

## Лабораторная работа № 8

### ⋮ Исследование асимметричных шифров ⋮ RSA и Эль-Гамала

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамала.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамала.

2. Разработать приложение для реализации асимметричного зашифрования/расшифрования на основе алгоритмов RSA и Эль-Гамала.

3. Выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамала.

4. Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

## 8.1. Теоретические сведения

### 8.1.1. Математические основы асимметричных шифров

 Как отмечалось выше, асимметричная криптография основана на сложности решения некоторых математических задач. По существу, таких задач две:

- разложение больших чисел на простые сомножители (задача факторизации);
- вычисление дискретного логарифма в конечном поле, а также вычислительные операции над точками эллиптической кривой.

Эти задачи объединяет то, что они используют операцию получения остатка от целочисленного деления.

В силу этого практически все системы асимметричного зашифрования/расшифрования основаны либо на проблеме факторизации (среди них – RSA), либо на проблеме дискретного логарифмирования (среди них – Эль-Гамала).

Базовые элементы перечисленных проблем мы рассмотрели с практической точки зрения при выполнении лабораторной работы № 1. Алгебраическая теория рассматриваемого класса криптосистем подробно изучена в [39]. Мы же здесь остановимся лишь на нескольких основных элементах этой теории.

**Теорема 1:** *основная теорема арифметики.* Всякое натуральное число  $N$ , кроме 1, можно представить как произведение простых множителей:

$$N = p_1 p_2 p_3 \dots p_z, z > 1. \quad (8.1)$$

**Определение 1.** *Задача дискретного логарифмирования* формулируется так: для данных целых чисел  $a$  и  $b$ ,  $1 < a, b < n$ , найти логарифм – такое целое число  $x$ , что

$$a^x \equiv b \pmod{n}, \quad (8.2)$$

если такое число существует.

По аналогии с вещественными числами используется обозначение  $x = \log_a b$ .

**Теорема 2:** *китайская теорема об остатках.* В общем случае если разложение числа  $N$  на простые множители представляет собой  $p_1 p_2 \dots p_t$  (некоторые простые числа могут встречаться несколько раз), то система уравнений

$$(x \bmod p_i) \equiv a_i, \quad (8.3)$$

где  $i = 1, 2, \dots, t$ , имеет единственное решение:  $x$ , *меньшее  $N$* .

Иными словами, число (меньшее, чем произведение нескольких простых чисел) однозначно определяется своими вычетами по модулю от этих простых чисел. Китайской теоремой об остатках можно воспользоваться для решения полной системы уравнений в том случае, если известно разложение числа  $N$  на простые множители.

### 8.1.2. Алгоритм RSA

Рассматриваемый алгоритм появился (1977 г.) после алгоритма ранца Меркла. Он стал первым полноценный алгоритмом с открытым

ключом, который впоследствии стал одним из основных для шифрования и для электронных цифровых подписей.

Из всех предложенных алгоритмов с открытыми ключами RSA проще всего понять и реализовать. Он назван в честь трех его создателей: Рона Ривеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Адлемана (Leonard Adleman).

Как было отмечено, безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Для генерации двух ключей: тайного и открытого (а по сути – двух взаимосвязанных частей одного ключа, т. е. ключа, принадлежащего одному физическому лицу (или группе лиц), либо одному юридическому лицу), используются два больших случайных простых числа  $p$  и  $q$ . Для максимальной большей криптостойкости нужно выбирать  $p$  и  $q$  равной длины. Рассчитывается произведение:  $n = pq$ . Это есть один из трех компонент ключа, состоящего из чисел  $n$ ,  $e$ ,  $d$ .

Затем случайным образом выбирается второй компонент ключа (открытый ключ или ключ зашифрования,  $e$ , такой что  $e$  и  $(p-1)(q-1)$  являются взаимно простыми числами; вспомним, что  $(p-1)(q-1) = \varphi(n)$  – функция Эйлера). Б. Шнайер [5] рекомендует число  $e$  выбирать из ряда: 3, 17,  $2^{16} + 1$ .

Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования  $d$  такого, что выполняется условие:

$$ed \equiv 1 \pmod{\varphi(n)}. \quad (8.4)$$

Другими словами:

$$d^{-1} \equiv e \pmod{\varphi(n)}. \quad (8.5)$$

Таким образом, сформирован ключ, состоящий из трех чисел, которые в свою очередь образуют две вышеупомянутые взаимосвязанные части: открытый (публичный) ключ ( $e$ ,  $n$ ) и тайный ключ ( $d$ ,  $n$ ; на самом деле, как видим, тайным здесь является лишь первое из пары чисел).

Примеры генерации ключевой информации, как и ее использования, можно найти в [3].

Для зашифрования/расшифрования используется ключ получателя: отправитель шифрует сообщение открытым ключом, а получатель расшифровывает шифртекст своим тайным ключом.

*Зашифрование.* Если шифруется сообщение  $M$ , состоящее из  $r$  блоков:  $m_1, m_2, \dots, m_i, \dots, m_r$ , то шифртекст  $C$  будет состоять из такого же числа ( $r$ ) блоков, представляемых числами:

$$c_i \equiv (m_i)^e \bmod n. \quad (8.6)$$

*Расшифрование.* Для расшифрования каждого зашифрованного блока производится вычисление вида:

$$m_i \equiv (c_i)^d \bmod n. \quad (8.7)$$

Разработаны несколько версий стандарта рассматриваемого алгоритма. Среди прочего, в этих документах обсуждаются размеры безопасного ключа. Доступна одна из последних версий стандарта RSA: RFC 3447.

Размер ключа в алгоритме RSA связан с размером модуля  $n$ . Два числа  $p$  и  $q$ , произведение которых равно  $n$ , должны иметь приблизительно одинаковую длину, поскольку в этом случае найти сомножители (факторы) сложнее, чем в случае, когда длина чисел значительно различается. Например, если предполагается использовать 768-битный модуль, то каждое число должно иметь длину приблизительно 384 бита. В 1999 г. 512-битный ключ был вскрыт за семь месяцев [5]. Это означает, что 512-битные ключи уже не обеспечивают достаточную криптостойкость. Сейчас в критических системах применяются ключи длиной 1024 и 2048 битов. Ссылочные представления этих чисел в десятичной системе счисления даны в [9].

### 8.1.3. Алгоритм Эль-Гамала

Предложен Т. Эль-Гамалем (Т. El-Gamal) в 1985 г. Он может быть использован для решения трех основных криптографических задач: для зашифрования/расшифрования данных, для формирования цифровой подписи и для согласования общего ключа. Кроме того, возможны модификации алгоритма для схем проверки пароля, доказательства идентичности сообщения и другие варианты.

**! Как подчеркивалось выше, безопасность алгоритма Эль-Гамала, как и безопасность алгоритма Диффи – Хеллмана, основана на трудности вычисления дискретных логарифмов.**

**Алгоритм Эль-Гамала фактически использует схему Диффи – Хеллмана, чтобы сформировать общий секретный ключ для абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ.**

И в случае шифрования, и в случае формирования цифровой подписи каждому пользователю необходимо сгенерировать пару ключей.

Рассматриваемый алгоритм отличается от алгоритма RSA несколькими параметрами и особенностями:

1) генерацией ключевой информации и числом компонент, составляющих ключ;

2) каждому блоку (символу) открытого сообщения в шифртексте на основе алгоритма Эль-Гамала соответствуют 2 блока (в RSA – один-один);

3) в алгоритме Эль-Гамала при зашифровании используется число (обозначим его  $k$ ), которое практически никак не связано с ключевой информацией получателя и которое принимает (по определению) различные значения при зашифровании различных блоков сообщения.

**Генерация ключевой информации.** Выбирается простое число  $p$ . Выбирается число  $(g, g < p)$ , являющееся первообразным корнем числа  $p$  – очень важный элемент с точки зрения безопасности алгоритма (см. ниже).

Далее выбирается число  $x$  ( $x < p$ ) и вычисляется последний компонент ключевой информации:

$$y \equiv g^x \bmod p. \quad (8.8)$$

Владельцу сформированной ключевой информации, состоящей из 4 чисел, может посылаться некоторый шифртекст, созданный с использованием открытого ключа получателя:  $p, g, y$ . Расшифрование шифртекста получатель производит своим тайным ключом:  $p, g, x$ .

Как видим, на самом деле тайным является лишь одно число (как и в RSA):  $x$ .

**Определение 2. Первообразный корень** (primary (residual) root) по модулю  $p$  является таким числом, что его степени ( $g^i, 1 \leq i \leq p - 1$ ) дают все возможные по модулю  $p$  вычеты (остатки), которые взаимно просты с  $p$ .

**Пример 1.** Следующие остатки по модулю 5 ( $p = 5$ ) от  $2^i$ : 2, 4, 3, 1. Они дают все возможные остатки. Число 2 является первообразным корнем по модулю 5.

**Пример 2.** Следующие остатки по модулю 7 от  $2^i$ : 2, 4, 1, 2, ... (они не дают всех возможных остатков). Число 2 не является первообразным корнем по модулю 7.

**Пример 3.** Для  $p = 17$  и  $g = 3$ : остатки по модулю 17 от  $3^i$ : 3, 9, 10, 13, 5, 15, 11, 16, 14, 8, 7, 4, 12, 2, 6, 1.

Число 3 является первообразным корнем по модулю 17.

Понятно, что для больших значений  $p$  количество всех повторяющихся остатков ( $p - 1$ ) будет также большим. А поскольку в уравнении (8.8) мы используем модуль  $p$  большого простого числа и находим первообразным корень от  $p$ , который имеет важное свойство: при использовании разных степеней ( $a^i = a^x$ ) решение будет равномерно распределяться от 0 до  $p - 1$ , то нахождение криптоаналитиком нужного  $x$  чрезвычайно затруднено. В этом заключается односторонность функции, задаваемой (8.8). И на этом основывается криптостойкость шифра Эль-Гамала.

Для схемы вероятностного шифрования само сообщение и ключ не определяют шифротекст однозначно.

**Зашифрование сообщения.** Как ранее, предположим, что сообщение  $M = \{m_i\}$ , где  $m_i$  –  $i$ -й блок сообщения.

Зашифрование отправителем (каждого отдельного блока  $m_i$  исходного сообщения) предусматривает использование, как это особо подчеркивалось выше, некоторого случайного числа  $k$  ( $1 < k < p - 1$ ).

**!** В силу использования случайной величины  $k$  шифр Эль-Гамала называют также **шифром многозначной замены, а также схемой вероятностного шифрования.**

Вероятностный характер шифрования является преимуществом для схемы Эль-Гамала по сравнению, например, с алгоритмом RSA.

Блок шифротекста ( $c_i$ ) состоит из двух чисел –  $a_i$  и  $b_i$ :

$$a_i \equiv g^k \pmod{p}; \quad (8.9)$$

$$b_i \equiv (y^k m_i) \pmod{p}. \quad (8.10)$$

Здесь стал очевидным упомянутый недостаток алгоритма шифрования Эль-Гамала: удвоение (реально – примерно в 1,5 раза) длины зашифрованного текста по сравнению с начальным текстом.

Случайное число  $k$  должно сразу после вычисления уничтожаться.

**Расшифрование  $c_i$ .** Выполняется по следующей формуле:



$$m_i \equiv (b_i(a_i)^x)^{-1} \bmod p \quad (8.11)$$

или

$$m_i \equiv (b_i(a_i)^{p-x-1}) \bmod p, \quad (8.12)$$

где  $(a^x)^{-1}$  – обратное значение числа  $a^x$  по модулю  $p$ .

Нетрудно проверить, что  $((a_i)^x)^{-1} \equiv g^{kx} \bmod p$ .

Еще раз возвратимся к криптостойкости рассмотренного алгоритма.

Если для зашифрования двух разных блоков ( $m_1$  и  $m_2$ ) некоторого сообщения использовать одинаковые  $k$ , то для соответствующих шифртекстов  $c_1 = (a_1, b_1)$  и  $c_2 = (a_2, b_2)$  выполняется соотношение  $b_1(b_2)^{-1} = m_1(m_2)^{-1}$ . Из этого выражения можно легко вычислить  $m_2$ , если известно  $m_1$ .

При примерно одинаковой размерности ключей рассмотренные алгоритмы обеспечивают примерно одинаковый уровень криптостойкости.

## 8.2. Практическое задание

1. С помощью простого консольного приложения составить табличную или графическую форму зависимости времени вычисления параметра  $y$ , функционально заданного выражением вида:

$$y \equiv a^x \bmod n,$$

от параметров:  $a$  (десятичные числа от 5 до 35; можно взять 1 или 2 числа),  $x$  (числа, желательно простые, из диапазона от  $10^3$  до  $10^{100}$ ; для примера взять 5–10 чисел, равномерно распределенных в указанном диапазоне),  $n$  (для примера взять числа, в двоичном виде состоящие из 1024 и 2048 битов).

2. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами.

В основе вычислений – кодировочные таблицы Base64 и ASCII.

Приложение должно реализовывать следующие операции:

- зашифрование и расшифрование текстовых документов на основе алгоритмов RSA и Эль-Гамала;
- определение времени выполнения операций.

Исходный текст для зашифрования – собственные фамилия, имя, отчество. Для численного представления блоков текста можно в том числе пользоваться указанными выше кодировочными таблицами.

Ключевую информацию для обоих алгоритмов можно сгенерировать самостоятельно либо воспользоваться, например, одной из утилит криптографической библиотеки OpenSSL, с помощью которой, в частности, можно сгенерировать ключевую информацию для алгоритма RSA.

3. Используя примерно одинаковый порядок ключевой информации, оценить производительность обоих алгоритмов и относительное изменение объемов криптотекстов (по отношению к объемам открытых текстов).

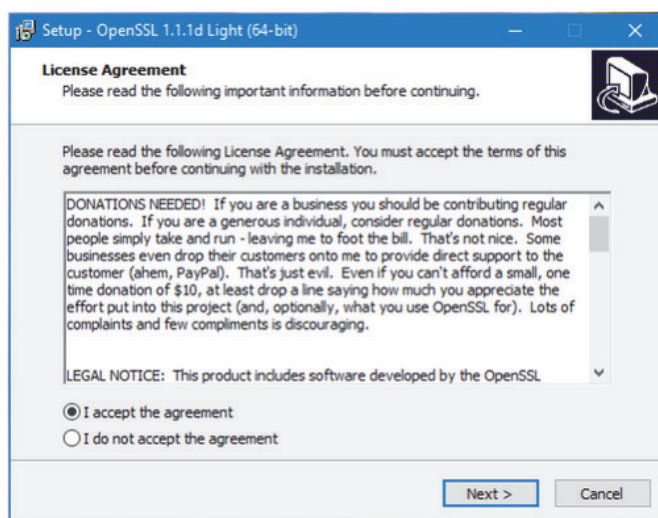
Ниже даны некоторые указания к инсталляции (при желании) библиотеки OpenSSL.

*OpenSSL* – это система защиты и сертификации данных; SSL (Secure Socket Layer – система безопасных сокетов). Внутри OpenSSL имеются отдельные компоненты (утилиты), отвечающие за то или иное действие. OpenSSL поддерживает в том числе много различных стандартов шифрования. К их числу относится RSA.

Краткую начальную информацию к процедуре инсталляции OpenSSL можно найти по адресу: [https://www.xolphin.com/support/OpenSSL/OpenSSL\\_-\\_Installation\\_under\\_Windows](https://www.xolphin.com/support/OpenSSL/OpenSSL_-_Installation_under_Windows).

Для загрузки OpenSSL можно также воспользоваться источником по адресу: <https://sourceforge.net/projects/openssl/>.

После скачивания выбранной версии OpenSSL запускается *exe*-файл, например, *Win64OpenSSL\_Light-1\_1\_1d.exe* (рис. 8.1).



**Рис. 8.1.** Начальное диалоговое окно инсталляции OpenSSL



Далее выполняем необходимые (достаточно традиционные) требования для установки.

Синтаксис OpenSSL следующий:

*\$ openssl<команда> [<опции команды>] [<параметры команды>]*

При вызове openssl без команды и параметров будет открыта интерактивная оболочка, из которой можно выполнять все те же команды, что и в неинтерактивном режиме. При вызове openssl с несуществующей командой будет выведен перечень команд и поддерживаемых криптоалгоритмов.

Вывод на экран списка доступных команд с неправильным ключом: *OpenSSL > help* или *OpenSSL > help -h*

Для нас в рамках выполнения данной лабораторной работы интерес представляют следующие команды:

*rsa* – обработка ключей RSA;

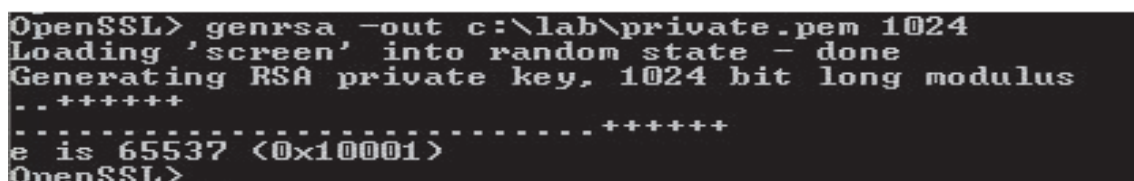
*speed* – вычисление скорости алгоритмов.

Для создания ключей алгоритма RSA используется команда *genrsa*: *openssl genrsa [-out file] [-des | -des3 | -idea] [-rand file] [bits]*

Команда *genrsa* создает секретный ключ длиной (дается в битах) в формате *PEM* (текстовый формат), зашифровывая его одним из алгоритмов (*des*, *des3*, *idea*). Опция *out* позволяет указать, в какой файл выполняется вывод созданного тайного ключа.

**Пример 4.** Команда *genrsa -out c:\lab\private.pem 1024* позволяет сгенерировать тайный ключ длиной 1024 бита, который будет записан в файл *key.txt*.

**Пример 5.** При генерации тайного ключа с записью его содержимого в файл *private.pem* пользователь увидит сообщение, показанное на рис. 8.2.



```
OpenSSL> genrsa -out c:\lab\private.pem 1024
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
..+++++
-----
e is 65537 (0x10001)
OpenSSL>
```

**Рис. 8.2.** Диалоговое окно OpenSSL при генерации тайного ключа RSA

Для создания открытого ключа на основе созданного секретного ключа используется команда *rsa*, которая имеет следующий синтаксис: *openssl rsa -in filename [-out file] [-des | -3des | -idea] [-check] [-pubout]*

В качестве параметров используются следующие ключи (дефисы перед ключами обязательны):

- pubout* – указывает на необходимость создания открытого ключа (public key) в файле, указанном в параметре -*out*;
- in* – указывает на файл с секретным ключом;
- pubin* – указание на то, что передается открытый ключ.

### **ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ**

1. Охарактеризовать алгоритмы RSA и Эль-Гамала. Для каких целей они могут применяться?
2. На чем основана криптостойкость алгоритмов RSA и Эль-Гамала?
3. Что такое первообразный корень?
4. Найти первообразные корни (если они существуют) чисел ( $p$ ): 13, 19, 23, 27, 31, 37, 39, 43.
5. Пусть пользователь А хочет передать пользователю В сообщение  $M$ , которое в некоторой кодировке соответствует числу 17 и зашифровано с помощью алгоритма RSA. Пользователь В имеет следующие ключевые параметры:  $p = 7$ ,  $q = 11$ ,  $d = 47$ . Описать процесс зашифрования сообщения пользователем А.
6. Пользователю системы RSA с ключевыми параметрами  $n = 33$ ,  $d = 3$  передано зашифрованное сообщение  $C$ , состоящее из блока цифр: 13. Расшифровать это сообщение (взломав систему RSA пользователя).
7. В системе связи, применяющей шифр Эль-Гамала, пользователь А желает передать сообщение  $M$  пользователю В. Найти недостающие параметры системы при следующих заданных параметрах:  $p = 19$ ,  $g = 2$ ,  $x = 3$ ,  $k = 5$ ,  $M = 10$ . Описать по шагам зашифрование сообщения и расшифрование шифртекста.
8. Положим, что в системе применяется алгоритм шифрования/расшифрования Эль-Гамала. Известны некоторые параметры системы:  $p = 167$ ,  $g = 5$ ,  $y = g^{29} \equiv 55 \pmod{p}$ . Используя указанные и недостающие (выбрать самостоятельно) параметры, зашифровать свое имя (в любом языке) в предположении: а) первая буква алфавита соответствует числу 0 и т. д.; б) первая буква алфавита соответствует числу 1. Проанализировать результат.