**Оглавление**

[Введение 2](#_Toc532669439)

[Глава 1. Разработка технического задания 20](#_Toc532669440)

[2. Основание для разработки 20](#_Toc532669441)

[3. Назначение 20](#_Toc532669442)

[4.Требования к программной системе 21](#_Toc532669443)

[5. Требования к программной документации 21](#_Toc532669444)

[Глава 2. Анализ предметной деятельности , уточнение спецификаций и разработка структурной схемы 22](#_Toc532669445)

[Глава 3. Инструкция пользователя. 26](#_Toc532669446)

[Библиография 28](#_Toc532669447)

[Используемая литература 28](#_Toc532669448)

[Интернет-ресурсы 28](#_Toc532669449)

[Листинг Программы 29](#_Toc532669450)

**Введение**

Шифрование текста свои истоки берет из древности во время войн, шпионажей и т. п. В чем же заключается шифрование информации. Мы, имея определенный алгоритм, зашифровываем текст, так чтобы кто-то посторонний не смог понять сообщение, которое зашифровано, ведь, не имея алгоритма почти невозможно расшифровать сообщение или же нужно много времени, но все это зависит от степени сложности алгоритма. В древности был изобретен метод называемый шифр Цезаря, который шифрует сообщение по ключу. В шифрование участвуют 2 вещи, а именно текст и алгоритм шифрования (ключ).

Шифротекст

Текст

Ключ

**Шифротекст** – зашифрованный текст, который мы будем передавать по каналу связи.

Bob

Alice

Шифротекст

Ключ

Как видим Алиса передает шифротекст с ключом Бобу, обычно ключ передают по защищенному каналу связи, так как открытый канал связи могут прослушивать, а некоторые и изменять данные. Если же злоумышленнику попадет шифротекст с ключом в руки, то он легко может прочитать сообщение. Даже если ключ не передается, то есть что ваш текст будет расшифрован, но на это потребуется не мало времени и ресурсов. А к тому времени, когда текст будет расшифрован в большинстве случаях он будет уже бесполезен.

Eva

Alice

Bob

До нашего времени шифрование усовершенствовалась. Было придумано много разных шифров: Шифр Виженера, Потоковые шифры, Блочные шифры и многое другое.

Также шифры распределяются на симметричные и асимметричные. Симметричные же делятся на блочные и поточные шифры.

Симметричные шифры основаны на открытом ключе, асимметричные же основаны на ключе, состоящем из 2 частей: открытой и закрытой. То есть в большинстве случаев, зная открытую часть ключа, сам ключ будет сложно расшифровать, потому что потребуется много времени и ресурсов

Шифры

Асимметричные

Симметричные

Синхронные

Асинхронные

**Шифр Цезаря**

Допустим у нас есть русский алфавит, который нумеруется:

0. A 17. Р

1. Б 18. С

2. В 19. Т

3. Г 20. У

4. Д 21. Ф

5. Е 22. Х

6. Ё 23. Ц

7. Ж 24. Ч

8. З 25. Ш

9. И 26. Щ

10. Й 27. Ь

11. К 28. Ы

12. Л 29. Ъ

13. М 30. Э

14. Н 31. Ю

15. О 32. Я

16. П

После мы в нашем сообщение на русском языке меняем буквы на цифры.

Исходный текст: Мама после собрания уставшая пришла домой.

Преобразованный текст: 13;0;13;0 16;15;18;12;5 18;15;1;17;0;14;9;32 20;18;19;0;2;25;0;32 16;17;9;25;12;0 4;15;13;15;10

Дальше мы имеем число(ключ). Обычно он имеет n значений от 0 до n-1 , где n – количество букв в использованном алфавите. Конечно же ключ может быть и больше n-1 но в этом нету смысла так как мы получим те же значения что и при ключе (key % n).

С помощью ключа мы изменяем исходный текст .Теперь он будет составляться из суммы ключа с числом, которым мы заменили букву в тексте по модулю числа букв в алфавите ((key+a[k]) % n), то есть при ключе равном 5 наш преобразованный текст станет таким:

18;5;18;5 21;20;23;17;10 23;20;6;22;5;19;4 25;23;24;5;7;30;5;4 21;22;14;30;17;5 9;20;18;20;15

А сам зашифрованный текст будет выглядеть так:

Сесе фуцрй цуёхетд шцчежэед фхнэре иусуо

Можно заметить, что зашифрованный текст непонятен: нет ни связи слов и словосочетаний, ни смысла. Но шифр Цезаря очень уязвим к атакам, ведь достаточно только узнать количество букв в алфавите и с помощью расписываний каждых из n ключей от 0 до n-1. Ведь почти всегда осмысленные слова встречаются в 2 или 3 ситуаций, а из них уже выбор не велик.

Сейчас будет показано наглядный алгоритм шифра цезаря с ключом равным 3.

Шифрование с использованием ключа {\displaystyle k=3}. Буква «Е» «сдвигается» на три буквы вперёд и становится буквой «З». Твёрдый знак, перемещённый на три буквы вперёд, становится буквой «Э», буква «Я», перемещённая на три буквы вперёд, становится буквой «В», и так далее:

Исходный алфавит: А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я

Шифрованный: Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я А Б В

Оригинальный текст:

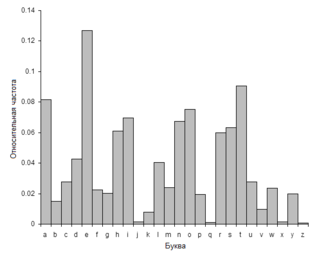
Съешь же ещё этих мягких французских булок, да выпей чаю.

Шифрованный текст получается путём замены каждой буквы оригинального текста соответствующей буквой шифрованного алфавита:

Фэзыя йз зьи ахлш пвёнлш чугрщцкфнлш дцосн, жг еютзм ъгб.

Шифр может быть взломан, используя те же самые методы что и для простого шифра подстановки, такие как [частотный анализ](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) и т. д. Используя эти методы, взломщик, вероятно, быстро заметит регулярность в решении и поймёт, что используемый шифр — это шифр Цезаря.

Ниже приведена гистограмма частотного анализа для английского алфавита.



**Шифр Виженера**

**Шифр Виженера**— метод полиалфавитного [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80) буквенного текста.[](https://commons.wikimedia.org/wiki/File:Confederate_cipher_disk.jpg?uselang=ru)

Репродукция шифровального диска Конфедерации

В 1466 году [Леон Альберти](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D0%B1%D0%B5%D1%80%D1%82%D0%B8,_%D0%9B%D0%B5%D0%BE%D0%BD_%D0%91%D0%B0%D1%82%D1%82%D0%B8%D1%81%D1%82%D0%B0), знаменитый архитектор и философ представил трактат о шифрах в папскую канцелярию. В трактате рассматриваются различные способы шифрования, в том числе маскировка открытого текста в некотором вспомогательном тексте. Работа завершается собственным шифром, который он назвал «шифр, достойный королей». Это был многоалфавитный шифр, реализованный в виде шифровального диска. Суть заключается в том, что в данном шифре используется несколько замен в соответствии с ключом. Позднее Альберти изобрел код с перешифровкой. Данное изобретение значительно опередило свое время, поскольку данный тип шифра стал применяться в странах Европы лишь 400 лет спустя.

В 1518 году в развитии криптографии был сделан новый шаг благодаря появлению в Германии первой печатной книги по криптографии. Аббат Иоганнес Тритемий, настоятель монастыря в Вюрцбурге, написал книгу «Полиграфия», в которой дается описание ряда шифров. Один из них использует «таблицу Тритемия» (ныне «таблицу Виженера») и развивает идею многоалфавитной замены. Система шифрования следующая: первая буква исходного текста шифруется по первой строке, вторая по второй и так далее. После использования последней строки следующая буква вновь шифруется по первой строке. В шифре Тритемия отсутствует ключ, секретом является сам способ шифрования.

Следующий шаг в развитии предложенного Тритемием способа шифрования был сделан итальянцем Джовани Белазо. В 1553 году выходит в свет его брошюра «Шифр синьора Белазо». В этом шифре ключом является так называемый пароль — фраза или слово. Пароль записывался периодически над буквами открытого текста. Буква пароля, стоящая над соответствующей буквой открытого текста, указывала номер строки в таблице Тритемия, по которой следует проводить замену (шифрование) это буквы.

В последующем идеи Тритемия и Белазо развил соотечественник Белазо [Джованни Батиста Порта](https://ru.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B8_%D0%91%D0%B0%D1%82%D1%82%D0%B8%D1%81%D1%82%D0%B0_%D0%9F%D0%BE%D1%80%D1%82%D0%B0). Он предложил отказаться от алфавитного порядка следования букв в первой строке таблицы Тритемия и заменить этот порядок на некоторый произвольный, являющийся ключом шифра. Строки таблицы по-прежнему циклически сдвигались. В своей книге «О тайной переписке», (вышедшей в 1563 году) Порта предложил [биграммный шифр](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80), а также привел описание механического дискового устройства, реализующего биграммную замену.

В середине XVI века в Италии появляется книга [Дж. Кардано](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE,_%D0%94%D0%B6%D0%B5%D1%80%D0%BE%D0%BB%D0%B0%D0%BC%D0%BE) «О тонкостях» с дополнением «О разных вещах». Там нашли отражение новые идеи криптографии: использование части самого передаваемого открытого текста в качестве ключа шифра (идея «самоключа») и новый способ шифрования, который вошел в историю как «[решетка Кардано](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D1%91%D1%82%D0%BA%D0%B0_%D0%9A%D0%B0%D1%80%D0%B4%D0%B0%D0%BD%D0%BE)».

Посол Франции в Риме [Блез де Виженер](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80,_%D0%91%D0%BB%D0%B5%D0%B7), познакомившись с трудами Тритемия, Белазо, Кардано, Порта, Альберти, также увлекся криптографией. В 1585 году он написал «Трактат о шифрах», в котором излагаются основы криптографии. В этом труде он замечает: «Все вещи в мире представляют собой шифр. Вся природа является просто шифром и секретным письмом». Эта мысль была позднее повторена [Блезом Паскалем](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C,_%D0%91%D0%BB%D0%B5%D0%B7) — одним из основоположников теории вероятностей, а в настоящее время и [Норбертом Винером](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BD%D0%B5%D1%80,_%D0%9D%D0%BE%D1%80%D0%B1%D0%B5%D1%80%D1%82) — «отцом кибернетики».

По сути дела Виженер объединил подходы Тритемия, Беллазо, Порта к шифрованию открытых текстов, по существу не внеся в них ничего оригинального. В наше время «шифр Виженера», состоящий в периодическом продолжении ключевого слова по таблице Тритемия, вытеснил имена его предшественников.[[4]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0#cite_note-:1-4) [Дэвид Кан](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD,_%D0%94%D1%8D%D0%B2%D0%B8%D0%B4) в своей книге «Взломщики кодов» отозвался об этом осуждающе, написав, что история «проигнорировала важный факт и назвала шифр именем Виженера, несмотря на то, что он ничего не сделал для его создания»[[7]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0#cite_note-:2-7).

Шифр Виженера имел репутацию исключительно стойкого к «ручному» взлому. Известный писатель и математик Чарльз Лютвидж Доджсон ([Льюис Кэрролл](https://ru.wikipedia.org/wiki/%D0%9B%D1%8C%D1%8E%D0%B8%D1%81_%D0%9A%D1%8D%D1%80%D1%80%D0%BE%D0%BB%D0%BB)) назвал шифр Виженера невзламываемым в своей статье «Алфавитный шифр» [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *The Alphabet Cipher*, опубликованной в детском журнале в 1868 году. В 1917 году [Scientific American](https://ru.wikipedia.org/wiki/Scientific_American)также отозвался о шифре Виженера, как о неподдающемся взлому.[[8]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0#cite_note-8) Это представление было опровергнуто после того, как [Касиски](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%B8%D1%81%D0%BA%D0%B8,_%D0%A4%D1%80%D0%B8%D0%B4%D1%80%D0%B8%D1%85) полностью взломал шифр в XIX веке, хотя известны случаи взлома этого шифра некоторыми опытными криптоаналитиками ещё в XVI веке.

Шифр Виженера достаточно прост для использования в полевых условиях, особенно если применяются шифровальные диски. Например, «конфедераты» использовали медный шифровальный диск для шифра Виженера в ходе [Гражданской войны](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B6%D0%B4%D0%B0%D0%BD%D1%81%D0%BA%D0%B0%D1%8F_%D0%B2%D0%BE%D0%B9%D0%BD%D0%B0_%D0%B2_%D0%A1%D0%A8%D0%90). Послания Конфедерации были далеки от секретных, и их противники регулярно взламывали сообщения. Во время войны командование Конфедерации полагалось на три ключевых словосочетания: «Manchester Bluff», «Complete Victory» и — так как война подходила к концу — «Come Retribution».

[Гилберт Вернам](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BB%D0%B1%D0%B5%D1%80%D1%82_%D0%92%D0%B5%D1%80%D0%BD%D0%B0%D0%BC) попытался улучшить взломанный шифр (он получил название шифр Вернама-Виженера в 1918 году), но, несмотря на его усовершенствования, шифр так и остался уязвимым к [криптоанализу](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7). Однако работа Вернама в конечном итоге всё же привела к получению [шифра Вернама](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B5%D1%80%D0%BD%D0%B0%D0%BC%D0%B0), который действительно невозможно взломать.

В [шифре Цезаря](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F) каждая буква алфавита сдвигается на несколько позиций; например в шифре Цезаря при сдвиге +3, A стало бы D, B стало бы E и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет такой вид:

ATTACKATDAWN

Человек, посылающий сообщение, записывает ключевое слово («LEMON») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

LEMONLEMONLE

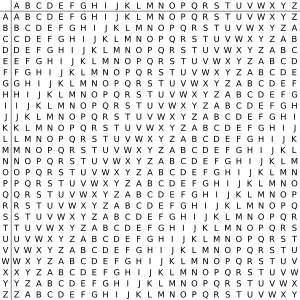
Первый символ исходного текста A зашифрован последовательностью L, которая является первым символом ключа. Первый символ L шифрованного текста находится на пересечении строки L и столбца A в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; то есть второй символ шифрованного текста X получается на пересечении строки E и столбца T. Остальная часть исходного текста шифруется подобным способом.

Исходный текст: ATTACKATDAWN

Ключ: LEMONLEMONLE

Зашифрованный текст: LXFOPVEFRNHR

Расшифровывание производится следующим образом: находим в таблице Виженера строку, соответствующую первому символу ключевого слова; в данной строке находим первый символ зашифрованного текста. Столбец, в котором находится данный символ, соответствует первому символу исходного текста. Следующие символы зашифрованного текста расшифровываются подобным образом. В компьютере такая операция соответствует сложению кодов ASCII символов сообщения и ключа по некоторому модулю. Кажется, что если таблица будет более сложной, чем циклическое смещение строк, то шифр станет надежнее. Это действительно так, если ее менять чаще, например, от слова к слову. Но составление таких таблиц, представляющих собой латинские квадраты, где любая буква встречается в строке или столбце один раз, трудоемко и его стоит делать лишь на ЭВМ. Для ручного же многоалфавитного шифра полагаются лишь на длину и сложность ключа, используя приведенную таблицу, которую можно не держать в тайне, а это упрощает шифрование и расшифровывание.



В 1881 году Жюль Верн написал роман [Жангада](https://ru.wikipedia.org/wiki/%D0%96%D0%B0%D0%BD%D0%B3%D0%B0%D0%B4%D0%B0._%D0%92%D0%BE%D1%81%D0%B5%D0%BC%D1%8C%D1%81%D0%BE%D1%82_%D0%BB%D1%8C%D1%91_%D0%BF%D0%BE_%D0%90%D0%BC%D0%B0%D0%B7%D0%BE%D0%BD%D0%BA%D0%B5). В данном романе автор использовал для зашифровки документа шифр Виженера. В качестве зашифрованного текста, автор использует следующий документ:

СГУЧПВЭЛЛЗИРТЕПНДНФГИНБОРГЙУГЛЧД

КОТХЖГУУМЗДХРЪСГСЮДТПЪАРВЙГГИЩВЧ

ЭЕЦСТУЖВСЕВХАХЯФБЬБЕТФЗСЭФТХЖЗБЗ

ЪГФБЩИХХРИПЖТЗВТЖЙТГОЙБНТФФЕОИХТ

ТЕГИИОКЗПТФЛЕУГСФИПТЬМОФОКСХМГБТ

ЖФЫГУЧОЮНФНШЗГЭЛЛШРУДЕНКОЛГГНСБК

ССЕУПНФЦЕЕЕГГСЖНОЕЫИОНРСИТКЦЬЕДБ

УБТЕТЛОТБФЦСБЮЙПМПЗТЖПТУФКДГ

По ходу истории герои находят фрагмент расшифрованного слова к этому документу: ОРТЕГА Герои догадались, что это имя может обозначать подпись в конце документа. Таким образом выходит:

О Р Т Е Г А

{\displaystyle {16\ 18\ 20\ 6\ 4\ 1}}

Т У Ф К Д Г

{\displaystyle {20\ 21\ 22\ 12\ 5\ 4}}

Следовательно, ключ: 432513. Зная ключ можно легко перевести данный документ:

НАСТОЯЩИЙ ВИНОВНИК КРАЖИ АЛМАЗОВ

{\displaystyle {43251343251343251343251343251}}

СГУЧПВЭЛЛ ЗИРТЕПНД НФГИН БОРГЙУГ

И УБИЙСТВА СОЛДАТ ОХРАНЫ В НОЧЬ НА

{\displaystyle {3432513432513432513432513432}}

Л ЧДКОТХЖГ УУМЗДХ РЪСГСЮ Д ТПЪА РВ

ДВАДЦАТЬ ВТОРОЕ ЯНВАРЯ ТЫСЯЧА

{\displaystyle {51343251343251343251343251}}

ЙГГИЩВЧЭ ЕЦСТУЖ ВСЕВХА ХЯФБЬБ

ВОСЕМЬСОТ ДВАДЦАТЬ ШЕСТОГО ГОДА

{\displaystyle {3432513432513432513432513432}}

ЕТФЗСЭФТХ ЖЗБЗЪГФБ ЩИХХРИП ЖТЗВ

НЕ ЖОАМ ДАКОСТА, НЕСПРАВЕДЛИВО ПРИ

{\displaystyle {51343251343251343251343251343}}

ТЖ ЙТГО ЙБНТФФЕ ОИХТТЕГИИОКЗП ТФЛ

ГОВОРЕННЫЙ К СМЕРТИ, А Я, НЕСЧАСТНЫЙ

{\displaystyle {25134325134325134325134325134}}

ЕУГСФИПТЬМ О ФОКСХМ Г Б ТЖФЫГУЧОЮН

СЛУЖАЩИЙ УПРАВЛЕНИЯ АЛМАЗНОГО

{\displaystyle {325134325134325134325134325}}

ФНШЗГЭЛЛ ШРУДЕНКОЛГ ГНСБКССЕУ

ОКРУГА; ДА, Я ОДИН, В ЧЕМ И ПОДПИСЫ

{\displaystyle {1343251343251343251343251}}

ПНФЦЕЕ ЕГ Г СЖНО И ЫИО Н РСИТКЦЬ

ВАЮСЬ СВОИМ НАСТОЯЩИМ ИМЕНЕМ,

{\displaystyle {3432513432513432513432513432513}}

ЕДБУБ ТЕТЛО ТБФЦСБЮЙП МПЗТЖП

ОРТЕГА

{\displaystyle {432513}}

ТУФКДГ

**Потоковые Шифры**

Потоковые шифры на базе сдвиговых [регистров](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_(%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) активно использовались в годы войны, ещё задолго до появления электроники. Они были просты в проектировании и реализации.

В 1965 году Эрнст Селмер, главный криптограф норвежского правительства, разработал теорию последовательности сдвиговых [регистров](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_(%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)). Позже [Соломон Голомб](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BB%D0%BE%D0%BC%D0%B1,_%D0%A1%D0%BE%D0%BB%D0%BE%D0%BC%D0%BE%D0%BD_%D0%92%D0%BE%D0%BB%D1%8C%D1%84), [математик](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA) [Агентства Национальной Безопасности США](https://ru.wikipedia.org/wiki/%D0%90%D0%B3%D0%B5%D0%BD%D1%82%D1%81%D1%82%D0%B2%D0%BE_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8), написал книгу под названием «Shift Register Sequences» («Последовательности сдвиговых регистров»), в которой изложил свои основные достижения в этой области, а также достижения Селмера.

Большую популярность потоковым шифрам принесла работа [Клода Шеннона](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD,_%D0%9A%D0%BB%D0%BE%D0%B4_%D0%AD%D0%BB%D0%B2%D1%83%D0%B4), опубликованная в 1949 году, в которой Шеннон доказал абсолютную стойкость [шифра Вернама](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B5%D1%80%D0%BD%D0%B0%D0%BC%D0%B0) (также известного, как одноразовый блокнот). В шифре Вернама [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) имеет длину, равную длине самого передаваемого сообщения. Ключ используется в качестве [гаммы](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), и если каждый [бит](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) ключа выбирается случайно, то вскрыть шифр невозможно (так как все возможные открытые тексты будут равновероятны). К настоящему времени создано большое количество алгоритмов потокового [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), таких как: [A3](https://ru.wikipedia.org/wiki/A3_(%D1%88%D0%B8%D1%84%D1%80)), [A5](https://ru.wikipedia.org/wiki/A5_(%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [A8](https://ru.wikipedia.org/wiki/A8_(%D1%88%D0%B8%D1%84%D1%80)), [MUGI](https://ru.wikipedia.org/wiki/MUGI), [PIKE](https://ru.wikipedia.org/wiki/PIKE), [RC4](https://ru.wikipedia.org/wiki/RC4), [SEAL](https://ru.wikipedia.org/wiki/SEAL_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC)).

Допустим, например, что в режиме гаммирования для потоковых шифров при передаче по [каналу связи](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB_%D1%81%D0%B2%D1%8F%D0%B7%D0%B8) произошло искажение одного знака шифротекста. Очевидно, что в этом случае все знаки, принятые без искажения, будут расшифрованы правильно. Произойдёт потеря лишь одного знака текста. А теперь представим, что один из знаков шифротекста при передаче по каналу связи был потерян. Это приведёт к неправильному расшифрованию всего текста, следующего за потерянным знаком.  
Практически во всех каналах передачи данных для потоковых систем шифрования присутствуют [помехи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BC%D0%B5%D1%85%D0%B0). Поэтому для предотвращения потери информации решают проблему синхронизации шифрования и расшифрования текста. По способу решения этой проблемы шифросистемы подразделяются на синхронные и системы с самосинхронизацией.

**Синхронные потоковые шифры**

**Определение:**

*Синхронные потоковые шифры (СПШ)* — шифры, в которых поток ключей генерируется независимо от [открытого текста](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82) и шифротекста.

При шифровании генератор потока ключей выдаёт биты потока ключей, которые идентичны битам потока ключей при дешифровании. Потеря знака шифротекста приведёт к нарушению синхронизации между этими двумя генераторами и невозможности расшифрования оставшейся части сообщения. Очевидно, что в этой ситуации отправитель и получатель должны повторно синхронизоваться для продолжения работы.

Обычно [синхронизация](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) производится вставкой в передаваемое сообщение специальных маркеров. В результате этого пропущенный при передаче знак приводит к неверному расшифрованию лишь до тех пор, пока не будет принят один из маркеров.

Заметим, что выполняться синхронизация должна так, чтобы ни одна часть потока ключей не была повторена. Поэтому переводить генератор в более раннее состояние не имеет смысла.

**Плюсы СПШ:**

* отсутствие эффекта распространения ошибок (только искажённый бит будет расшифрован неверно);
* предохраняют от любых вставок и удалений шифротекста, так как они приведут к потере синхронизации и будут обнаружены.

**Минусы СПШ:**

* уязвимы к изменению отдельных бит шифрованного текста. Если злоумышленнику известен [открытый текст](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82), он может изменить эти биты так, чтобы они расшифровывались, как ему надо.

**Самосинхронизирующиеся потоковые шифры**

*Основная статья:*[***Самосинхронизирующийся потоковый шифр***](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B8%D0%B9%D1%81%D1%8F_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80)

Основная идея построения была запатентована в 1946 г. в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%A8%D1%82%D0%B0%D1%82%D1%8B_%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B8).

**Определение:**

*Самосинхронизирующиеся потоковые шифры (асинхронные потоковые шифры (АПШ))* — шифры, в которых ключевой поток создаётся функцией ключа и фиксированного числа знаков шифртекста.

Итак, внутреннее состояние генератора потока ключей является функцией предыдущих N битов шифротекста. Поэтому расшифрующий генератор потока ключей, приняв N битов, автоматически синхронизируется с шифрующим генератором.

Реализация этого режима происходит следующим образом: каждое сообщение начинается случайным заголовком длиной N битов; заголовок шифруется, передаётся и расшифровывается; расшифровка является неправильной, зато после этих N бит оба генератора будут синхронизированы.

**Плюсы АПШ:**

* Размешивание статистики открытого текста. Так как каждый знак открытого текста влияет на следующий шифротекст, статистические свойства открытого текста распространяются на весь шифротекст. Следовательно, АПШ может быть более устойчивым к атакам на основе избыточности открытого текста, чем СПШ.

**Минусы АПШ:**

* распространение ошибки (каждому неправильному биту шифротекста соответствуют N ошибок в открытом тексте);
* [чувствительны](https://ru.wikipedia.org/wiki/%D0%A7%D1%83%D0%B2%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%B0)) к вскрытию повторной передачей.

**RC4**

**RC4** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Rivest cipher 4* или *Ron’s code*), также известен как **ARC4** или **ARCFOUR** (*alleged RC4*) — [потоковый шифр](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80), широко применяющийся в различных системах защиты информации в компьютерных сетях (например, в протоколах [SSL](https://ru.wikipedia.org/wiki/SSL) и [TLS](https://ru.wikipedia.org/wiki/TLS), алгоритмах обеспечения безопасности беспроводных сетей [WEP](https://ru.wikipedia.org/wiki/WEP) и [WPA](https://ru.wikipedia.org/wiki/WPA)).

Шифр разработан компанией [RSA Security](https://ru.wikipedia.org/w/index.php?title=RSA_Security&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/RSA_Security), и для его использования требуется [лицензия](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F).

Алгоритм RC4, как и любой [потоковый шифр](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80), строится на основе [генератора псевдослучайных](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB) [битов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82). На вход генератора записывается ключ, а на выходе читаются псевдослучайные биты. Длина ключа может составлять от 40 до 2048 бит[[1]](https://ru.wikipedia.org/wiki/RC4#cite_note-KLEIN_ATTACKS-1). Генерируемые биты имеют [равномерное](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D0%B2%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) [распределение](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9).

Основные преимущества шифра:

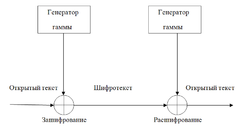
* высокая скорость работы;
* переменный размер ключа.

RC4 довольно уязвим, если:

* используются не случайные или связанные ключи;
* один ключевой поток используется дважды.

Эти факторы, а также способ использования могут сделать [криптосистему](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) небезопасной (например, [WEP](https://ru.wikipedia.org/wiki/WEP)).

Ядро алгоритма [поточных шифров](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) состоит из функции — [генератора псевдослучайных](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB) [битов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) ([гаммы](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), который выдаёт поток битов ключа (ключевой поток, гамму, последовательность псевдослучайных битов).

[](https://commons.wikimedia.org/wiki/File:Gamma12.PNG?uselang=ru)

Режим [гаммирования](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) для [поточных шифров](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80)

Алгоритм шифрования.

1. Функция генерирует последовательность битов
2. Затем последовательность битов посредством операции «[суммирование по модулю два](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2)» (xor) объединяется с открытым текстом . В результате получается [шифрограмма](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82)

{\displaystyle c\_{i}=m\_{i}\oplus k\_{i}}.

Алгоритм расшифровки.

1. Повторно создаётся (регенерируется) поток битов ключа (ключевой поток).
2. Поток битов ключа складывается с [шифрограммой](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82)операцией «[xor](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2)». В силу свойств операции «[xor](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2)» на выходе получается исходный (незашифрованный) текст.

{\displaystyle m\_{i}=c\_{i}\oplus k\_{i}=(m\_{i}\oplus k\_{i})\oplus k\_{i}}

RC4 — фактически класс алгоритмов, определяемых размером блока (в дальнейшем [S-блока](https://ru.wikipedia.org/wiki/S-%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8)). Параметр *n* является размером слова для алгоритма и определяет длину *S-блока*. Обычно, *n* = 8, но в целях анализа можно уменьшить его. Однако для повышения безопасности необходимо увеличить эту величину. В алгоритме нет противоречий на увеличение размера *S-блока* . При увеличении *n*, допустим, до 16 бит, элементов в *S-блоке* становится 65 536 и соответственно время начальной итерации будет увеличено. Однако, скорость шифрования возрастёт[[5]](https://ru.wikipedia.org/wiki/RC4#cite_note-5).

Внутреннее состояние RC4 представляется в виде массива размером 2*n* и двух счётчиков. Массив известен как *S-блок*, и далее будет обозначаться как S. Он всегда содержит перестановку 2*n* возможных значений слова. Два счётчика обозначены через i и j.

Инициализация RC4 состоит из двух частей:

1. инициализация *S-блока*;
2. генерация псевдослучайного слова K.

### Инициализация *S-блока*

Алгоритм также известен как «key-scheduling algorithm» или «KSA». Этот алгоритм использует ключ, подаваемый на вход пользователем, сохранённый в Key, и имеющий длину L байт. Инициализация начинается с заполнения массива S, далее этот массив перемешивается путём перестановок, определяемых ключом. Так как только одно действие выполняется над S, то должно выполняться утверждение, что S всегда содержит один набор значений, который был дан при первоначальной инициализации (*S[i] := i*).

**for** i **from** 0 **to** 255

S[i] := i

**endfor**

j := 0

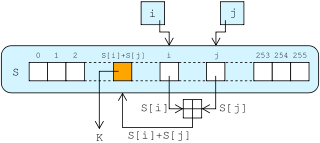
**for** i **from** 0 **to** 255

j := ( j + S[i] + Key[ i mod L ] ) mod 256 // *n* = 8 ; 2*8* = 256

поменять местами S[i] и S[j]

**endfor**

### Генерация псевдослучайного слова *K*

[](https://commons.wikimedia.org/wiki/File:RC4.svg?uselang=ru)

Генератор ключевого потока RC4

Эта часть алгоритма называется генератором псевдослучайной последовательности ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ***p****seudo-****r****andom****g****eneration****a****lgorithm*, *PRGA*). Генератор ключевого потока RC4 переставляет значения, хранящиеся в S. В одном цикле RC4 определяется одно *n*-битное слово K из ключевого потока. В дальнейшем ключевое слово будет [сложено по модулю два](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2) с исходным текстом, которое пользователь хочет зашифровать, и получен зашифрованный текст.

i := 0

j := 0

**while** Цикл генерации:

i := ( i + 1 ) mod 256

j := ( j + S[i] ) mod 256

поменять местами S[i] и S[j]

t := ( S[i] + S[j] ) mod 256

K := S[t]

сгенерировано псевдослучайное слово K (для *n* = 8 будет сгенерирован один байт)

**endwhile**

# 

В отличие от современных шифров (таких, как [eSTREAM](https://ru.wikipedia.org/wiki/ESTREAM)), RC4 не использует [nonce](https://ru.wikipedia.org/wiki/Nonce) (оказию) наряду с ключом. Это значит, что если один ключ должен использоваться в течение долгого времени для шифрования нескольких потоков, сама криптосистема, использующая RC4, должна комбинировать оказию и долгосрочный ключ для получения потокового ключа для RC4. Один из возможных выходов — генерировать **новый** ключ для RC4 с помощью [хэш](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)-функции от долгосрочного ключа и [nonce](https://ru.wikipedia.org/wiki/Nonce). Однако многие приложения, использующие RC4, просто конкатенируют ключ и [nonce](https://ru.wikipedia.org/wiki/Nonce). Из-за этого и слабого расписания ключей, используемого в RC4, приложение может стать уязвимым. Поэтому он был признан устаревшим многими софтверными компаниями, такими как [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Например, в [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft)отсутствует реализация RC4.

Здесь будут рассмотрены некоторые атаки на шифр и методы защиты от них.

### Исследования Руза и восстановление ключа из перестановки

В 1995 году Андрю Руз ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Andrew Roos*) экспериментально пронаблюдал, что первый байт ключевого потока коррелирован с первыми тремя байтами ключа, а первые несколько байт перестановки после алгоритма расписания ключей ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *KSA*) коррелированы с некоторой линейной комбинацией байт ключа[[9]](https://ru.wikipedia.org/wiki/RC4#cite_note-9). Эти смещения не были доказаны до 2007 года, когда Пол, Рафи и Мэйтрэ доказали коррелированность ключа и ключевого потока. Также Пол и Мэйтрэ доказали коррелированность перестановки и ключа. Последняя работа также использует коррелированность ключа и перестановки для того, чтобы создать первый алгоритм полного восстановления ключа из последней перестановки после KSA, не делая предположений о ключе и векторе инициализации ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *IV*, ***i****nitial****v****ector*). Этот алгоритм имеет постоянную вероятность успеха в зависимости от времени, которая соответствует квадратному корню из сложности полного перебора. Позднее было сделано много работ о восстановлении ключа из внутреннего состояния RC4.

### Атака Флурера, Мантина и Шамира (ФМШ)

В 2001 году Флурер, Мантин и Шамир опубликовали работу об уязвимости ключевого расписания RC4. Они показали, что первые байты ключевого потока среди всех возможных ключей неслучайны. Из этих байтов можно с высокой вероятностью получить информацию об используемом шифром ключе. И если долговременный ключ и [nonce](https://ru.wikipedia.org/wiki/Nonce) просто склеиваются для создания ключа шифра RC4, то этот долговременный ключ может быть получен с помощью анализа достаточно большого количества сообщений, зашифрованных с использованием данного ключа. Эта уязвимость и некоторые связанные с ней эффекты были использованы при взломе шифрования [WEP](https://ru.wikipedia.org/wiki/WEP) в беспроводных сетях стандарта [IEEE 802.11](https://ru.wikipedia.org/wiki/IEEE_802.11). Это показало необходимость скорейшей замены [WEP](https://ru.wikipedia.org/wiki/WEP), что повлекло за собой разработку нового стандарта безопасности беспроводных сетей [WPA](https://ru.wikipedia.org/wiki/WPA).

Криптосистему можно сделать невосприимчивой к этой атаке, если отбрасывать начало ключевого потока. Таким образом, модифицированный алгоритм называется «RC4-drop[n]», где n — количество байтов из начала ключевого потока, которые следует отбросить. Рекомендовано использовать n = 768, консервативная оценка составляет n = 3072.

Атака базируется на слабости [инициализационного вектора](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D0%BD%D0%B8%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8&action=edit&redlink=1). Зная первое псевдослучайное слово K и m байтов входного ключа Key, используя слабость в алгоритме генерации псевдо-случайного слова K , можно получить m + 1 байт входного ключа. Повторяя шаги добывается полный ключ. При атаке на [WEP](https://ru.wikipedia.org/wiki/WEP), для n = 8 IV имеет вид (B; 255; N), где B — от 3 до 8, а N любое число . Для определения около 60 вариантов *N* потребуется перехватить примерно 4 миллиона пакетов.

### Атака Ванхофа и Писсенса (2015)

Летом 2015 года Мэти Ванхоф (Mathy Vanhoef) и Франк Писсенс (Frank Piessens) из университета Левена в Бельгии продемонстрировали реальную атаку на протокол [TLS](https://ru.wikipedia.org/wiki/TLS), использующий RC4 для шифрования передаваемых данных. Идея взлома базируется на принципе [MITM](https://ru.wikipedia.org/wiki/%D0%90%D1%82%D0%B0%D0%BA%D0%B0_%D0%BF%D0%BE%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D0%BA%D0%B0). Встроившись в канал передачи данных, атакующая сторона генерирует серверу большое количество запросов, вынуждая его в ответ возвращать [куки](https://ru.wikipedia.org/wiki/Magic_cookie), зашифрованные одним и тем же ключом. Имея в распоряжении около 9x227 ~ 230 пар {открытый текст, шифротекст}, атакующая сторона получила возможность на основе статистических методов [Флюрер-Макгрю](https://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BB%D1%8E%D1%80%D0%B5%D1%80-%D0%9C%D0%B0%D0%BA%D0%B3%D1%80%D1%8E&action=edit&redlink=1) и [ABSAB](https://ru.wikipedia.org/w/index.php?title=ABSAB&action=edit&redlink=1) с вероятностью 94 % восстановить ключ и, следовательно, зашифрованные куки. Практические временные затраты составили около 52 часов, верхняя же оценка потребного времени на момент демонстрации составила около 72 часов.

# **Глава 1. Разработка технического задания**

1. Название разработки

«Программа для описания 3 видов шифров: шифр Цезаря, шифр Виженера, Потоковый шифр RC4»

# **2. Основание для разработки**

Система разрабатывается в качестве курсовой работы по курсу «Высокоуровневые методы информатики и программирования» в соответствии с учебным планом.

# **3. Назначение**

Основным назначением программы является упрощение шифрования текста такими алгоритмами, как шифр Цезаря, шифр Виженера, Потоковый шифр RC4.

# **4.Требования к программной системе**

4.1. Требования к функциональным характеристикам

Система должна обеспечивать выполнение следующих функций:

* 1. Принять данные;
  2. Преобразовать для их дальнейшего шифрования.
  3. Вывести в текстовом документе информацию о входных и выходных данных

4.2 Требования к надежности

4.3 Требования к составу и параметрам технических средств

4.3.1 Программная система должна работать на персональных компьютерах типа IBM PC.

4.3.2 Минимальная аппаратная конфигурация:

* ПК на базе компьютера Intel Pentium, оснащенный ОЗУ объемом не менее 128Мб
* Установленная на ПК операционная система должна иметь visual studio 2017

4.4 Требования к информационной и программной совместимости

Программа запустится на любой платформе имеющий компилятор с++

# **5. Требования к программной документации**

5.1 Разрабатываемые программные модули должны быть самодокументированны, то есть тексты программ должны содержать все необходимые комментарии.

5.2 Разрабатываемая система должна включать справочную информацию.

5.3 В состав сопровождающей документации должны входить:

5.3.1 Пояснительная записка на 25-30 листах, содержащая описание разработки.

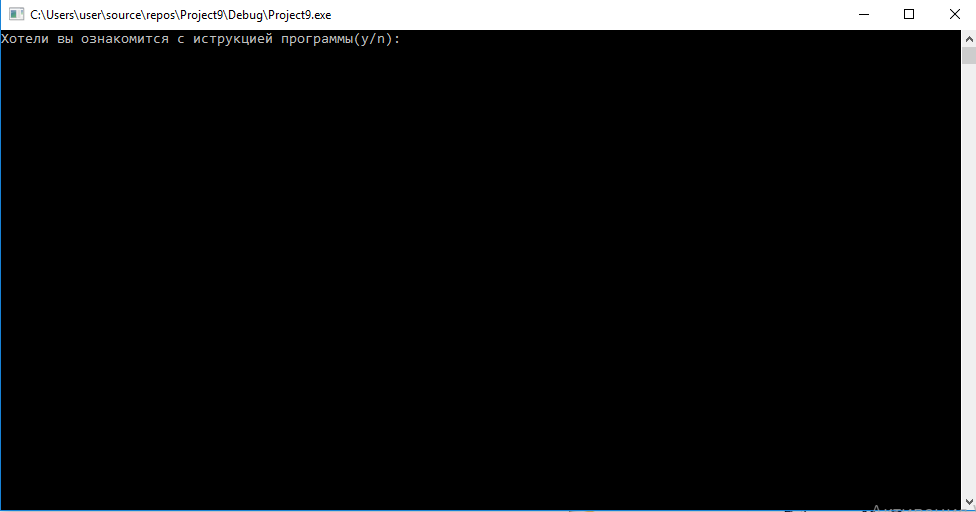
5.3.2 Руководство пользователя.

5.3.3 Тексты программ.

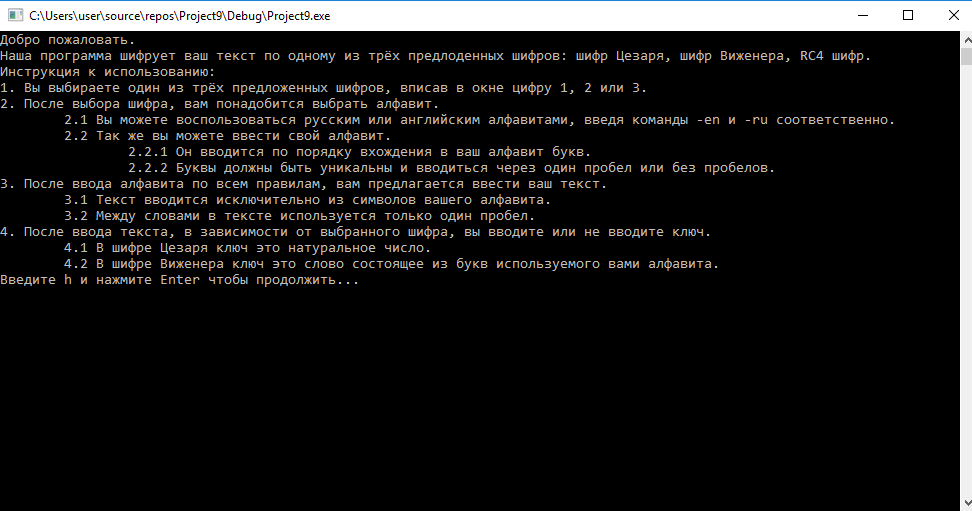
# 

# **Глава 2. Описание теории и выявление необходимых элементов программы. Главные задачи и их решения**

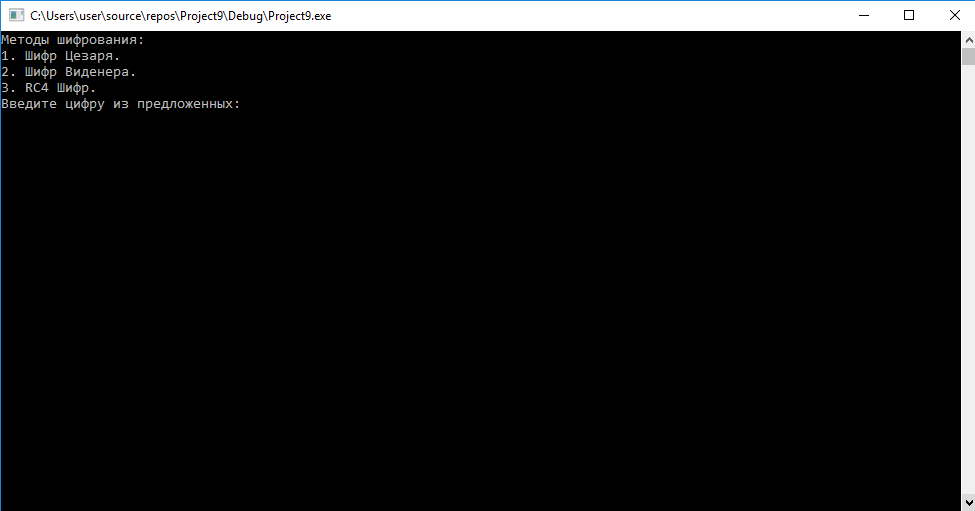
В начале программы пользователю предлагают прочитать инструкцию программы.

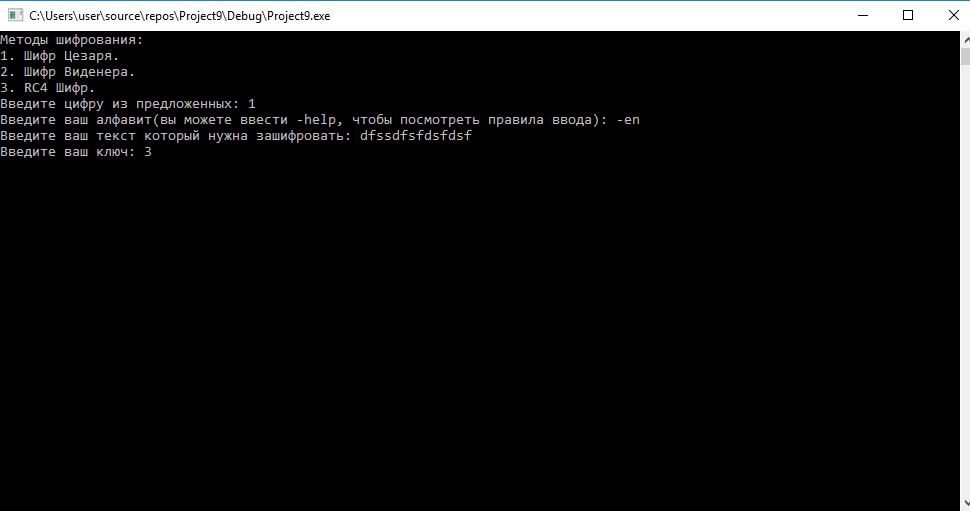


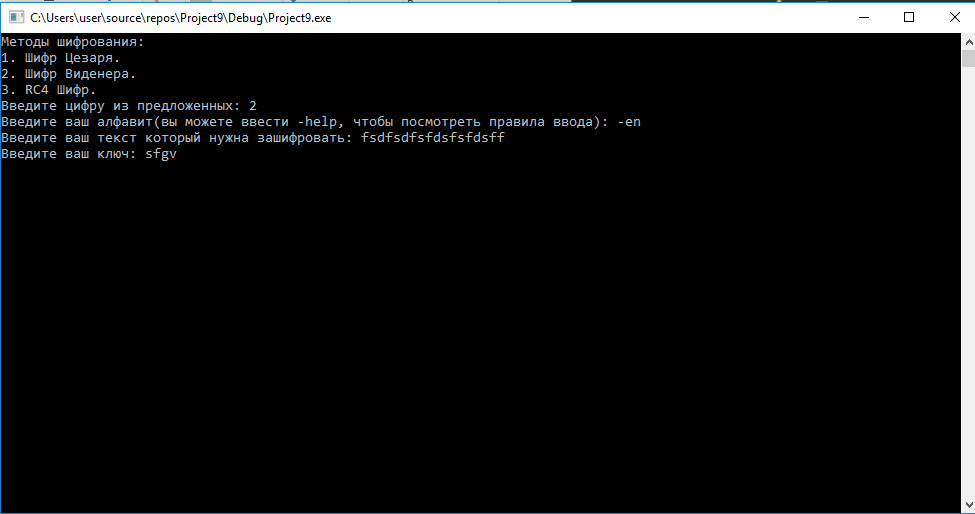
Если пользователь захотел ознакомиться с инструкцией, то ему выведет окно с инструкцией.

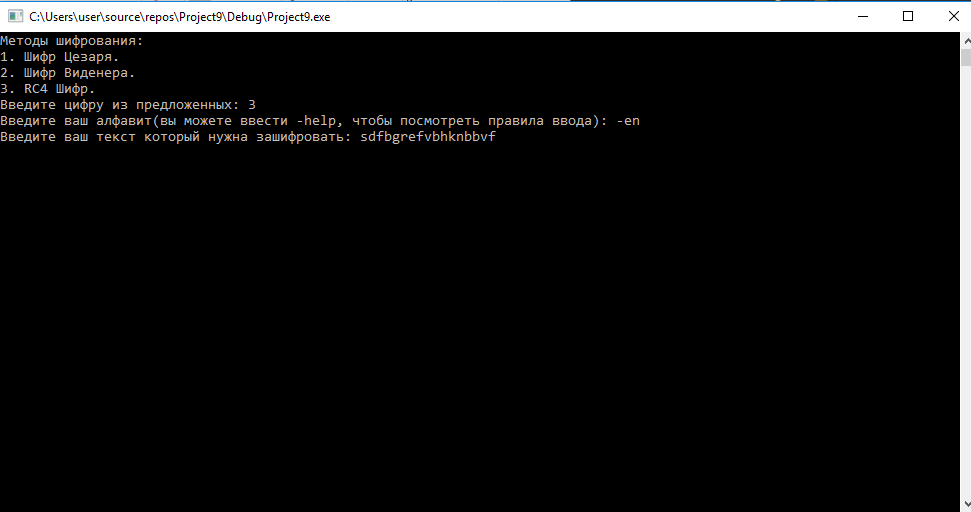


После прочтения инструкции пользователь может продолжить работу с программой. По мере выбора пользователем алгоритма, который он хочет использовать, пользователь вводит свои данные.









После ввода нужных данных для каждого из 3 шифров консоль закрывается и пользователь может посмотреть результат шифрования в текстовом файле “Выходные данные”. Так же для удобства входные данные тоже записываются в текстовом файле “Выходные данные”. Это сделано для того, чтобы пользователь мог проверить работу или вспомнить входные данные.

**Пользователь**

**Чтение инструкции**

**Ввод соответствующих данных для выбранного алгоритма**

**Выбор Алгоритма**

**Обработка и вычисление по данным, введенным пользователем**

**Вывод результата**

-

**Взаимодействие пользователь-программа**

Теперь рассмотрим взаимодействие пользователя с программой. Оно происходит посредством консоли. Всю информацию пользователь вносит в неё.

После ввода нужных данных пользователь должен обратиться к файлам, в которых он может увидеть входные данные и выходные.

# **Глава 3. Инструкция пользователя.**

Несмотря на интуитивную понятность программы, имеет смысл пояснить, как ей управлять. И так:

1. Запустите программу.
2. В открывшем окне выберите нужный пункт введя его.
3. Введите соответствующие данные.
4. Откройте текстовый файл “Выходные данные”.

**Заключение**

Результат выполнения курсовой работы стало создания программы на языке C++. В программе реализованы работы 3 видов шифров: шифр Цезаря, шифр Виженера, потоковый шифр RC4.

**Библиография**

Использованная литература:

1) Иванов М.А., Чугунков И.В. Теория, применение и оценка качества   
генераторов псевдослучайных последовательностей. - М.: КУДИЦ-ОБРАЗ,   
2003. (СКБ - специалисту компьютерной безопасности)  
 Интернет-ресурсы:

1) https://ru.wikipedia.org/wiki/Шифр\_Цезаря

2­) https://ru.wikipedia.org/wiki/Шифр\_Виженера

3) <https://ru.wikipedia.org/wiki/RC4>

**Листинг Программы**

Программа состоит из 3 файлов: main.cpp, interfase.h, Crypt.h.

**Main.cpp:**

#include<iostream>

#include<string>

#include<cmath>

#include<vector>

#include<fstream>

#include"Crypt.h"

#include<windows.h>

#include"Interfase.h"

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

start();

return 0;

}

**Interfase.h:**

void start();

void help();

void start\_code();

void en();

void ru();

void start() {

char c = 'n';

int l = 1;

while (l) {

cout << "Хотели вы ознакомится с иструкцией программы(y/n): ";

cin >> c;

if (c == 'n' || c == 'y') {

l = 0;

system("cls");

}

}

if (c == 'y') {

system("cls");

help();

}

else {

system("cls");

start\_code();

}

}

void en(string& s) {

s = "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";

}

void ru(string& s) {

s = "а б в г д е ё ж з и й к л м н о п р с т у ф х ц ш щ ь ы ъ э ю я";

}

void help() {

char k;

cout << "Добро пожаловать.\nНаша программа шифрует ваш текст по одному из трёх предлоденных шифров: шифр Цезаря, шифр Виженера, RC4 шифр.";

cout << "\nИнструкция к использованию:\n1. Вы выбираете один из трёх предложенных шифров, вписав в окне цифру 1, 2 или 3.";

cout << "\n2. После выбора шифра, вам понадобится выбрать алфавит.\n 2.1 Вы можете воспользоваться русским или английским алфавитами, введя команды -en и -ru соответственно.";

cout << "\n 2.2 Так же вы можете ввести свой алфавит.\n 2.2.1 Он вводится по порядку вхождения в ваш алфавит букв.";

cout << "\n 2.2.2 Буквы должны быть уникальны и вводиться через один пробел или без пробелов.";

cout << "\n3. После ввода алфавита по всем правилам, вам предлагается ввести ваш текст.\n 3.1 Текст вводится исключительно из символов вашего алфавита.";

cout << "\n 3.2 Между словами в тексте используется только один пробел.";

cout << "\n4. После ввода текста, в зависимости от выбранного шифра, вы вводите или не вводите ключ.\n 4.1 В шифре Цезаря ключ это натуральное число.";

cout << "\n 4.2 В шифре Виженера ключ это слово состоящее из букв используемого вами алфавита.";

cout << "\nВведите h и нажмите Enter чтобы продолжить...";

cin >> k;

system("cls");

start\_code();

}

void start\_code() {

int a = 0;

int l = 1;

while (l) {

cout << "Методы шифрования: \n1. Шифр Цезаря. \n2. Шифр Виденера. \n3. RC4 Шифр.";

cout << "\nВведите цифру из предложенных: ";

cin >> a;

if (a == 1 || a == 2 || a == 3) {

l = 0;

}

if (l == 1) {

system("cls");

}

}

if (a == 1) {

string s;

string s1;

int key;

cout << "Введите ваш алфавит(вы можете ввести -help, чтобы посмотреть правила ввода): ";

cin.ignore();

getline(cin, s1);

if (s1 == "-help") {

system("cls");

help();

}

else {

if (s1 == "-en") {

en(s1);

}

if (s1 == "-ru") {

ru(s1);

}

cout << "Введите ваш текст который нужна зашифровать: ";

getline(cin, s);

cout << "Введите ваш ключ: ";

cin >> key;

ofstream fout("Выходные данные.txt");

fout << "Ваш Ключ: " << key << "\nВаш алфавит: " << s1 << "\nВаш текст: " << s;

cezar(s, s1, key);

fout << "\nВаш зашифрованный текст: " << s;

fout.close();

}

}

if (a == 2) {

string s;

string s1;

string key;

cout << "Введите ваш алфавит(вы можете ввести -help, чтобы посмотреть правила ввода): ";

cin.ignore();

getline(cin, s1);

if (s1 == "-help") {

system("cls");

help();

}

else {

if (s1 == "-en") {

en(s1);

}

if (s1 == "-ru") {

ru(s1);

}

cout << "Введите ваш текст который нужна зашифровать: ";

getline(cin, s);

cout << "Введите ваш ключ: ";

getline(cin, key);

ofstream fout("Выходные данные.txt");

fout << "Ваш Ключ: " << key << "\nВаш алфавит: " << s1 << "\nВаш текст: " << s;

vegener(s, s1, key);

fout << "\nВаш зашифрованный текст: " << s;

fout.close();

}

}

if (a == 3) {

string s;

string s1;

cout << "Введите ваш алфавит(вы можете ввести -help, чтобы посмотреть правила ввода): ";

cin.ignore();

getline(cin, s1);

if (s1 == "-help") {

system("cls");

help();

}

else {

if (s1 == "-en") {

en(s1);

}

if (s1 == "-ru") {

ru(s1);

}

cout << "Введите ваш текст который нужна зашифровать: ";

getline(cin, s);

int\* key = new int[s.size()];

ofstream fout("Выходные данные.txt");

fout << "Ваш Ключ: " << key << "\nВаш алфавит: " << s1 << "\nВаш текст: " << s;

rc4\_code(s, s1, key);

fout << "\nВаш зашифрованный текст: " << s;

fout.close();

}

}

}

**Crypt.h:**

void cezar(string& s, string& s1, int& key)

{

int n = 0;

string s2;

for (int i = 0; i < s1.size(); i++) {

if (s1[i] != ' ') {

n++;

s2 += s1[i];

}

}

int\* k = new int[n];

int\* k1 = new int[n];

for (int i = 0; i < n; i++) {

k[i] = i;

}

for (int i = 0; i < n; i++) {

k1[i] = k[i] + key;

while (k1[i] < 0 || k1[i] > n - 1) {

if (k1[i] < 0) { k1[i] = k1[i] + n; }

else {

if (k1[i] > n - 1) {

k1[i] = k1[i] - n;

}

}

}

}

string s3;

string s4;

s3 = s;

int m = s.size();

for (int v = 0; v < s2.size(); v++) {

while (s.find(s2[v]) != -1) {

int l = s.find(s2[v]);

s3[l] = s2[k1[v]];

for (int j = 0; j < s.size(); j++) {

if (j != l) {

s4 += s[j];

}

else { s4 = s4 + ' '; }

}

s = s4;

s4.clear();

}

}

s = s3;

}

int binary(int val) {

int val1 = 0;

int val2 = 0;

int i = 0;

while (val != 0) {

val1 = val % 2;

val = val / 2;

val2 += pow(10, i)\*val1;

i++;

}

return val2;

}

int binary\_size(int val) {

int val1 = 0;

int n = 0;

val1 = binary(val);

if (val1 == 0) {

n = 1;

}

else {

while (val1 != 0) {

val1 = val1 / 10;

n++;

}

}

return n;

}

void vegener(string& s, string& s1, string& key) {

string s2;

string s3;

int n = 0;

for (int i = 0; i < s1.size(); i++) {

if (s1[i] != ' ') {

n++;

s2 += s1[i];

}

}

int\*\* k = new int\*[key.size()];

for (int i = 0; i < key.size(); i++) {

k[i] = new int[s2.size()];

for (int j = 0; j < s2.size(); j++) {

k[i][j] = j;

}

}

int\* key1 = new int[key.size()];

for (int i = 0; i < key.size(); i++) {

for (int j = 0; j < s2.size(); j++) {

if (key[i] == s2[j]) {

key1[i] = j;

}

}

}

cout << endl;

for (int i = 0; i < key.size(); i++) {

for (int j = 0; j < s2.size(); j++) {

k[i][j] = k[i][j] + key1[i];

while (k[i][j] >= s2.size()) {

k[i][j] = k[i][j] - s2.size();

}

}

}

string s4;

s3 = s;

s4 = s;

int m = 0;

for (int i = 0; i < s2.size(); i++) {

while (s4.find(s2[i]) != -1) {

int l = s4.find(s2[i]);

s3[l] = s2[k[l % (key.size())][i]];

s4[l] = ' ';

}

}

s = s3;

}

void rc4\_code(string& s3, string& alf, int\*& key\_1) {

vector<char> s;

vector<string> s1;

vector<int> k;

int m = 0;

int n = 0;

for (int i = 0; i < alf.size(); i++) {

if (alf[i] != ' ') {

m++;

}

}

s1.reserve(m);

s.reserve(m);

k.reserve(m);

for (int i = 0; i < alf.size(); i++) {

if (alf[i] != ' ') {

s.push\_back(alf[i]);

}

}

for (int i = 0; i < m; i++) {

k.push\_back(binary(i));

}

string s2;

for (int i = 0; i < m; i++) {

for (int j = binary\_size(i); j < binary\_size(m - 1); j++) {

s2 += "0";

}

s1.push\_back(s2 + to\_string(k[i]));

s2.clear();

}

int l = 256;

int\* s6 = new int[l];

for (int i = 0; i < l; i++) {

s6[i] = i;

}

int j = 0;

int temp;

for (int i = 0; i < l; i++) {

j = (j + s6[i]) % 256;

temp = s6[i];

s6[i] = s6[j];

s6[j] = temp;

}

j = 0;

for (int i = 0; i < m; i++) {

for (int j = 0; j < s.size(); j++) {

while (s3.find(s[i]) != -1) {

string s4;

int k = s3.find(s[i]);

for (int l = 0; l < k; l++) {

s4 += s3[l];

}

s4 = s4 + s1[i];

for (int l = k + 1; l < s3.size(); l++) {

s4 += s3[l];

}

s3 = s4;

s4.clear();

}

}

}

int u;

int\* k2 = new int[s3.size()];

string s7;

key\_1 = new int[s3.size()];

for (int i = 0; i < s3.size(); i++) {

if (s3[i] != ' ') {

j = (j + s6[i]) % 256;

u = s6[(s6[j] + s6[i]) % 256] % 2;

k2[i] = u;

if ((s3[i] == '0') && (u == 0)) {

s7 += '0';

}

else {

if ((s3[i] == '0') && (u == 1)) {

s7 += '1';

}

else {

if ((s3[i] == '1') && (u == 0)) {

s7 += '1';

}

else {

if ((s3[i] = '1') && (u = 1)) {

s7 += '0';

}

}

}

}

}

else {

s7 += ' ';

}

}

int o = 0;

string s8;

string s9;

for (int i = 0; i < s7.size(); i++) {

if (s7[i] != ' ') {

for (int j = 0; j < s1[0].size(); j++) {

s8 += s7[i];

i++;

}

o = 0;

for (int k = 0; k < m; k++) {

if (s8 == s1[k]) {

s9 += s[k];

o = 1;

break;

}

}

if (o == 0) {

s9 += s8;

}

i--;

s8.clear();

}

else {

s9 += ' ';

}

}

cout << s3 << endl;

for (int i = 0; i < s3.size(); i++) {

key\_1[i] = k2[i];

}

s3 = s9;

}