Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

Отчёт о практической работе №4

по дисциплине «Математические основы защиты информации»

«Криптосистема RSA»

Студент гр. БИБ202

Шадрунов А. С.

«10» июня 2022 г.

Руководитель

Заведующий кафедрой информационной безопасности киберфизических систем

канд. техн. наук, доцент

О. О. Евсютин

«___» __________ 2022 г.

Москва, 2022

Содержание

1 Задание на практическую работу **3**

2 Краткая теоретическая часть **4**

2.1 Криптография с открытым ключом 4

2.2 Криптосистема RSA 4

2.3 Криптоанализ системы RSA 7

2.3.1 Бесключевое чтение RSA 7

3 Примеры шифрования **8**

3.1 Зашифрование 8

3.2 Расшифрование 9

4 Программная реализация криптосистемы RSA **10**

4.1 Генерация ключей 10

4.2 Зашифрование 10

4.3 Расшифрование 10

5 Примеры криптоанализа **13**

6 Выводы о проделанной работе **14**

Список использованных источников **15**

1. Задание на практическую работу

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

В рамках практической работы необходимо выполнить следующее:

1. Написать программную реализацию криптосистемы RSA;
2. Изучить методы криптоанализа криптосистемы RSA;
3. Реализовать (вручную или программно) не менее одной атаки на криптосистему RSA, исключая наивную переборную атаку, для случая, когда параметры криптосистемы не являются большими числами;
4. Подготовить отчет о выполнении работы.

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. Отправной точкой для развития этой области можно считать публикацию статьи М. Хеллмана и У. Диффи "Новые направления в криптографии" [1].

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. Примеры других сложных вычислительных задач — вычисление логарифма в конечном поле и вычисление корней алгебраических уравнений.

2.2. Криптосистема RSA

Данная криптосистема является первой криптосистемой с открытым ключом. Она основывается на сложности проблемы факторизации целых чисел, то есть разложении 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 $n = p \cdot q$, которое называют *модулем алгоритма*.

3. Пользователь A вычисляет значение функции Эйлера для n по формуле:

$$\phi(n) = (p - 1)(q - 1).$$

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

5. Пользователь A применяет расширенный алгоритм Евклида к паре чисел e и $\phi(n)$ и вычисляет значение d, удовлетворяющее соотношению $e{\cdot}d \equiv 1 \, mod \, \phi(n)$. 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 $c = m^e \; mod \; n$.

4. Зашифрованное сообщение отправляется пользователю A.

Алгоритм расшифрования

1. Пользователь A получает криптограмму c от пользователя B.

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

Замечание. Для нахождения обратного элемента по модулю натурального числа применяется расширенный алгоритм Евклида.

Алгоритмы работы с большими числами

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: $a, k \in \mathbf{Z_n}, k = \sum\limits_{i=0}^{t} k_i \cdot 2^i$.
- Выход: $a^k \; mod \, n$.

1. $b = 1$. Если $k == 0$, то переход к шагу 5.
2. $A = a$.
3. Если $k_0 == 1$, то $b = a$.
4. Для $i = \overline{1, t}$:
   - $A = A^2 \; mod \, n$.
   - Если $k_i == 1$, то $b = (A \cdot b) \; mod \, n$.
5. Вернуть b.

Простые числа

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

Малая теорема Ферма

Пусть p — простое число, $a \neq 0$ и $a \in \mathbf{Z_p}$. Тогда $a^{p-1} \equiv 1 \; mod \, p$.

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

Тест Ферма

- Вход: нечетное число n
- Выход: n — простое?

1. Для $i = \overline{1, t}$:
   - Выбираем случайное целое число $a \in [2; n-1]$.
   - Вычисляем $r = a^{n-1} \; mod \, n$ с помощью алгоритма возведения в степень по модулю.

– Если $r \neq 1$, то вернуть `False`.

2. Вернуть `True`.

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

2.3. Криптоанализ системы RSA

Криптосистема RSA является достаточно стойкой, если в качестве ключа используются достаточно большие числа. В 2009 году исследователям удалось дешифровать сообщение, зашифрованное при помощи криптографического ключа стандарта RSA длиной 768 бит [2]. Для этого были задействованы значительные вычислительные ресурсы.

Более выполнимыми оказываются атаки на RSA с неправильным подбором параметров (так называемые атака Хастада, атака Франклина-Рейтера) [3].

2.3.1. Бесключевое чтение RSA

Рассмотрим атаку методом бесключевого чтения RSA [4]. Злоумышленник известны открытый ключ (e, n) и шифротекст c. Злоумышленник должен найти такое число j, что $c^{e^j} \equiv c \ mod \ n$. Затем злоумышленник вычисляет $c^{e^{j-1}} \ mod \ n$. Утверждается, что это и есть открытый текст m. В самом деле, согласно алгоритму зашифрования, $c = m^e \ mod \ n$. Видно, что $(c^{e^{j-1}})^e \ mod \ n = c$, и значит $c^{e^{j-1}}$ и есть m.

3. Примеры шифрования

3.1. Зашифрование

Зашифруем слово "Shadrunov" с помощью криптосистемы RSA. Представим его в битовом виде с помощью таблицы ASCII.

$$X = \text{Shadrunov} = \begin{pmatrix} 083 & 104 & 097 & 100 & 114 & 117 & 110 & 111 & 118 \end{pmatrix} =$$
$$= (1010011 \mid 1101000 \mid 1100001 \mid 1100100 \mid 1110010 \mid 1110101 \mid 1101110 \mid 1101111 \mid 1110110)$$

Выберем параметры шифрования:

1. Пара простых чисел: $p = 31, q = 37$;
2. Модуль алгоритма: $n = p \cdot q = 31 \cdot 37 = 1147$;
3. Значение функции Эйлера: $\phi(n) = (31 - 1)(37 - 1) = 1080$;
4. Экспонента зашифрования: $e = 17$;
5. Экспонента расшифрования: $17 \cdot d \equiv 1 \; mod \; 1080, d = 953$;

Открытый ключ $(e, n) = (17, 1147)$, закрытый ключ $(d, n) = (953, 1147)$.

Рассчитаем длину блока: $\lfloor \log_2 1147 \rfloor = 10$. Разобьём открытый текст на блоки: 0000000101 | 0011110100 | 0110000111 | 0010011100 | 1011101011 | 1011101101 | 1111110110 Теперь переведём каждый блок в десятичный вид: $(5 \mid 244 \mid 391 \mid 156 \mid 747 \mid 749 \mid 1014)$. Зашифруем каждый блок по формуле $c_i = m^e \; mod \; n$:

- $c_1 = 5^{17} \; mod \; 1147 = 614$
- $c_2 = 244^{17} \; mod \; 1147 = 449$
- $c_3 = 391^{17} \; mod \; 1147 = 733$
- $c_4 = 156^{17} \; mod \; 1147 = 652$
- $c_5 = 747^{17} \; mod \; 1147 = 53$
- $c_6 = 749^{17} \; mod \; 1147 = 366$
- $c_7 = 1014^{17} \; mod \; 1147 = 291$

Выходная битовая последовательность: $(1001100110 \mid 0111000001 \mid 1011011101 \mid 1010001100 \mid 0000110101 \mid 0101101110 \mid 0100100011)$. Переведём в ASCII: $(76 \mid 103 \mid 3 \mid 55 \mid 52 \mid 48 \mid 26 \mid 86 \mid 114 \mid 35) = (L\, g\, ETX\, 7\, 4\, 0\, DLE\, V\, r\, \#)$. Видно, что в результате появились не только символы английского алфавита, но и непечатные символы.

3.2. Расшифрование

Проделаем обратное преобразование ASCII-символов в битовую последовательность. Разобьём её на блоки по 10 бит. Каждый блок представим в виде десятичного числа (элемента кольца классов вычетов по модулю n). Получим: $(614 \mid 449 \mid 733 \mid 652 \mid 53 \mid 366 \mid 291)$. Теперь каждый элемент шифртекста расшифруем по формуле $m_i = c^d \, mod \, n$:

- $m_1 = 614^{953} \, mod \, 1147 = 5$
- $m_2 = 449^{953} \, mod \, 1147 = 244$
- $m_3 = 733^{953} \, mod \, 1147 = 391$
- $m_4 = 652^{953} \, mod \, 1147 = 156$
- $m_5 = 53^{953} \, mod \, 1147 = 747$
- $m_6 = 366^{953} \, mod \, 1147 = 749$
- $m_7 = 291^{953} \, mod \, 1147 = 1014$

Выходная числовая последовательность: $(5 \mid 244 \mid 391 \mid 156 \mid 747 \mid 749 \mid 1014)$. Она совпадает с числовой последовательностью при зашифровании, то есть при представлении в виде ASCII-символов получаем "Shadrunov".

4. Программная реализация криптосистемы RSA

Опишем особенности программной реализации криптосистемы RSA. Реализация на языке Python **доступна по ссылке** на GitHub.

4.1. Генерация ключей

На вход программа получает длину простых чисел в битах. На выходе программа выдаёт параметры ключей: e, n, d, p, q. В качестве генератора простых чисел используется случайная битовая последовательность, которая затем проверяется на простоту с помощью теста Ферма [5].

Примеры генерации ключей представлен на рисунке 1.

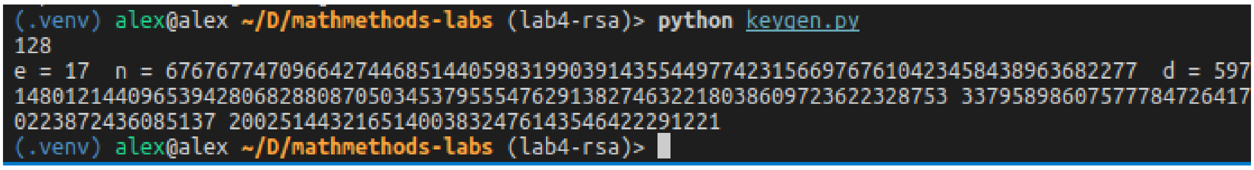

Рис. 1 – Пример генерации ключей

4.2. Зашифрование

На вход программа получает открытый текст (поддерживаются символы ASCII), открытый ключ (e, n). На выходе программа выдаёт шифртекст в виде символов ASCII, числа в десятичном и двоичном представлении. Важно учесть, что после зашифрования длина блока может превышать исходную, поэтому для хранения битовой записи блоков шифртекста используем увеличенную длину.

Пример зашифрования ручного примера представлен на рисунке 2.

4.3. Расшифрование

На вход программа получает шифртекст в виде символов ASCII или битовой последовательности (не все символы ASCII можно набрать на клавиатуре), закрытый ключ (d, n). На выходе программа выдаёт открытый текст в виде символов ASCII, числа в десятичном и двоичном представлении. После расшифрования длину блока можно уменьшить на один, чтобы получить исходные последовательности.

Пример расшифрования ручного примера представлен на рисунке 2.

```
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python encrypt.py
type your message: Shadrunov
enter e: 17
enter n: 1147
plain bits:  101001111010001100001110010011100101110101110111011011111110110
S | 83 | 1010011
h | 104 | 1101000
a | 97 | 1100001
d | 100 | 1100100
r | 114 | 1110010
u | 117 | 1110101
n | 110 | 1101110
o | 111 | 1101111
v | 118 | 1110110
plain numbers:  [5, 244, 391, 156, 747, 749, 1014]
ciphertext:  'L''g''\x03''7''4''0''\x1a''V''r''#'
cipher bits:  1001100110011100000110110111011010001100000011010101011011100100100011
cipher numbers:  [614, 449, 733, 652, 53, 366, 291]
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python decrypt.py
Bits / Text?
b
type your bits (0 & 1): 1001100110011100000110110111011010001100000011010101011011100100100011
enter d: 953
enter n: 1147
1001100110011100000110110111011010001100000011010101011011100100100011
1001100110 | 614 | 5 | 0000000101
0111000001 | 449 | 244 | 0011110100
1011011101 | 733 | 391 | 0110000111
1010001100 | 652 | 156 | 0010011100
0000110101 | 53 | 747 | 1011101011
0101101110 | 366 | 749 | 1011101101
0100100011 | 291 | 1014 | 1111110110
plain text:  '\x00''S''h''a''d''r''u''n''o''v'
plain bits:  0000000101001111010001100001110010011100101110101110111011011111110110
plain numbers:  [5, 244, 391, 156, 747, 749, 1014]
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)>
```

Рис. 2 – Пример шифрования ручного примера

Пример шифрования более длинной последовательности представлен на рисунках 3—4.

```
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa) [0|SIGINT]> python keygen.py
20
e = 5  n = 382800455761  d = 153119687213 604957 632773
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python encrypt.py
type your message: A little party never killed nobody
enter e: 5
enter n: 382800455761
plain bits:  1000001010000001101100110100111101001110100110110011001010100000011100000110000111100101110100111100101
0000011011101100101111011011001011110010010000011010111101001110110011011001100101110010001000001101110110111110
0010110111111001001111001
A | 65 | 1000001
  | 32 | 0100000
l | 108 | 1101100
i | 105 | 1101001
t | 116 | 1110100
t | 116 | 1110100
l | 108 | 1101100
e | 101 | 1100101
  | 32 | 0100000
p | 112 | 1110000
a | 97 | 1100001
r | 114 | 1110010
t | 116 | 1110100
y | 121 | 1111001
  | 32 | 0100000
n | 110 | 1101110
e | 101 | 1100101
v | 118 | 1110110
e | 101 | 1100101
r | 114 | 1110010
  | 32 | 0100000
k | 107 | 1101011
i | 105 | 1101001
l | 108 | 1101100
l | 108 | 1101100
e | 101 | 1100101
d | 100 | 1100100
  | 32 | 0100000
n | 110 | 1101110
o | 111 | 1101111
b | 98 | 1100010
o | 111 | 1101111
d | 100 | 1100100
y | 121 | 1111001
block_length:  38 238 True 28
00000000000000000000000000001000001010 | 522 | 94364761771 | 0010101111110001001001111111111010101011
00001101100110100111101001110100110110 | 14606572854 | 363640369354 | 1010100101010101010011110110000110 01010
01100101010000001110000011000011110010 11 | 108723749835 | 139952514087 | 0100000100101011101000111100100001001 11
10100111100101000000110111011001011110 1 | 179936095421 | 214522622390 | 0110001111100101000101101110001101 10110
10110010111100100100000110101111010011 | 192142142419 | 145793217374 | 0100001111100011111001111110011010 11110
10110011011001100101110010001000001101 | 192629121549 | 36430532046 | 000100001111011011011011100110111001110
11011011111100010110111111001001111001 | 236162118265 | 109359690251 | 0011001011101100101100001110010000 01011
plain numbers:  [522, 14606572854, 108723749835, 179936095421, 192142142419, 192629121549, 236162118265]
ciphertext:  '\x15''|''$''\x7f''j'']''%''*''O''0''e''\x10''%'':''\x1e''!''\x1d''G''e''\x0b''8''m''H''>''\x1f''\x1
f''M''<''\x10''{''6''s''9''c''\x17''2''a''d''\x0b'
cipher bits:  0010101111110001001001111111111010101011101010010101010101001111011000011001010010000001001010111 0100
01111001000010011101100011111001010001011011100011011011001000011111000111110011111100110101111000010000111101101
1011011100110111001110001100101110110010110000111001000001011
cipher numbers:  [94364761771, 363640369354, 139952514087, 214522622390, 145793217374, 36430532046, 109359690251]
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)>
```

Рис. 3 – Генерация ключа и зашифрование длинной последовательности

```
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python decrypt.py
Bits / Text?
b
type your bits (0 & 1): 0010101111110001001001111111111010101011101010010101010101001111011000011001010010000001001
0101110100011110010000100111011000111110010100010110111000110110110010000111110001111100111111001101011110000 1000
01111011011011011100110111001110001100101110110010110000111001000001011
enter d: 153119687213
enter n: 382800455761
block_length:  39 False 39
0010101111110001001001111111111010101011101010010101010101001111011000011001010010000001001010111010001111001000010
0111011000111110010100010110111000110110110010000111110001111100111111001101011110000100001111011011011011100110 1
1100111000110010111011001011000011100100000 1011
001010111111000100100111111111101010101 1 | 94364761771 | 522 | 000000000000000000000000000001000001010
101010010101010101001111011000011001010 | 363640369354 | 14606572854 | 000011011001101001111010011101001101 10
010000010010101110100011110010000100111 | 139952514087 | 108723749835 | 011001010100000011100000110000111100101 1
011000111110010100010110111000110110110 | 214522622390 | 179936095421 | 101001111001010000001101110110010111101
010000111110001111100111111001101011110 | 145793217374 | 192142142419 | 101100101111001001000001101011110100 11
000100001111011011011011100110111001110 | 36430532046 | 192629121549 | 101100110110011001011100100010000 01101
001100101110110010110000111001000001011 | 109359690251 | 236162118265 | 110110111111000101101111110010011110 01
plain text:  '\x00''\x00''\x00''\x00''A'' ''l''i''t''t''l''e'' ''p''a''r''t''y'' ''n''e''v''e''r'' ''k''i''l''l''
e''d'' ''n''o''b''o''d''y'
plain bits:  00000000000000000000000000001000001010000001101100110100111101001110100110110011001010100000011100000 11
000011110010111010011110010100000011011101100101111011011001011110010010000011010111101001110110011011001100101110
0100010000001101110110111111000101101111110010011110 01
plain numbers:  [522, 14606572854, 108723749835, 179936095421, 192142142419, 192629121549, 236162118265]
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)>
```

Рис. 4 – Расшифрование длинной последовательности

5. Примеры криптоанализа

Выполним простую атаку на криптосистему RSA: бесключевое чтение [4]. Для этого напишем скрипт `crack.py`, последовательно возводящий шифртекст в степень e, пока не будет достигнуто совпадение с самим шифртекстом. Открытый текст при этом оказывается на предыдущей итерации.

Пример зашифрования и дешифрования показан на рисунках 5 — 7.

```
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python keygen.py
10
e = 5  n = 601229  d = 479741 827 727
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python encrypt.py
type your message: A
enter e: 5
enter n: 601229
plain bits:  1000001
A | 65 | 1000001
block_length:  19 7 True 12
0000000000001000001 | 65 | 519884 | 01111110111011001100
plain numbers:  [65]
ciphertext:  '\x1f''']''L'
cipher bits:  01111110111011001100
cipher numbers:  [519884]
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)>
```

Рис. 5 – Генерация ключей и зашифрование символа "A"

```
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)> python crack.py
1 | 540707 | 519884
1 | 540707 | 519884
2 | 40845 | 519884
3 | 97089 | 519884
4 | 205079 | 519884
5 | 294514 | 519884
6 | 515989 | 519884
7 | 381081 | 519884
8 | 435871 | 519884
9 | 544588 | 519884
10 | 171651 | 519884
11 | 47801 | 519884
12 | 422520 | 519884
13 | 309373 | 519884
14 | 396280 | 519884
```

Рис. 6 – Начало работы взломщика

```
857 | 29940 | 519884
858 | 408972 | 519884
859 | 81489 | 519884
860 | 262526 | 519884
861 | 361138 | 519884
862 | 354909 | 519884
863 | 3306 | 519884
864 | 535864 | 519884
865 | 485715 | 519884
866 | 388037 | 519884
867 | 411615 | 519884
868 | 478037 | 519884
869 | 65 | 519884
870 | 519884 | 519884
plain number:  65
(.venv) alex@alex ~/D/mathmethods-labs (lab4-rsa)>
```

Рис. 7 – Открытый текст найден на 869 итерации

Данная атака осуществима только при малых размерах ключа, иначе перебор займёт очень много времени.

6. Выводы о проделанной работе

В данной работе мы изучили криптосистему RSA: генерацию ключей, алгоритмы зашифрования, расшифрования, а также алгоритмы для работы с большими числами (алгоритм возведения в степень по модулю) и с простыми числами (тест Ферма). Также удалось реализовать простую атаку — бесключевое чтение RSA. Эта атака, как и почти все известные атаки, осуществимы только при малых параметрах криптосистемы.

После выполнения работы стало ясно, что криптосистема RSA является стойкой при правильном выборе параметров.

Список использованных источников

[1] Antipov Alexander. Публикация статьи М. Хеллмана и У. Диффи ”Новые направления в криптографии”. — 1976. — Jun. — Access mode: https://www.securitylab.ru/informer/240713.php.

[2] Factorization of a 768-Bit RSA Modulus / Kleinjung Thorsten, Aoki Kazumaro, Franke Jens, Lenstra Arjen K., Thomé Emmanuel, Bos Joppe W., Gaudry Pierrick, Kruppa Alexander, Montgomery Peter L., Osvik Dag Arne, te Riele Herman, Timofeev Andrey, and Zimmermann Paul // Advances in Cryptology – CRYPTO 2010 / ed. by Rabin Tal. — Berlin, Heidelberg : Springer Berlin Heidelberg. — 2010. — P. 333–350.

[3] Википедия. Криптоанализ RSA — Википедия, свободная энциклопедия. — 2022. — [Онлайн; загружено 16 июня 2022]. Access mode: https://ru.wikipedia.org/?curid=3131537&oldid=120755392.

[4] Stalker. Атаки на RSA. — Access mode: http://algolist.ru/defence/attack/rsa.php.

[5] Prudhomme Antoine. How to generate big prime numbers — Miller-Rabin. — 2018. — May. — Access mode: https://medium.com/@prudywsh/how-to-generate-big-prime-numbers-miller-rabin-49e6e6af32fb.