Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 3
по дисциплине «Математические основы защиты информации»
ШИФР ВИЖЕНЕРА

Студент гр. БИБ 191
_____И.Г. Тюрин
"_"_____2021 г.

"
Руководитель
Заведующий кафедрой информационной безопасности киберфизических систем
канд. техн. наук, доцент
_____О.О. Евсютин
"_"_____2021 г.

Москва 2021

СОДЕРЖАНИЕ

1. Задание на практическую работу

Целью данной работы является исследование асимметричной криптосистемы RSA, основанной на проблеме факторизации целых чисел. Задание:

1) написать программную реализацию криптосистемы RSA;

2) изучить методы криптоанализа криптосистемы RSA;

3) реализовать (вручную или программно) не менее одной атаки на криптосистему RSA для случая, когда параметры криптосистемы не являются большими числами;

4) подготовить отчет о выполнении работы.

Программа должна обладать следующей функциональностью:

1) принимать на вход произвольную последовательность символов, вводимую пользователем в качестве открытого текста или шифртекста;

2) принимать на вход ключевую пару (открытый ключ, закрытый ключ);

3) давать пользователю возможность сгенерировать ключевую пару;

4) осуществлять зашифрование или расшифрование введенного текста по выбору пользователя.

2. Краткая теоретическая часть

2.1. Криптография с открытым ключом

Концепция криптографии с открытым ключом появилась в середине 1970-х годов как возможное решение проблемы распределения ключей шифрования, актуальной для симметричных алгоритмов шифрования.

Криптосистемы с открытым ключом, также называемые асимметричными криптосистемами, используют два разных ключа: открытый ключ зашифрования и закрытый ключ расшифрования. Открытый ключ в общем случае доступен всем желающим, а закрытый ключ известен только законному владельцу. Оба ключа связаны между собой некоторой математической зависимостью. При этом данная зависимость такова, что, зная один ключ, вычислить другой практически невозможно.

Первые криптографические алгоритмы с открытым ключом основаны на задаче факторизации целых чисел.

2.2. Криптосистема RSA

Данная криптосистема является первой криптосистемой с открытым ключом. Она основывается на сложности проблемы факторизации целых чисел, то есть чисел на простые множители.

Алгоритм генерации ключей.

1. Пользователь A генерирует два больших простых числа p и q, отличных друг от друга. При этом $|p - q|$ – большое число, хотя p и q имеют приблизительно одинаковый битовый размер.

2. Держа p и q в секрете, Пользователь A вычисляет их произведение $n = pq$, которое называют модулем алгоритма.

3. Пользователь A вычисляет значение функции Эйлера для n по формуле $\varphi(n) = (p - 1)(q - 1)$.

4. Пользователь A выбирает целое число e, взаимно простое со значением функции $\varphi(n)$. Это число называется экспонентой зашифрования.

5. Пользователь A применяет расширенный алгоритм Евклида к паре чисел e и $\varphi(n)$ и вычисляет значение d, удовлетворяющее соотношению $ed \equiv 1 \bmod \varphi(n)$. Это значение называется экспонентой расшифрования.

6. Пара (e, n) публикуется в качестве открытого ключа пользователя A, d является закрытым ключом и держится в секрете.

Алгоритм зашифрования.

1. Пользователь B получает аутентичную копию открытого ключа пользователя A – пару (e, n).

2. Пользователь B представляет сообщение в виде числа m, меньшего модуля алгоритма. В общем случае сообщение может быть разбито на блоки, каждый из которых представляется своим числом.

3. Пользователь B вычисляет $c = m^e \pmod n$.

4. Зашифрованное сообщение отправляется пользователю A.

Алгоритм расшифрования.

Московский институт электроники и математики им. А.М. Тихонова

1. Пользователь A получает криптограмму c от пользователя B.

2. Пользователь A вычисляет $m = c^d \pmod n$.

2.3. Нахождение обратного элемента по модулю простого числа

Для нахождения обратного элемента по модулю натурального числа, что необходимо при вычислении экспоненты расшифрования, применяется расширенный алгоритм Евклида.

Вход: целые числа $a \geq b > 0$.

Выход: $d = НОД(a, b)$ и целые x, y, такие, что $ax + by = d$.

1. Полагаем $x2 \leftarrow 1$, $x1 \leftarrow 0$, $y2 \leftarrow 0$, $y1 \leftarrow 1$.

2. Пока $b > 0$, выполнять следующее:

2.1. $q \leftarrow [a/b]$, $r \leftarrow a - qb$, $x \leftarrow x2 - qx1$, $y \leftarrow y2 - qy1$;

2.2. $a \leftarrow b$, $b \leftarrow r$, $x2 \leftarrow x1$, $x1 \leftarrow x$, $y2 \leftarrow y1$, $y1 \leftarrow y$.

3. d ← a, x ← x2, y ← y2 и возврат (d, x, y).

Чтобы найти $a^{-1} \bmod n$, необходимо подать на вход алгоритма Евклида пару n a, и если НОД(a, n) = 1, вернуть в качестве a^{-1} значение y2.

2.4. Нахождение обратного элемента по модулю простого числа

Криптографические алгоритмы с открытым ключом при их использовании на практике оперируют числами большой битовой длины (или просто большими числами), когда речь идет о сотнях и тысячах бит. Для некоторых операцией над такими числами созданы специальные алгоритмы. В случае криптосистемы RSA необходимо иметь алгоритм, который позволит осуществлять быстрое возведение в степень по модулю.

Данный алгоритм представлен ниже.

Алгоритм возведения в степень по модулю.

Вход: a, $k \in Zn$, $k = \sum ki \cdot 2\ t\ i\ i=0$

Выход: $a^k \bmod n$.

1. b ← 1. Если k = 0, то переход к шагу 5.

2. A ← a.

3. Если $k_0 = 1$, то b ← a.

4. Для i = (1, t) выполняем следующее:

4.1. $A \leftarrow A^2 \bmod n$.

4.2. Если $k_i = 1$, то $b \leftarrow (A \cdot b) \bmod n$.

5. Возврат b.

Еще одним важным аспектом криптографии с открытым ключом является использование простых чисел, в частности, как было рассмотрено, в криптосистеме RSA элементом открытого ключа является произведение двух больших простых чисел n = pq

Наиболее развитые вероятностные алгоритмы проверки чисел на простоту основаны на малой теореме Ферма.

Малая теорема Ферма.

Пусть p — простое число, $a \neq 0$ и $a \in Zp$. Тогда $a^{p-1} \equiv 1 \pmod p$.

Соотношение, приведенное в теореме, используется в тесте, проверяющем, являетсяли заданное число составным. Этот тест называют тестом Ферма.

Тест Ферма.

Вход: нечетное число n.

Выход: ответ на вопрос «является ли n простым».

1. Для i = (1, t) выполняем следующее:

1.1. Выбираем случайное целое число $a \in [2; n - 1]$.

1.2. Вычисляем $r = a^{n-1} \bmod n$ с помощью алгоритма возведения в степень по модулю.

1.3. Если $r \neq 1$, то возврат «n — составное».

Тест Ферма по основанию a определяет простоту n с вероятностью 1/2, после t итераций вероятность ошибки составляет $1/2^t$.

3. Примеры шифрования

3.1. Шифрование по буквам

В качестве примера рассмотрим небольшие числа, но на практике используют числа длины 1024 и больше. Будем передавать сообщение: «ARTISTICALLY». Представим открытый текст в виде последовательности чисел из диапазона 1...26.

Открытый текст: ARTISTICALLY

Закодированный текст: [1, 18, 20, 9, 19, 20, 9, 3, 1, 12, 12, 25]

Пусть p = 199, q = 167. Тогда N = 33233, а функция Эйлера φ(n) = (p – 1) (q – 1) = 32868. В качестве шифрующей экспоненты выбирают число, взаимно простое с φ(n). Положим, E = 65537: НОД(65537, 32868) = 1. Тогда d = 31877, т.к. d – обратный к 65537 элемент в кольце классов вычетов по модулю N.

Открытый ключ: (N, E) = (33233, 65537).
Секретный ключ: (d, p, q) = (31887, 199, 167).

Зашифрование

Приступаем к зашифрованию сообщения m по формуле: $C_i = m_i{}^E \ (mod\ N)$.

C1 = 1^{65537} (mod 33233) = 1
C2 = 18^{65537} (mod 33233) = 60
C3 = 20^{65537} (mod 33233) = 6
C4 = 9^{65537} (mod 33233) = 9
C5 = 19^{65537} (mod 33233) = 33
C6 = 20^{65537} (mod 33233) = 6
C7 = 9^{65537} (mod 33233) = 9
C8 = 3^{65537} (mod 33233) = 3
C9 = 1^{65537} (mod 33233) = 1
C10 = 12^{65537} (mod 33233) = 12
C11 = 12^{65537} (mod 33233) = 12
C12 = 25^{65537} (mod 33233) = 2

Шифртекст: [1, 60, 9, 9, 33, 6, 9, 3, 1, 12, 12, 25]

Расшифрование

Расшифровать шифртекст можно, обладая секретным ключом. Расшифрование производится по формуле: $m_i = C_i^d \ (mod\ N)$.

m1 = 1^{31877} (mod 33233) = 1

m2 = 60^{31877} (mod 33233) = 18

m3 = 6^{31877} (mod 33233) = 20

m4 = 9^{31877} (mod 33233) = 9

m5 = 33^{31877} (mod 33233) = 19

m6 = 6^{31877} (mod 33233) = 20

m7 = 9^{31877} (mod 33233) = 9

m8 = 3^{31877} (mod 33233) = 3

m9 = 1^{31877} (mod 33233) = 1

m10 = 12^{31877} (mod 33233) = 12

m11 = 12^{31877} (mod 33233) = 12

m12 = 25^{31877} (mod 33233) = 25

Получим исходный текст:

Полученный текст: [1, 18, 20, 9, 19, 20, 9, 3, 1, 12, 12, 25]

Переведём обратно в текст: «ARTISTICALLY».

4. Программная реализация шифров

Результаты работы программы для рассмотренного примера:

1) Шифр RSA

Открытый текст: ARTISTICALLY

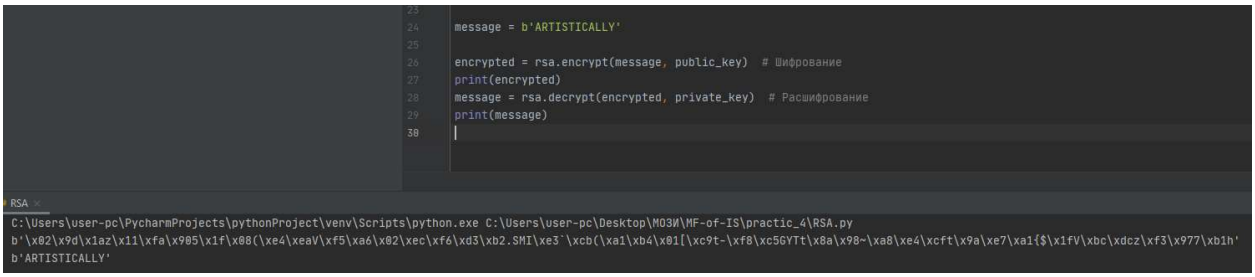

Рисунок 4.1 – Результаты работы Алгоритма, реализующего шифр RSA

Ссылка на код:

5. Криптоанализ шифра RSA

Атака на RSA (атака Виенера, Wiener attack)

Теоретическая часть

Рассматривается открытый RSA ключ ***(e, N),*** по которому необходимо определить акрытую экспоненту ***d***. Если известно, что$d<1/3N^{1/4}$, то это возможно сделать по следующему алгоритму:

1. Разложить дробь ***e/N*** в непрерывную дробь [a1, a2, …].
2. Для непрерывной дроби [a1, a2, …]. найти множество всех возможных подходящих дробей k_n/d_n.
3. Исследовать подходящую дробь k_n/d_n.
    3.1. Определить возможное значение $\phi(N)$, вычислив $(ed_n-1)/k_n$.
    3.2. Решив уравнение $x^2-((N-f_n)+1)x+N=0$, получить пару корней (p_n, q_n).
4. Если для пары корней (p_n, q_n) выполняется равенство $N = p_n*q_n$, то закрытая экспонента найдена $d = d_n$

Если условие не выполняется или не удалось найти пару корней, то необходимо исследовать следующую подходящую дробь, вернувшись к шагу 3.

Практическая часть

Ссылка на код:

Пример атаки Виенера можно реализовать следующим образом:

Рисунок 5.1 – Атака Виенера на шифр RSA с малыми параметрами (16 бит)

6. Выводы о проделанной работе

В результате выполнения данной работы я могу подвести следующие итоги:

Криптосистема RSA, первая криптосистема асимметричного шифрования, выдержала испытание временем и по сей день активно используются во всем мире. Это можно считать аргументом в пользу криптостойкости данной системы. Хотя аргумент этот слабый, ведь даже если задача RSA эквивалентна проблеме факторизации (а это утверждение не доказано), то с развитием вычислительной техники разумным решением будет отказаться от этой криптосистемы.

Кроме того, даже с настоящим положением вычислительной мощности компьютеров необходимо быть на чеку и тщательно подбирать параметры шифрования. Изучив методы криптоанализа системы RSA, я могу сделать следующие выводы:

1) Нельзя использовать общий модуль n для группы пользователей

2) Сообщения перед шифрованием необходимо дополнять случайными числами

3) Показатель дешифрования должен быть большим.

7. Список использованных источников

1. Традиционные шифры с симметричным ключом [электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Атака_Винера
2. Cryptanalysis of the Hill Cipher [электронный ресурс] – URL: http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-hill-cipher/
3. Хилл шифр – Hill cipher [электронный ресурс] – URL: https://ru.qaz.wiki/wiki/Hill_cipher

ПРИЛОЖЕНИЕ А.

Шифр Виженера

```
def encryptText(text, key, abc):
    return ''.join([abc[(abc.index(j) + abc.index(key[i])) % 26 +
abc.index('A')] for i, j in enumerate(text)])


def decrtptText(decodedText, key, abc):
    return ''.join([abc[(abc.index(j) - abc.index(key[i])) % 26 +
abc.index('A')] for i, j in enumerate(decodedText)])


def createKey(key, text, myType, abc):
    l = len(text)
    k = len(key)

    if myType == 'repeat':
        res = key * int((l // k + 1))
    elif myType == 'open':
        res = key[0] + text[:-1]
    elif myType == 'cipher':
        key = key[:1]
        for j, i in enumerate(text):
            c = abc[(abc.index(i) + abc.index(key[j])) % len(abc)]
            key += c
        res = key

    return res


abc = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

text = 'SOMETEXT'
key = 'KEY'
myType = 'cipher'  # repeat | open | cipher

k = createKey(key, text, myType, abc)

encryptedText = encryptText(text, k, abc)
print('Encrypted text:', encryptedText)
print('Decrypted text:', decrtptText(encryptedText, k, abc))
```