МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8 НА ТЕМУ:

Исследование ассиметричных шифров *RSA* и Эль-Гамаля

Выполнила:

Студентка 3 курса 1 группы ФИТ

Шимчёнок Елизавета Константиновна

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

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамаля.
2. Разработать приложение для реализации асимметричного зашифрования/расшифрования на основе алгоритмов RSA и Эль-Гамаля.
3. Выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамаля.
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.

**Теоретические сведения**

Асимметричная криптография основана на сложности решения некоторых математических задач:

- разложение больших чисел на простые сомножители (задача факторизации);

- вычисление дискретного логарифма в конечном поле, а также вычислительные операции над точками эллиптической кривой.

Все системы асимметричного зашифрования основаны на:

- проблеме факторизации (*RSA*);

- проблеме дискретного логарифма (Эль-Гамаля).

*RSA*

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.

Используются два больших случайных числа *p* и *q* (для максимальной криптостойкости нужно выбирать их равной длины)для генерации тайного и открытого ключа (двух взаимосвязанных частей ключа, принадлежащего одному физическому лицу). Рассчитываем *n* = *pq.* Ключ состоит из *n*, *e*, *d*.

Выбираем случайным образом *e,* такое что *e* и (*p* – 1)(*q* – 1) являются взаимно простыми числами.

Затем находим *d*:





В результате получаем ключ из трех чисел, которые образуют две взаимосвязи:

публичный ключ (*e*, *n*); тайный ключ (*d*, *n*).

Зашифрование RSA: если шифруется сообщение *М*, состоящее из *r* блоков: *m*1, *m*2, …, *mi*, …, mr, то шифртекст *С* будет состоять из такого же числа (*r*) блоков, представляемых числами:



Расшифрование RSA:



Эль-Гамаль

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

Алгоритм Эль-Гамаля 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.

Генерация ключевой информации:

Выбирается простое число *р*. Выбирается число (*g*, *g* < *p*), являющееся первообразным корнем числа *р* – очень важный элемент с точки зрения безопасности алгоритма.

Далее выбирается число *х* (*х* < *p*) и вычисляется последний компонент ключевой информации:



Отправка сообщения, зашифрованного ключами *p*, *g*, *y*. Расшифровка сообщения ключами *p*, *g*, *x*. Как видим, тайным является только *x*.

При больших *p* нахождение *x* затруднено.

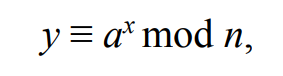
Алгоритм Эль-Гамаля предназначен для шифрования сообщений и создания цифровой подписи. Этот алгоритм основан на сложности вычисления дискретного логарифма в конечном поле.

Хотя в теории возможно использование алгоритма Эль-Гамаля для расшифрования сообщений, это не является его основной функцией и не является практически осуществимым в общем случае. Для расшифрования сообщений, зашифрованных алгоритмом Эль-Гамаля, необходимо знать закрытый ключ, который был использован для шифрования сообщения.

Поэтому алгоритм Эль-Гамаля обычно используется только для шифрования сообщений и создания цифровой подписи, а не для расшифровки сообщений, зашифрованных им.

**Ход работы**

В ходе лабораторной работы было написано консольное приложение, составляющее табличную форму зависимости времени вычисления параметра *у*, функционально заданного выражением вида:



от параметров: *а* (десятичные числа: 5 и 10), *х* (числа из диапазона от 103 до 10100), *n* (числа в двоичном виде, состоящие из 1024 и 2048 битов).

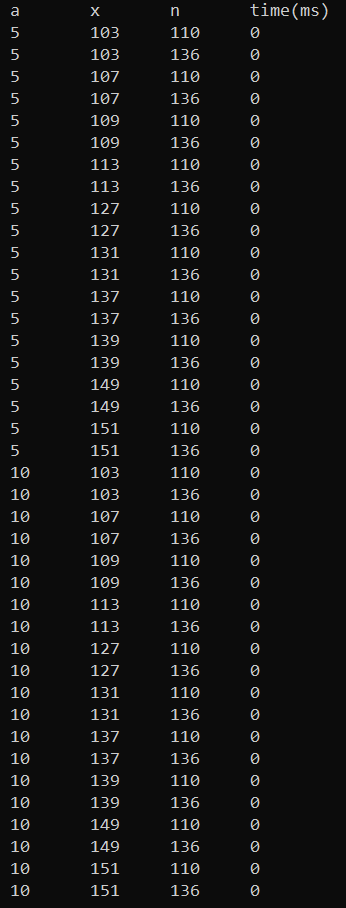


Рис. 1 – Таблица зависимости

Эль-Гамаль

Для реализации генерации ключевой информации, выполнены следующие действия.

Во-первых, разработанное приложение генерирует простое число p случайным образом в диапазоне [2000; 2500]. Реализация процесса генерации числа p продемонстрирована на рисунке 2.

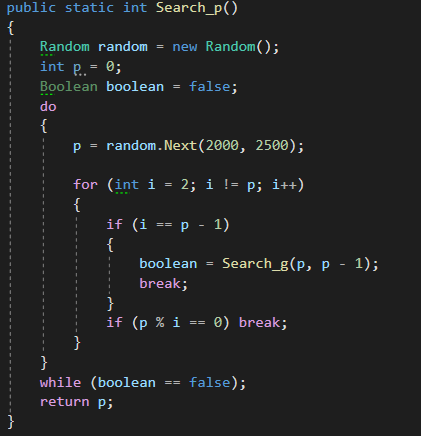


Рис. 2 – Генерация числа *p*

Во-вторых, выбираем число *g*, такое что *g* < *p* и является первообразным по модулю числа *p*, т.е., степени числа *g* (*gi*, 1≤ *i* ≤ *p*-1) дают все возможные по модулю *p* остатки, которые взаимно-просты с *р*.

В-третьих, генерируем закрытый ключ *х* случайным образом в диапазоне [1; *p*-1], т.е. *x*<*p*, который будет использоваться для дальнейших вычислений.

В-четвертых, производим вычисление открытого ключа *y* по формуле  
*y* = *gX* *mod p*. Теперь у нас есть вся необходимая ключевая информация (*p, g, x, y*) для осуществления операций зашифрования и расшифрования.

Зашифрования каждого отдельного блока mi исходного сообщения (в нашем случае 1 блок равен 1 символу) предусматривает использование некоторого случайного числа *k* (1<*k*<*p*-1).

Блок шифротекста (*ci*) состоит из двух чисел *ai* и *bi*:

*ai = gk mod p*;

*bi = (yk \*mi) mod p*;

Вычисление всех указанных выше действий зашифрования реализованы в функции, представленной на рисунке 3.

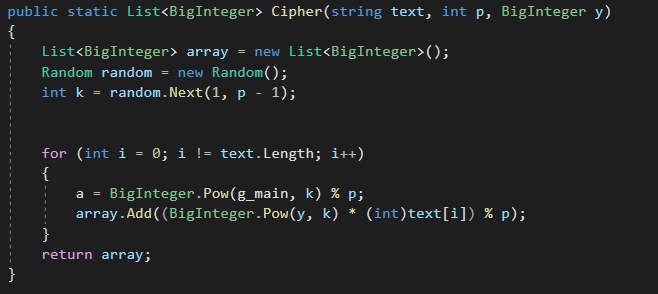


Рис. 3 – Реализация зашифрования

Полученный шифротекст следует расшифровать. Для этого вычислим каждый блок исходного сообщения по формуле:

*mi* = (*bi* \*(*ai*)*p*-*x*-1) *mod p*.

Реализация расшифрования представлена на рисунке 4.

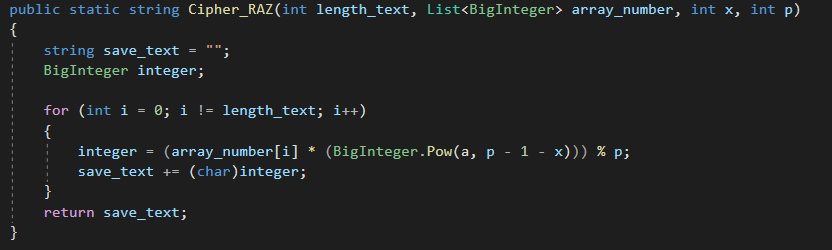


Рис. 4 – Реализация расшифрования

Протестируем разработанное программное средство. Результат работы приложения представлен на рисунке 5. При каждом запуске консольного приложения числа *p*, *g*, *y* будут разными, т.к. они основаны на генерации случайных чисел. Из рисунка видно, что приложение работает исправно, а время операций зашифрования и расшифрования составляют всего 260 и 9 мс, соответственно, что говорит о хороших результатах и большой производительности данного алгоритма. Из разницы размеров исходного и зашифрованного сообщений стал очевидным недостаток алгоритма: удвоение длины текста, связанный с использованием 2 чисел, соответствующих одному блоку исходного текста.

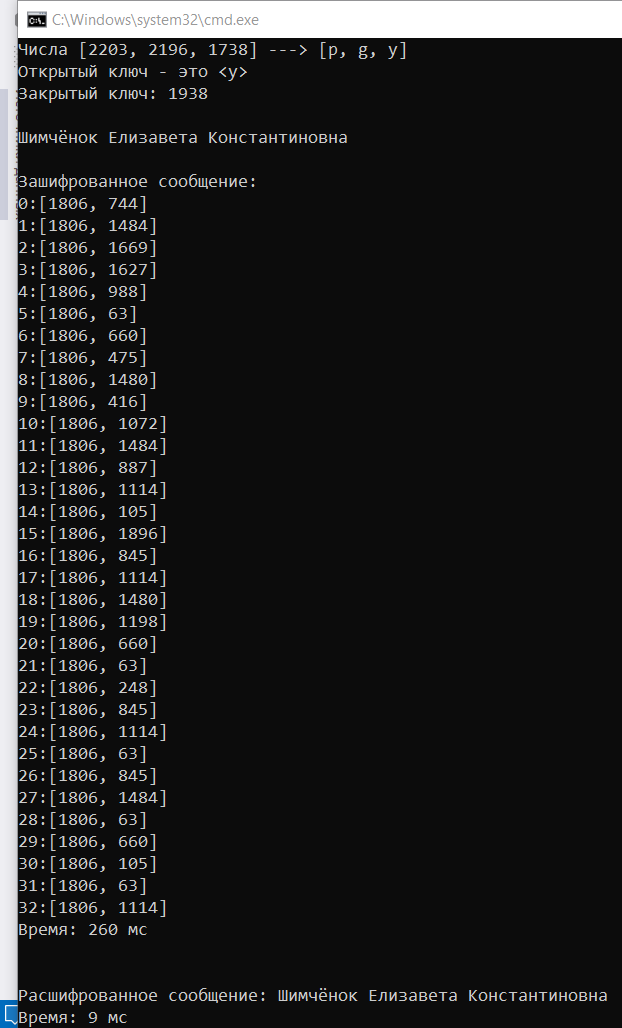
**

Рис. 5 – Результат алгоритма Эль-Гамаля

Алгоритм *RSA*

Далее рассмотрим программную реализацию зашифрования и расшифрования на основе алгоритма *RSA*. Данный алгоритм гораздо проще для понимания и реализации чем предыдущий.

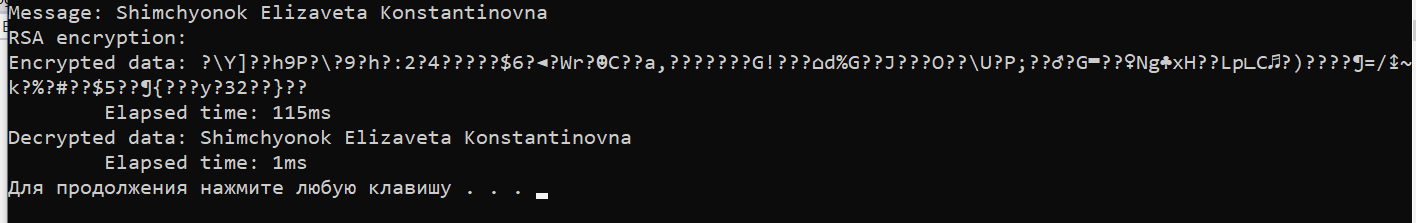


Рис. 6 – Результат алгоритма RSA

Действие алгоритма заключается в следующих шагах.

Во-первых, пользователю необходимо ввести большие простые числа *p*, *q*. Желательно, чтобы они были равной длины, тогда алгоритм станет еще более криптостойким за счет того, что найти сомножители будет труднее. Если пользователь введет не простое число в поле ввода *p*, *q*, система предупредит его об этом.

Во-вторых, необходимо выбрать число *е*, взаимно простое с функцией Эйлера ф(*n*) = (*p*-1)(*q*-1). Пара (*e*, *n*) будет являться открытым ключом алгоритма. Реализация нахождения числа *е* представлена на рисунке 7.

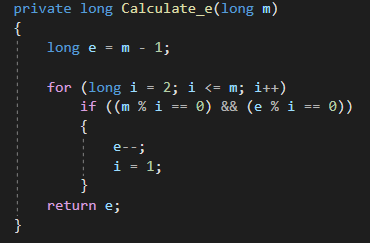
**

Рис. 7 – Нахождение числа е

В-третьих, необходимо вычислить число *d* по формуле Евклида, такое что *ed* = 1 *mod* (ф(*n*)). Пара (*d*, *n*) будет являться закрытым ключом алгоритма. Реализация нахождения числа *d* продемонстрирована на рисунке 8.

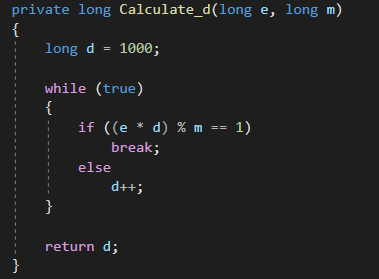
**

Рис. 8 – Нахождение числа d

Теперь у нас есть вся ключевая информация. Рассмотрим алгоритм зашифрования сообщения. Каждый блок шифротекста вычисляется отдельно по формуле *ci* = (*mi*)*e* *mod n*, где *mi* – блок исходного сообщения. Часть программного кода, непосредственно реализующая данные вычисления, представлена на рисунке 9.

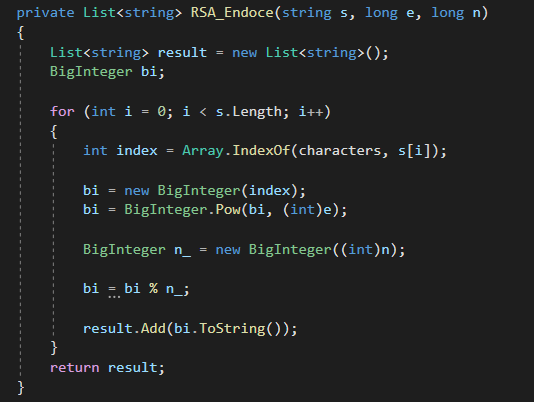
**

Рис. 9 – Реализация зашифрования

Расшифрование проводится подобным образом по формуле *mi =* (*ci*)*d* *mod n*, с использованием закрытого ключа *d*.

**Ответы на вопросы**

**1. Охарактеризовать алгоритмы RSA и Эль-Гамаля. Для каких целей они могут применяться?**

*RSA* и Эль-Гамаля являются асимметричными криптографическими алгоритмами, которые используются для шифрования и подписи данных.

Алгоритм *RSA* основан на сложности факторизации больших целых чисел и позволяет генерировать открытый и закрытый ключи для шифрования и дешифрования данных. *RSA* может использоваться для защиты конфиденциальности данных, аутентификации и цифровой подписи.

Алгоритм Эль-Гамаля также используется для шифрования и подписи данных, но основан на сложности вычисления дискретного логарифма в конечных полях. Он позволяет генерировать открытый и закрытый ключи для шифрования и дешифрования данных. Эль-Гамаля может использоваться для защиты конфиденциальности данных и цифровой подписи.

Оба алгоритма широко применяются для защиты конфиденциальности данных, аутентификации и цифровой подписи в различных сферах, таких как финансы, банковское дело, электронная коммерция и т.д.

**2. На чем основана криптостойкость алгоритмов RSA и Эль-Гамаля?**

Все системы асимметричного зашифрования основаны:

– на проблеме факторизации (*RSA*)

– на проблеме дискретного логарифма (Эль-Гамаля)

**3. Что такое первообразный корень?**

Первообразный корень по модулю *р* является таким числом, что его степени (*gi*, 1 ≤ *i* ≤ *p* – 1) дают все возможные по модулю *р* вычеты (остатки), которые взаимно просты с *p*.

**4. Найти первообразные корни (если они существуют) чисел (*р*): 13, 19, 23, 27, 31, 37, 39, 43.**

Для того чтобы найти первообразный корень по модулю *p*, где *p* - простое число, нужно найти наименьшее целое число *g*, такое что все числа от 1 до *p*-1 можно представить в виде gkmod p, где *k* - натуральное число.

Для числа 13 первообразных корней нет.

Для числа 19 первообразный корень равен 2.

Для числа 23 первообразный корень равен 5.

Для числа 27 первообразных корней нет.

Для числа 31 первообразный корень равен 3.

Для числа 37 первообразный корень равен 5.

Для числа 39 первообразных корней нет.

Для числа 43 первообразный корень равен 3.

**5. Пусть пользователь А хочет передать пользователю В сообщение *М*, 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.**

Пользователь A шифрует сообщение М:

Находим модуль *n* = *pq*=77;

Находим значение функции Эйлера от числа *n*:

ф(*n*)= (*p*-1)(*q*-1)=60;

Находим открытый ключ пользователя В, который состоит из 2-х чисел: модуля *n* и закрытой экспоненты *e*, равной взаимнопростому числу ф(*n*).

*e*=7; НОД(60,7) = 1;

Открытый ключ пользователя В – *n* и *e*.

Пользователь А должен преобразовать сообщение М в целое число, соответствующее его кодировке. У меня 17.

Пользователь А шифрует сообщение М, используя открытый ключ пользователя В. Для этого используется формула: *C*=*Me(mod n*), *C*=52;

*C* – есть зашифрованное сообщение пользователем А, которое может передаваться В.

**6. Пользователю системы RSA с ключевыми параметрами *n* = 33, *d* = 3 передано зашифрованное сообщение С, состоящее из блока цифр: 13. Расшифровать это сообщение (взломав систему RSA пользователя).**

*de* ≡ 1 (*mod* (*n*)).

1) Находим закрытый ключ *e*: ищем значение функции Эйлера от *n*.

*n* = 33 => функция Эйлера от *n* будет равна φ(33) = (11 - 1) \* (3 - 1) = 20.

2) Находим такое значение *e*, что *d* \* *e* ≡ 1 (*mod* (20)).

Используя расширенный алгоритм Евклида, можно найти решение этого уравнения: *de* + *k* \* 20 = 1, где *k* - целое число.

3 \* 7 + (-4) \* 20 = 1 => *e* = 7;

Теперь для расшифровки сообщения С необходимо возвести его в степень *d*, по модулю *n*:

133 *mod* 33 = 13;

Ответ: Расшифрованный текст равен 13.

**7. В системе связи, применяющей шифр Эль-Гамаля, пользователь А желает передать сообщение *М* пользователю В. Найти недостающие параметры системы при следующих заданных параметрах: *p* = 19, *g* = 2, *х* = 3, *k* = 5, *М* = 10. Описать по шагам зашифрование сообщения и расшифрование шифртекста.**

1. Вычисляем открытый ключ *y* пользователя А:

y=*gx* *mod p*= 23 *mod* 19=8;

1. Генерируем случайное число *k*, взаимно простое с *p*-1: *k* = 5;
2. Вычисляем первую составляющую шифртекста:

*a*=*gk mod p*= 25 *mod* 19=32 *mod* 19=13;

1. Вычисляем вторую составляющую шифртекста:

*b*=*M*\**yk mod p*= 10\*85 *mod* 19=10\*32768 *mod* 19=14;

1. Пользователь А передает пользователю В пару (*a*, *b*).

Пользователь В выполняет следующие шаги для расшифрования полученного шифртекста:

1. Вычисляем значение общего секретного ключа *s*:

*s*=*admod* *p*= 1311 *mod* 19=8;

1. Вычисляем мультипликативно обратное значение *y*:

*y*-1=*sp*-1-*xmod p*= 85 *mod* 19=7;

1. Вычисляем исходное сообщение *М*:

*M*=*b*\**y*-1*mod* *p*= 14\*7-5 *mod* 19=10;

Ответ: Полученное исходное сообщение *М* равно 10.

**8. Положим, что в системе применяется алгоритм шифрования/расшифрования Эль-Гамаля. Известны некоторые параметры системы: р = 167, g = 5, y = g29 ≡ 55 mod 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.**

Для шифрования сообщения в системе Эль-Гамаля необходимо выбрать случайное число *k*, которое должно быть взаимно простым с *p*-1, и вычислить два числа:

1. Вычисляем открытый ключ отправителя, который состоит из двух чисел: *p* и *g*, и закрытый ключ, который состоит из числа *x*. Закрытый ключ должен храниться в секрете.
2. Вычисляем *y = gx mod p*, где *x* - закрытый ключ отправителя.
3. Выбираем случайное число *k*, которое должно быть взаимно простым с *p*-1.
4. Вычисляем первое число *a = gk mod p*.
5. Вычисляем второе число *b = myk mod p*, где *m* – это сообщение, которое мы хотим зашифровать.

Теперь давайте зашифруем сообщение "*ALICE*":

1. *p* = 167, *g* = 5, *x* = 23 (случайное число, выбранное в качестве закрытого ключа)
2. *y = gx mod p* = 5^23 *mod* 167 = 94
3. Выберем случайное число *k* = 7
4. *a = gk mod p* = 5^7 *mod* 167 = 124
5. *m* = 01100001 01001100 01001001 01000011 01000101 (в двоичном виде)

*b = m \* yk mod p* = 01100001 \* 94^7 *mod* 167 = 9

Теперь мы получили зашифрованное сообщение "*ALICE*". Давайте перейдем к дешифрованию.

Дешифрование осуществляется с помощью закрытого ключа x, который хранится только у получателя:

1. Вычисляем числа *s* = *ax mod p*.
2. Вычисляем числа *m* = *b* \* *s*(*p*-1-*x*) *mod p*.

Давайте дешифруем сообщение "*ALICE*":

1. *s* = *ax* *mod p* = 124^23 *mod* 167 = 153
2. *m* = *b* \* *s*(*p*-1-*x*) *mod p* = 9 \* 153(167-1-23) *mod* 167 = 01100001 01001100 01001001 01000011 01000101 (в двоичном виде).

Результатом дешифрования является исходное сообщение "ALICE" в двоичном виде. Если первая буква алфавита соответствует числу 0, то это означает, что первая буква "A" кодируется как 01000001 в двоичном виде. Аналогично, если первая буква алфавита соответствует числу 1, то это означает, что первая буква "A" кодируется как 00110001 в двоичном виде.

**Вывод:** в ходе лабораторной работы были приобретены практические навыки разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля. Было разработано приложение для реализации методов генерации ключевой информации и ее использования. Также была оценена скорость зашифрования/расшифрования.