18,4,3,1,25,13,11,21,22,8,0,7,15,14,24,20,2,23,17,9,10,19,16,6};
6. const uint8\_t rotor1Values[2][26]={{13,25,20,22,11,1,17,23,4,9,2,5,18,16,6,21,3,8,15,7,12,24,10,0,14,19},{23,5,10,16,8,11,14,19,17,9,22,4,20,0,24,18,13,6,12,25,2,15,3,7,21,1}};
7. const uint8\_t rotor2Values[2][26]={{11,14,22,21,25,1,2,13,23,4,9,8,15,18,17,20,3,19,0,12,10,7,16,6,5,24},{18,5,6,16,9,24,23,21,11,10,20,0,19,7,1,12,22,14,13,17,15,3,2,8,25,4}};
8. const uint8\_t rotor3Values[2][26]={{21,11,25,22,19,17,24,6,5,13,18,9,20,14,16,2,3,0,4,23,15,12,10,8,1,7},{17,24,15,16,18,8,7,25,23,11,22,1,21,9,13,20,14,5,10,4,12,0,3,19,6,2}};
9. int rotorPos[3]={0,0,0};
10. void rotorPosInput(){
11. cout << "Enter first rotor position (1-26): ";
12. while(!(cin >> rotorPos[0])){
13. cin.clear();
14. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
15. cout << "Invalid input. Try again: ";
16. }
17. while ((rotorPos[0]>26)||(rotorPos[0]<0)){
18. cout << "Invalid input. Try again: ";
19. while(!(cin >> rotorPos[0])){
20. cin.clear();
21. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
22. cout << "Invalid input. Try again: ";
23. }
24. }
25. rotorPos[0]-1;
26. cout << "Enter second rotor position (1-26): ";
27. while(!(cin >> rotorPos[1])){
28. cin.clear();
29. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
30. cout << "Invalid input. Try again: ";
31. }
32. while ((rotorPos[1]>26)||(rotorPos[1]<0)){
33. cout << "Invalid input. Try again: ";
34. while(!(cin >> rotorPos[1])){
35. cin.clear();
36. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
37. cout << "Invalid input. Try again: ";
38. }
39. }
40. rotorPos[1]-1;
41. cout << "Enter third rotor position (1-26): ";
42. while(!(cin >> rotorPos[2])){
43. cin.clear();
44. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
45. cout << "Invalid input. Try again: ";
46. }
47. while ((rotorPos[2]>26)||(rotorPos[2]<0)){
48. cout << "Invalid input. Try again: ";
49. while(!(cin >> rotorPos[2])){
50. cin.clear();
51. cin.ignore(numeric\_limits<streamsize>::max(), '\n');
52. cout << "Invalid input. Try again: ";
53. }
54. }
55. rotorPos[2]-1;
56. }
57. void changeRotorPos(){
58. rotorPos[0]++;
59. if (rotorPos[0]>25){
60. rotorPos[1]++;
61. rotorPos[0]=0;
62. if (rotorPos[1]>25){
63. rotorPos[2]++;
64. rotorPos[1]=0;
65. if (rotorPos[2]>25){
66. rotorPos[2]=0;
67. }
68. }
69. }
70. }
71. uint8\_t reflector(uint8\_t input){
72. return 65+reflectorValues[input-65];
73. }
74. uint8\_t rotor1(uint8\_t direction, uint8\_t input){
75. input=input-65+rotorPos[0];
76. if (input>25){
77. input=input-26;
78. }
79. uint8\_t result=65-rotorPos[0]+rotor1Values[direction][input];
80. if (result<65){
81. result=result+26;
82. }
83. return result;
84. }
85. uint8\_t rotor2(uint8\_t direction, uint8\_t input){
86. input=input-65+rotorPos[1];
87. if (input>25){
88. input=input-26;
89. }
90. uint8\_t result=65-rotorPos[1]+rotor2Values[direction][input];
91. if (result<65){
92. result=result+26;
93. }
94. return result;
95. }
96. uint8\_t rotor3(uint8\_t direction, uint8\_t input){
97. input=input-65+rotorPos[2];
98. if (input>25){
99. input=input-26;
100. }
101. uint8\_t result=65-rotorPos[2]+rotor3Values[direction][input];
102. if (result<65){
103. result=result+26;
104. }
105. return result;
106. }
107. int main()
108. {
109. rotorPosInput();
110. char symbol;
111. cout << "Input: "; // You can input several symbols if you want, not only one
112. cin >> symbol;
113. symbol = toupper(symbol); // We'll use only UPPER CASE.
114. uint8\_t asciiVal=uint8\_t(symbol);
115. while (asciiVal!=46){ // Ascii 46 = dot .
116. asciiVal=rotor1(0,asciiVal); // Forward, vpered -->>
117. asciiVal=rotor2(0,asciiVal);
118. asciiVal=rotor3(0,asciiVal);
119. asciiVal=reflector(asciiVal);
120. asciiVal=rotor3(1,asciiVal); // Backward, nazad <<--
121. asciiVal=rotor2(1,asciiVal);
122. asciiVal=rotor1(1,asciiVal);
123. changeRotorPos();
124. cout << "Output: " << asciiVal << endl << "Input: ";
125. cin >> symbol;
126. symbol = toupper(symbol);
127. asciiVal=uint8\_t(symbol);
128. }
129. return 0;
130. }

**ЛИСТ НОРМОКОНТРОЛЯ**

1. Лист является обязательным приложением к пояснительной записке курсового проекта.

2. Нормоконтролер имеет право возвращать документацию без рассмотрения в случаях:

-нарушения установленной комплектности,

-отсутствия обязательных подписей,

-нечеткого выполнения текстового и графического материала.

3. Устранение ошибок, указанных нормоконтролером, обязательно.

**ПЕРЕЧЕНЬ**

замечаний и предложений нормоконтролера по дипломному курсовому проекту,

студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(группа, инициалы, фамилия)

|  |  |  |
| --- | --- | --- |
| Лист  (страница) | Условное обозна-  чение (код ошибок) | Содержание замечаний и предложений со ссылкой на нормативный документ, стандарт или типовую документацию |
|  |  |  |

Дата «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись) фамилия, инициалы