Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Контрольная работа №2

по дисциплине «Методы защиты информации»

Студент 4 курса

Группы № 493551

Рыбак Артем Владимирович

Минск, 2018

**Содержание**

**[Введение 3](#_Toc467176293)**

**[История появления и развития 4](#_Toc467176294)**

**[Описание алгоритма 7](#_Toc467176295)**

**[Применение на практике 11](#_Toc467176296)**

**[Результат работы программы 14](#_Toc467176297)**

**[Список используемых источников 15](#_Toc467176298)**

**[Приложение 16](#_Toc467176299)**

**Введение**

Концепция криптографии с открытым ключом была предложена Уитфилдом Диффи (Whitfield Diffie) и Мартином Хеллманом (Martin Hellman), и, независимо от них, Ральфом Мерклом (Ralph Merkle). Основная идея заключается в том, чтобы использовать ключи парами, состоящими из ключа зашифрования и ключа расшифрования, которые невозможно вычислить один из другого. В 1976 г. вышла основополагающая работа [1]. С этого времени было создано много алгоритмов, использующих концепцию открытых ключей. Алгоритм является общедоступным, нет необходимости в секретных каналах связи. Общая схема выглядит следующим образом:

1. Каждый пользователь генерирует пару ключей: один для шифрования, другой для дешифрования.

2. Каждый пользователь публикует свой ключ шифрования, размещает его в открытом для всех доступе. Второй ключ, соответствующий открытому, сохраняется в секрете.

3. Если пользователь ***A*** собирается послать сообщение пользователю ***B***, он шифрует сообщение открытым ключом пользователя ***B***.

4. Когда пользователь ***B*** получает сообщение, он дешифрует его с помощью своего личного (секретного) ключа. Другой получатель не сможет дешифровать сообщение, поскольку личный ключ ***B*** знает только ***B***.

**История появления и развития**

Первые криптографические системы с открытым ключом или ассиметричные криптосистемы появились в конце 1970-х годов. От классических симметричных алгоритмов они отличаются тем, что для шифрования данных используется один ключ, обычно его называют открытый или публичный ключ, а для дешифрования ‑ другой, секретный или закрытый, ключ.

Диффи и Хеллман, которые впервые предложили и описали криптосистему с открытым ключом, выявляют следующие требования к криптосистемам:

1. Вычислительно легко создавать пару открытый ключ (Ko), закрытый ключ (Kc).

2. Вычислительно легко, имея открытый ключ и незашифрованное сообщение М, создать соответствующее зашифрованное сообщение: С= Еko[М].

3. Вычислительно легко дешифровать сообщение, используя закрытый ключ: М= DKc[C] = DKc [EKo[M]].

4. Вычислительно невозможно, зная открытый ключ Ko, определить закрытый ключ Kc.

5. Вычислительно невозможно, зная открытый ключ Ko и зашифрованное сообщение С, восстановить исходное сообщение М.

Можно добавить шестое требование, хотя оно не выполняется для всех алгоритмов с открытым ключом:

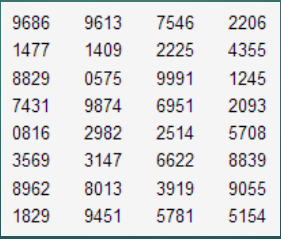
6. Шифрующие и дешифрующие функции могут применяться в любом порядке, т.е. М= ЕKo[DKc[M]] = DKc[EKo[M]].

Таким образом, данные, зашифрованные открытым ключом, можно расшифровать только секретным ключом. Следовательно, открытый ключ может распространяться через обычные коммуникационные сети и другие открытые каналы, что устраняет главный недостаток стандартных криптографических алгоритмов: необходимость использовать специальные каналы связи для распределения ключей.

Опубликованная в ноябре 1976 года статья Уитфилда Диффи и Мартина Хеллмана «Новые направления в криптографии» перевернула представление о криптографических системах, заложив основы криптографии с открытым ключом. Разработанный впоследствии алгоритм Диффи Хеллмана позволял двум сторонам получить общий секретный ключ, используя незащищенный канал связи. Однако этот алгоритм не решал проблему аутентификации. Без дополнительных средств пользователи не могли быть уверены, с кем именно они сгенерировали общий секретный ключ.

Изучив эту статью, трое учёных Рональд Ривест, Ади Шамир и Леонард Адлеман из Массачусетского технологического института приступили к поискам математической функции, которая бы позволяла реализовать сформулированную Уитфилдом Диффи и Мартином Хеллманом модель криптографической системы с открытым ключом. После работы над более чем 40 возможными вариантами, им удалось найти алгоритм, основанный на различии в том, насколько легко находить большие простые числа и насколько сложно раскладывать на множители произведение двух больших простых чисел, получивший впоследствии название RSA. Система была названа по первым буквам фамилий её создателей.

В августе 1977 года в колонке «Математические игры» Мартина Гарднера в журнале Scientific American, с разрешения Рональда Ривеста появилось первое описание криптосистемы RSA. Читателям также было предложено дешифровать английскую фразу, зашифрованную описанным алгоритмом:



В качестве открытых параметров системы были использованы числа n = 1143816...6879541 (129 десятичных знаков, 425 бит, также известно как RSA-129 и e = 9007. За расшифровку была обещана награда в 100 долларов США. По заявлению Ривеста, для факторизации числа потребовалось бы более 40 квадриллионов лет. Однако чуть более чем через 15 лет, 3 сентября 1993 года было объявлено о старте проекта распределённых вычислений с координацией через электронную почту по нахождению сомножителей числа RSA ‑ 129 и решению головоломки. На протяжении полугода более 600 добровольцев из 20 стран жертвовали процессорное время 1600 машин (две из которых были факс-машинами). В результате были найдены простые множители и расшифровано исходное сообщение, которое представляет собой фразу «THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE» («Волшебные слова это брезгливый ягнятник»). Полученную награду победители пожертвовали в фонд свободного программного обеспечения.

После публикации Мартина Гарднера полное описание новой криптосистемы любой желающий мог получить, выслав по почте запрос Рональду Ривесту, с приложенным конвертом с обратным адресом и марками на 35 центов. Полное описание новой криптосистемы было опубликовано в журнале «Communications of the ACM» в феврале 1978 года. Заявка на патент была подана 14 декабря 1977 года, в качестве владельца был указан MIT. Патент 4405829 был выдан 20 сентября 1983 года, а 21 сентября 2000 года срок его действия истёк. Однако за пределами США у изобретателей патента на алгоритм не было, так как в большинстве стран его необходимо было получить до первой публикации.

В 1982 году Ривест, Шамир и Адлеман организовали компанию RSA Data Security. В 1989 году RSA, вместе с симметричным шифром DES, упоминается в RFC 1115, тем самым начиная использование алгоритма в зарождающейся сети Internet. В 1990 году министерство обороны США начинает использовать алгоритм. В ноябре 1993 года открыто публикуется версия 1.5 стандарта PKCS1, описывающего применение RSA для шифрования и создания электронной подписи. Последние версии стандарта также доступны в виде RFC (RFC 2313 1.5, 1993 год; RFC 2437 2.0, 1998 год; RFC 3447 2.1, 2002 год).

**Описание алгоритма**

Алгоритм RSA предполагает, что посланное закодированное сообщение может быть прочитано адресатом и только им. В этом алгоритме используется два ключа ‑ открытый и секретный. Данный алгоритм привлекателен также в случае, когда большое число субъектов (N) должно общаться по схеме все-со-всеми. В случае симметричной схемы шифрования каждый из субъектов каким-то образом должен доставить свои ключи всем остальным участникам обмена, при этом суммарное число используемых ключей будет достаточно велико при большом значении N. Применение асимметричного алгоритма требует лишь рассылки открытых ключей всеми участниками, суммарное число ключей равно N.

Сообщение представляется в виде числа **M**. Шифрование осуществляется с помощью общедоступной функции **f(M),** и только адресату известно, как выполнить операцию **f-1**. Адресат выбирает два больших простых (prime) числа **p** и **q**, которые делает секретными. Он объявляет **n=pq** и число **d**, c **(d,p-1)=(d,q-1)=1** (один из возможных способов выполнить это условие, выбрать **d** больше чем **p/2** и **q/2**). Шифрование производится по формуле:

**f(M)  Md mod n,**

где **M** и **f(M)** оба  **n-1**. Как было показано, может быть вычислено за разумное время, даже если **M**, **d** и **n** содержит весьма большое число знаков. Адресат вычисляет **M** на основе **Md**, используя свое знание **p** и **q**. В соответствие со следствием 6, если

**dc (p-1)1,** тогда **(Md)e p1.**

Исходный текст M получается адресатом из зашифрованного F(M) путем преобразования: M = (F(M))e (mod pq). Здесь как исходный текст, так и зашифрованный рассматриваются как длинные двоичные числа.

Аналогично **(Md)e  qM,** если **dc  (q-1)1. e** удовлетворяет этим двум условиям, если **cd  (p-1) (q-1)1.** Теорема 1 гласит, что мы можем позволить **e=x**, когда **x** является решением уравнения **dx + (p-1)(q-1)y = 1**.

Так как **(Md)e ‑ M** делимо на **p** и **q**, оно делимо и на **pq**, следовательно, мы можем определить **M**, зная **Md**, вычислив его значение в степени **e** и определив остаток от деления на **pq**. Для соблюдения секретности важно, чтобы, зная **n,** было нельзя вычислить **p** и **q**. Если **n** содержит 100 цифр, подбор шифра связан с перебором ~1050 комбинаций. Данная проблема изучается уже около 100 лет. RSA-алгоритм запатентован (20 сентября 1983, действует до 2000 года).

Теоретически можно предположить, что возможно выполнение операции **f-1**, не вычисляя **p** и **q**. Но в любом случае задача эта не проста и разработчики считают ее трудно факторизуемой.

Предположим, что мы имеем зашифрованный текст **f(M)** и исходный текст **M**, и мы хотим найти значения **p** и **q**. Нетрудно показать, что таких исходных данных для решения задачи недостаточно ‑ надо знать все возможные значения ***Mi***.

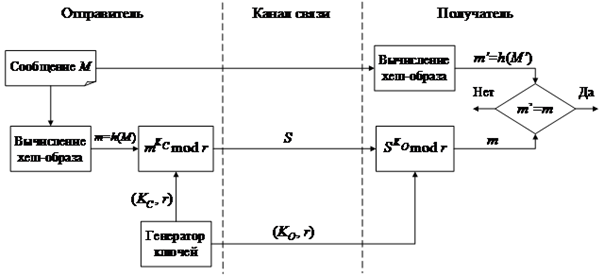


Рисунок 1 ‑ Алгоритм шифрования

Проясним использование алгоритма RSA на конкретном примере. Выбираем два простые числа **p=7; q=17** (на практике эти числа во много раз длиннее). В этом случае **n = p\*q** будет равно 119. Теперь необходимо выбрать **e**, выбираем **e=5**. Следующий шаг связан с формированием числа **d** так, чтобы d\*e=1 mod [(p-1)(q-1)]. **d=77** (использован расширенный алгоритм Эвклида). **d** ‑ секретный ключ, а **e** и **n** характеризуют открытый ключ. Пусть текст, который нам нужно зашифровать представляется M=19. **С = Memod n**. Получаем зашифрованный текст C=66. Этот “текст” может быть послан соответствующему адресату. Получатель дешифрует полученное сообщение, используя **М= Cdmod n** и C = 66. В результате получается M = 19.

На практике общедоступные ключи могут помещаться в специальную базу данных. При необходимости послать партнеру зашифрованное сообщение можно сделать сначала запрос его открытого ключа. Получив его, можно запустить программу шифрации, а результат ее работы послать адресату. На использовании общедоступных ключей базируется и так называемая электронная подпись, которая позволяет однозначно идентифицировать отправителя. Сходные средства могут применяться для предотвращения внесения каких-либо корректив в сообщение на пути от отправителя к получателю. Быстродействующие аппаратные 512-битовые модули могут обеспечить скорость шифрования на уровне 64 кбит в сек. Готовятся ИС, способные выполнять такие операции со скоростью 1 Мбайт/сек. Разумный выбор параметра **e** позволяет заметно ускорить реализацию алгоритма.

**Способы взлома криптосистемы RSA**

Существует несколько способов взлома RSA. Наиболее эффективная атака: найти секретный (private) ключ, соответствующий необходимому открытому (public) ключу. Это позволит нападающему читать все сообщения, зашифрованные открытым (public) ключом и подделывать подписи. Такую атаку можно провести, найдя главные сомножители (факторы) общего модуля ***n*** ‑ ***p*** и ***q***. На основании ***p***, ***q*** и ***e*** (общий показатель), нападающий может легко вычислить частный показатель ***d***. Основная сложность ‑ поиск главных сомножителей (факторинг) ***n***; безопасность RSA зависит от разложения на сомножители (факторинга), что является трудноразрешимой задачей, не имеющей эффективных способов решения.

Фактически, задача восстановления секретного (private) ключа эквивалентна задаче разложения на множители (факторинга) модуля: можно использовать ***d*** для поиска сомножителей ***n***, и наоборот можно использовать ***n*** для поиска ***d***. Надо отметить, что усовершенствование вычислительного оборудования само по себе не уменьшит стойкость криптосистемы RSA, если ключи будут иметь достаточную длину. Фактически же совершенствование оборудования увеличивает стойкость криптосистемы.

Другой способ взломать RSA состоит в том, чтобы найти метод вычисления корня степени ***e*** из mod ***n***. Поскольку ***С*** = ***M***\****e***\*(mod ***n***), то корнем степени ***e*** из (mod ***n***) является сообщение ***M***. Вычислив корень, можно вскрыть зашифрованные сообщения и подделывать подписи, даже не зная частный (private) ключ. Такая атака не эквивалентна факторингу, но в настоящее время неизвестны методы, которые позволяют взломать RSA таким образом. Однако, в особых случаях, когда на основе одного и того же показателя относительно небольшой величины шифруется достаточно много связанных сообщений, есть возможность вскрыть сообщения. Упомянутые атаки ‑ единственные способы расшифровать все сообщения, зашифрованные данным ключом RSA. Существуют и другие типы атак, позволяющие, однако, вскрыть только одно сообщение и не позволяющие нападающему вскрыть прочие сообщения, зашифрованные тем же ключом.

Самое простое нападение на единственное сообщение ‑ атака по предполагаемому открытому тексту. Нападающий, имея зашифрованный текст, предполагает, что сообщение содержит какой-то определенный текст, например, «Нападение на рассвете», затем шифрует предполагаемый текст открытым (public) ключом получателя и сравнивает полученный текст с имеющимся зашифрованным текстом. Такую атаку можно предотвратить, добавив в конец сообщения несколько случайных битов. Другая атака единственного сообщения применяется в том случае, если кто-то посылает одно и то же сообщение M трем корреспондентам, каждый из которых использует общий показатель ***e*** = 3. Зная это, нападающий может перехватить эти сообщения и расшифровать сообщение ***M***.

Такую атаку можно предотвратить вводя в сообщение перед каждым шифрованием несколько случайных бит. Также существуют несколько атак по зашифрованному тексту (или атаки отдельных сообщений с целью подделки подписи), при которых нападающий создает некоторый зашифрованный текст и получает соответствующий открытый текст, например, заставляя обманным путем зарегистрированного пользователя расшифровать поддельное сообщение.

Что же касается затруднения взлома увеличением размера ключа, то удвоение длины модуля в среднем увеличивает время операций открытого (public) ключа (шифрование и проверка подписи) в четыре раза, а время операций секретного (private) ключа (расшифровка и подпись) в восемь раз. Разница между временем работы отрытого и секретного ключей возникает потому, что открытый показатель может оставаться неизменным, в то время как модуль будет увеличен, а длина частного показателя будет увеличена пропорционально увеличению длины ключа. Время создания ключей при удвоении модуля увеличивается в 16 раз, но это нечасто выполняемая операция и потому на общей производительности это практически не сказывается.

Надо отметить, что размеры ключей в криптосистеме RSA (а также и в других криптосистемах открытого (public) ключа) намного больше размеров ключей систем блокового шифрования типа DES, но надежность ключа RSA несравнима с надежностью ключа аналогичной длины другой системы шифрования.

Разумеется, существуют и атаки, нацеленные не на криптосистему непосредственно, а на уязвимые места всей системы коммуникаций в целом; такие атаки не могут рассматриваться как взлом RSA, так как говорят не о слабости алгоритма RSA, а скорее об уязвимости его конкретной реализации.

Например, нападающий может завладеть секретным (private) ключом, если тот хранится без должных предосторожностей. Необходимо подчеркнуть, что для полной защиты недостаточно защитить выполнение алгоритма RSA и принять меры вычислительной безопасности, то есть использовать ключ достаточной длины. На практике же наибольший успех имеют атаки на незащищенные этапы управления ключами системы RSA.

**Применение на практике**

На практике криптосистема RSA часто используется вместе с криптографической системой секретного ключа типа DES для зашифровывания сообщения ключом RSA посредством цифрового конверта.

Предположим, что Алиса посылает зашифрованное сообщение Бобу. Сначала она шифрует сообщение по алгоритму DES, используя случайно выбранный ключ DES и затем шифрует ключ DES открытым (public) ключом RSA Боба. Сообщение зашифрованное ключом DES и ключ DES зашифрованный в свою очередь ключом RSA вместе формируют цифровой конверт RSA и отсылаются Бобу. Получив цифровой конверт, Боб расшифровывает ключ DES с помощью своего частного (private) ключа, а затем использует ключ DES, чтобы расшифровать само сообщение.

**Применение алгоритма RSA для установления подлинности и цифровых подписей**

Криптосистема RSA может использоваться также и для подтверждения подлинности или идентификации другого человека или юридического лица. Это возможно потому, что каждый зарегистрированный пользователь криптосистемы имеет свой уникальный частный (private) ключ, который (теоретически) больше никому недоступен. Именно это делает возможным положительную и уникальную идентификацию

Предположим, Алиса желает послать подписанное сообщение Бобу. Она хеширует сообщение (применяет к сообщению хеш-функцию), чтобы создать дайджест сообщения, который является как бы “цифровым отпечатком” сообщения.

Затем Алиса шифрует дайджест сообщения своим частным (private) ключом, создавая цифровую подпись, которую посылает Бобу непосредственно вместе с сообщением.

Получив сообщение и подпись, Боб расшифровывает подпись открытым (public) ключом Алисы и получает таким образом даджест сообщения. Затем он обрабатывает сообщение той же хеш-функцией что и Алиса и сравнивает результат с дайджестом сообщения, полученным при расшифровке подписи. Если они совпадают точно, то это означает успешную проверку подписи и Боб может быть уверен, что сообщение действительно послано Алисой. Если же результаты не одинаковы, то это означает, что либо сообщение пришло не от Алисы, либо было изменено при передаче (то есть после того, как Алиса его подписала). Подпись Алисы может проверить любой, кто получил или перехватил это сообщение.

Если же Алиса хочет сохранить содержание документа в тайне, то она подписывает документ, а затем зашифровывает его открытым (public) ключом Боба. Боб расшифровывает сообщение своим частным (private) ключом и проверяет подпись на восстановленном сообщении, используя открытый (public) ключ Алисы. Либо ‑ если, например, необходимо, чтобы посредник мог подтвердить целостность сообщения, не получая доступ к его содержанию ‑ вместо дайджеста открытого текста может быть рассчитан дайджест зашифрованного сообщения.

На практике же общий показатель алгоритма RSA обычно много меньше показателя частного и потому проверка подписи осуществляется быстрее чем подписание. Это является оптимальным так как сообщение подписывается только однажды, а проверка подписи может быть неоднократной.

Для обеспечения секретности обмена информацией необходимо исключить для нападающего возможность во-первых получить открытое сообщение, соответствующее хешированному, а во-вторых получить два различных хешированных сообщения, имеющих одно значение так как в любом из этих случаев нападающий имеет возможность присоединить к подписи Алисы ложное сообщение. Специально для этого разработаны функции хеширования MD5 и SHA, которые делают такое сопоставление невозможным.

Цифровая подпись может сопровождаться одним или несколькими сертификатами. Сертификат ‑ заверенный подписью документ, подтверждающий принадлежность открытого (public) ключа определенному владельцу, благодаря чему предотвращается возможность имитации отправителя. При наличии сертификата, получатель (или третье лицо) имеет возможность удостовериться в принадлежности ключа автору сообщения, то есть ключ позволяет удостоверить сам себя.

**Использование криптосистемы RSA в настоящее время**

Криптосистема RSA используется в самых различных продуктах, на различных платформах и во многих отраслях. В настоящее время криптосистема RSA встраивается во многие коммерческие продукты, число которых постоянно увеличивается. Также ее используют операционные системы Microsoft, Apple, Sun и Novell. В аппаратном исполнении RSA алгоритм применяется в защищенных телефонах, на сетевых платах Ethernet, на смарт-картах, широко используется в криптографическом оборудовании. Кроме того, алгоритм входит в состав всех основных протоколов для защищенных коммуникаций Internet, в том числе S/MIME, SSL и S/WAN, а также используется во многих учреждениях, например, в правительственных службах, в большинстве корпораций, в государственных лабораториях и университетах. На осень 2000 года технологии с применением алгоритма RSA были лицензированы более чем 700 компаниями.

Технологию шифрования RSA BSAFE используют около 500 миллионов пользователей всего мира. Так как в большинстве случаев при этом используется алгоритм RSA, то его можно считать наиболее распространенной криптосистемой общего (public) ключа в мире и это количество имеет явную тенденцию к увеличению по мере роста Internet.

**Стандарты с применением RSA**

Криптосистема RSA ‑ часть многих стандартов. Стандарт ISO 9796 описывает RSA как совместимый криптографический алгоритм, соотвествующий стандарту безопасности ITU-T X.509. Кроме этого криптосистема RSA является частью стандартов SWIFT, ANSI X9.31 rDSA и проекта стандарта X9.44 для американских банков. Австралийский стандарт управления ключами AS2805.6.5.3 также включает систему RSA.

Алгоритм RSA используется в Internet, в частности он входит в такие протоколы как S/MIME, IPSEC (Internet Protocol Security) и TLS (которым предполагается заменить SSL), а также в стандарт PKCS, применяемый в важных приложениях.   
Для разработчиков приложений с применением PKCS организация OSI Implementers' Workshop (OIW) выпустила соглашение, которое в частности посвящено алгоритму RSA.

Множество других разрабатываемых в настоящее время стандартов включают в себя либо сам алгоритм RSA или его поддержку либо рекомендуют криптосистему RSA для обеспечения секретности и/или установления подлинности (аутентификации). Например, включают в себя систему RSA рекомендации IEEE P1363 и WAP WTLS.

**Криптосистема RSA в мире**

На начало 2001 года криптосистема RSA являлась наиболее широко используемой асимметричной криптосистемой (криптосистемой открытого (public) ключа) и зачастую называется стандартом де факто. Вне зависимости от официальных стандартов существование такого стандарта чрезвычайно важно для развития электронной коммерции и вообще экономики. Единая система открытого (public) ключа допускает обмен документами с электронно-цифровыми подписями между пользователями различных государств, использующими различное программное обеспечение на различных платформах; такая возможность насущно необходима для развития электронной коммерции. Распространение системы RSA дошло до такой степени, что ее учитывают при создании новых стандартов. При разработке стандартов цифровых подписей, в первую очередь в 1997 был разработан стандарт ANSI X9.30, поддерживающий Digital Signature Standard (стандарт Цифровой подписи). Годом позже был введен ANSI X9.31, в котором сделан акцент на цифровых подписях RSA, что отвечает фактически сложившейся ситуации в частности для финансовых учреждений.

Недостатки защищенной аутентификации (установления подлинности) были главным препятствием для замены бумажного документооборота электронным; почти везде контракты, чеки, официальные письма, юридические документы все еще выполняются на бумаге. Именно это ‑ необходимость элементов бумажного документооборота ‑ не позволяло полностью перейти к электронным транзакциям. Предлагаемая RSA цифровая подпись ‑ инструмент, который позволит перевести наиболее существенные бумажные документопотоки в электронно-цифровой вид. Благодаря цифровым подписям многие документы ‑ паспорта, избирательные бюллетени, завещания, договора аренды ‑ теперь могут существовать в электронной форме, а любая бумажная версия будет в этом случае только копией электронного оригинала. Все это стало возможным благодаря стандарту цифровых подписей RSA.

**Результат работы программы**

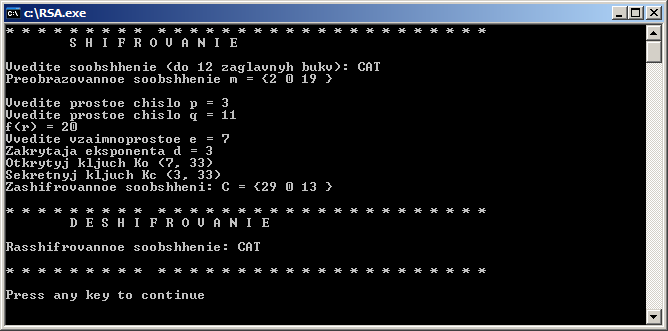


Рисунок 1 ‑ Работа программы

**Приложение**

***Листинг программы***

#include <iostream.h> //Подключаемые файлы

#include <conio.h>

#include <stdlib.h>

const N = 12;

char simv[26] = {'A','B','C','D','I','F','G','H','I','J',

'K','L','M','N','O','P','Q','R','S','T',

'U','V','W','X','Y','Z'};

void rashEvklid(int , int , int \*, int \*); //Расширенный метод Евклида

int mod(int , int);

int bystrVozvStep(int , int , int); //Быстрое возведение в степень

void main(void)

{

char ch[N];

int p, q, r, i, fr, e, d, j;

int x1, y1, \*px1 = &x1, \*py1 = &y1;

int m[N], c[N], m1[N];

for (i = 0; i < N; i++)

m[i] = c[i] = m1[i] = -1;

cout << "\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*"<< endl;

cout << "\tS H I F R O V A N I E\n"<< endl;

cout << "Vvedite soobshhenie (do " << N << " zaglavnyh bukv): ";

cin >> ch;

cout << "Preobrazovannoe soobshhenie m = {";

i = 0;

while (ch[i] != '\0')

{

for (j = 0; j < 26; j++)

if (simv[j] == ch[i])

m[i] = j;

cout << m[i] << " ";

i++;

}

cout << "}" <<endl;

cout << "\nVvedite prostoe chislo p = ";

cin >> p;

cout << "Vvedite prostoe chislo q = ";

cin >> q;

r = p \* q;

fr = (p - 1) \* (q - 1);

cout << "f(r) = " << fr << "\nVvedite vzaimnoprostoe e = ";

cin >> e;

rashEvklid(fr, e, px1, py1);

if (y1 < 0)

y1 += fr;

d = y1;

cout << "Zakrytaja eksponenta d = " << d << endl;

cout << "Otkrytyj kljuch Ko (" << e << ", " << r << ")" << endl;

cout << "Sekretnyj kljuch Kc (" << d << ", " << r << ")" << endl;

cout << "Zashifrovannoe soobshheni: C = {";

i = 0;

while ((m[i] >= 0)&&(i < N))

{

c[i] = bystrVozvStep(m[i],e, r);

cout << c[i] << " ";

i++;

}

cout << "}" << endl;

cout << "\n\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*"<< endl;

cout << "\tD E S H I F R O V A N I E\n"<< endl;

i = 0;

while ((c[i] >= 0)&&(i < N))

{

m1[i] = bystrVozvStep(c[i], d, r);

i++;

}

cout << "Rasshifrovannoe soobshhenie: ";

i = 0;

while ((m[i] >= 0)&&(i < N))

{

cout << simv[m1[i]];

i++;

}

cout << endl;

cout << "\n\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*"<< endl;

cout << "\nPress any key to continue" << endl;

getch();

}

int mod(int a, int n)

{

int q;

q = a%n;

if (q < 0) q+=n;

return q;

}

void rashEvklid(int a, int n, int \*x1, int \*y1) //Расширенный метод Евклида

{

int q, i; //Вспомогательные переменные

int u[3] = {0, 1, n}, v[3] = {1, 0, a}, t[3] = {0}; //Массивы для вычислений

while (u[2] != 1)

{

q=u[2]/v[2];

for (i = 0; i<3; i++)

t[i] = u[i] - v[i]\*q;

for (i = 0; i<3; i++)

{

u[i] = v[i];

v[i] = t[i];

}

}

\*x1 = u[0]; //Сохранение вычислений

\*y1 = u[1];

}

int bystrVozvStep(int a, int k, int n) //Быстрое возведение в степень

{

int b = 1;

while (k)

{

if (k%2 == 0)

{

k /= 2;

a = mod((a\*a), n);

}

else

{

k--;

b = mod((b\*a),n);

}

}

return b;

}