Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование асимметричных шифров  
 RSA и Эль-Гамаля

Студент: Шулаков А.А.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

## 1. Описание приложения

Приложение написано на языке программирования C# и позволяет выполнить следующие задачи:

* генерация ключевой информации
* зашифрование текстовых документов на основе алгоритма RSA;
* расшифрование на основе алгоритма RSA;
* зашифрование на основе алгоритма Эль-Гамаля;
* расшифрование на основе алгоритма Эль-Гамаля;
* оценка времени выполнения операций зашифрования и расшифрования.

## 2. Методика выполнения поставленных задач

## 2.1. Алгоритм Эль-Гамаля

Для реализации генерации ключевой информации, выполнены следующие действия.

Во-первых, разработанное ПС генерирует простое число *p* случайным образом в диапазоне [2000; 2500]. Реализация процесса генерации числа *p* продемонстрирована на рисунке 2.1.

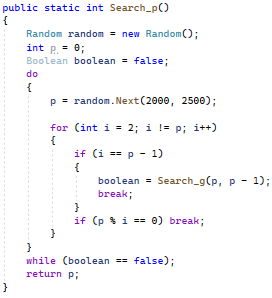


Рис. 2.1 – Генерация числа *p*

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 2.2.

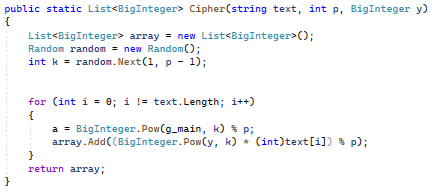


Рис. 2.2 – Реализация зашифрования

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 2.3.

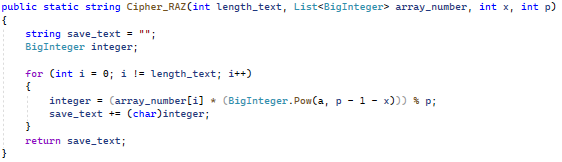


Рис. 2.3 – Реализация расшифрования

Протестируем разработанное программное средство. Результат работы приложения представлен на рисунке 2.4. 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 100 и 30 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.

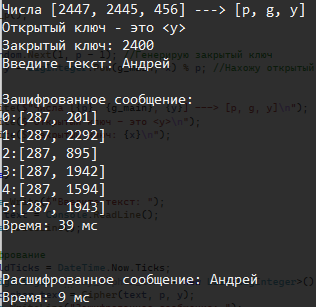


Рис. 2.4 – Результат алгоритма Эль-Гамаля

## 2.2. Алгоритм RSA

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

При первоначальном запуске приложения, открывается форма, изображенная на рисунке 2.5.

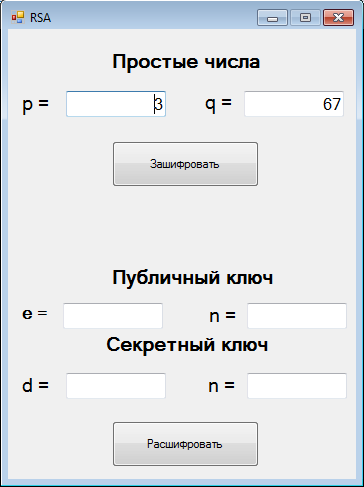


Рис. 2.5 – Форма алгоритма RSA

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 *ȹ(n)* = (*p*-1)×(*q*-1). Пара (*e,n*) будет являться открытым ключом алгоритма. Реализация нахождения числа е представлена на рисунке 2.6.

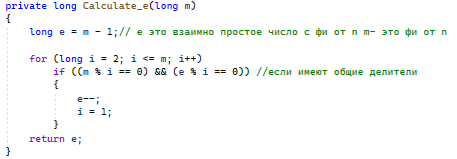


Рис. 2.6 – Нахождение числа е

В-третьих, необходимо вычислить число *d* по формуле Евлида, такое что *ed* = 1 mod (*ȹ*(*n*)). Пара (*d,n*) будет являться закрытым ключом алгоритма. Реализация нахождения числа d продемонстрирована на рисунке 2.7.

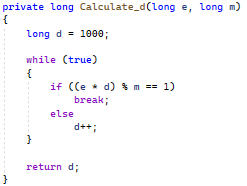


Рис. 2.7 – Нахождение числа d

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 2.8.

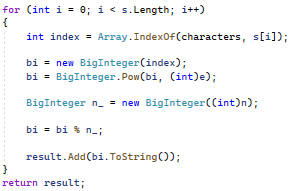


Рис. 2.8 – Реализация зашифрования

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

Заполним поля ввода *p, q* и нажмем кнопку «Зашифровать». Исходный текст из файла in.txt будет считан, а результат его зашифрования запишется в выходной файл out1.txt, который будет автоматически открыт приложением.

Далее, нажмем на кнопку «Расшифровать». Зашифрованный текст будет расшифрован по выше изложенному алгоритму, а результат будет записан в файл out2.txt, который также будет открываться автоматически. Результат работы приложения представлен на рисунке 2.9.

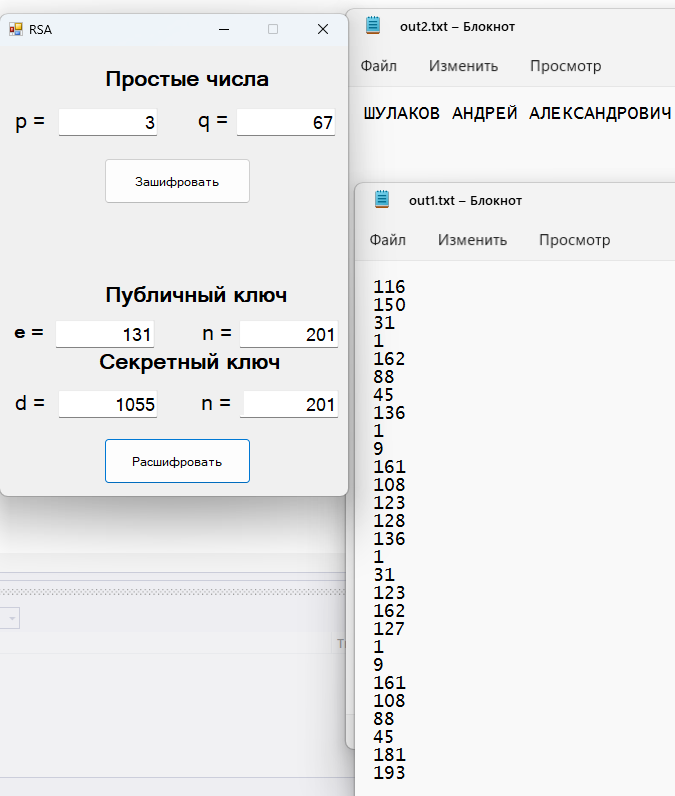


Рис. 2.9 – Результат алгоритма RSA

Графическая форма зависимости времени вычисления параметра *у*, функционально заданного выражением вида: *у ≡ ax mod n.* Реализация представлена на рисунке 2.10 и результат выполнения представлен на рисунке 2.11.

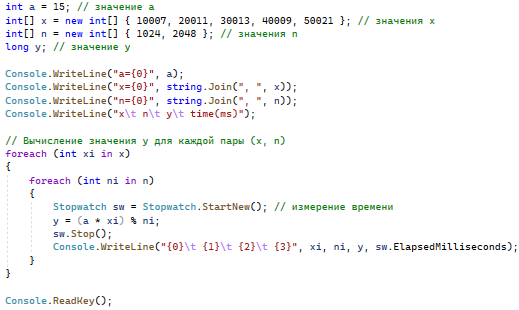


Рис. 2.10 – Реализация параметра *y* из выражения *у ≡ ax mod n*

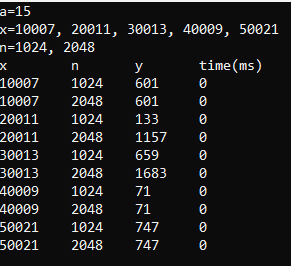


Рис. 2.11 – Результат выполнения выражения *у ≡ ax mod n*

Оценка выполнения шифрования и дешифрования алгоритмов RSA и Эль-Гамаля, представлено на рисунке 2.12. Из рисунка видно, что приложение работает исправно, график говорит о хороших результатах и большой производительности алгоритма Эль-Гамаля по сравнению с RSA.

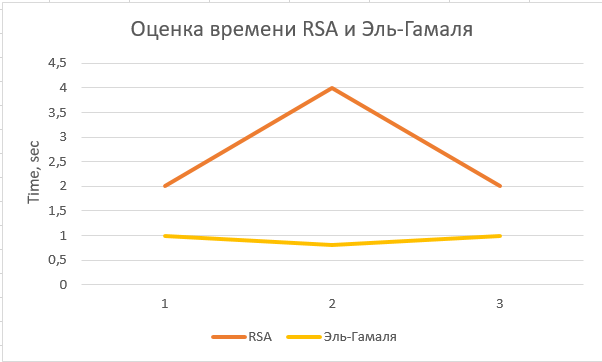


Рис. 2.12 – Оценка времени шифрования и дешифрования алгоритмов RSA и Эль-Гамаля

Оценка объема файлов шифротекстов двух алгоритмов, измерения в килобайтах, исходный текст – 0,140 KB, RSA – 2 KB, Эль-Гамаля – 3 KB. На рисунке 2.13 представлена гисторгамма размеров файлов каждого шифратекста, исходный текст помечен 1, RSA – 2, Эль-Гамаля – 3. Из рисунка разницы размеров исходного и зашифрованного сообщений стал очевидным недостаток алгоритма Эль-Гамаля: удвоение длины текста, связанный с использованием 2 чисел, соответствующих одному блоку исходного текста.



Рис. 2.13 – Оценка объема двух шифротекстов алгоритмов RSA и EDH

## 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.