МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование алгоритмов генерации и верификации электронной цифровой подписи

Студент: Евсеенко В. П.

ФИТ 3 курс 4 группа

Преподаватель: Савельева М. Г.

Минск 2025

Содержание

[Введение 3](#_Toc201143042)

[1 Теоретические сведения 4](#_Toc201143043)

[2 Практическое задание 6](#_Toc201143044)

[Оценка времени 11](#_Toc201143045)

[Вывод 13](#_Toc201143046)

# **Введение**

Цель лабораторной работы:

* изучение алгоритмов генерации и верификации электронной цифровой подписи и приобретение практических навыков их реализации.

Задачи:

* закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций генерации и верификации электронной цифровой подписи (ЭЦП);
* получить навыки практической реализации методов генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эль-Гамаля и Шнорра, а также DSA;
* разработать приложение для реализации заданных алгоритмов генерации и верификации ЭЦП;
* оценить скорость генерации и верификации ЭЦП.

# **1 Теоретические сведения**

Электронная цифровая подпись – контрольная характеристика сообщения, которая вырабатывается с использованием личного ключа, проверяется с использованием открытого ключа, служит для контроля целостности и подлинности сообщения и обеспечивает невозможность отказа от авторства.

Таким образом, ЭЦП выполняет те же функции, что и собственноручная (поставленная «от руки») подпись:

• аутентифицирование лица, подписавшего сообщение;

• контроль целостности подписанного сообщения;

• защита сообщения от подделок;

• доказательство авторства лица, подписавшего сообщение, если это лицо отрицает свое авторство.

Важнейшие отличительные особенности ЭЦП:

• ЭЦП представляет собой бинарную последовательность (в отличие от графического образа, каковым является подпись от руки);

• указанная бинарная последовательность зависит от содержания подписываемого сообщения.

Электронная цифровая подпись – бинарная (или в ином виде) последовательность символов, являющаяся реквизитом электронного документа, зависящая от содержания этого документа и предназначенная для подтверждения целостности и подлинности электронного документа.

ЭЦП на основе RSA. Здесь можно рассматривать две ситуации:

• сообщение *М*o подписывается и передается в открытом (незашифрованном) виде;

• сообщение *М*o подписывается и передается в зашифрованном виде.

При этом подпись *S* вычисляется на основе соотношения:

*S* ≡ (*H*(*M*o))dо mod*n*o.

Передаваемое сообщение *М*' = *М*o||*S*. Соответственно, операция расшифрования на приемной стороне (получатель анализирует *М*п||*S*) производится с известной модификацией ключей:

*H*(*M*o) ≡ (*S*) ео mod*n*o.

Далее вычисляется *Н*(*M*п). Если *H*(*M*o) = *H*(*M*п), подпись верифицирована.

Если подписываемое сообщение *М*(*М*') также должно передаваться в зашифрованном виде, то обычно *М*' шифруется на стороне отправителя стандартным образом: с помощью открытого ключа получателя (*е*п и *n*п), который перед основным процессом верификации подписи расшифровывает послание своим тайным ключом: *d*п и *n*п. Далее осуществляются вычисления и анализ, как и в первом случае.

Основное отличие в применении расчетов в алгоритме Эль-Гамаля состоит в том, что результатом зашифрования является только одна пара чисел, а не пара для каждого блока исходного сообщения. Причем в рассматриваемом случае таким сообщением является хеш подписываемого документа: Н(Mо).

Алгоритм ЭЦП К. Шнорра (К. Schnorr) является вариантом алгоритма ЭЦП Эль-Гамаля. Одной из особенностей ЭЦП Эль-Гамаля является то, что число *p* должно быть очень большим, чтобы сделать действительно трудной проблему дискретного логарифма. Рекомендуемая длина *p* должна составлять по крайней мере 1024 бита. Чтобы уменьшить размер подписи, Шнорр предложил новую схему, но с уменьшенным размером подписи.

Ключевая информация: *p* – простое число в диапазоне от 512 до 1024 битов; *q* –160-битное простое число, делитель (*p* – 1); любое число *g* (*g* ≠ 1) такое, что

*gq* ≡ 1 mod *p*.

Числа *p*, *g*, *q* являются открытыми и могут применяться группой пользователей. Выбирается число *х* < *q* (*х* является тайным ключом) и вычисляется последний элемент открытого ключа:

*y* ≡ *g*–*х* mod *p*.

Секретный ключ имеет длину не менее 160 битов.

# **2 Практическое задание**

Было необходимо разработать приложение, выполняющее генерацию и верификацию ЭЦП с помощью алгоритмов RSA, Эль-Гамаля и Шнорра.

Метод createDigitalSignature класса RSA принимает на вход оригинальный текст и подписывает его с помощью алгоритма RSA. Код метода представлен на рисунке 2.1.

|  |
| --- |
| public createDigitalSignature(*text*: string): BigInteger {          const hash = crypto.createHash('sha256').update(*text*, 'utf8').digest();          return bigInt(hash.readBigInt64LE()).modPow(this.d, this.n);      } |

Листинг 2.1 – Метод генерации ЭЦП алгоритмом RSA

Метод verifyDigitalSignature класса RSA принимает на вход полученный от отправителя текст, электронную подпись и осуществляет верификацию подписи с помощью алгоритма RSA. Код метода представлен в листинге 2.2.

|  |
| --- |
| public verifyDigitalSignature(*text*: string, *digitalSign*: BigInteger): boolean {          const signBytes: BigInteger = *digitalSign*.modPow(this.e, this.n);          const receivedHash = crypto.createHash('sha256').update(*text*, 'utf8').digest();          return bigInt(receivedHash.readBigInt64LE()).eq(signBytes);      } |

Листинг 2.2 – Метод верификации ЭЦП алгоритмом RSA

Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.1.

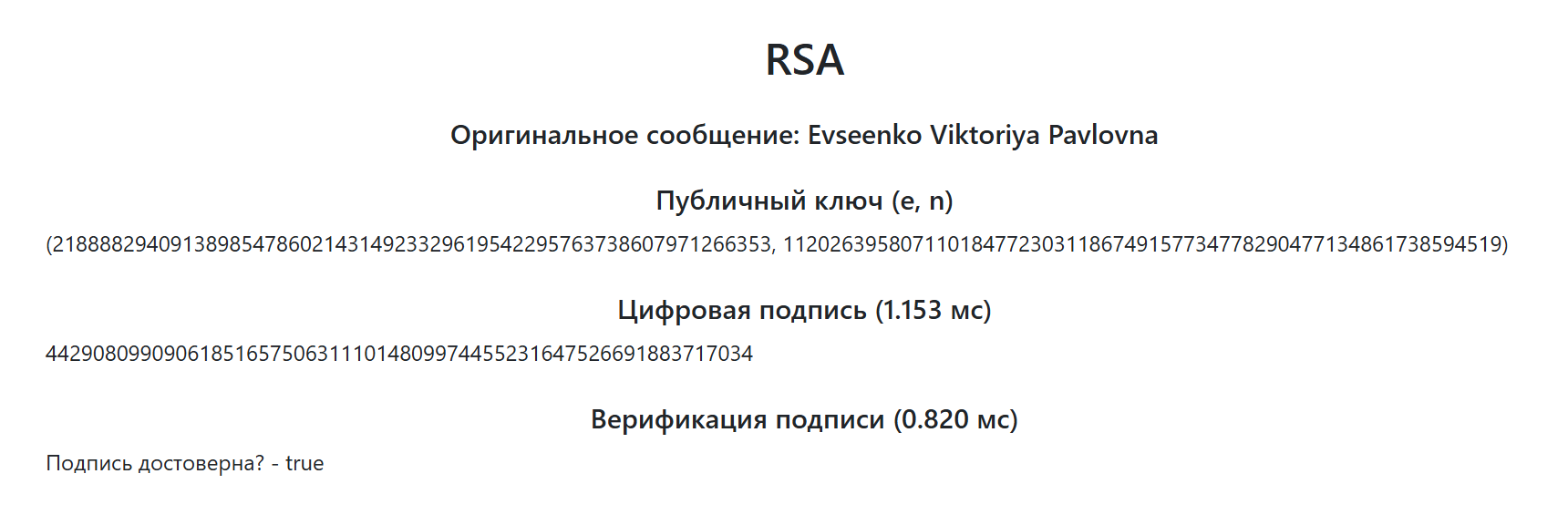


Рисунок 2.1 – Результат работы приложения

Метод createDigitalSignature класса ElGamal принимает на вход оригинальный текст и подписывает его с помощью алгоритма Эль-Гамаля. Подписью являются два числа *а* и *b* (*S* = {*a*, *b*}). Число *а* вычисляется по формуле: *a* ≡ *gk* mod *p*. При вычислении параметра *b*, решается уравнение *Н*(*M*о) ≡ (*x*\**a* + *k*\**b*) mod (*p* – 1). Код метода представлен в листинге 2.3.

|  |
| --- |
| public createDigitalSignature(*message*: string): BigInteger[] {          const hash = crypto.createHash('sha256').update(*message*, 'utf8').digest();          let digitalSignI: BigInteger[];          do {              let k = bigInt.randBetween(bigInt(2), this.p.subtract(2));              while (!isCoprime(k, this.p.subtract(1))) {                  k = bigInt.randBetween(bigInt(2), this.p.subtract(2));              }              digitalSignI = [];              digitalSignI[0] = this.g.modPow(k, this.p);              let temp = bigInt(hash.readBigInt64LE()).subtract(this.x.multiply(digitalSignI[0]));              temp = temp.multiply(k.modInv(this.p.subtract(1))).mod(this.p.subtract(1));              if (temp.isNegative()) {                  temp = this.p.subtract(1).subtract(temp.abs());              }              digitalSignI[1] = temp;          } while (digitalSignI[1].equals(0));          return digitalSignI;      } |

Листинг 2.3 – Метод генерации ЭЦП алгоритмом Эль-Гамаля

Метод verifyDigitalSignature класса ElGamal принимает на вход полученный от отправителя текст, электронную подпись (массив из двух параметров: *а* и *b*) и осуществляет верификацию подписи с помощью алгоритма Эль-Гамаля. Для верификации подписи вычисляется хеш полученного сообщения *Н*(*М*п) = *h*. Далее нужно убедиться, что выполняется равенство *yaab* ≡ *gh*mod *p*. Код метода представлен в листинге 2.4.

|  |
| --- |
| public verifyDigitalSignature(*message*: string, *digitalSignature*: BigInteger[]): boolean {          const hash = crypto.createHash('sha256').update(*message*, 'utf8').digest();          const leftPart = this.g.modPow(hash.readBigInt64LE(), this.p);          const rightPart = this.y.modPow(*digitalSignature*[0], this.p)              .multiply(*digitalSignature*[0].modPow(*digitalSignature*[1], this.p))              .mod(this.p);          return leftPart.equals(rightPart);      } |

Листинг 2.4 – Метод верификации ЭЦП алгоритмом Эль-Гамаля

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

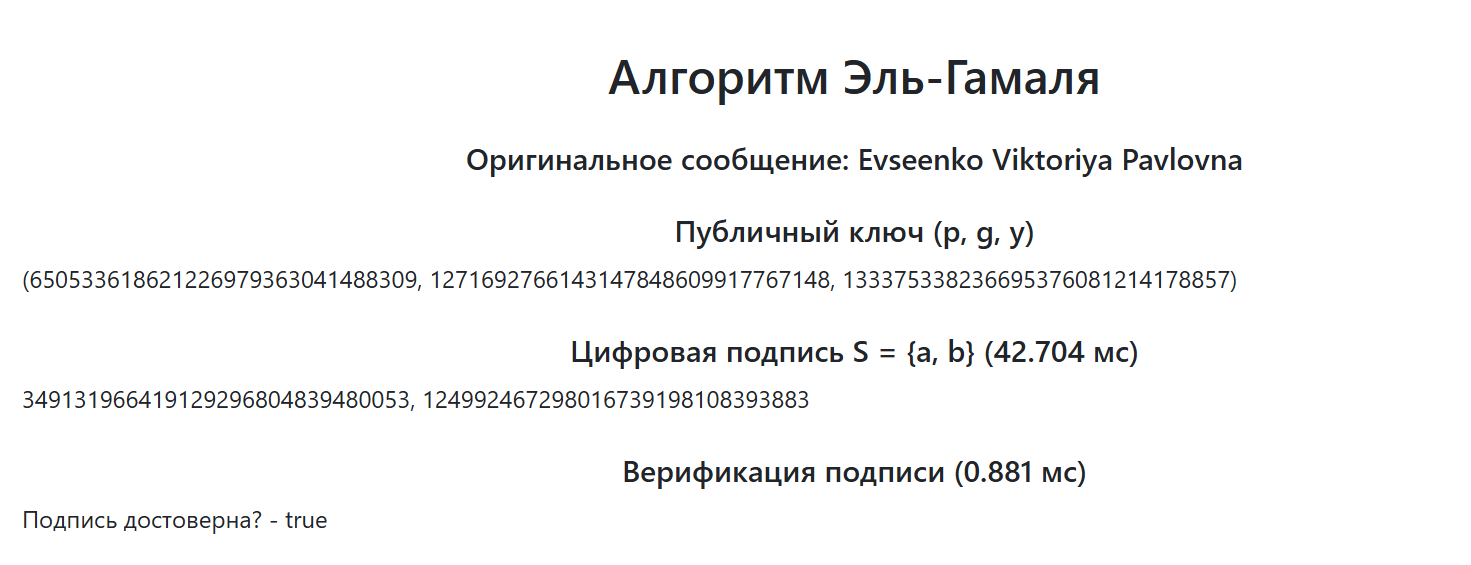


Рисунок 2.2 – Результат работы приложения

Метод createDigitalSignature класса Schorr принимает на вход оригинальный текст и подписывает его с помощью алгоритма Шнорра. Ключевой информацией в алгоритме Шнорра являются числа *p*, *g*, *q*, *y*. *p* – простое число в диапазоне от 512 до 1024 битов; *q* –160-битное простое число, делитель (*p* – 1); любое число *g* (*g* ≠ 1) такое, что *gq* ≡ 1 mod *p*, число *y* ≡ *g*–*х* mod *p*. Для подписи сообщения *М*о выбирается случайное число *k* (1 < *k* < *q*) и вычисляет параметр *а*: *а* ≡ *gk* mod *p*. Далее вычисляется хеш от канкатенации сообщения *М*о и числа *а*: *h* = *H*(*M*o||*a*). Далее вычисляется значение *b*: *b* ≡ (*k* + *x*\**h*) mod *q*. Получателю отправляются *М*' = *М*о||*S*; *S* = {*h*, *b*}. Код метода представлен в листинге 2.5.

|  |
| --- |
| public generateDigitalSignature(message: string): BigInteger[] {          let k: BigInteger;          do {              k = bigInt.randBetween(2, this.q.subtract(1));          } while (!(k.compare(1) > 0 && k.compare(this.q) < 0));          const a = this.g.modPow(k, this.p);          message += a.toString();          const hash = crypto.createHash('sha256').update(message, 'utf8').digest();          return [              bigInt(hash.readBigUInt64LE()),              k.add(this.x.multiply(bigInt(hash.readBigUInt64LE()))).mod(this.q)          ];      } |

Листинг 2.5 – Метод генерации ЭЦП алгоритмом Шнорра

Метод verifyDigitalSignature класса Schorr принимает на вход полученный от отправителя текст, электронную подпись и осуществляет верификацию подписи с помощью алгоритма Шнорра. Для проверки подписи вычисляется *Х* ≡ *gbyh* (mod *p*). Затем он проверяет выполнение равенства: *h* = *Н*(*M*п||*Х*). Подпись достоверна, если равенство выполняется. Код метода представлен в листинге 2.6.

|  |
| --- |
| public verifyDigitalSignature(*message*: string, *digitalSignature*: BigInteger[]): boolean {          let x = this.g.modPow(*digitalSignature*[1], this.p)              .multiply(this.y.modPow(*digitalSignature*[0], this.p)).mod(this.p);  *message* += x.toString();          const receivedHash = crypto.createHash('sha256').update(*message*, 'utf8').digest();          return *digitalSignature*[0].equals(receivedHash.readBigUInt64LE());      } |

Листинг 2.6 – Метод верификации ЭЦП алгоритмом Шнорра

Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.3.

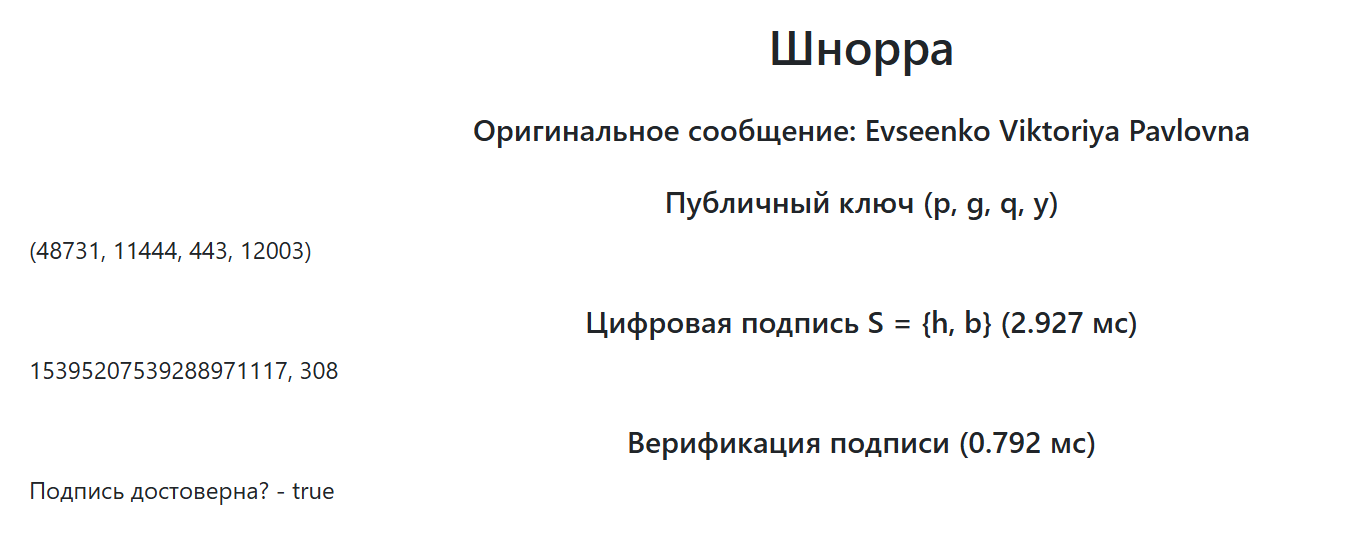


Рисунок 2.3 – Результат работы приложения

# **Оценка времени**

Приложение выполняет оценку скорости алгоритма генерации и верификации ЭЦП разных алгоритмов. Результат нескольких попыток хеширования представлен на рисунке 2.4, 2.5, 2.6.

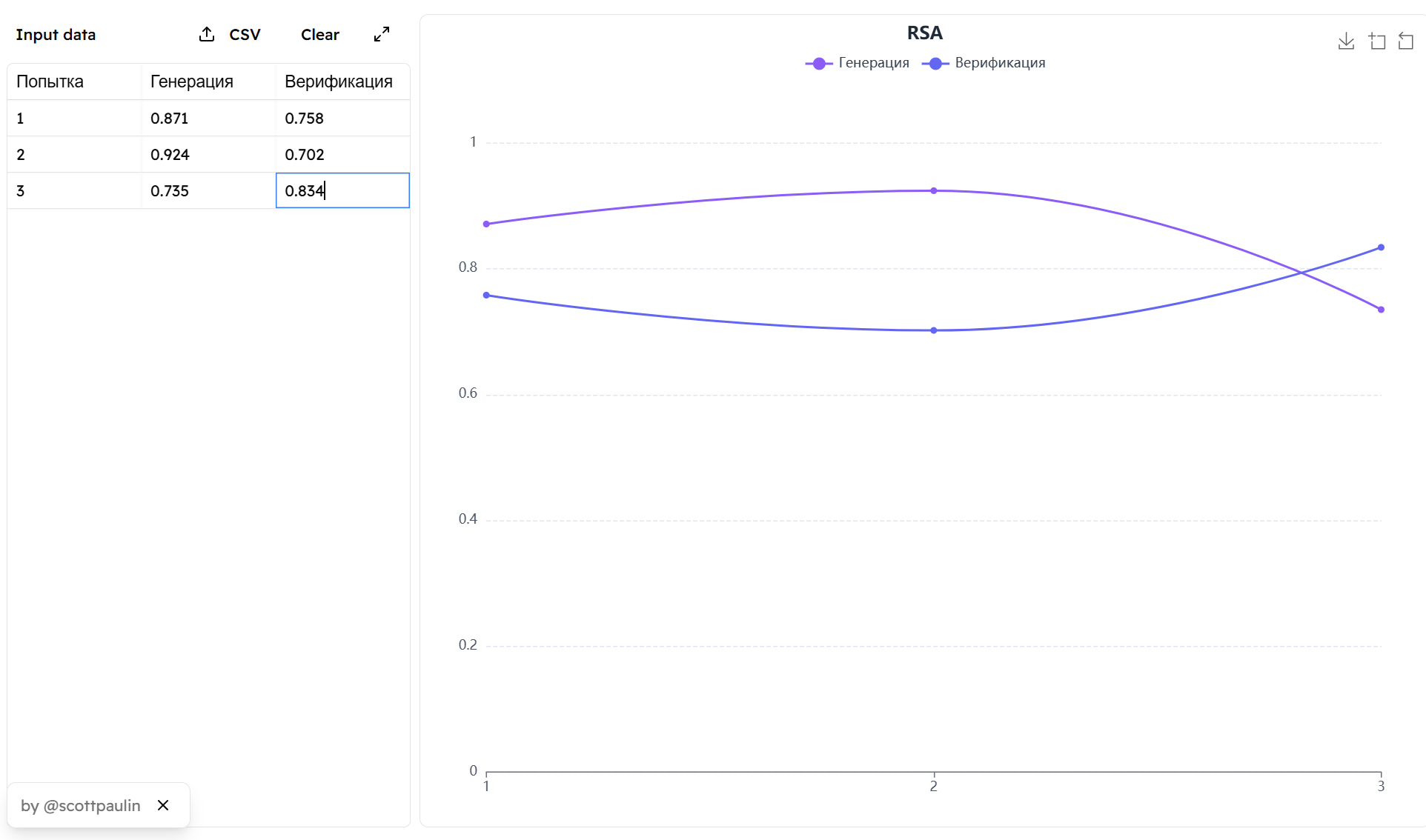


Рисунок 2.4 – Оценка времени операций алгоритма RSA

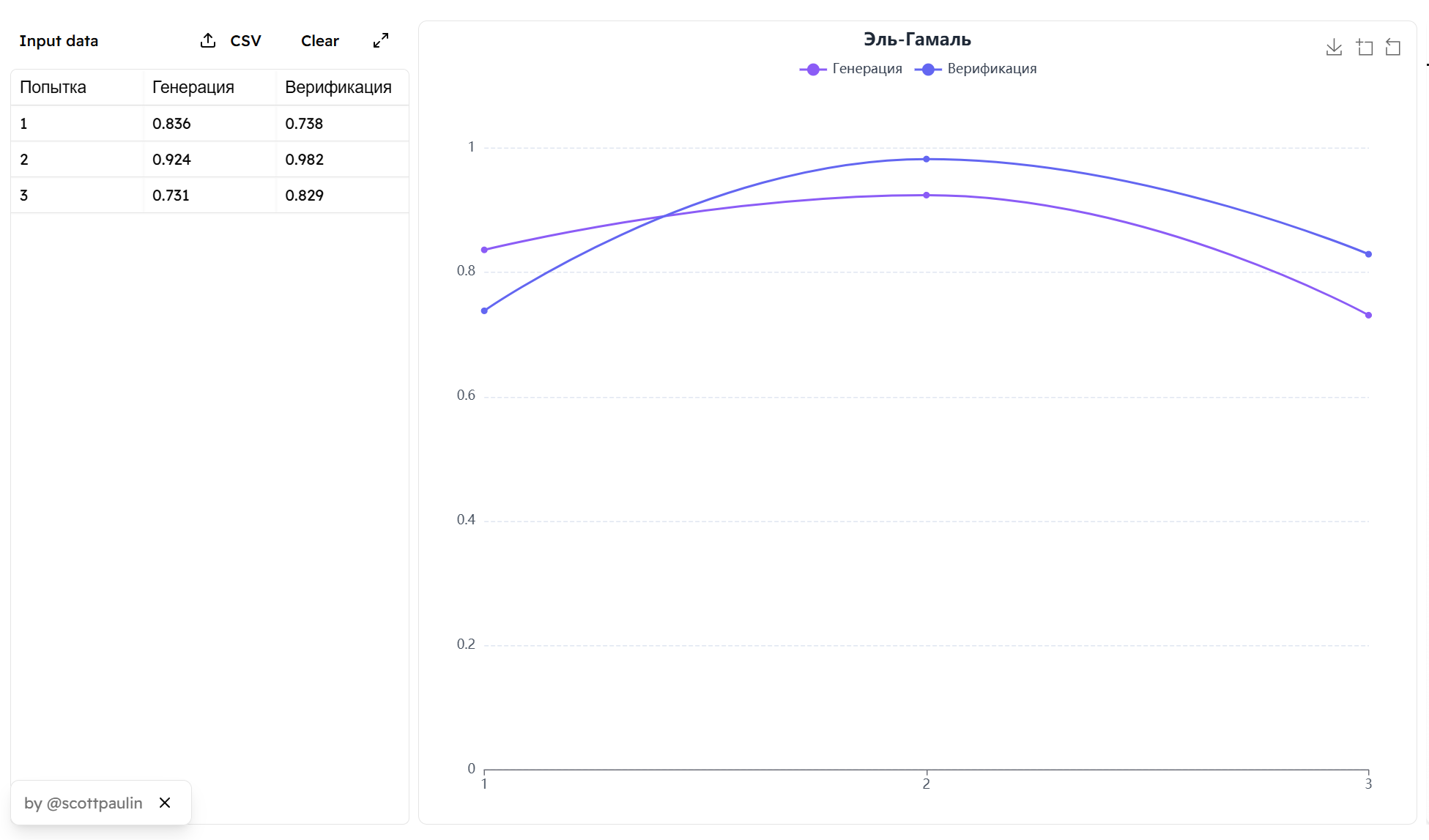


Рисунок 2.5 – Оценка времени операций алгоритма Эль-Гамаля

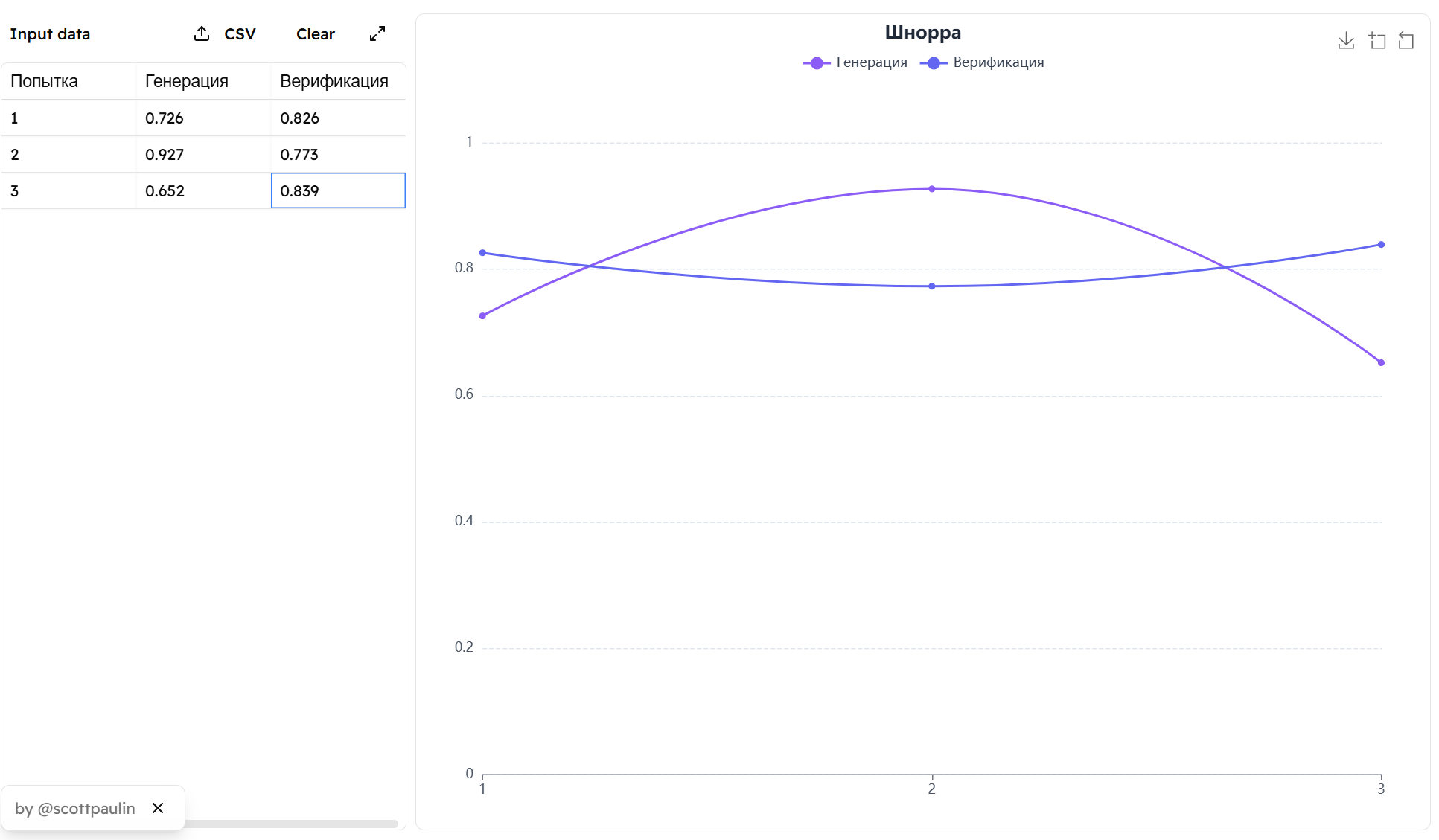


Рисунок 2.6 – Оценка времени операций алгоритма Шнорра

# **Вывод**

В ходе выполнения лабораторной работы были изучены и реализованы алгоритмы генерации и верификации электронной цифровой подписи (ЭЦП) с использованием методов RSA, Эль-Гамаля, Шнорра и DSA. Были закреплены теоретические знания по алгебраическому описанию и принципам функционирования данных алгоритмов, а также получены практические навыки их программной реализации.

Разработано приложение, позволяющее выполнять генерацию ключей, создание цифровой подписи и проверку подлинности подписанных сообщений. При реализации использовались методы хеширования сообщений с целью обеспечения целостности и неподдельности данных.

В ходе экспериментов была произведена оценка времени генерации и верификации ЭЦП для каждого из реализованных алгоритмов. Полученные результаты позволили сравнить эффективность и производительность разных методов.

Таким образом, достигнуты все цели и решены поставленные задачи. Работа способствовала углублению понимания принципов криптографии, расширению практических навыков программирования и повышению уровня знаний в области информационной безопасности.