МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование асимметричных шифров RSA и Эль-Гамаля

Студент: Евсеенко В. П.

ФИТ 3 курс 4 группа

Преподаватель: Савельева М. Г.

Минск 2025

Содержание

[Введение 3](#_Toc197604416)

[1 Теоретические сведения 4](#_Toc197604417)

[2 Практическое задание 6](#_Toc197604418)

[Задание 1 6](#_Toc197604419)

[Задание 2 7](#_Toc197604420)

[Анализ криптостойкости шифров 11](#_Toc197604421)

[Оценка времени 12](#_Toc197604422)

[Производительность обоих алгоритмов 13](#_Toc197604423)

[Относительное изменение объемов криптотекста 13](#_Toc197604424)

[Вывод 14](#_Toc197604425)

# **Введение**

Цель лабораторной работы:

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

Задачи:

* закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамаля;
* разработать приложение для реализации асимметричного зашифрования/расшифрования на основе алгоритмов RSA и Эль Гамаля;
* выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамаля;
* оценить скорость зашифрования/расшифрования реализованных шифров.

# **1 Теоретические сведения**

Асимметричная криптография основана на сложности решения некоторых математических задач. По существу, таких задач две:

Задачи:

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

Задача дискретного логарифмирования формулируется так: для данных целых чисел *а* и *b*, 1 < *а*, *b* < *n*, найти логарифм – такое целое число *х*, что

*ax* ≡ *b* (mod *n*),

если такое число существует.

Китайская теорема об остатках. В общем случае если разложение числа *N* на простые множители представляет собой *p*1*p*2…*pt* (некоторые простые числа могут встречаться несколько раз), то система уравнений

(*x* mod *pi*) ≡ *ai*,

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

Алгоритм RSA. 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; (*p* – 1)(*q* – 1) = φ(*n*) – функция Эйлера.

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

*ed* ≡ 1 (mod φ(*n*)).

Зашифрование. 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:

*ci* ≡ (*mi*)*e* mod *n.*

Расшифрование. Для расшифрования каждого зашифрованного блока производится вычисление вида:

*mi* ≡ (*ci*)*d*mod *n.*

Алгоритм Эль-Гамаля. 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.

# **2 Практическое задание**

# **Задание 1**

С помощью простого консольного приложения составить табличную или графическую форму зависимости времени вычисления параметра *у*, функционально заданного выражением вида:

*у* ≡ *ax* mod *n*,

от параметров: *а* (десятичные числа от 5 до 35; можно взять 1 или 2 числа), *х* (числа, желательно простые, из диапазона от 103 до 10100; для примера взять 5–10 чисел, равномерно распределенных в указанном диапазоне), *n* (для примера взять числа, в двоичном виде со стоящие из 1024 и 2048 битов).

Код приложения представлен в листинге 2.1.

|  |
| --- |
| const xValues: *BigInteger*[] = [      bigInt(1000),      bigInt(3000),      bigInt(5000),      bigInt(10000),      bigInt(15000)  ];  const nValues: *BigInteger*[] = [      bigInt(2).pow(1024),      bigInt(2).pow(2048)  ];  const aValues = [5, 10];  aValues.forEach(*a* => {      nValues.forEach(*n* => {          console.log("=================================");          console.log(`a: ${*a*}, n: ${*n*}`);          console.log("=================================");          console.log("   x       | Время вычисления (мс)");          console.log("=================================");          xValues.forEach(*x* => {              const timerStart = performance.now();              const y = bigInt(*a*).modPow(*x*, *n*);              const timerEnd = performance.now();              const elapsedMilliseconds = timerEnd - timerStart;              console.log(`${*x*}   |${elapsedMilliseconds}`);          });      });  }); |

Листинг 2.1 – Код приложения

Результат работы приложения приведен на рисунке 2.1.

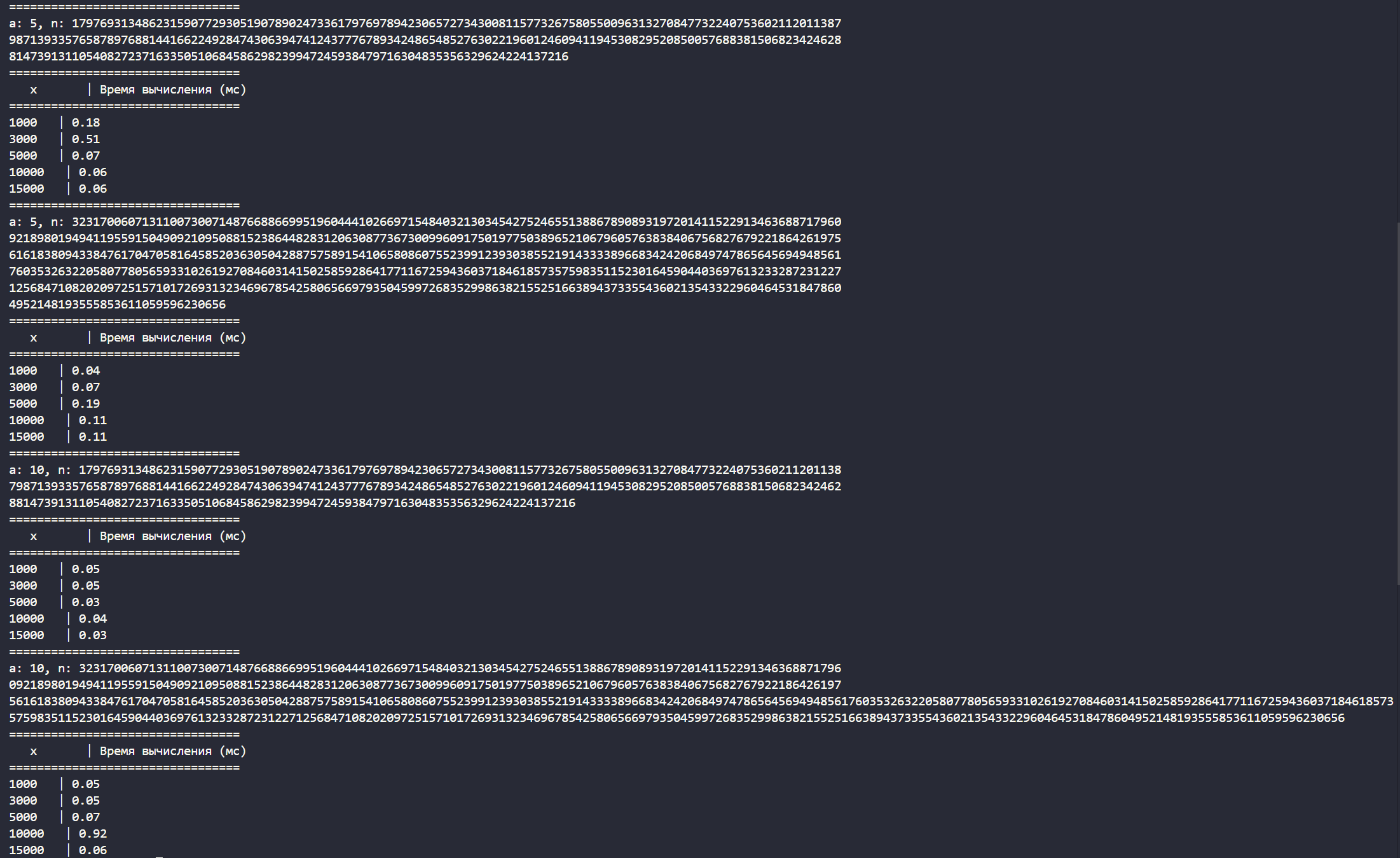


Рисунок 2.1 – Результат работы приложения

# **Задание 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-Гамаля;
* 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.

Функция rsaEncrypt принимает на вход оригинальный текст и публичный ключ и осуществляет зашифрование с помощью алгоритма RSA. Код функции представлен в листинге 2.2.

|  |
| --- |
| static rsaEncrypt(*data*: *string*, *publicKey*: *string*): *string* {const buffer = Buffer.from(*data*);          const encryptedData = *crypto*.publicEncrypt(*publicKey*, buffer);          return encryptedData.toString('base64');} |

Листинг 2.2 – Код функции rsaEncrypt

Функция rsaDecrypt принимает на вход зашифрованный текст и приватный ключ и осуществляет расшифрование с помощью алгоритма RSA. Код функции представлен в листинге 2.3.

|  |
| --- |
| static rsaDecrypt(*data*: *string*, *privateKey*: *string*): *string* {          const buffer = Buffer.from(*data*, 'base64');          const decryptedData = *crypto*.privateDecrypt(*privateKey*, buffer);          return decryptedData.toString('utf-8');      } |

Листинг 2.3 – Код функции rsaDecrypt

Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.2.

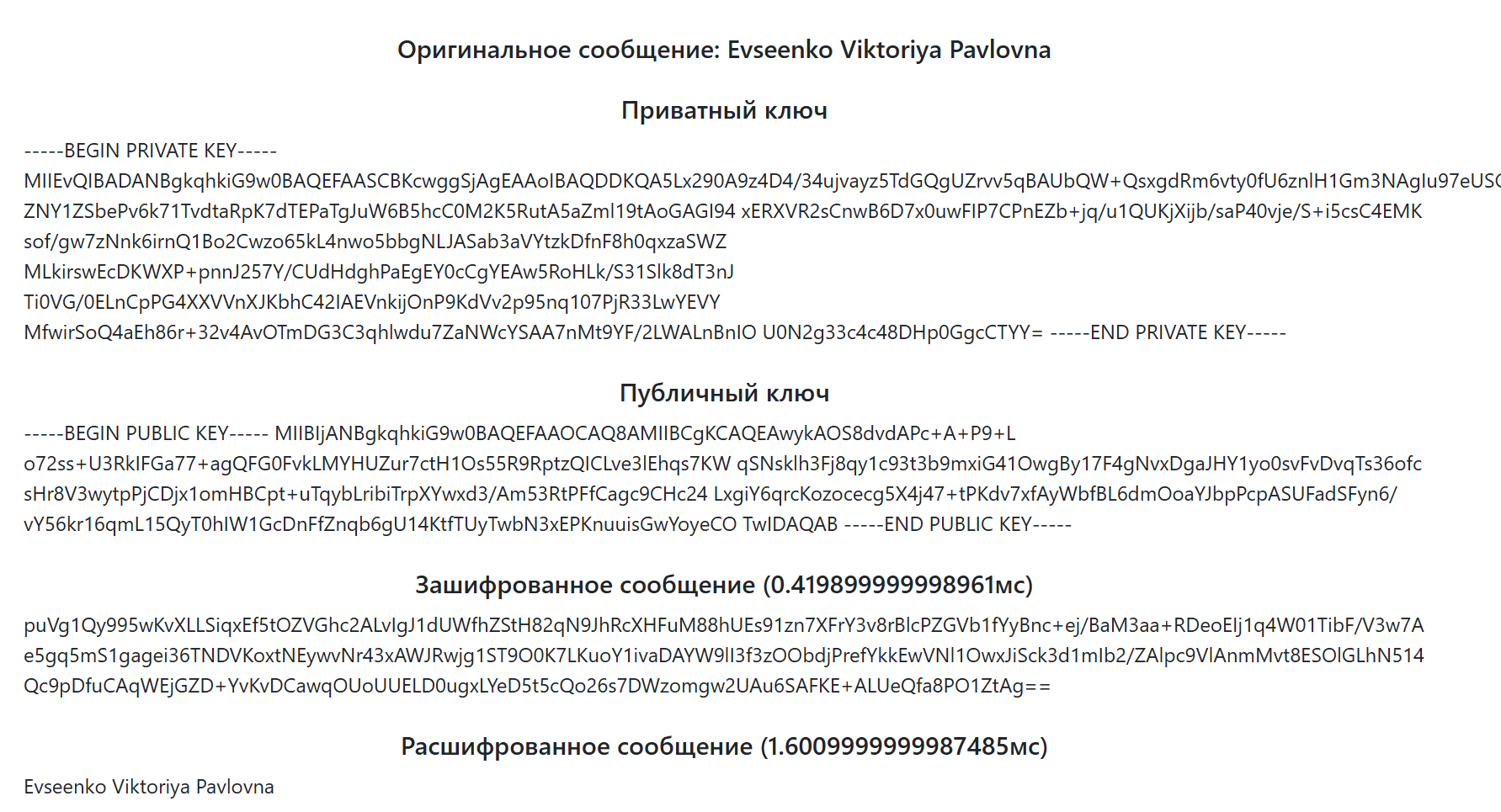


Рисунок 2.2 – Результат работы приложения

Функция encryptElGamal принимает на вход оригинальный текст и параметры публичного ключа и осуществляет зашифрование с помощью алгоритма Эль-Гамаля. Код функции представлен в листинге рисунке 2.4.

|  |
| --- |
| static encryptElGamal(*p*: *number*, *g*: *number*, *x*: *number*, *originalString*: *string*): *string* {          let result = "";          const y = *this*.exponentiation(*g*, *x*, *p*);          for (const char of *originalString*) {              const code = char.charCodeAt(0);              if (code > 0) {                  const k = Math.floor(Math.random() \* (*p* - 2)) + 1;                  const a = *this*.exponentiation(*g*, k, *p*);                  const b = *this*.multiplication(*this*.exponentiation(y, k, *p*), code, *p*);                  result += `${a} ${b} `;              }          }          return result;      } |

Листинг 2.4 – Функция шифрования алгоритмом Эль-Гамаля

Функция decryptElGamal принимает на вход зашифрованный текст и приватный параметр *x* и осуществляет расшифрование с помощью алгоритма Эль-Гамаля. Код функции представлен в листинге 2.5.

|  |
| --- |
| static decryptElGamal(*p*: *number*, *x*: *number*, *encryptedText*: *string*): *string* {          let result = "";          const arr = *encryptedText*.split(' ').filter(*xx* => *xx* !== "");          for (let i = 0; i < arr.length; i += 2) {              const a = parseInt(arr[i]);              const b = parseInt(arr[i + 1]);              if (a !== 0 && b !== 0) {                  const deM = *this*.multiplication(b, *this*.exponentiation(a, *p* - 1 - *x*, *p*), *p*);                  const m = *String*.fromCharCode(deM);                  result += m;              }          }          return result;      } |

Листинг 2.3 – Код функции decryptElGamal

Алгоритм Эль-Гамаля использует несколько параметров, которые являются ключевыми. Для начало было выбрано просто число *p* = 241. Далее было выбрано число *g* = 56, меньшее числа *p* и являющееся первообразным корнем числа *p*. Затем было выбрано число *x* = 9. Число *x* должно быть меньше, чем число *p*. Компонент *y* вычисляется внутри функции шифрования. Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.3

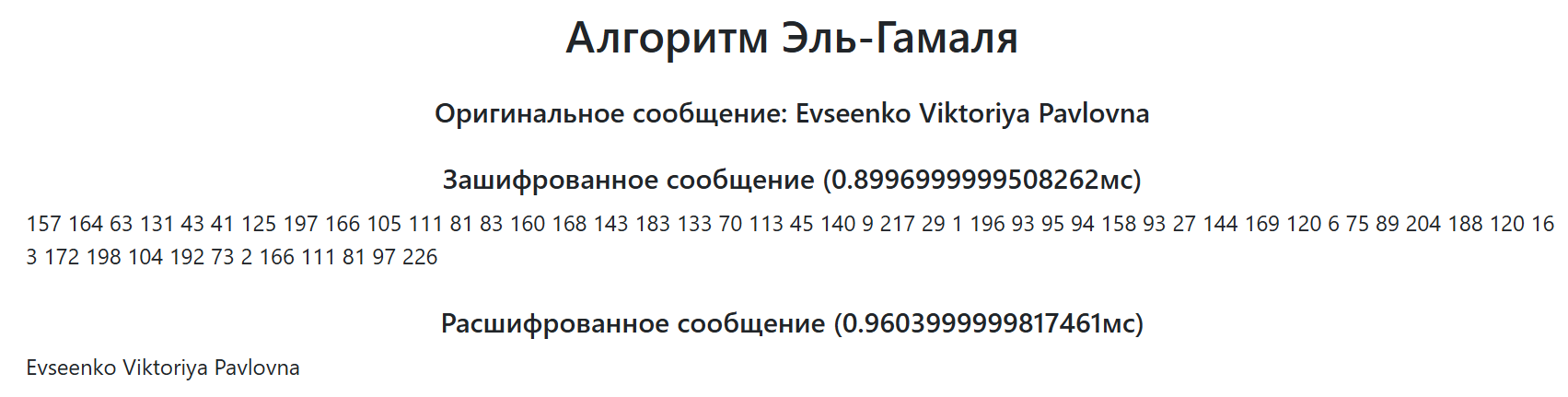


Рисунок 2.3 – Результат работы приложения

# **Анализ криптостойкости шифров**

Криптостойкость асимметричных алгоритмов основана на трудности решения определённых математических задач. В случае RSA — это задача факторизации больших составных чисел, а в основе алгоритма Эль-Гамаля лежит вычисление дискретного логарифма в конечном поле.

RSA остаётся одним из наиболее популярных асимметричных алгоритмов благодаря своей надёжности при использовании достаточно длинных ключей (не менее 2048 бит). Его безопасность обусловлена тем, что факторизация больших чисел требует значительных вычислительных ресурсов. Однако при использовании коротких ключей (например, меньше 1024 бит) криптостойкость RSA существенно снижается, и он становится уязвим для атак. Также существуют специфические уязвимости RSA, такие как атаки при малой публичной экспоненте или по открытому тексту. Эти риски минимизируются применением схемы дополнения, например, OAEP.

Алгоритм Эль-Гамаля опирается на сложность задачи дискретного логарифмирования, которая также считается вычислительно трудоёмкой при работе с большими простыми числами. Тем не менее, этот шифр чувствителен к повторному использованию случайных значений: утечка такого параметра может привести к раскрытию секретного ключа. Эль-Гамаль — вероятностный шифр, что делает его более устойчивым к криптоанализу, но требует строгого контроля над генерацией случайных чисел. При недостаточной длине ключей возможны атаки, такие как метод «baby-step – giant-step» или алгоритм Полларда.

При правильной реализации и соблюдении современных стандартов оба алгоритма обеспечивают высокий уровень безопасности. Однако потенциальная угроза со стороны квантовых компьютеров остаётся актуальной: такие технологии в перспективе смогут эффективно решать задачи, лежащие в основе RSA и Эль-Гамаля. Пока квантовые вычисления не достигли широкого применения, при корректной генерации ключей и случайных параметров оба алгоритма считаются надёжными для большинства криптографических целей.

# **Оценка времени**

Приложение выполняет оценку скорости шифрования и дешифрования сообщения шифров RSA и Эль-Гамаля. Результат нескольких попыток шифрования представлен на рисунках 2.4 и 2.5.

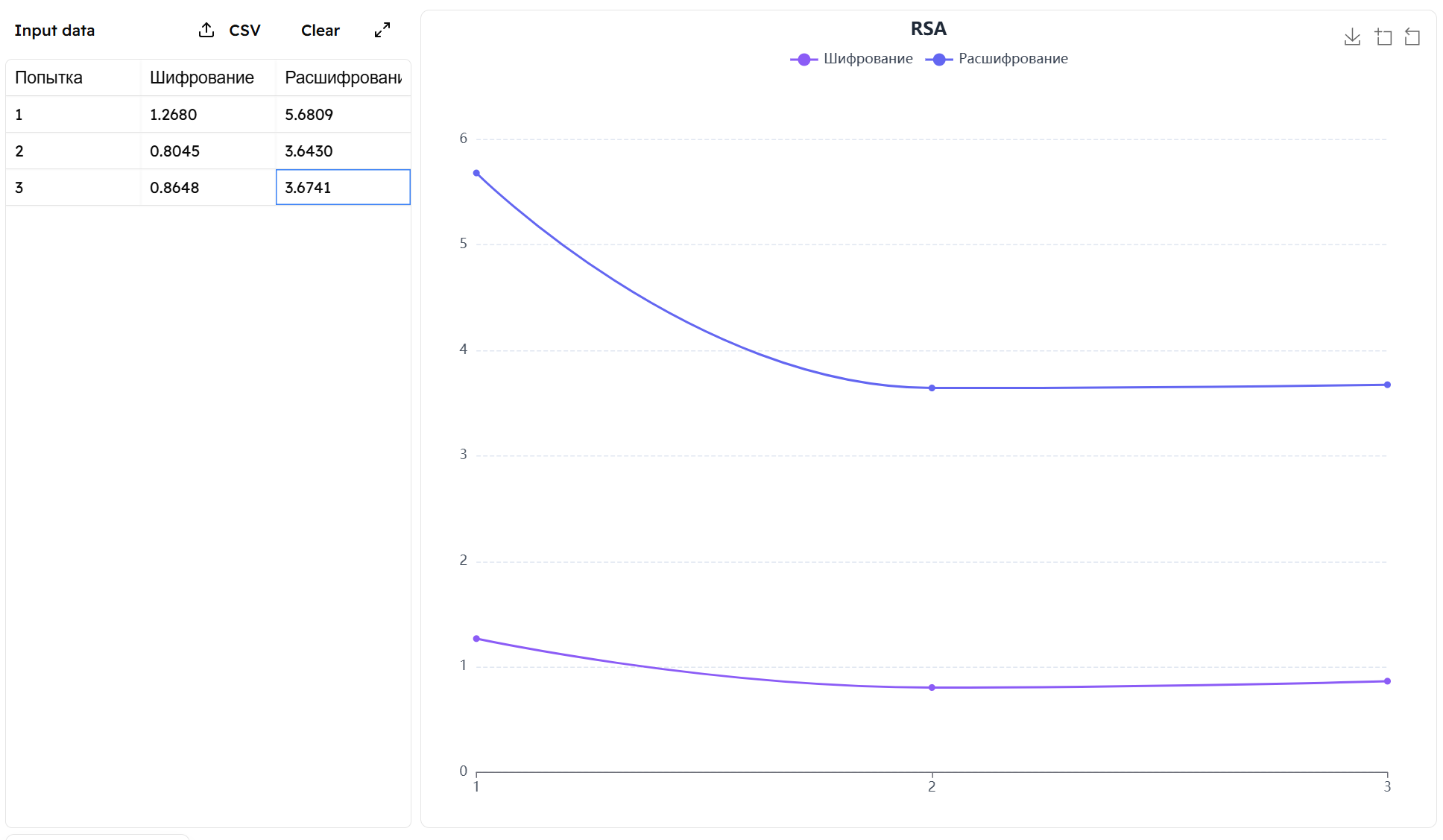


Рисунок 2.4 – Оценка времени операций зашифрования/расшифрования шифра RSA

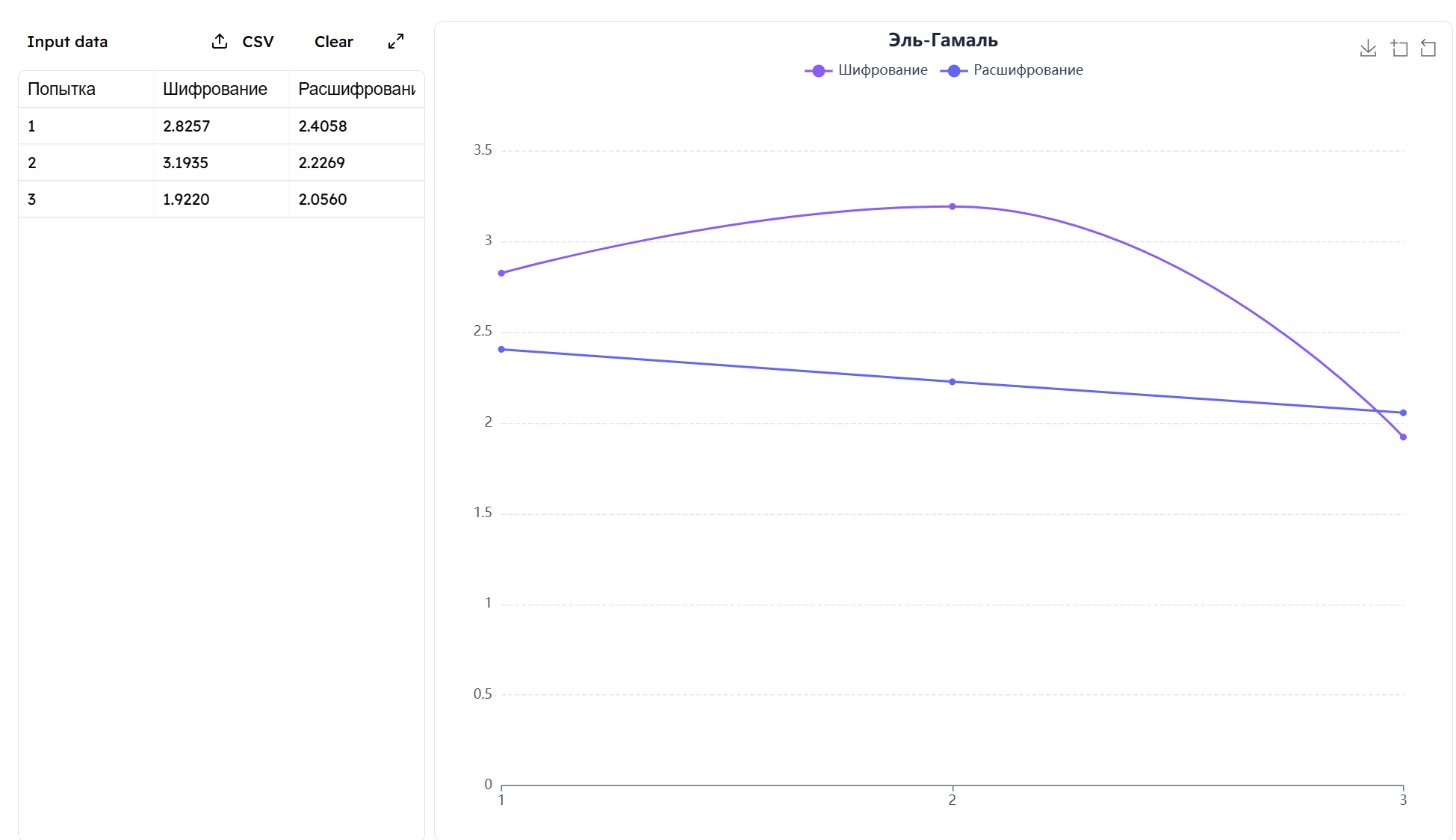


Рисунок 2.5 – Оценка времени операций зашифрования/расшифрования шифра Эль-Гамаля

# **Производительность обоих алгоритмов**

RSA. Алгоритм RSA характеризуется высокой скоростью шифрования, особенно при использовании малого публичного экспонента (например, *e* = 3 или 65537). Однако операция расшифрования относительно медленная, так как требует вычисления модульной экспоненты с большим приватным ключом. Важным преимуществом RSA является то, что он позволяет прямое шифрование без необходимости дополнительных случайных параметров.

Эль-Гамаль. Алгоритм Эль-Гамаля в целом медленнее RSA, поскольку каждая операция шифрования требует генерации случайного числа и двух экспоненциальных операций (для вычисления пары значений в шифртексте). Расшифровка также требует модульной экспоненты, но она проще, чем у RSA. Заметно, что Эль-Гамаль эффективнее в плане расшифрования, но проигрывает по скорости при шифровании.

# **Относительное изменение объемов криптотекста**

RSA. Объем шифртекста при использовании RSA обычно приблизительно равен размеру модуля *n*. То есть, если шифруемый блок открытого текста меньше модуля, то шифртекст будет по размеру равен модулю. Например, при использовании 2048-битного ключа каждый зашифрованный блок будет 256 байт. Таким образом, RSA увеличивает объем данных, но незначительно – примерно в 1.1–2 раза, в зависимости от способа кодирования и длины сообщения.

Эль-Гамаль. Алгоритм Эль-Гамаля производит два значения в результате шифрования одного блока текста: (*c*1, *c*2). Это означает, что объем шифртекста примерно в два раза превышает объем открытого текста (если не учитывать дополнительное кодирование). При использовании 2048-битного ключа каждая часть (*c*1, *c*2) будет около 256 байт, и общий объем шифртекста составит 512 байт на блок – то есть рост объема данных в ~2 раза.

Вывод: RSA обычно обеспечивает более высокую производительность при шифровании и умеренное увеличение объема данных.Эль-Гамаль требует больше вычислений при шифровании и увеличивает объем шифртекста примерно в два раза.При выборе между этими алгоритмами важно учитывать требования к скорости, объему данных и необходимости наличия вероятностного шифрования (что обеспечивает Эль-Гамаль).

# **Вывод**

В ходе лабораторной работы были изучены и реализованы два широко известных алгоритма асимметричного шифрования – RSA и Эль-Гамаль. Получены практические навыки разработки программных приложений для выполнения операций зашифрования и расшифрования, включая работу с ключами, открытым и закрытым доступом.

Выполнена оценка скорости выполнения криптографических операций, показавшая, что время зашифрования и расшифрования зависит от длины ключей и сложности алгоритма. Также проведён сравнительный анализ криптостойкости: RSA обеспечивает высокую надёжность при использовании больших ключей (2048 бит и более), а шифр Эль-Гамаля обладает высокой криптостойкостью за счёт сложности задачи дискретного логарифмирования.

Таким образом, цели и задачи лабораторной работы были достигнуты. Полученные результаты подтвердили теоретические особенности алгоритмов и продемонстрировали важность выбора параметров ключей для обеспечения безопасности и производительности шифрования.