Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

КАФЕДРА № 51

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		А.М.Буланов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Алгоритмы с открытым ключом

по курсу: Криптографические методы защиты информации

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.№	5721		Ковалева А.Е.
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

Цель работы: Изучение и программирование алгоритмов с открытым ключом.

Задание: Реализовать криптографическую систему RSA.

Требования к работе:
Система должна работать в двух режимах: шифрования и постановки подписи.
При постановке подписи использовать хеш-функцию SHA1.
В системе должна быть реализована возможность выбора длины ключа шифрования.
Ключи в системе должны генерироваться автоматически: случайным образом или по паролю.
Отчет должен содержать описание системы с указанием особенностей реализации, примеры работы программы.

Описание шифра:

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Описание алгоритма RSA

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

- если известно x, то f(x) вычислить относительно просто;
- если известно y=f(x), то для вычисления x нет простого (эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел.

Для шифрования используется операция возведения в степень по модулю большого числа.

Для дешифрования (обратной операции) за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложение числа на простые множители.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. public key), так и закрытым ключом (англ. private key). В криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их.

Открытый и закрытый ключи каждого участника обмена сообщениями в криптосистеме RSA образуют «согласованную пару» в том смысле, что они являются взаимно обратными, то есть:

$\forall$ допустимых пар открытого и закрытого ключей (p,s)

$\exists$ соответствующие функции шифрования $E_p(x)$ и расшифрования $D_s(x)$ такие, что

$\forall$ сообщения $m \in M$, где M — множество допустимых сообщений, $m = D_s(E_p(m)) = E_p(D_s(m))$.

Создание открытого и секретного ключей

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
2. Вычисляется их произведение n= pq, которое называется модулем.
3. Вычисляется значение функции Эйлера от числа n:

   $\varphi(n)=(p-1)\,(q-1)$.

4. Выбирается целое число e ($1<e<\varphi(n)$), взаимно простое со значением функции $\varphi(n)$. Обычно в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи.
   - Число e называется открытой экспонентой (англ. *public exponent*)
   - Время, необходимое для шифрования с использованием быстрого возведения в степень, пропорционально числу единичных бит в e.
   - Слишком малые значения e, например, 3, потенциально могут ослабить безопасность схемы RSA.
5. Вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению:

   $d\cdot e\equiv 1 \pmod{\varphi(n)}$.

   Примечание

   Сравнение двух целых чисел по модулю натурального числа m — математическая операция, позволяющая ответить на вопрос о том, дают ли два выбранных целых числа при делении на m один и тот же остаток. Любое целое число при делении на m дает один из m m возможных остатков: число от 0 до $m-1$.

   - Число d называется секретной экспонентой. Обычно, оно вычисляется при помощи расширенного алгоритма Евклида.
6. Пара {e,n} публикуется в качестве открытого ключа RSA (англ. RSA *public key*).
7. Пара {d,n} играет роль закрытого ключа RSA (англ. RSA *private key*) и держится в секрете.

Передача ключей

Предположим, что Боб хочет отправить Алисе информацию.

Если они решат использовать RSA, Боб должен знать открытый ключ Алисы для того чтобы зашифровать сообщение, а Алиса должна использовать свой закрытый ключ для дешифрования сообщения.

Чтобы позволить Бобу отправлять свои зашифрованные сообщения, Алиса передает свой открытый ключ Бобу через надежный, но не обязательно секретный маршрут.

Закрытый ключ Алисы никогда никому не передается.

1 Шифрование

Предположим, Боб хочет послать Алисе сообщение m.

Сообщениями являются целые числа в интервале от 0 до n−1, то есть m∈Zn.

Алгоритм:

- Взять открытый ключ (e,n) Алисы
- Взять открытый текст m
- Зашифровать сообщение с использованием открытого ключа Алисы: $c = E(m) = m^e \pmod{n}$

2 Дешифрование

Алгоритм:

- Принять зашифрованное сообщение c
- Взять свой закрытый ключ (d,n)
- Применить закрытый ключ для дешифрования сообщения:

$$m = D(c) = c^d \pmod{n}$$

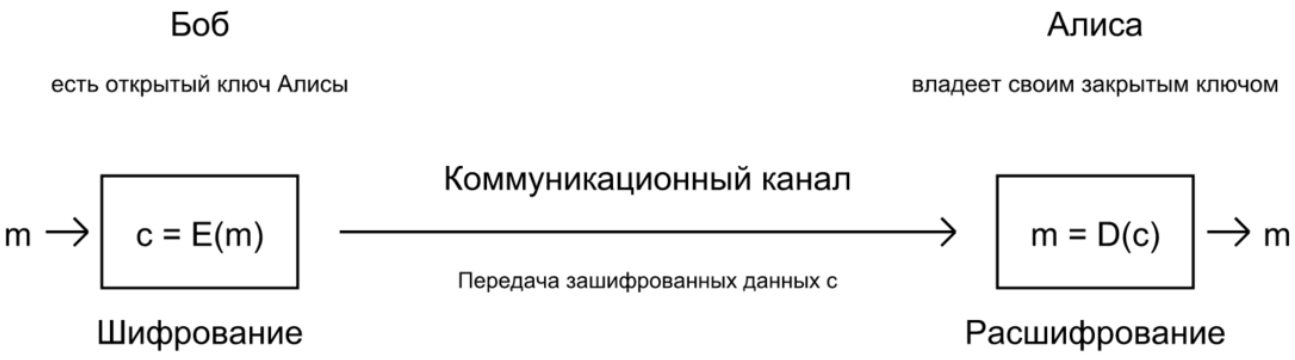


Рисунок 1 – Алгоритм шифрования и дешифрования

Цифровая подпись

Система RSA может использоваться не только для шифрования, но и для цифровой подписи.

Предположим, что Алисе (стороне A) нужно отправить Бобу (стороне B) сообщение m, подтверждённое электронной цифровой подписью.

Рисунок 2 – Схема цифровой подписи

Алгоритм:

1 Взять открытый текст m

2 Создать цифровую подпись s с помощью своего секретного ключа $\{d, n\}$: $s = S_A(m) = m^d \mod n$.

3 Передать пару $\{m, s\}$,состоящую из сообщения и подписи.

Алгоритм:

1 Принять пару $\{m, s\}$.

2 Взять открытый ключ Алисы $\{e, n\}$.

3 Вычислить прообраз сообщения из подписи:

$m' = P_A(s) = s^e \mod n$

4 Проверить подлинность подписи

(и неизменность сообщения),

сравнив m и m'

Поскольку цифровая подпись обеспечивает как аутентификацию автора сообщения, так и подтверждение целостности содержимого подписанного сообщения, она служит аналогом подписи, сделанной от руки в конце рукописного документа.

Важное свойство цифровой подписи заключается в том, что её может проверить каждый, кто имеет доступ к открытому ключу её автора. Один из участников обмена сообщениями после проверки подлинности цифровой подписи может передать подписанное сообщение ещё кому-то, кто тоже в состоянии проверить эту подпись. Например, сторона A может переслать стороне B электронный чек. После того как сторона B проверит подпись стороны A на чеке, она может передать его в свой банк, служащие которого также имеют возможность проверить подпись и осуществить соответствующую денежную операцию.

Заметим, что подписанное сообщение m не зашифровано. Оно пересылается в исходном виде и его содержимое не защищено от нарушения конфиденциальности. Путём совместного применения представленных выше схем шифрования и цифровой подписи в системе RSA можно создавать сообщения, которые будут и зашифрованы, и содержать цифровую подпись. Для этого автор сначала должен добавить к сообщению свою цифровую подпись, а затем — зашифровать получившуюся в результате пару (состоящую из самого сообщения и подписи к нему) с помощью открытого ключа, принадлежащего получателю. Получатель расшифровывает полученное сообщение с помощью своего секретного ключа[18]. Если проводить аналогию с пересылкой обычных бумажных документов, то этот процесс похож на то, как если бы автор документа поставил под ним свою печать, а затем положил его в бумажный конверт и запечатал, с тем чтобы конверт был распечатан только тем человеком, кому адресовано сообщение.

Описание программы:

Для генерации ключей был реализован класс RSA, в котором хранятся необходимые для работы алгоритма переменные, а также реализованы следующие методы:

1. Переменный под два различных случайных простых числа p и q заданного размера
   BigInteger p = null;
   BigInteger q = null;

2. Переменная под модуль (*n=p*q)*
   BigInteger n = p.multiply(q);

3. Переменная под m = (p-1) (q-1)
   BigInteger m = (p.subtract(BigInteger.*ONE*)).multiply(q.subtract(BigInteger.*ONE*));

4. Переменная под e
   BigInteger e = *getKey*(m);

5. Переменная под d как модульная инверсию e и m
   BigInteger d = e.modInverse(m);

6. Метод для теста Ферма на примитивность числа
   public boolean fermatest (BigInteger b)

7. Метод генерации e на основе m
   public static BigInteger getKey(BigInteger m)

8. Метод для печати ключей в файл
   public void printToFile(String file, BigInteger a, BigInteger b)

Для реализации подписи был реализован класс RSAsign, в котором хранятся необходимые для работы алгоритма переменные, а также реализованы следующие методы:

1. Метод для преобразования байтового массива в шестнадцатеричную строку
   private static String convertByteArrayToHexString(byte[] arrayBytes)

2. Метод для преобразования файла в массив байтов
   private static byte[] hashFile(String file, String algorithm) throws Exception

3. При постановке подписи используем хеш-функцию SHA-2
   fileBytes = *hashFile*(filePath, "SHA-256");

4. Читаем открытый ключ
   BufferedReader br1 = new BufferedReader(new FileReader("e_n.txt"));

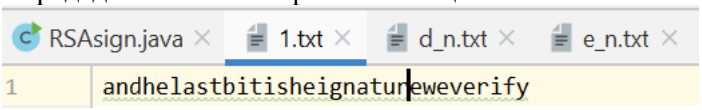
5. Читаем закрытый ключ
   br1 = new BufferedReader(new FileReader("d_n.txt"));

6. Метод шифрования
   origBig = origBig.modPow(e, n);

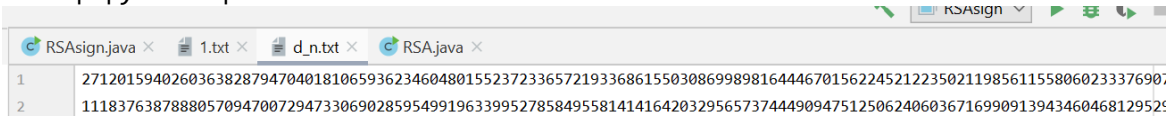
7. Метод дешифрования
   message = message.modPow(d, n);

Примеры:

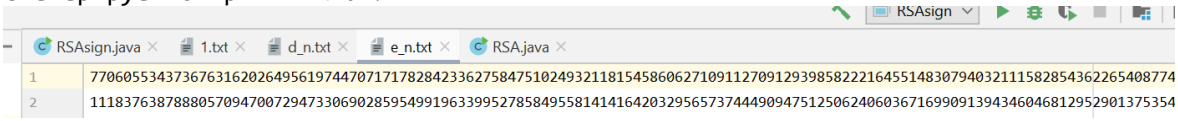
1. Передадим текстовый файл с сообщением:



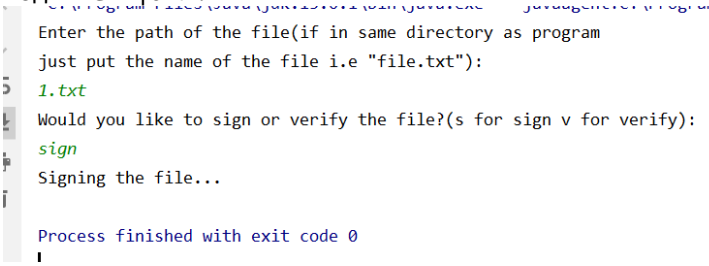
2. Сгенерируем закрытый ключ:



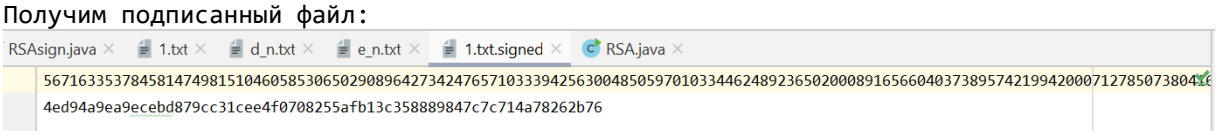
3. Сгенерируем открытый ключ:



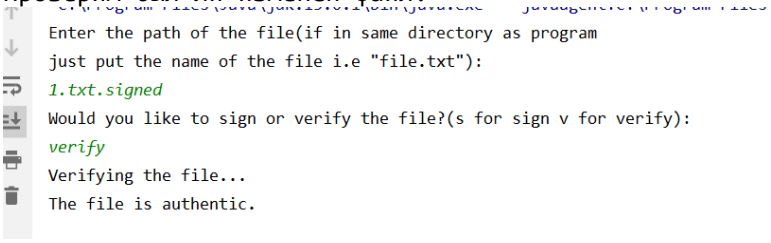
4. Подпишем файл:



5. Получим подписанный файл:



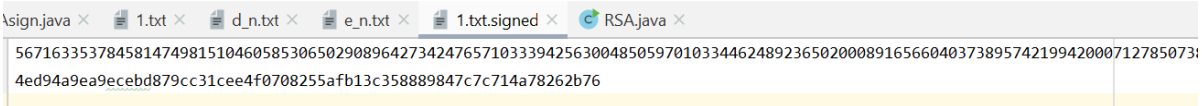
6. Проверим был ли изменен файл:



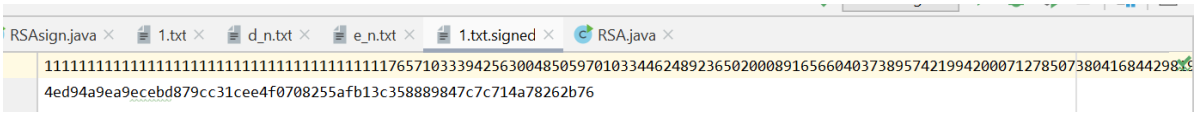
(файл изменен не был)

7. Изменим символы в подписанном файле:

До:



После:



Получили сообщение, что в файле были допущены изменения:

```
Enter the path of the file(if in same directory as program
just put the name of the file i.e "file.txt"):
1.txt.signed
Would you like to sign or verify the file?(s for sign v for verify):
verify
Verifying the file...
The file has been modified.

Process finished with exit code 0
```

(Получили сообщение, что в файле были допущены изменения)

Выводы:

Был реализована криптографическая система RSA.
Система работает в двух режимах: шифрования и постановки подписи.
При постановке подписи использована хеш-функция SHA2.
В системе реализована возможность выбора длины ключа шифрования.