МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и
прикладная математика»
Кафедра: 806 «Вычислительная математика и
программирование»

Реферат
по курсу «Фундаментальная
информатика»
I семестр
Тема:

«Язык программирования C++ на примере разработки программы, реализующей шифрование файлов с открытым ключом»

Группа:	М8О- 109Б-22
Студент:	Степанов А.Н.
Преподаватель:	Сысоев М.А
Оценка:	
Дата:	

Москва, 2020

Содержание

1.История C++ и сферы применения

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Синтаксис C++ унаследован от языка C. Изначально одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Где применяется сейчас?

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает *создание операционных систем*, *разнообразных прикладных программ, драйверов устройств*, п*риложений для встраиваемых систем, высокопроизводительных серверов, а также компьютерных игр*. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

2.Алгоритм шифрования RSA

Как многим известно, основой современной тайнописи и криптографии является криптографический алгоритм - пара связанных математический функций, которые реализуют шифрование и дешифрование на основе переданного им закрытого (секретного ключа) и открытого(публичного) ключей, если это ассиметричный алгоритм шифрования, или использую только закрытый ключ (симметричное шифрование). Более ранние алгоритмы, ограниченные(restricted), основанные на принципе секретности самого алгоритма, с приходом компьютерной эры утратили свое влияние или какой-либо смысл из-за широкого доступа средств декомпиляции, позволяющих вытащить код из исполняемого файла, а также необходимости изменять алгоритм всякий раз, когда сужается группа пользователей шифрованием. Криптографические алгоритмы, использующие ключ как основу своей безопасности, обладают большей устойчивостью к криптоанализу или утечки информации: даже зная применённый алгоритм, злоумышленник может получить доступ к зашифрованной информации только при наличии конкретного значения ключа. Мною рассматривался ассиметричный метод шифрования, который требует только передачи в устойчивых к подмене сообщений канале открытого ключа, необходимого только для шифрования данных и бесполезного при дешифрации.
Одним из самых распространенных, общепринятых и выдержавших проверку временем алгоритмов шифрования с открытым ключом (и цифровой подписи) является алгоритм RSA (аббревиатура фамилий создателей шифра: Рональда Линн Ривеста, Ади Шамира, Леонарда Макс Адлемана), впервые представленным публике в 1997 году, в последствие используемым минобороны США. Его работа основана на односторонней функции, задействующей двух фактах из теории чисел:
1 задача проверки числа на простоту является сравнительно легкой

2 задача разложения чисел вида n = pq (p и q — простые числа); на множители является очень трудной, если мы знаем только n, а p и q — большие числа(факторизация больших чисел)

Итак, рассмотрим сам алгоритм. Пусть отправитель А захочет передать получателю Б зашифрованное сообщение.

Тогда Б генерирует открытый и закрытый ключ, отсылая открытый отправителю А.

Для генерации выбираются 2 простых больших числа P и Q, а затем вычисляется произведение N:

$N = PQ$.

После этого определяется вспомогательное число f:

$f = (P - l)(Q - 1)$.

Затем случайным образом выбирается число $d < f$ и взаимно простое с f(открытая экспонента)

Далее ему необходимо найти число e, такое, что

$ed \bmod f = 1$.

N и d - открытый ключ, e - закрытый

Так как пользователь Б хочет получить зашифрованное сообщение от пользователя А, значит пользователь Б должен отправить свой открытый ключ (d, N) пользователю А. Числа P и Q больше не нужны, однако их нельзя никому сообщать; лучше всего их вообще забыть.

Если абонент А хочет передать некоторые данные абоненту Б, он должен представить свое сообщение в цифровом виде и разбить его на блоки m_1, m_2, m_3, ... , где $m_i < N$. Зашифрованное сообщение будет состоять из блоков c_i.

Абонент А шифрует каждый блок своего сообщения по формуле

$c_i = m_i^d \bmod N$

используя *открытые параметры* пользователя Б, и пересылает зашифрованное сообщение $C=(c_1, c_2, c_3, ...)$ по открытой линии.

Абонент Б, получивший зашифрованное сообщение, расшифровывает все блоки полученного сообщения по формуле

$m_i = c^e \bmod N$

Все расшифрованные блоки будут точно такими же, как и исходящие от пользователя А.

Злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших значениях P и Q.

Пример вычислений по формуле:

P = 3, Q = 11, N = 3x11 = 33.

f = (P - l)(Q - 1) = (3-1)(11-1) = 20.

d = 13(делители 13 и 1 не являются делителями числа 20)

e=17. (Проверяем: 13*17 mod 20 = 221 mod 20 = 1.)

Пользователь А, получивший открытый ключ (13, 33), увидев, что N=33, разбивает исходное сообщение на три блока, причем значение каждого меньше N. Например, пусть имеется три блока m_1=8, m_2=27, m_3,=5

$c_1=8^{13}$ mod 33 = 17c_2 = 27^{13} mod 33 = 15c_3 = 5^{13} mod 33 = 26

Зашифрованное сообщение, состоящее из трех блоков (17, 15, 26), передается пользователю Б, который, используя свой закрытый ключ e = 17 и N=33, расшифровывает сообщение:

m_1 = 17^{17} mod 33 = 8m_2 = 15^{17} mod 33 = 27m_3 = 26^{17} mod 33 = 5

3.Приложение, реализующие цикл шифрования RSA

Для программной реализации алгоритма была использована библиотека Openssl для языка C/C++, которая доказала свою эффективность в защите информации, в отличие от других менее популярных вариантов реализации этого алгоритма (в том числе и моего, ибо у меня отсутствуют знания и система проверки крипто устойчивости моей библиотеки). Сама программа разбита на 3 независимые части, что позволяет не устанавливать некоторые устройства шифратор и программу генерация ключей, усложняя возможным злоумышленникам возможную атаку с использованием подобранного шифротекста.

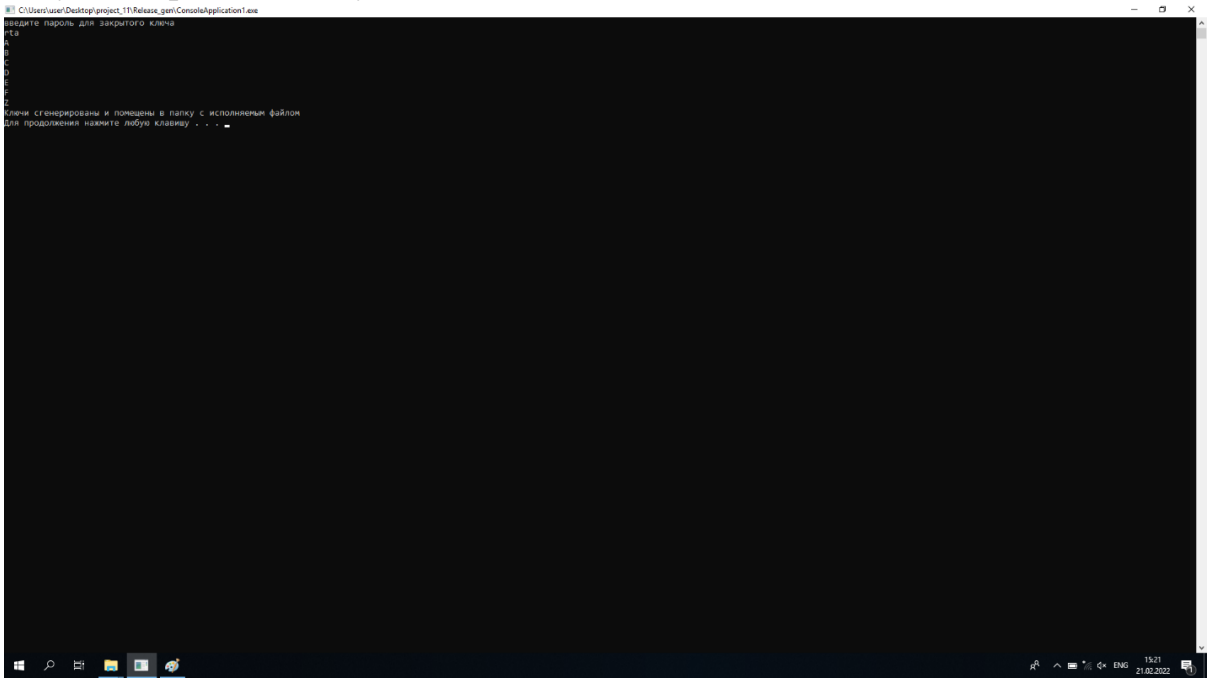
Использование языка C++ обусловлено обновленным стандартным потоком ввода и вывода, более удобным функционалом работы с массивами char и отсутствием разночтений между Си и C++, ибо всякий метод, конструкция или библиотека, реализованные в Си, также спокойно смогут работать и в C++.

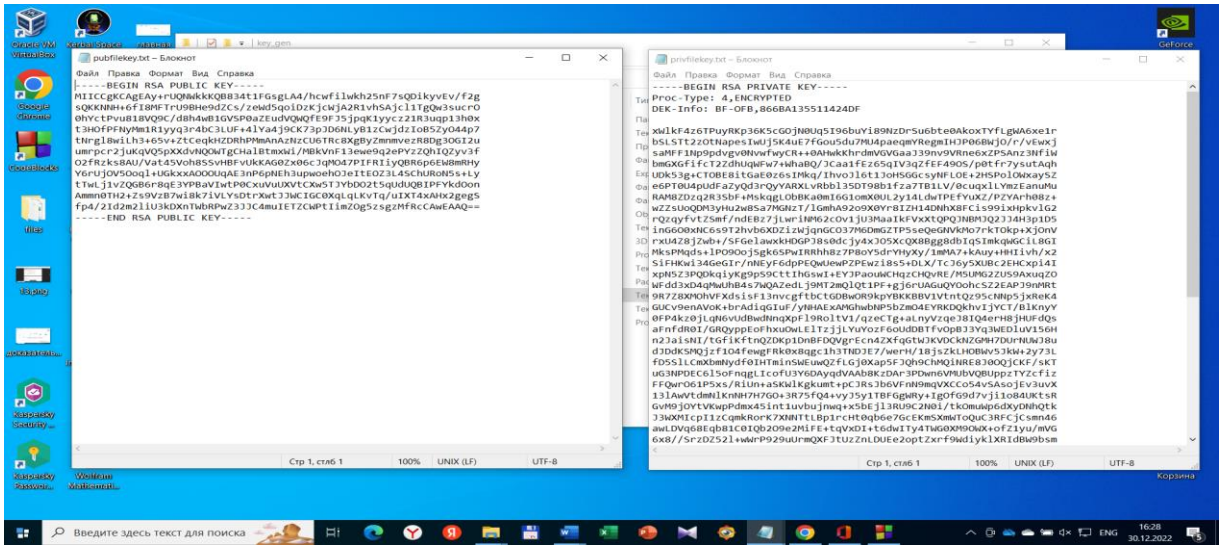
Это приложение нельзя назвать бесполезным, особенно для рядового пользователя, ведь, реализуя полноценный функционал алгоритма Rsa, оно позволяет нам держать в секрете личную информацию, обеспечить конфиденциальность деловой переписки или использовать его для долговременного хранения конфиденциальной информации на дисках или съемных носителях (что рекомендуется делать осторожно, как минимум не забывать их периодически сызнова перешифровывать)

4.Генератор ключей

Для генерации ключей (открытого и закрытого) нам потребуется пароль от пользователя, который будет необходим для доступа к закрытому ключу, что составляет дополнительную степень защиты, усложняя на некоторое время дешифровку при утечке закрытого ключа. Получив пароль с стандартного потока ввода, мы кладем его в массив char длины 500 символов, что будет достаточно для большинства нужд пользователя, и подаем указатель на него как аргумент для функции create_keys, которая использует структуру Rsa, представленную в библиотеке openssl в заголовочном файле rsa.h, используя вначале нулевые указатели для ключевых файлов, которые в дальнейшем будут заменены на относительные пути к ним. Подключая контекст шифрования мы генерируем на основе пароля в структуре закрытый ключ с большой экспонентой шифрования(защита от атака Винера) и открытый ключ ,математически связанный с закрытым, записывая их в файлы соответствующих ключей с дополнительной информацией о них, после чего модуль завершает свою работу.

Результат работы модуля:



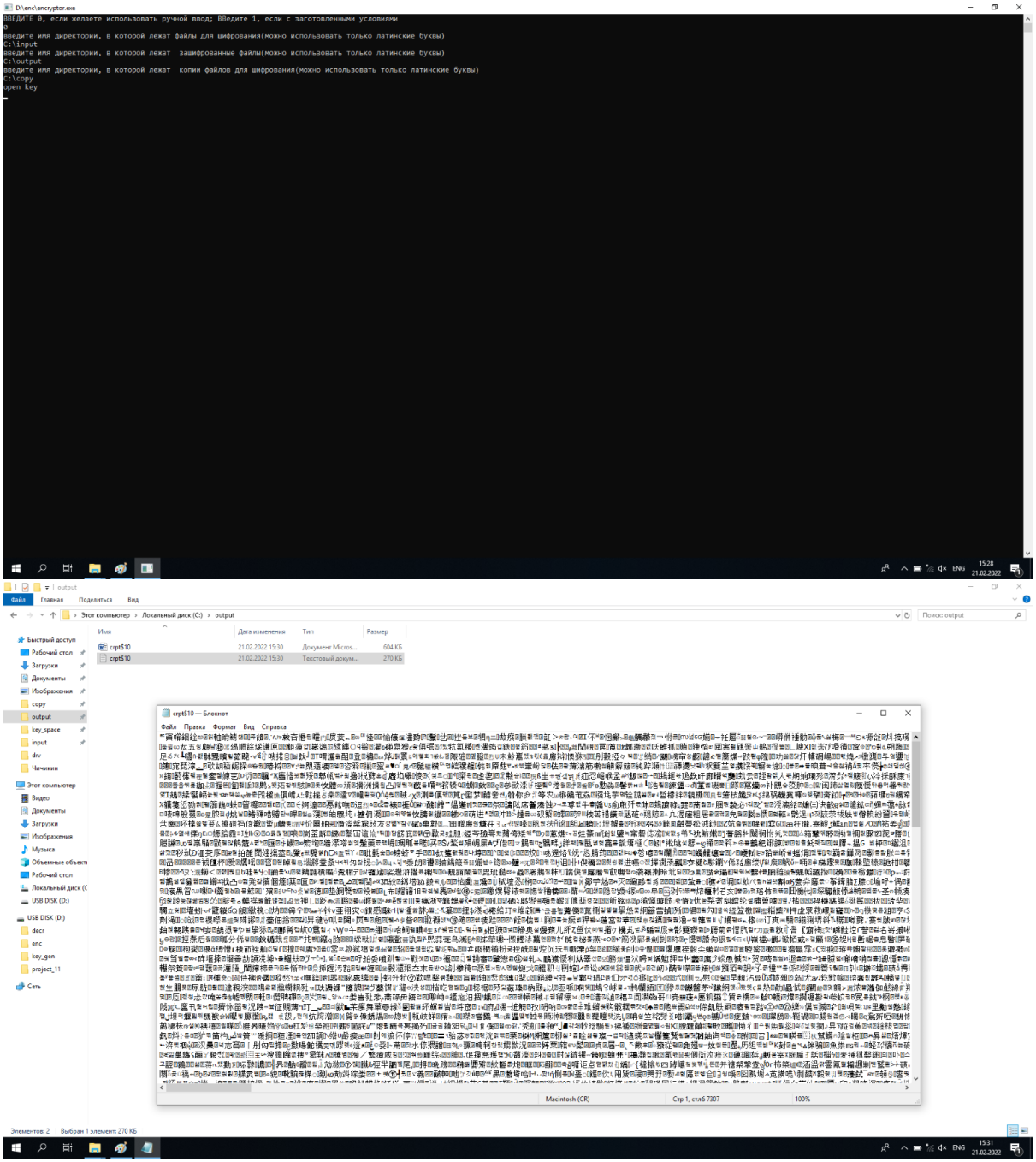

pubfilekey.txt – Блокнот
Файл Правка Формат Вид Справка
-----BEGIN RSA PUBLIC KEY-----
MIICCgKCAgEAy+rUQNWkkKQB834t1FGsgLA4/hcwfilwkh25nF7sQDikyvEv/f2g
sQKKNNH+6fI8MFTrU9BHe9dZCs/zeWd5qoiDzKjcWjA2R1vhSAjcl1TgQw3sucrO
0hYctPvu818VQ9C/d8h4wB1GVSP0aZEudVQWQfE9FJ5jpqK1yyczz1R3uqp13h0x
t3HOfPFAyYm1R1yyq3r4bC3LUF+4lYa4j9CK73pJD6NLyB1zCwjdz1o8SZy044p7
tNrgl8wilh3+65v+ztCeqkHZDRhPMmAnAzNzCU6TRc8XgByZmnmverR8Dg3OGI2u
umrpcr2juKqVQ5pXXdvNQOwTgChalBtmxWi/MBkVnFi3ewe9q2ePYzZQhIQZyv3f
O2fRzks8AU/Vat45Voh8SSvHBFvUkKAG0Zx06cJqMO47PIFRIiyQBR6p6EW8mRHy
Y6rUjOV5Ooql+UGkxxAOOUqAE3nP6pNEh3upwoehOJeItEOZ3L4SChURoN5s+Ly
tTwLj1vZQGB6r8qE3YPBaVIwtP0CxuVuUXVtCXw5TJYbDO2t5qUdUQBIPFYkd0on
Ammn0TH2+Zs9VzB7wi8k7iVLYsDtrXwtJWCIGC0XqLqLKvTq/uIXT4xAHx2geg5
fp4/2Id2m2liU3kDXnTWbRPwZ3JJC4muIETZCWPtIimZOg5zsgzMfRcCAwEAAQ==
-----END RSA PUBLIC KEY-----
Стр 1, стлб 1
100%
UNIX (LF)
UTF-8
privfilekey.txt – Блокнот
Файл Правка Формат Вид Справка
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: BF-OFB,866BA135511424DF
xWlkF4z6TPuyRKp36K5cGOjN0Uq5I96buYi89NzDrSu6bte0AkoxTYfLgWA6xe1r
bSLSTt2zOtNapesIWUj5K4uE7fGou5du7MU4paeqmYRegmIHJP06BWjO/r/vEwxj
saMFF1Np9pdvgv0NvwfwyCR++0AHwkKhrdmVGVGaaJ39nv9VRne6xZPSAnz3NfiW
bmGXGfifcT2ZdhUqWFw7+WhaBQ/JCaa1fEz65qIV3qZfEF49OS/p0tfr7ysutAqh
UDk53g+CTOBE8itGaE0z6sIMkq/IhvoJl6t1JoHSGGcsyNFLOE+2HSPolOWxay5Z
e6PT0U4puUFaZyQd3rQyYARXLvRbbl35DT98b1fza7TBlLV/0cuqxllYmzEanuMu
RAM8ZDzq2R3SbF+MskqgLObBKa0mI6G1omX0UL2y14LdwTPEfYuXZ/PZYArh08z+
wZZsUoQDM3yHu2wBSa7MGNzT/lGmhA92o9X0Yr8IZH14DNhX8FCis99ixHpkvlG2
rQzqyfvtZSmf/ndEBz7jLwriNM62cOv1jU3MaaIkFVxXtQPQJNBMJQ2JJ4H3p1D5
inG6O0xNC6s9T2hvb6XDZizWjqnGCO37M6DmGZTP5seQeGNVkMo7rkTOkp+XjOnV
rmU4Z8jZwb+/SFGelawxkHDGPJ8s8dcjy4x3O5XcQX88ggBdb1qSImkqW6Cil8GI
MksPMqds+lPO9OojSgk65PwIRRhh8z7P8oY5drYHyXy/1mMA7+kAuy+HHIivh/x2
SiFHKw134GeGIr/nNEyF6dpPEQwUewPZPEwzi8s5+DLX/TcJ6y5XUBc2EHCxpi4I
xpN5Z3PQDkqiyKg9pS9CttIh6swI+EYJPaouWCHqzCHQvRE/M5UMGZZUS9AxuqZO
WFdd3xD4qMwUhB4s7aQAZedLj9MTzmQlQt1PF+gj6rUAGuQYOohcSZ2EAP39nNRt
9R7Z8XMOHVFXdsisf13nvcgfTbCtGDBwOR9kpYBKKBBV1VtntQz95cNNp5jxReK4
GUCv9enAVoK+brAdiqGIuF/yNHAExAMGhwbNP5bZmO4EYRKDQkhvIjYCT/BlKnyY
0FP4kz0jLqN6vUdBwdNnqXpFl9RoltV1/qzeCTg+aLnyVzqeJ8IQ4erH8jHUFdQs
aFnfdR0I/GRQyppEoFhxuOwLElTzjjLYuYozF6oUdDBTfvOpBJ3Yq3WEDluV156H
n2JaisNI/tGfiKftnQZDKp1OnBFDQVgrEcn4ZXfqGtWJKVDCkNZGMH7DUrNUwJ8u
dJDdK5MQjzf1O4fewgFRk0x8qgc1h3TNDJE7/werH/18jsZkLHOBWv5JkW+2y73L
fD5SlLCmXbmNydf0IHTminSWEuwQZfLGj0Xap5FJQh9ChMQiNRE8J0OQjCKF/sKT
uG3NPDEC6l5oFnqgLIcofU3Y6DAyqdVAAb8KzDAr3PDwn6VMUbVQBUppzTYZcfiz
FFQwrO61P5xs/RiUn+a5KWlKgkumt+pCJRs3b6VFnN9mqVXCCo54vSAsoJEv3uvX
13lAwVtdmWlKmNH7H7GO+3R75fQ4+vyJ5y1TBFGgWRy+IgOf69d7vji1o84UKtsR
GvM9jOYtVKwpPdmx45int1uvbujnwq+x5bEjl3RU9C2N0i/tkOmuWp6dXyDNhQtk
J3WXMIcpI1zCqmkRorK7XNNTtLBp1rcHt0qb6e7GcEKmSXmWToQuC3RFCjCsmn46
awLDVq68EqbB1C0IQb2O9e2MiFE+tqVxDI+t6dwITy4TWG0XM9OWX+ofZ1yu/mVG
6x8//SrzDZ52l+WwrP929uUrmQXFJtUzZnLDUEe2optZxrf9WdiyklXRIdBW9bsm
Стр 1, стлб 1
100%
UNIX (LF)
UTF-8
Введите здесь текст для поиска
ENG
16:28
30.12.2022

5.Encryptor

Вторым модулем программы является encryptor или шифратор, который принимает на вход абсолютные пути 3 директорий:
1 - где находятся файлы, которые пользователь хочет зашифровать.

2 - куда необходимо переместить уже зашифрованные файлы

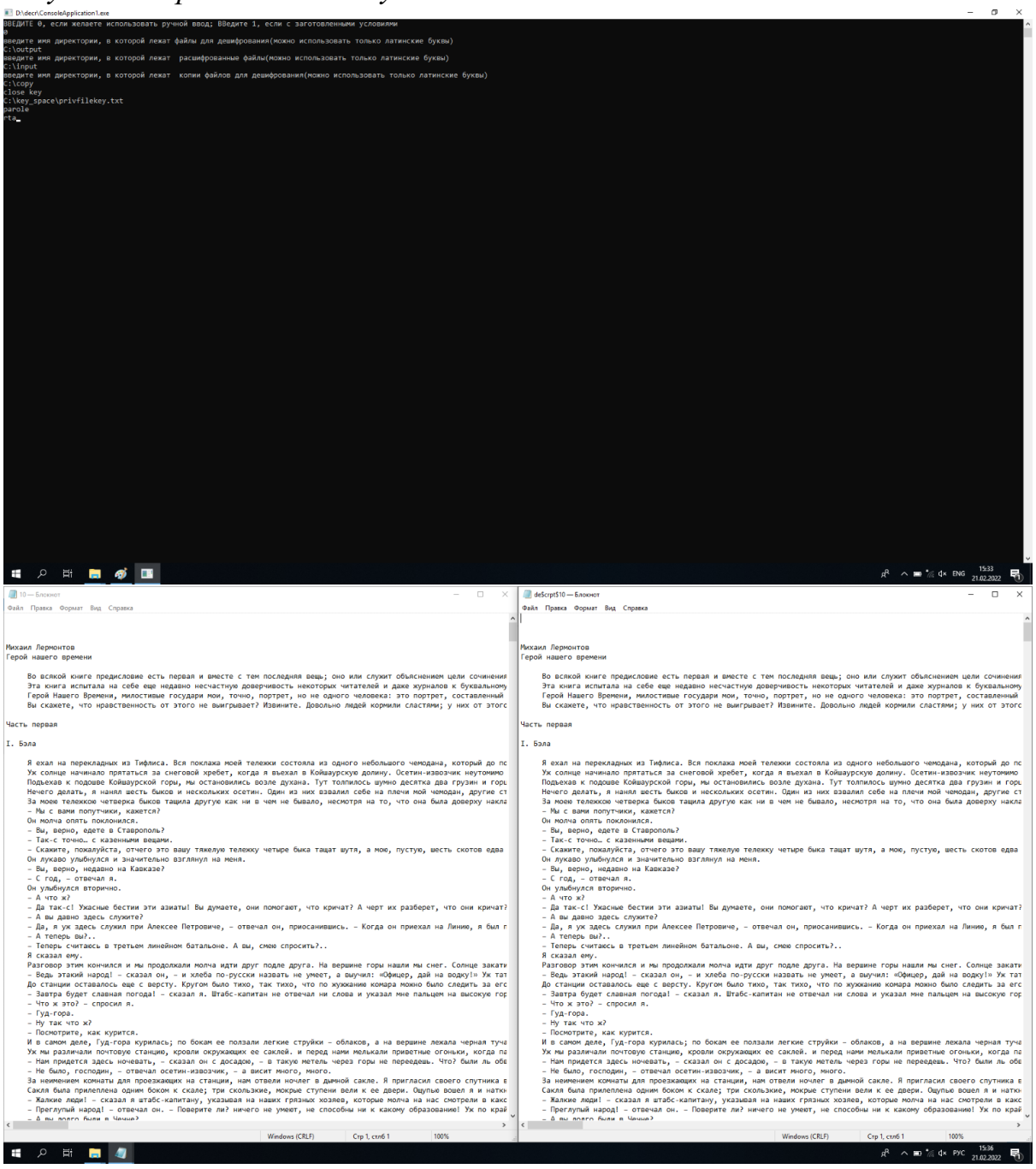
3 - путь к безопасной директории, где будут храниться уже зашифрованные файлы;
Также нам потребуется абсолютный адрес открытого ключа для дальнейшего процесса шифрования.
Использую методы библиотек dirent и direct переходим в необходимую нам директорию и считываем все ее содержимое, отделяя файлы от директорий и обрабатывая их поблочно(блок по размеру равен key_size), шифруя при помощи функции RSA_public_encrypt и записывая в новый файл с приставкой crpt, создаваемый с помощью функции _open из заголовочного файла io.h, предоставляющий функции низкоуровневого вводы-вывода. Универсальность шифрования достигается за счет флагов O_BINARY| O_APPEND, названия которых говорит само за себя. После того, как все символы были считаны(входящий блок не является ненулевым), мы копируем файлы шифрованный и исходный в соответствующие директории, после чего удаляем их из рабочей директории, используя для этого библиотеку Windows.h(мало компьютеров на unix пока, к сожалению). Итогом работы модуля будут служить шифрованные файлы.
Результат работы модуля шифрования:


D:\enc\encryptor.exe
ВВЕДИТЕ 0, если желаете использовать ручной ввод; ВВедите 1, если с заготовленными условиями
0
введите имя директории, в которой лежат файлы для шифрования(можно использовать только латинские буквы)
C:\input
введите имя директории, в которой лежат зашифрованные файлы(можно использовать только латинские буквы)
C:\output
введите имя директории, в которой лежат копии файлов для шифрования(можно использовать только латинские буквы)
C:\copy
open key
output
Файл
Главная
Поделиться
Вид
Этот компьютер
Локальный диск (C:)
output
Поиск: output
Имя
Дата изменения
Тип
Размер
crpt510
21.02.2022 15:30
Документ Micros...
604 КБ
crpt510
21.02.2022 15:30
Текстовый докум...
270 КБ
Быстрый доступ
Рабочий стол
Загрузки
Документы
Изображения
copy
output
key_space
input
drv
Чичкин
Этот компьютер
Видео
Документы
Загрузки
Изображения
Музыка
Объемные объекты
Рабочий стол
Локальный диск (C
USB DISK (D:)
USB DISK (D:)
decr
enc
key_gen
project_11
Сеть
Элементов: 2
Выбран 1 элемент: 270 КБ
crpt510 — Блокнот
Файл
Правка
Формат
Вид
Справка
Macintosh (CR)
Стр 1, стлб 7307
100%
ENG
15:28
21.02.2022
ENG
15:31
21.02.2022

6.Decryptor

Структура дешифратора аналогична структуре программы для шифрования, за исключением того, что при шифровании от нас потребуется ранее введенный пароль, и если его ввести неверно, то программа аварийно себя заблокирует и завершится. В результате применения этого модуля мы получаем файл de$crpt,который полностью соответствует исходному файлу, до шифрования.

Результат применения модуля:



7.Список литературы

1)Справка о C++- https://ru.wikipedia.org/wiki/C%2B%2B ;

2) Брюс Шнайер:”Прикладная криптография. Протоколы, алгоритмы и исходный код на C”,2-е юбилейное издание,2019, ООО ”Диалектика”;

3) Виктор Де Касто.:” Просто криптография”

4)Репозиторий с кодом - https://github.com/Alexstepano/Rsa_realised

8.Приложения.

Приложение 1. Код программы генерации ключа.

```
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <iostream>
#include <openssl/applink.c>
#include <cstdlib> // для system
/* Имена ключевых файлов */


using namespace std;

void create_keys(char* passphrase)
{
   //Указатель на структуру для хранения ключей
   RSA* rsa = NULL;
   //Длина ключа в битах
   unsigned long bits = 4096;//2^12
   FILE* priv_key_file = NULL, * pub_key_file = NULL;
   //контекст алгоритма шифрования
   const EVP_CIPHER* cipher = NULL;

   priv_key_file = fopen("privfilekey.txt", "wb");
   cout << "A" << endl;

   pub_key_file = fopen("pubfilekey.txt", "wb");
   cout << "B" << endl;
   //Генерируем ключи
   rsa = RSA_generate_key(bits, 65537, NULL, NULL);
   cout << "C" << endl;
   //Формируем контекст алгоритма шифрования
   OpenSSL_add_all_ciphers();//
```

```
    cipher = EVP_get_cipherbyname("bf-ofb");
    cout << "D" << endl;
    //Получаем из структуры rsa открытый и секретный ключи и
сохраняем в файлах.
    //Секретный ключ шифруем с помощью парольной фразы
"hello"
    PEM_write_RSAPrivateKey(priv_key_file, rsa, cipher, NULL, 0,
NULL, passphrase);
    PEM_write_RSAPublicKey(pub_key_file, rsa);
    cout << "E" << endl;
    //Освобождаем память, выделенную под структуру rsa
    RSA_free(rsa);
    cout << "F" << endl;
    //закрываем файлы с ключами
    fclose(priv_key_file);
    fclose(pub_key_file);
    cout << "Z" << endl;
}

int main()
{
    setlocale(LC_ALL, "Russian");
    char parole[500] = {};
    cout << "введите пароль для закрытого ключа"<<endl;
    cin >> parole;
    create_keys(parole);
 cout << "Ключи сгенерированы и помещены в папку с
исполняемым файлом" << endl;
 system("pause");
 return 0;
}
```

Приложение 2. Код программы шифрования

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
```

```
#include <string.h>
#include <dirent.h>
#include <stdio.h>
#include <direct.h>
#include<io.h>
#include "sys/stat.h"
#include <fstream>
#include <clocale>
#include <windows.h>
#include <openssl/applink.c>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <fcntl.h>
#include <stdlib.h>
#include <openssl/evp.h>
using namespace std;
int main(void)
{
   setlocale(LC_ALL, "rus");//включаем русский язык
   char path1[255];
   char path2[255];
   char path3[255];
   char pathz[255];
   char costil[255];
   char path[255];//создаем буффер,куда мы печатаем название
деректории
   int h = 0;
   cout << "ВВЕДИТЕ 0, если желаете использовать ручной
ввод; ВВедите 1, если с заготовленными условиями" << endl;
   cin >> h;
   if (h == 0) {
      gets_s(costil, 250);
      cout << "введите имя директории, в которой лежат файлы
для шифрования(можно использовать только латинские буквы)"
<< endl;
```

```
        gets_s(path, 255);//получаем название деректории откуда берем файлы
        cout << "введите имя директории, в которой лежат зашифрованные файлы(можно использовать только латинские буквы)" << endl;
        gets_s(path1, 255);//получаем название деректории куда отправляем файлы
        cout << "введите имя директории, в которой лежат  копии файлов для шифрования(можно использовать только латинские буквы)" << endl;
        gets_s(path2, 255);
        cout << "open key" << endl;
        gets_s(path3, 250);
    }
    else
    {//получаем имя файла с конфигурацией, откуда считываем требуемую информацию
        gets_s(costil, 250);
        gets_s(pathz, 250);
        fstream fil(pathz);
        fil.getline(path, 255);
        fil.getline(path1, 255);
        fil.getline(path2, 255);
        fil.getline(path3, 255);
        cout << path << 211 << endl;
        cout << path1 << endl;
        cout << path2 << endl;
        cout << path3 << endl;
        fil.close();
    }
    RSA* pubKey = NULL;
    unsigned char* ptext, * ctext;
    int inlen, outlen;
    FILE* pub_key_file;
    struct stat s;
```

```
   DIR* dir = opendir(path);//получая указатель на поток к папке
и вставляем наш путь к ней, открывая ее
   struct dirent* ent;//структура данных в папке
   pub_key_file = fopen(path3, "rb");
   if (pub_key_file == NULL) {
      cout << "err:filekey do not open ";
      return -1;
   }
   pubKey = PEM_read_RSAPublicKey(pub_key_file, NULL, NULL,
NULL);
   fclose(pub_key_file);
   int key_size = RSA_size(pubKey);
   cout << key_size << endl;
   ctext = (unsigned char*)malloc(key_size);
   ptext = (unsigned char*)malloc(key_size);

   _chdir(path);//меняет рабочую директорию на указанный путь
   if (dir)//если открылось
   {

      while ((ent = readdir(dir)) != NULL)
      {
         cout << (ent->d_name) << "\n";//из структуры мы
выуживаем имя файла
         if (stat((ent->d_name), &s) == 0) {

            if (s.st_mode & S_IFDIR)//это директория
            {
               cout << "direct" << "\n"; cout <<
"_____" << endl;

            }
            else if (s.st_mode & S_IFREG)
            {
```

```
char nome[500] = { "crpt" };//если убрать в начало, то все ломается

char l_pyt1[1500] = { "crpt" };//пути
char l_pyt2[1500] = { "crpt" };
char l_pyt3[1500] = { "crpt" };
char l_pyt4[1500] = { "crpt" };
cout << "file" << "\n";//it a file
strcpy(nome, "crpt$");//добавляем crpt в название файла
strcat(nome, ent->d_name);//а потом приписываем имя
cout << nome << endl;

//______закрипт.откуда_____________________________________
_____________________________________________________________
strcpy(l_pyt1, path);
strcat(l_pyt1, "\\");
strcat(l_pyt1, nome);
cout << l_pyt1 << endl;
//________сейв. директория_____________________________________
_____________________________________
strcpy(l_pyt2, path1);
strcat(l_pyt2, "\\");
strcat(l_pyt2, nome);
cout << l_pyt2 << endl;
//________откуда. директория_____________________________________
_____________________________________
strcpy(l_pyt3, path);
strcat(l_pyt3, "\\");
strcat(l_pyt3, ent->d_name);
cout << l_pyt3 << endl;
//________копия. деректория_____
```

```
                strcpy(l_pyt4, path2);
                strcat(l_pyt4, "\\");
                strcat(l_pyt4, ent->d_name);
                cout << l_pyt4 << endl;
                //FILE* file;
                //FILE* ofs;
               // file = fopen(ent->d_name, "rb");//открываем для
чтения
                //ofs = fopen(nome, "wb");//для записи и шифровки
                int out = _open(nome, O_CREAT | O_TRUNC |
O_RDWR | O_BINARY| O_APPEND, 0600);
                int in = _open(ent->d_name, O_RDWR | O_BINARY);
                if (in == NULL) {//проверяем,что открылся
                   cout << "err:file do not open ";
                   break;
                }
                else {
                   cout << "ok" << "\n";
                }
                while (1) {//читаем файл до конца файла, делим на
блоки, шифруем и записываем в файл.
                   inlen = _read(in, ptext, key_size - 11);
                   if (inlen <= 0) break;
                   outlen = RSA_public_encrypt(inlen, ptext, ctext,
pubKey, RSA_PKCS1_PADDING);
                   if (outlen != RSA_size(pubKey)) exit(-1);
                   _write(out, ctext, outlen);



                }
                //fclose(file);
                //fclose(ofs);

                CopyFileA(l_pyt1, l_pyt2, FALSE);//перемещаем
зашифрованный файл в директорию для отправления
```

```
            CopyFileA(l_pyt3, l_pyt4, FALSE);//перемещаем файл в директорию для копий

            DeleteFileA(l_pyt1);//удаляем лишний закрипт.файл из раб. директории
            DeleteFileA(l_pyt3);//удаляем лишний исходный файл из раб. директории
            cout << "_____" << endl;
        }
        else {
            cout << "unknown" << "\n";//неизвестный объект
            cout << "_____" << endl;
        }

      }
    }
  }
  else
  {
    cout << "Error opening directory" << endl;
  }
  return 0;
}
```

Приложение 3. Код программы дешифрования

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <dirent.h>
#include <stdio.h>
#include <direct.h>
#include<io.h>
```

```
#include "sys/stat.h"
#include <fstream>
#include <clocale>
#include <windows.h>
#include <openssl/applink.c>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <fcntl.h>
#include <stdlib.h>
#include <openssl/evp.h>
using namespace std;
int main(void)
{
   setlocale(LC_ALL, "rus");//включаем русский язык
   char path1[255];
   char path2[255];
   char path3[255];
   char parole [255] ;
   char pathz[255];
   char costil[255];
   char path[255];//создаем буфферы,куда мы печатаем название деректории
   int h = 0;
   cout << "ВВЕДИТЕ 0, если желаете использовать ручной ввод; ВВедите 1, если с заготовленными условиями" << endl;
   cin >> h;
   if (h == 0) {
      gets_s(costil, 250);
      cout << "введите имя директории, в которой лежат файлы для дешифрования(можно использовать только латинские буквы)" << endl;
      gets_s(path, 255);//получаем название деректории откуда берем зашифрованные файлы
```

```
        cout << "введите имя директории, в которой лежат расшифрованные файлы(можно использовать только латинские буквы)" << endl;
        gets_s(path1, 255);//получаем название деректории куда кладем дешифрованные файлы
        cout << "введите имя директории, в которой лежат  копии файлов для дешифрования(можно использовать только латинские буквы)" << endl;
        gets_s(path2, 255);
        cout << "close key" << endl;
        gets_s(path3, 250);
    }
    else {
        gets_s(costil, 250);
        gets_s(pathz, 250);
        fstream fil(pathz);
        fil.getline(path, 255);
        fil.getline(path1, 255);
        fil.getline(path2, 255);
        fil.getline(path3, 255);
        cout << path << 211 << endl;
        cout << path1 << endl;
        cout << path2 << endl;
        cout << path3 << endl;
        fil.close();
    }
    cout << "parole" << endl;
    gets_s(parole, 250);
    RSA* privKey = NULL;
    FILE* privKey_file;
    unsigned char* ptext, * ctext;
    int inlen, outlen;
    struct stat s;
    DIR* dir = opendir(path);//получая указатель на поток к папке и вставляем наш путь к ней, открывая ее
```

```
   struct dirent* ent;//структура данных в папке
   privKey_file = fopen(path3, "rb");
 if (privKey_file == NULL) {
      cout << "err:filekey do not open ";
      return -1;
   }
 privKey = PEM_read_RSAPrivateKey(privKey_file, NULL, NULL,
parole);

  fclose(privKey_file);
  int key_size = RSA_size(privKey);
  ptext = (unsigned char*)malloc(key_size);
  ctext = (unsigned char*)malloc(key_size);

   cout << key_size << endl;
   cout << ptext << endl;
   cout << ctext << endl;

   _chdir(path);//меняет рабочую директорию на указанный путь
   if (dir)//если открылось
   {

      while ((ent = readdir(dir)) != NULL)
      {
         cout << (ent->d_name) << "\n";//из структуры мы
получаем имя
         if (stat((ent->d_name), &s) == 0) {

            if (s.st_mode & S_IFDIR)//это директория
            {
               cout << "direct" << "\n"; cout <<
"_____" << endl;

            }
```

```
            else if (s.st_mode & S_IFREG)
            {
                char nome[500] = { "crpt" };//если убрать в начало,то
все ломается

                char l_pyt1[1500] = { "crpt" };//пути
                char l_pyt2[1500] = { "crpt" };
                char l_pyt3[1500] = { "crpt" };
                char l_pyt4[1500] = { "crpt" };
                cout << "file" << "\n";//it a file
                strcpy(nome, "de$");// добавляем crpt в начало
                strcat(nome, ent->d_name);//а потом приписываем
имя
                cout << nome << endl;

//______закрипт.откуда_____________________________________
_____________________________________________________________
                strcpy(l_pyt1, path);
                strcat(l_pyt1, "\\");
                strcat(l_pyt1, nome);
                cout << l_pyt1 << endl;
                //________сейв
директория_____________________________________________
_____________________________________________
                strcpy(l_pyt2, path1);
                strcat(l_pyt2, "\\");
                strcat(l_pyt2, nome);
                cout << l_pyt2 << endl;
                //________куда.
директория_____________________________________________
_____________________________________________
                strcpy(l_pyt3, path);
                strcat(l_pyt3, "\\");
                strcat(l_pyt3, ent->d_name);
                cout << l_pyt3 << endl;
```

```
                //_________копия директория_____
                strcpy(l_pyt4, path2);
                strcat(l_pyt4, "\\");
                strcat(l_pyt4, ent->d_name);
                cout << l_pyt4 << endl;
                //FILE* file;
                //FILE* ofs;
               // file = fopen(ent->d_name, "rb");//открываем для
чтения
                //ofs = fopen(nome, "wb");//для записи и шифровки
                int out = _open(nome, O_CREAT | O_TRUNC |
O_RDWR| O_BINARY | O_APPEND, 0600);
                int in = _open(ent->d_name, O_RDWR | O_BINARY);
                if (in == NULL) {//проверяем, что открылся файл
                   cout << "err:file do not open ";
                   break;
                }
                else {
                   cout << "ok" << "\n";
                }
                while (1) {//читаем файл до конца файла и
дешифруем по блокам
                   inlen = _read(in, ctext, key_size);
                   cout << inlen;
                   cout << "goood" << endl;
                   if (inlen <= 0) break;
                   cout << "goood" << endl;
                   cout << ptext << endl;
                   cout << ctext << endl;
                   outlen =  RSA_private_decrypt(inlen, ctext, ptext,
privKey, RSA_PKCS1_PADDING);
                   cout << "goood" << endl;
                   if (outlen < 0) { cout << "mistake" << endl; break; }
                   cout << "goood" << endl;
                   _write(out, ptext, outlen);
```

```
                cout << "goood" << endl;


                }
                //fclose(file);
                //fclose(ofs);

                CopyFileA(l_pyt1, l_pyt2, FALSE);//перемещаем дешифрованный файл в защищенную директорию
                CopyFileA(l_pyt3, l_pyt4, FALSE);//перемещаем файл в директорию для копии

               // DeleteFileA(l_pyt1);//удаляем лишний закрипт.файл из раб. директории
                //DeleteFileA(l_pyt3);//удаляем лишний исходный файл из раб. директории
                cout << "_____" << endl;
            }
            else {
                cout << "unknown" << "\n";//неизвестный объект
                cout << "_____" << endl;
            }

        }
    }
  }
  else
  {
    cout << "Error opening directory" << endl;
  }
  return 0;
}
```