Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №12

**ИССЛЕДОВАНИЕ АЛГОРИТМОВ ГЕНЕРАЦИИ И ВЕРИФИКАЦИИ ЭЛЕКТРОННОЙ ЙИФРОВОЙ ПОДПИСИ**

Студент: Сосновец М.И.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О.А.

Минск 2025

**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. Остановимся на важнейших свойствах и иных информационных и фактологических характеристиках ЭЦП. Более подробные сведения из предметной области можно найти в [2, 4, 29, 50].

Определение 1. 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.

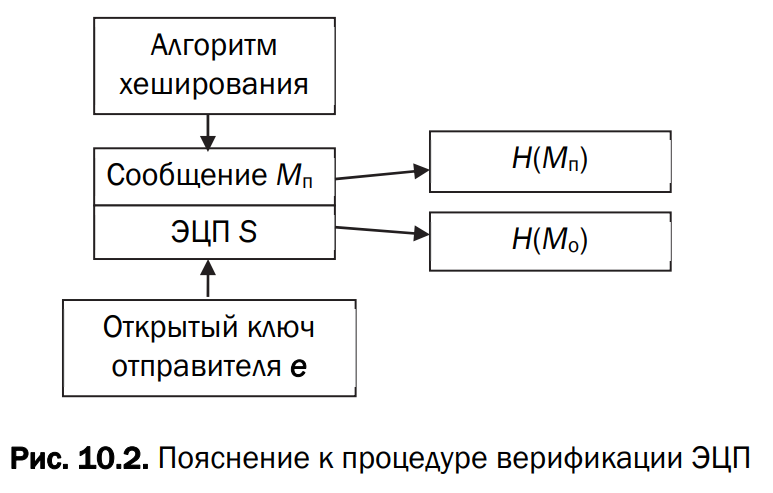
Важнейшие отличительные особенности ЭЦП:

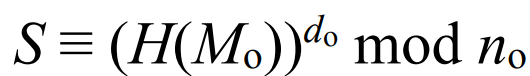
* ЭЦП представляет собой бинарную последовательность (в отличие от графического образа, каковым является подпись от руки);
* указанная бинарная последовательность зависит от содержания подписываемого сообщения.

Определение 2. 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.

Классическая технология использования ЭЦП предусматривает подписание не самого сообщения (обозначим его здесь Мо), а его хеша, H(Mо). Это сокращает время генерации/верификации подписи и снижает вероятность появления случайных ошибок в итоговом документе. Основу рассматриваемых протоколов составляют методы асимметричной криптографии и эллиптических кривых.

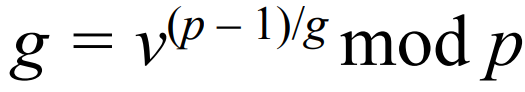
Важное свойство цифровой подписи заключается в том, что ее может проверить (верифицировать) каждый, кто имеет доступ к открытому ключу ее автора. На рис. 10.2 показан в общем виде порядок процесса верификации (без учета использования служебной информации). Заметим, что в общем случае версии исходного документа (Мо) и полученного (Мп) могут отличаться.



ЭЦП на основе RSA. Здесь можно рассматривать две ситуации: • сообщение Мo подписывается и передается в открытом (незашифрованном) виде; • сообщение Мo подписывается и передается в зашифрованном виде. Первый случай соответствует схеме и операциям, представленным на рис. 10.1 и рис. 10.2. При этом подпись S вычисляется на основе известного из лабораторной работы № 8 соотношения (8.5): 

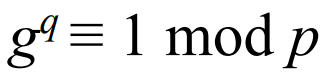
Соответственно, операция расшифрования на приемной стороне (получатель анализирует Мп||S) будет производиться в соответствии с формулой (8.6) с известной модификацией ключей: 

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 (еп и nп), который перед основным процессом верификации подписи расшифровывает послание своим тайным ключом: dп и nп. Далее осуществляются вычисления и анализ, как и в первом случае.

ЭЦП на основе DSA. Алгоритм DSA (Digital Signature Algorithm – алгоритм цифровой подписи), или DSS (Digital Signature Standard – 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: 



ЭЦП Эль-Гамаля. Ключевая информация отправителя для ЭЦП создается точно так же, как это описано в материалах к лабораторной работе № 8. Она 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(Mо). Итак, ключевая информация отправителя: открытый ключ: y, g и р; тайный ключ: х. Чтобы подписать сообщение Мо, обладатель используемых для ЭЦП 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}):

ЭЦП Шнорра. Рассматриваемая схема является основой стандарта ЭЦП в Беларуси. 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 .

**Практическое задание**

1. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться результатами выполнения предыдущих лабораторных работ, а также доступными библиотеками либо программными кодами.

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.

Для вычисления хешей можно также воспользоваться доступными online-средствами, например katvin (https://katvin.com/tools/ hash-generator.html).

2. Для выполнения необходимых операций передачи (по сети)/верификации информации обменяться открытой ключевой информацией с получателем подписанного сообщения для каждого исследуемого алгоритма (по согласованию с преподавателем).

В лабораторной работе было реализовано консольное приложение, которое выполняет генерацию и верификацию ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнорра.

Алгоритм RSA работает на основе сложности разложения больших чисел на множители. Сначала генерируются два больших простых числа, из которых создаются открытый и закрытый ключи. Подпись формируется путем шифрования хеша документа закрытым ключом, а проверка — расшифровкой открытым ключом и сравнением с хешем.

Алгоритм RSA выполняется так:

1. Генерация ключей:

* Используется функция crypto.generateKeyPairSync для создания пары RSA-ключей длиной 2048 бит
* Открытый ключ (publicKey) кодируется в формате PEM/SPKI
* Закрытый ключ (privateKey) кодируется в формате PEM/PKCS8
* Замеряется время генерации ключей

1. Создание цифровой подписи:

* Сообщение хешируется с помощью SHA-256
* Создается объект подписи (createSign) с алгоритмом SHA-256
* Хеш сообщения передается в объект подписи
* Подпись создается с использованием закрытого ключа и кодируется в base64
* Замеряется время создания подписи

1. Проверка подписи:

* Сообщение снова хешируется с помощью SHA-256
* Создается объект проверки (createVerify) с алгоритмом SHA-256
* Хеш сообщения передается в объект проверки
* Подпись проверяется с использованием открытого ключа
* Замеряется время проверки подписи

Реализация генерации ключей алгоритма RSA представлена в листинге 1.1.

|  |
| --- |
| console.log("Генерация ключей RSA...\n");  const startKeyGen = process.hrtime();  const { publicKey, privateKey } = crypto.generateKeyPairSync('rsa', {    modulusLength: 2048,    publicKeyEncoding: { type: 'spki', format: 'pem' },    privateKeyEncoding: { type: 'pkcs8', format: 'pem' }  }); |

Листинг 1.1 - Реализация генерации ключей алгоритма RSA

Создание цифровой подписи алгоритма RSA показано в листинге 1.2.

|  |
| --- |
| function createDigitalSignature(*message*) {    const startTime = process.hrtime();     const hash = crypto.createHash('sha256').update(*message*).digest('hex');      const signer = crypto.createSign('sha256');    signer.update(hash);    signer.end();      const signature = signer.sign(privateKey, 'base64');      const endTime = process.hrtime(startTime);    console.log(`Время создания подписи: ${endTime[0]}s ${endTime[1] / 1000000}ms`);      return signature;  } |

Листинг 1.2 - Реализация цифровой подписи алгоритма RSA

Проверка цифровой подписи алгоритма RSA показана в листинге 1.3.

|  |
| --- |
| function verifyDigitalSignature(*message*, *signature*) {    const startTime = process.hrtime();      const hash = crypto.createHash('sha256').update(*message*).digest('hex');      const verifier = crypto.createVerify('sha256');    verifier.update(hash);    verifier.end();      const isValid = verifier.verify(publicKey, *signature*, 'base64');      const endTime = process.hrtime(startTime);    console.log(`Время проверки подписи: ${endTime[0]}s ${endTime[1] / 1000000}ms`);      return isValid;  } |

Листинг 1.3 - Реализация проверки цифровой подписи алгоритма RSA

Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма RSA, а также оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах показан на рисунке 1.1.

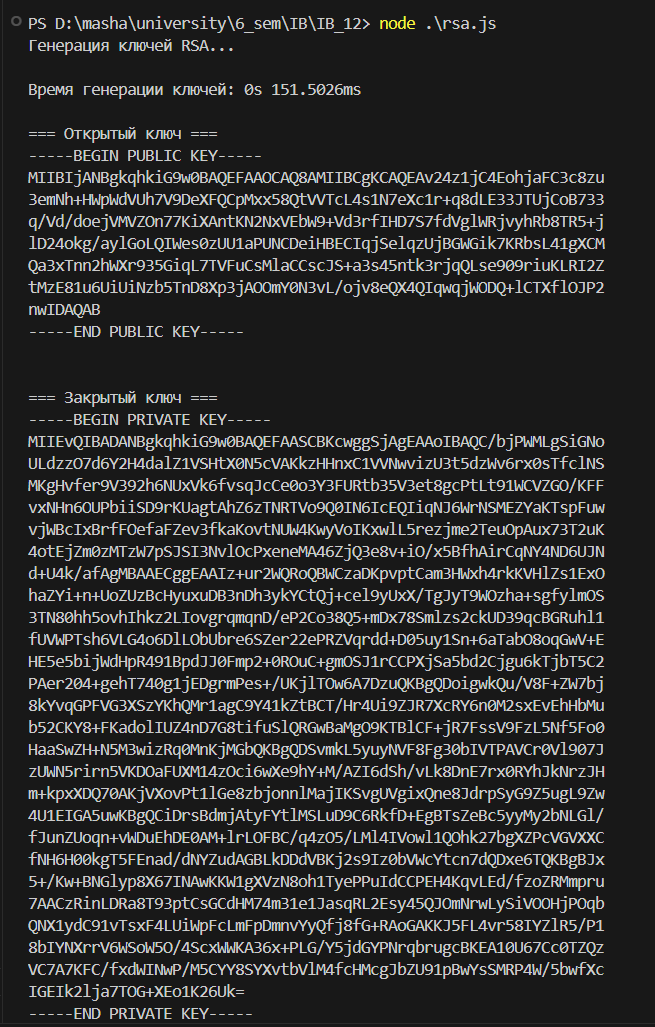
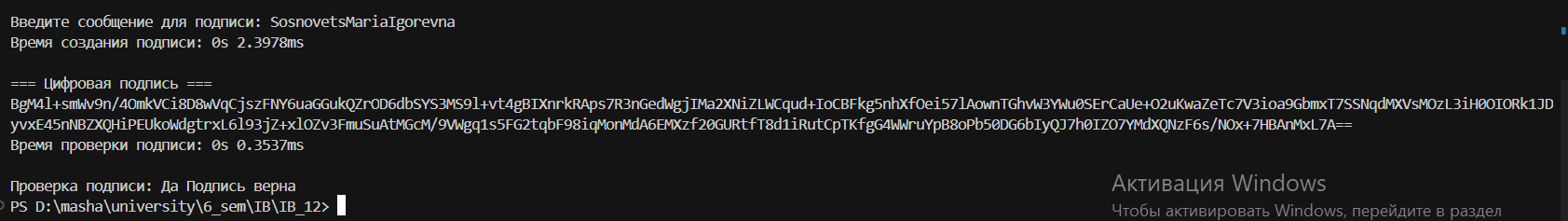
 

Рисунок 1.1 - Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма RSA

Эль-Гамаль использует сложность решения задачи дискретного логарифмирования. Здесь также есть открытый и закрытый ключи, но подпись создается с помощью случайного числа и состоит из двух частей. Проверка подписи требует выполнения специального равенства с использованием открытого ключа.

Алгоритм Эль-Гамаля выполняется так:

1. Генерация ключей:

* Выбирается большое простое число p и его первообразный корень g
* Секретный ключ x выбирается случайно в диапазоне от 2 до p-2
* Открытый ключ y вычисляется как g^x mod p

1. Создание подписи:

* Сообщение хешируется в число с помощью SHA-256
* Выбирается случайное число k, взаимно простое с p-1
* Вычисляется первая часть подписи: a = g^k mod p
* Вычисляется вторая часть подписи: b = (h - x\*a) \* k^{-1} mod (p-1)
* Подпись состоит из пары (a, b)

1. Проверка подписи:

* Сообщение хешируется в число с помощью SHA-256
* Проверяется равенство: (y^a \* a^b) mod p == g^h mod p
* Если равенство выполняется, подпись считается верной

Реализация генерации ключей алгоритма Эль-Гамаля представлена в листинге 1.4.

|  |
| --- |
| function generateKeys(bits = 256) {      const { p, g } = generatePG(bits);      const x = bigInt.randBetween(2, p.minus(2));      const y = g.modPow(x, p);      return { p, g, y, x };  } |

Листинг 1.4 - Реализация генерации ключей алгоритма Эль-Гамаля

Создание цифровой подписи алгоритма Эль-Гамаля показано в листинге 1.5.

|  |
| --- |
| function sign(message, keys) {      const { p, g, x } = keys;      const h = hashMessage(message).mod(p.minus(1));      let k;      do {          k = bigInt.randBetween(2, p.minus(2));      } while (bigInt.gcd(k, p.minus(1)).notEquals(1));      const a = g.modPow(k, p);      const kInv = k.modInv(p.minus(1));      const b = kInv.multiply(h.minus(x.multiply(a))).mod(p.minus(1));      return { a: a.toString(), b: b.toString() };  } |

Листинг 1.5 - Реализация цифровой подписи алгоритма Эль-Гамаля

Проверка цифровой подписи алгоритма Эль-Гамаля показана в листинге 1.6.

|  |
| --- |
| function verify(message, signature, keys) {      const { p, g, y } = keys;      const { a, b } = signature;      const h = hashMessage(message).mod(p.minus(1));      const left = y.modPow(bigInt(a), p).multiply(bigInt(a).modPow(bigInt(b), p)).mod(p);      const right = g.modPow(h, p);      return left.equals(right);  } |

Листинг 1.6 - Реализация проверки цифровой подписи алгоритма Эль-Гамаля

Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма Эль-Гамаля, а также оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах показан на рисунке 1.2.



Рисунок 1.2 - Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма Эль-Гамаля

Шнорр — это более эффективная модификация схемы Эль-Гамаля. В ней подпись компактнее, а проверка быстрее. Алгоритм тоже основан на дискретном логарифмировании, но использует дополнительную подгруппу простого порядка, что делает его более удобным для некоторых применений, например, в блокчейне.

Алгоритм Шнорр  выполняется так:

1. Генерация ключей:

* Выбираются простые числа p и q, где q делит p-1
* Выбирается элемент g порядка q
* Секретный ключ x выбирается случайно в диапазоне от 1 до q-1
* Открытый ключ y вычисляется как g^x mod p

1. Создание подписи:

* Выбирается случайное число k от 1 до q-1
* Вычисляется a = g^k mod p
* Вычисляется хеш h = H(message || a) mod q
* Вычисляется b = (k + x\*h) mod q
* Подпись состоит из пары (h, b)

1. Проверка подписи:

* Вычисляется X = g^b \* y^{-h} mod p
* Вычисляется h2 = H(message || X) mod q
* Подпись верна, если h2 == h

Реализация генерации ключей алгоритма Шнорр представлена в листинге 1.7.

|  |
| --- |
| function generateKeys() {      const x = bigInt.randBetween(1, q.minus(1));      const y = g.modPow(x, p);      return { p, q, g, x, y };  } |

Листинг 1.7 - Реализация генерации ключей алгоритма Шнорр

Создание цифровой подписи алгоритма Шнорр показано в листинге 1.8.

|  |
| --- |
| function sign(message, keys) {      const { p, q, g, x } = keys;      let k;      do {          k = bigInt.randBetween(1, q.minus(1));      } while (k.isZero());      const a = g.modPow(k, p);      const h = hashMessage(message + a.toString()).mod(q);      const b = k.add(x.multiply(h)).mod(q);      return { h: h.toString(), b: b.toString(), a: a.toString(), k: k.toString() };  } |

Листинг 1.8 - Реализация цифровой подписи алгоритма Шнорр

Проверка цифровой подписи алгоритма Шнорр показана в листинге 1.9.

|  |
| --- |
| function verify(message, signature, keys) {      const { p, q, g, y } = keys;      const { h, b } = signature;      const y\_inv\_h = y.modPow(q.minus(bigInt(h)), p);      const X = g.modPow(bigInt(b), p).multiply(y\_inv\_h).mod(p);      const h2 = hashMessage(message + X.toString()).mod(q);      console.log("X:", X.toString());      console.log("h:", h, "h2:", h2.toString());      return h2.equals(bigInt(h));  } |

Листинг 1.9 - Реализация проверки цифровой подписи алгоритма Шнорр

Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма Шнорр, а также оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах показан на рисунке 1.3.

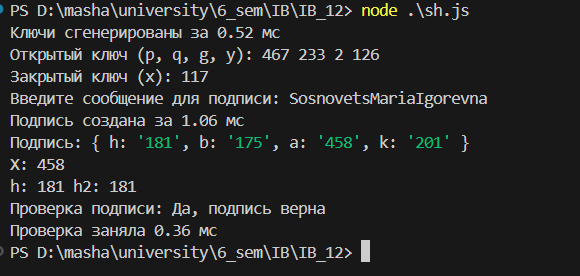


Рисунок 1.3 - Результат работы приложения, реализующего генерацию и верификацию ЭЦП на основе алгоритма Шнорр

Так же была проведена оценка времени создания подписи алгоритмов и проверка подписи. Результат сравнения создания подписи показан на рисунке 1.4.

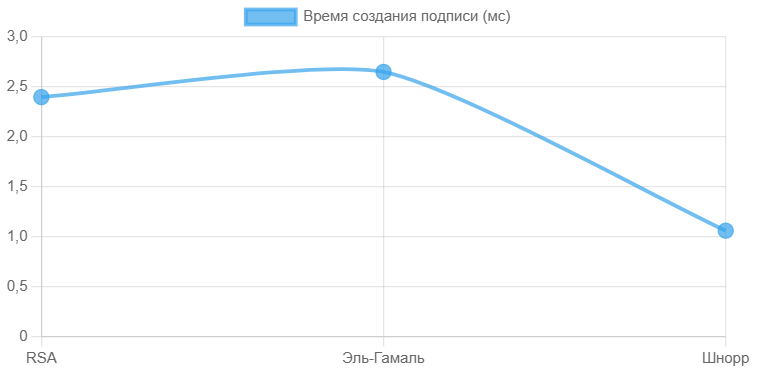


Рисунок 1.4 - Результат сравнения создания подписи

Алгоритм Шнорра демонстрирует наилучшую производительность при создании подписи, опережая RSA на 36% и Эль-Гамаль на 48%. Это связано с его компактной математической схемой на основе хеш-функций.

Результат сравнения проверки подписи показан на рисунке 1.5.

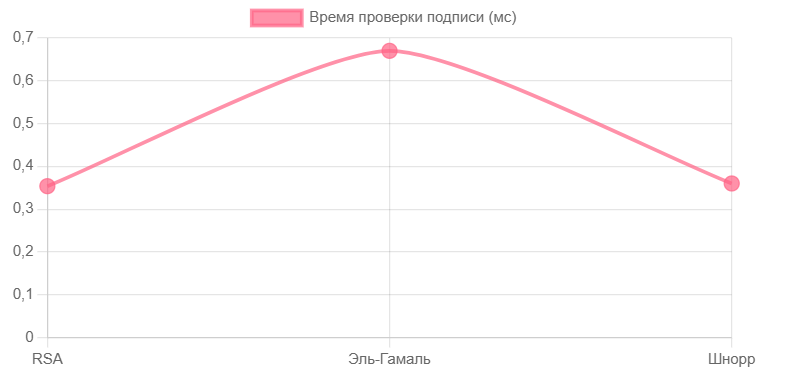


Рисунок 1.5 - Результат сравнения проверки подписи

RSA и Эль-Гамаль показывают схожую скорость проверки (~0.6 мс), тогда как Шнорр немного отстает (на 25% медленнее). Это может объясняться дополнительными вычислениями при верификации (например, повторным хешированием).

1. Для частого подписания документов (где важна скорость создания):  
   Шнорр — оптимален благодаря минимальному времени подписи (1.8 мс).
2. Для систем с упором на проверку (например, сервисы верификации):  
   RSA или Эль-Гамаль — из-за быстрой проверки (~0.6 мс).
3. Для баланса производительности:  
   RSA — несмотря на долгую генерацию ключей, он демонстрирует стабильные результаты в обоих тестах.

**Вывод:** В ходе лабораторной работы были изучены и реализованы алгоритмы ЭЦП (RSA, Эль-Гамаля, Шнорра), что позволило закрепить их теоретические основы и получить практические навыки работы с цифровыми подписями. Экспериментальная оценка показала, что Шнорр быстрее создаёт подписи (1.8 мс), а RSA и Эль-Гамаль эффективнее при проверке (~0.6 мс). Результаты подтвердили, что выбор алгоритма зависит от конкретных задач — скорости подписания или верификации.