

**Министерство науки и высшего образования
Липецкий государственный технический университет**

Факультет автоматизации и информатики
Кафедра прикладной математики

Отчет по лабораторной работе № 3
по дисциплине «Безопасности компьютерных систем»
на тему «Асимметричные криптосистемы»

Студент

Группа ПМ-19-1

Руководитель

к.т.н., доцент

учёная степень, учёное звание

подпись, дата

подпись, дата

Пестова А.Ю.

фамилия, инициалы

Сысоев А.С.

фамилия, инициалы

Липецк 2023 г.

Оглавление

Задание кафедры	2
1 Теоретическая часть	3
1.1 RSA	3
1.2 SHA-256	3
2 Практическая часть	5
3 Контрольные вопросы	5
Заключение	9

Задание кафедры

Реализовать алгоритм RSA. На контрольном примере проверить правильность работы алгоритмов шифрования и дешифрования.

1 Теоретическая часть

1.1 RSA

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 $n = p \cdot q$, которое называется модулем;
3. вычисляется значение функции Эйлера от числа n :

$$\varphi(n) = (p - 1) \cdot (q - 1)$$

4. выбирается целое число e ($1 < e < \varphi(n)$), взаимно простое со значением функции $\varphi(n)$;
5. вычисляется число d , мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению:

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

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.

1.2 SHA-256

Хэш генерируется следующим образом:

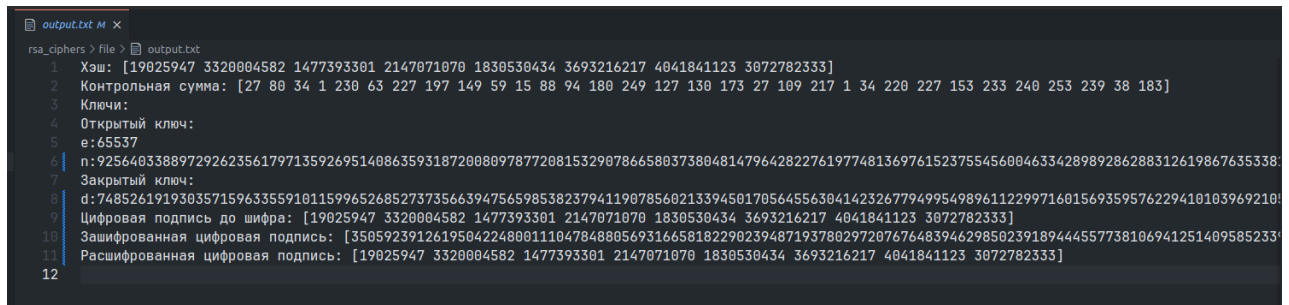
1. Преобразовать исходный текст в двоичный код.
2. Добавить 1.
3. Дополнить код нулями, пока данные не станут равны 512 бит, минус 64 бита (в результате 448 бит).
4. Добавить 64 бита в конец в виде целого числа с порядком байтов от старшего к младшему (big-endian), представляющего длину входного сообщения в двоичном формате.
5. Создать 8 хэш-значений. Это жестко запрограммированные константы, которые представляют собой первые 32 бита дробных частей квадратных корней из первых восьми простых чисел: 2, 3, 5, 7, 11, 13, 17, 19.

6. Создать еще несколько констант. На этот раз их будет 64. Каждое значение (0—63) представляет собой первые 32 бита дробных частей кубических корней первых 64 простых чисел (2—311).
7. Для каждого 512-битного «фрагмента» из наших входных данных:
 - 7.1 Скопировать входные данные из шага 1 в новый массив, где каждая запись представляет собой 32-битное слово.
 - 7.2 Добавить еще 48 слов, инициализированных нулем, чтобы у нас получился массив $w[0 \dots 63]$.
 - 7.3 Изменить обнуленные индексы в конце массива, используя следующий алгоритм. Для i из $w[16 \dots 63]$:
 - 7.3.1 $s_0 = (w[i-15] \text{ rightrotate } 7) \text{ xor } (w[i-15] \text{ rightrotate } 18) \text{ xor } (w[i-15] \text{ rightshift } 3)$
 - i. $s_1 = (w[i-2] \text{ rightrotate } 17) \text{ xor } (w[i-2] \text{ rightrotate } 19) \text{ xor } (w[i-2] \text{ rightshift } 10)$
 - ii. $w[i] = w[i-16] + s_0 + w[i-7] + s_1$
 - 7.4 Инициализировать переменные a, b, c, d, e, f, g, h и установить их равными текущим значениям хэш-функции соответственно $h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7$.
 - 7.5 Запустите цикл сжатия, который изменит значения $a \dots h$. Для i от 0 до 63:
 - 7.5.1 $S1 = (e \text{ rightrotate } 6) \text{ xor } (e \text{ rightrotate } 11) \text{ xor } (e \text{ rightrotate } 25)$
 - 7.5.2 $ch = (e \text{ and } f) \text{ xor } ((\text{not } e) \text{ and } g)$
 - 7.5.3 $\text{temp1} = h + S1 + ch + k[i] + w[i]$
 - 7.5.4 $S0 = (a \text{ rightrotate } 2) \text{ xor } (a \text{ rightrotate } 13) \text{ xor } (a \text{ rightrotate } 22)$
 - 7.5.5 $\text{maj} = (a \text{ and } b) \text{ xor } (a \text{ and } c) \text{ xor } (b \text{ and } c)$
 - 7.5.6 $\text{temp2} := S0 + \text{maj}$
 - 7.5.7 $h = g$
 - 7.5.8 $g = f$
 - 7.5.9 $e = d + \text{temp1}$
 - 7.5.10 $d = c$
 - 7.5.11 $c = b$
 - 7.5.12 $b = a$
 - 7.5.13 $a = \text{temp1} + \text{temp2}$

7.6 После цикла сжатия, во время цикла фрагментов, изменяются хеш-значения, добавляя к ним соответствующие переменные $a-h$. Как и ранее, все сложение производится по модулю 2^{32} .

8. Финальный хэш

2 Практическая часть



```
rsa_ciphers > File > output.txt
1  Хэш: [19025947 3320004582 1477393301 2147071070 1830530434 3693216217 4041841123 3072782333]
2  Контрольная сумма: [27 80 34 1 230 63 227 197 149 59 15 88 94 180 249 127 130 173 27 109 217 1 34 220 227 153 233 240 253 239 38 183]
3  Ключи:
4  Открытый ключ:
5  e:65537
6  n:925640338897292623561797135926951408635931872008097877208153290786658037380481479642822761977481369761523755456004633428989286288312619867635338
7  Закрытый ключ:
8  d:748526191930357159633559101159965268527373566394756598538237941190785602133945017056455630414232677949954989611229971601569359576229410103969210
9  Цифровая подпись до шифра: [19025947 3320004582 1477393301 2147071070 1830530434 3693216217 4041841123 3072782333]
10 Зашифрованная цифровая подпись: [35059239126195042248001110478488056931665818229023948719378029720767648394629850239189444557738106941251409585233]
11 Расшифрованная цифровая подпись: [19025947 3320004582 1477393301 2147071070 1830530434 3693216217 4041841123 3072782333]
12
```

Рисунок 1 – Результаты работы алгоритма

3 Контрольные вопросы

1. Асимметричные криптосистемы: принцип работы.

Для зашифрования данных используется открытый ключ, который может быть опубликован для использования всеми пользователями системы. Расшифрование с помощью открытого ключа невозможно.

Для расшифрования используется секретный ключ, который не может быть определен из ключа зашифрования.

2. Схема RSA (рис. 2).

3. Схема Полига-Хеллмана (рис. 3).

4. Схема Эль-Гамала (рис. 4).

5. Комбинированный метод шифрования (рис. 5).

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом.

При таком подходе криптосистема с открытым ключом применяется для шифрования, передачи и последующего расшифрования только секретного ключа симметричной криптосистемы, а симметричная крипто-

ШИФРОВАНИЕ И РАСШИФРОВАНИЕ

Шаг 1. Пользователь **В** выбирает два произвольных больших простых числа **P** и **Q**.

Шаг 2. Пользователь **В** вычисляет значения модуля $N = P \cdot Q$.

Шаг 3. Пользователь **В** вычисляет функцию Эйлера $\varphi(N) = (P - 1)(Q - 1)$ и выбирает случайным образом значения открытого ключа K_B с учетом выполнения условий $1 < K_B \leq \varphi(N)$, $\text{НОД}(K_B, \varphi(N)) = 1$.

Шаг 4. Пользователь **В** вычисляет значение секретного ключа k_B , используя расширенный алгоритм Евклида при решении сравнения $k_B = K_B^{-1} \bmod \varphi(N)$.

Шаг 5. Пользователь **В** пересылает пользователю **А** пару чисел (N, K_B) по незащищенному каналу. Если пользователь **А** хочет передать пользователю **В** сообщение **M**, он выполняет шаг 6.

Шаг 6. Пользователь **А** разбивает исходный открытый текст **M** на блоки, каждый из которых может быть представлен в виде числа $M_i = 0, 1, 2, \dots, N - 1$.

Шаг 7. Пользователь **А** шифрует текст, представленный в виде последовательности чисел M_i по формуле $C_i = M_i^{K_B} \bmod N$ и отправляет криптограмму $C_1, C_2, C_3, \dots, C_i, \dots$ пользователю **В**.

Шаг 8. Пользователь **В** расшифровывает принятую криптограмму $C_1, C_2, C_3, \dots, C_i, \dots$, используя секретный ключ k_B по формуле $M_i = C_i^{k_B} \bmod N$.

В результате будет получена последовательность чисел M_i , которые представляют собой исходное сообщение **M**.

Рисунок 2 – Схема RSA

4.5. Схема шифрования Полига-Хеллмана

Схема шифрования Полига-Хеллмана сходна со схемой шифрования RSA. Она представляет собой несимметричный алгоритм, поскольку используются различные ключи для шифрования и расшифрования. В то же время эту схему нельзя отнести к классу криптосистем с открытым ключом, т.к. ключи шифрования и расшифрования легко выводятся один из другого. Оба ключа (шифрования и расшифрования) нужно держать в секрете.

Аналогично схеме RSA криптограмма **C** и открытый текст **P** определяются из соотношений:

$$C = P^e \bmod n,$$

$$P = C^d \bmod n,$$

где $e \cdot d \equiv 1$ (по модулю некоторого составного числа).

В отличие от алгоритма RSA в этой схеме число **n** не определяется через два больших простых числа; число **n** должно оставаться только частью секретного ключа. Если кто-либо узнает значение **e** и **n**, он сможет вычислить значение **d**.

Не зная значений **e** и **d**, противник будет вынужден вычислять значение

$$e = \log_P C \bmod n.$$

Известно, что это является трудной задачей.

Рисунок 3 – Схема Полига-Хеллмана

4.6. Схема шифрования Эль Гамала

Безопасность схемы Эль Гамала обусловлена сложностью вычисления дискретных логарифмов в конечном поле.

- Для того, чтобы сгенерировать пару ключей (открытый ключ, секретный ключ), сначала выбирают некоторое большое простое число P и большое целое число G , причем $G < P$. Числа P и G могут быть распространены среди группы пользователей.
- Затем выбирают случайное целое число X , причем $X < P$. Число X является секретным ключом и должно храниться в секрете.
- Вычисляют $Y = G^X \bmod P$. Число Y является открытым ключом.
- Для того чтобы зашифровать сообщение M , выбирают случайное целое число K , $1 < K < P - 1$, такое, что числа K и $(P - 1)$ являются взаимно простыми.
- Вычисляют числа $a = G^K \bmod P$, $b = Y^K M \bmod P$.
- Пара чисел (a, b) является шифротекстом. Длина шифротекста вдвое больше длины исходного открытого текста M .
- Для того чтобы расшифровать шифротекст (a, b) , вычисляют $M = \frac{b}{a^X} \bmod P$.

Рисунок 4 – Схема Эль-Гамала

система применяется для шифрования и передачи исходного открытого текста.

6. Однонаправленные функции и их виды.

Большинство хэш-функций строится на основе однонаправленной функции f , которая образует выходное значение длиной n при задании двух входных значений длиной n . Этими входами являются блок исходного текста M_i и хэш-значение H_{i-1} предыдущего блока текста.

Хэш-значение последнего блока - хэш-значение всего сообщения M .

7. Хэш-функции

Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит.

Хэш-функция h принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение $h(M) = H$ фиксированной длины.

Значение хэш-функции $h(M)$ сложным образом зависит от документа M и не позволяет восстановить сам документ M .

4.7. Комбинированный метод шифрования

Если пользователь **A** хочет передать зашифрованное комбинированным методом сообщение **M** пользователю **B**, то порядок его действий таков:

1. Создать (например, сгенерировать случайным образом) симметричный ключ, называемый в этом методе *сеансовым ключом*.
2. Зашифровать сообщение **M** на сеансовом ключе K_S .
3. Зашифровать сеансовый ключ K_S на открытом ключе K_B пользователя **B**.
4. Передать по открытому каналу связи в адрес пользователя **B** зашифрованное сообщение вместе с зашифрованным сеансовым ключом.

Действия пользователя **B** при получении зашифрованного сообщения и зашифрованного сеансового ключа должны быть обратными:

5. Расшифровать на своем секретном ключе k_B сеансовый ключ K_S .
6. С помощью полученного сеансового ключа K_S расшифровать и прочитать сообщение **M**.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь **B** сможет правильно расшифровать ключ K_S и прочитать сообщение **M**.

Рисунок 5 – Комбинированный метод шифрования

Хэш-функция должна быть чувствительна к всевозможным изменениям в тексте **M**, таким как вставки, выбросы, перестановки и т.п.

Хэш-функция должна обладать свойством необратимости, т.е. задача подбора документа **M**, который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима.

Вероятность того, что значения хэш-функций двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала.

8. Расширенный алгоритм Евклида.

Используется для вычисления секретного ключа.

Заключение

В ходе выполнения данной работы был реализован алгоритм RSA. На контрольном примере были проверены правильность и корректность работы приведённого алгоритма шифрования и дешифрования.