Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Информационная безопасность**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №12. Исследование алгоритмов генерации и верификации электронной цифровой подписи**

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

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций генерации и верификации электронной цифровой подписи (ЭЦП).
2. Получить навыки практической реализации методов генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эль-Гамаля и Шнорра, а также DSA.
3. Разработать приложение для реализации заданных алгоритмов генерации и верификации ЭЦП.
4. Оценить скорость генерации и верификации ЭЦП.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Электронная цифровая подпись (ЭЦП) является важным элементом современных информационных систем, использующих методы и технологии криптографического преобразования информации.

*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.

*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.

Перечисленные выше функции ЭЦП могут быть реализованы на основе классических методов зашифрования/расшифрования:

* на основе симметричных систем (с тайным ключом);
* на основе симметричных систем и посредника;
* на основе асимметричных систем (с открытым ключом).

Классическая технология использования ЭЦП предусматривает подписание не самого сообщения (обозначим его здесь *М*о), а его хеша, *H*(*M*о). Это сокращает время генерации/верификации подписи и снижает вероятность появления случайных ошибок в итоговом документе.

При использовании *ЭЦП на основе RSA* можно рассматривать две ситуации:

* сообщение *М*o подписывается и передается в незашифрованном виде;
* сообщение *М*o подписывается и передается в зашифрованном виде.

Подпись *S* вычисляется на основе соотношения:

*S* ≡ (*H*(*M*o))*d*о mod *n*o.

Передаваемое сообщение *М'* = *М*o||*S*.

Соответственно, операция расшифрования на приемной стороне (получатель анализирует *М*п||*S*) будет производиться в соответствии с формулой с известной модификацией ключей:

*H*(*M*o) ≡ (*S*)*е*о 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.

*Алгоритм DSA* (алгоритм цифровой подписи), или *DSS* (стандарт цифровой подписи), является одним из известных, нередко и сейчас применяемых. В алгоритме используются следующие параметры: *p* – простое число длиной от 64 до 1024 битов (число должно быть кратно 64); *q –* 160-битный простой множитель (*р* – 1). Далее вычисляется число *g*:

*g = v*(*p* – 1)/*g* mod *p*,

где *v –* любое число, меньшее (*р* – 1), для которого выполняется условие:

*v*(*p* – 1)/*g* mod *p* > 1*.*

Числа *p*, *q*, *v* могут использоваться группой лиц. Еще один элемент открытого ключа *у* вычисляется в соответствии с выражением

*y* ≡ *gx* mod *p*,

где *x* < *q*; *х* – закрытый ключ.

Ключевая информация отправителя для *ЭЦП Эль-Гамаля* состоит из тех же элементов, что и ключи в DSA. Основное отличие в применении расчетов 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*(*M*о).

Итак, ключевая информация отправителя: открытый ключ: *y*, *g* и *р*; тайный ключ: *х*. Чтобы подписать сообщение *М*о, обладатель используемых ключей должен выбрать, как и в предыдущей схеме, случайное число *k*, взаимно простое с (*р* – 1). Затем вычисляется числа *а* и *b*, являющиеся цифровой подписью (*S* = {*a*, *b*}):

*a* ≡ *gk* mod *p*;

для вычисления b с помощью расширенного алгоритма Евклида решается уравнение

*Н*(*M*о) ≡ (*xa + kb*) mod (*p* – 1).

Получателю отправляется сообщение *М'* = *М*о||*S*.

Для верификации подписи вычисляется хеш полученного сообщения *Н*(*М*п) = *h*. Далее нужно убедиться, что выполняется равенство

*yaab* ≡ *gh* mod *p*.

Если равенство выполняется, подпись верифицируется

Алгоритм *ЭЦП К. Шнорра* является вариантом алгоритма ЭЦП Эль-Гамаля.

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.

Для *подписи сообщения М*о выбирается случайное число *k* (1 *< k < q*) и вычисляет параметр *а*:

*а* ≡ *gk* mod *p*.

Далее вычисляется хеш от конкатенации сообщения *М*о и числа *а*: *h = H*(*M*o*||a*). Обратим внимание, что хэш-функция непосредственно не применяется к сообщению. Создается хеш-образ подписываемого сообщения, спереди присоединенного к числу *а*. Далее вычисляется значение *b*:

*y* ≡ (*k + xh*) mod *q*.

Получателю отправляются *М'* = *М*о||*S*; *S* = {*h*, *b*}.

Для *проверки подписи* получатель вычисляет

*Х* ≡ *gbyh* (mod *p*)*.*

Затем он проверяет выполнение равенства*: h = Н*(*M*п*||Х*). Подпись достоверна, если равенство выполняется.

**Ход работы**

**Задание 1.** Разработать авторское оконное приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции: генерацию и верификацию ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнорра; оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах.

Метод createDigitalSignature класса RSA принимает на вход оригинальный текст и подписывает его с помощью алгоритма RSA.

createDigitalSignature(text) {

    const hash = crypto.createHash('sha256').update(text,

'utf8').digest();

    return bigInt(hash.readBigInt64LE()).modPow(this.d, this.n);

}

Листинг 12.1 – Функция createDigitalSignature класса RSA

Метод verifyDigitalSignature класса RSA принимает на вход полученный от отправителя текст, электронную подпись и осуществляет верификацию подписи с помощью алгоритма RSA.

verifyDigitalSignature(text, digitalSign, e, n) {

    const signBytes = digitalSign.modPow(e, n);

    const receivedHash = crypto.createHash('sha256').update(text,

'utf8').digest();

    return bigInt(receivedHash.readBigInt64LE()).eq(signBytes);

}

Листинг 12.2 – Функция verifyDigitalSignature класса RSA

Результат работы данных функций с исходным сообщением «Kozeka Elizaveta Maksimovna» представлен на рисунке 12.1.

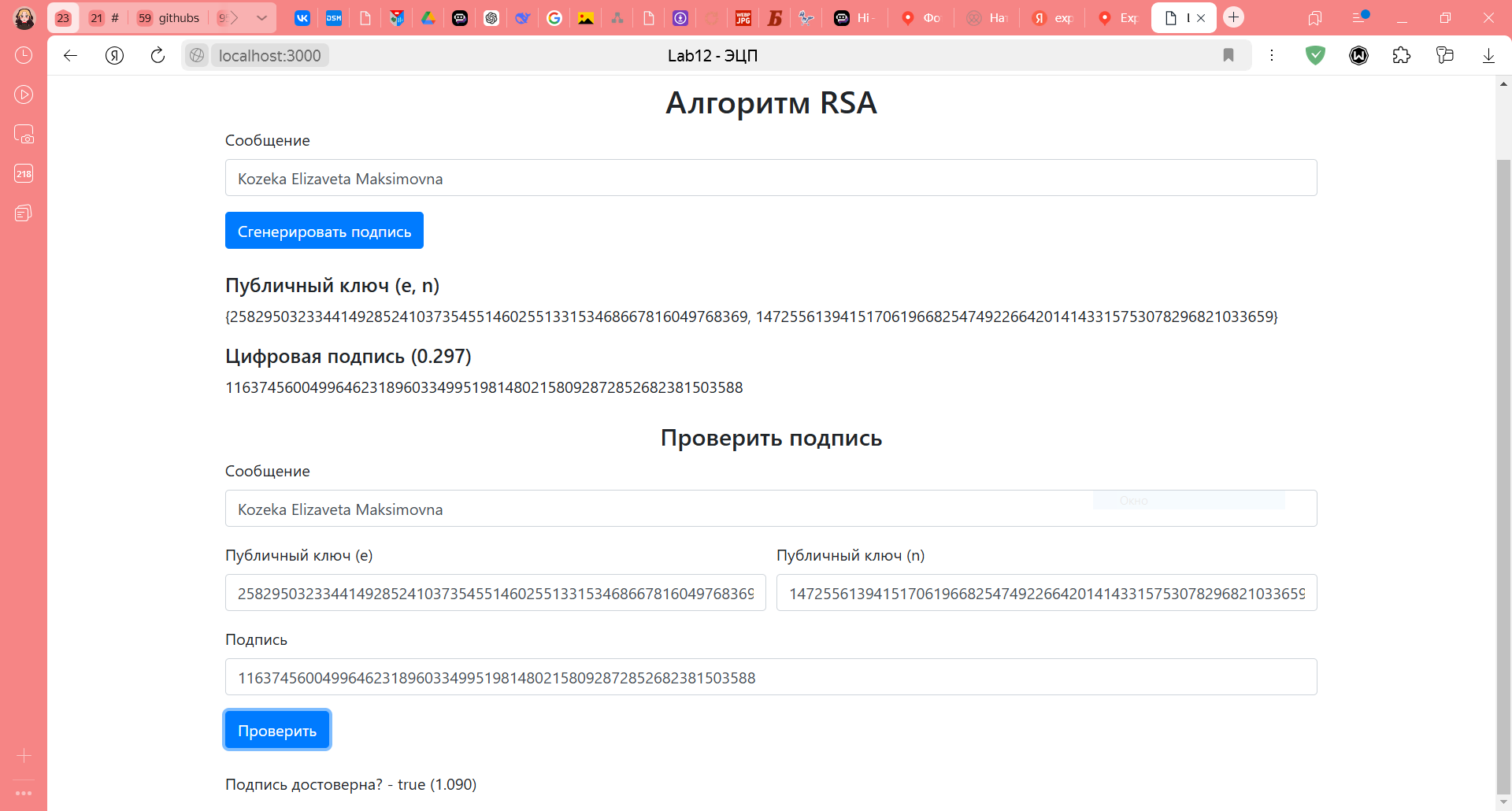


Рисунок 12.1 – Результат генерации и проверки ЭЦП (алгоритм RSA)

Генерация ЭЦП заняла примерно 0,3 мс. Процесс верификации занял немного больше времени – 1 мс.

Метод createDigitalSignature класса ElGamal принимает на вход оригинальный текст и подписывает его с помощью алгоритма Эль-Гамаля.

createDigitalSignature(message) {

    const hash = crypto.createHash('sha256').update(message,

'utf8').digest();

    let digitalSignI;

    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 12.3 – Функция createDigitalSignature класса ElGamal

Метод verifyDigitalSignature класса ElGamal принимает на вход полученный от отправителя текст, электронную подпись и осуществляет верификацию подписи с помощью алгоритма Эль-Гамаля.

verifyDigitalSignature(message, digitalSignature, p, g, y) {

    const hash = crypto.createHash('sha256').update(message,

'utf8').digest();

    const leftPart = g.modPow(hash.readBigInt64LE(), p);

    const rightPart = y.modPow(digitalSignature[0], p)

        .multiply(digitalSignature[0].modPow(digitalSignature[1], p))

        .mod(p);

    return leftPart.equals(rightPart);

}

Листинг 12.4 – Функция verifyDigitalSignature класса ElGamal

Результат работы данных функций с исходным сообщением «Kozeka Elizaveta Maksimovna» приведен на рисунке 12.2.

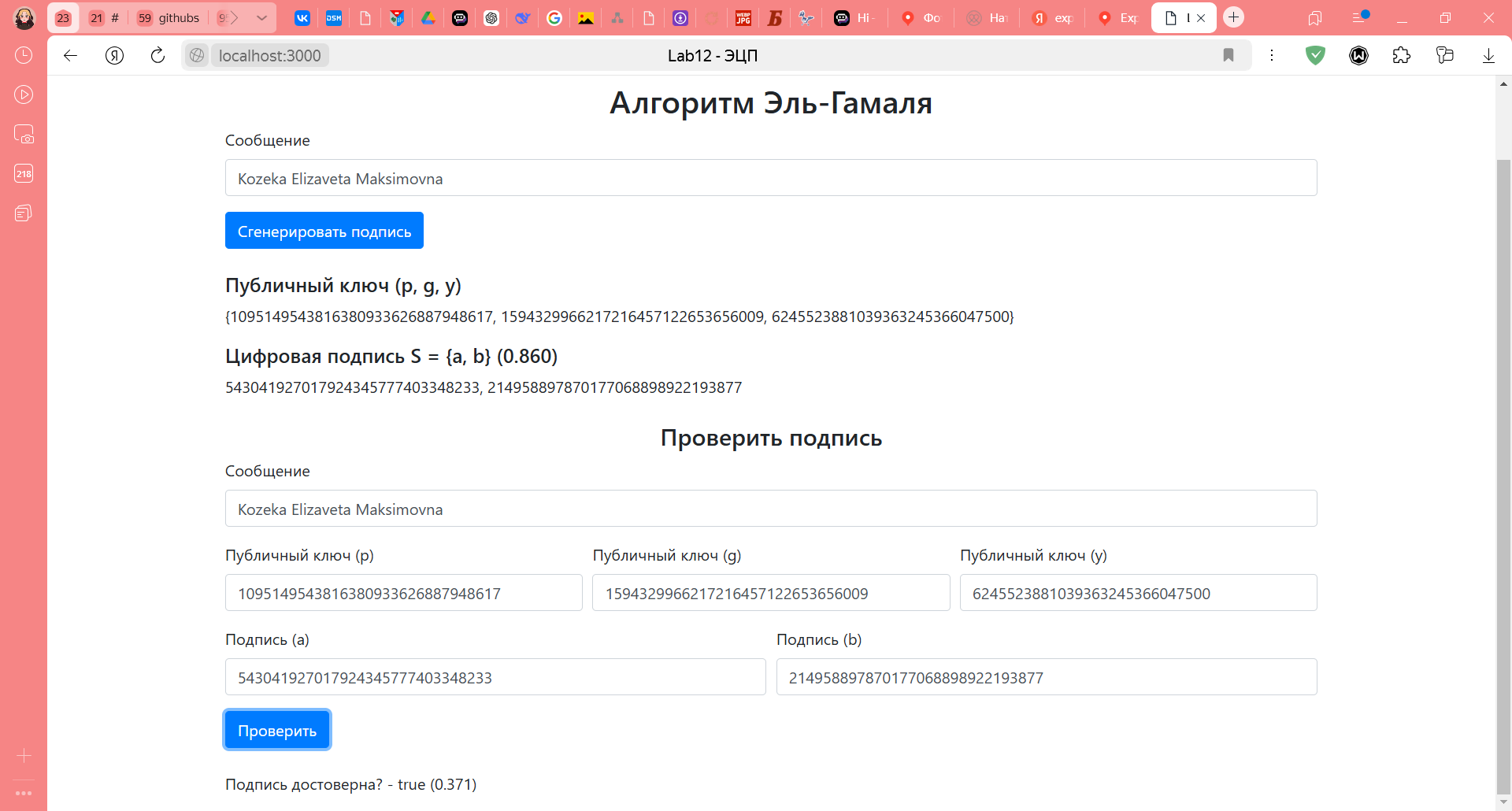


Рисунок 12.2 – Результат генерации и проверки ЭЦП (алгоритм Эль-Гамаля)

Генерация ЭЦП заняла 0,86 мс. Процесс верификации занял меньше времени по сравнению с алгоритмом RSA – 0,37 мс.

Метод generateDigitalSignature класса Schnorr принимает на вход оригинальный текст и подписывает его с помощью алгоритма Шнорра.

generateDigitalSignature(message) {

    let k;

    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 12.5 – Функция generateDigitalSignature класса Schnorr

Метод verifyDigitalSignature класса Schnorr принимает на вход полученный от отправителя текст, электронную подпись и осуществляет верификацию подписи с помощью алгоритма Шнорра.

verifyDigitalSignature(message, digitalSignature, p, g, q, y) {

    let x = g.modPow(digitalSignature[1], p)

        .multiply(y.modPow(digitalSignature[0], p)).mod(p);

    message += x.toString();

    const receivedHash = crypto.createHash('sha256').update(message,

'utf8').digest();

    return digitalSignature[0].equals(receivedHash.readBigUInt64LE());

}

Листинг 12.6 – Функция verifyDigitalSignature класса Schnorr

Результат работы данных функций с исходным сообщением «Kozeka Elizaveta Maksimovna» приведен на рисунке 12.3.

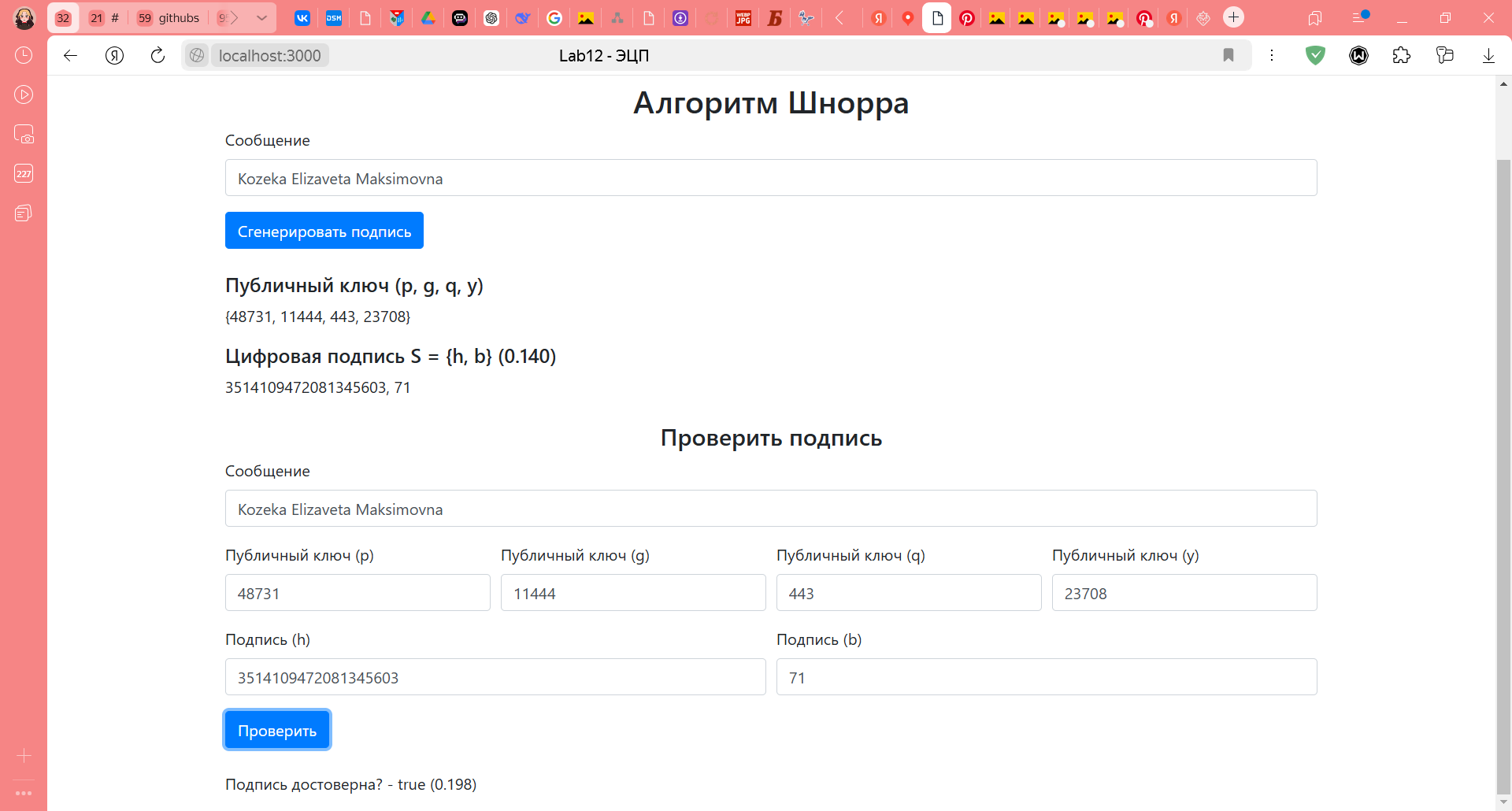


Рисунок 12.3 – Результат генерации и проверки ЭЦП (алгоритм Шнорра)

Генерация ЭЦП заняла 0,14 мс. Процесс верификации занял практически столько же времени – 0,19 мс.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип генерации и верификации ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнорра. Также было разработано приложение, выполняющее генерацию и верификацию ЭЦП на основе данных алгоритмов. Была оценена скорость генерации и верификации подписи.