МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12 НА ТЕМУ:**

**ИССЛЕДОВАНИЕ АЛГОРИТМОВ ГЕНЕРАЦИИИ ВЕРИФИКАЦИИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ**

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

При генерации ЭЦП с использованием алгоритма RSA можно рассматривать две ситуации:

* сообщение *Мо* подписывается и передается в открытом (незашифрованном) виде;
* сообщение *Мо* подписывается и передается в зашифрованном виде.

Первый случай соответствует ситуации, когда этом подпись *S* вычисляется на основе соотношения:

*S* ≡ (*H*(*Mo*))*d*о mod *n*o

где *dо* и *no* – элементы тайного ключа отправителя. Передаваемое сообщение *М*' = *М*o||*S*.

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

*H*(*Mo*) ≡ (*S*)ео mod *n*o

Далее вычисляется *Н*(*M*п). Если *H*(Mo) = *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 генерации ЭЦП на основе алгоритма RSA реализован в функции rsaSignData, код которой представлен в листинге 1.1. Алгоритм верификации ЭЦП на основе алгоритма RSA реализован в функции rsaVerifyData, код которой представлен в листинге 1.2.

function rsaSignData(data, privateKey) {

const md = forge.md.sha256.create();

  md.update(data, 'utf8');

  return privateKey.sign(md);

}

Листинг 1.1 – Функция генерации ЭЦП на основе алгоритма RSA

function rsaVerifyData(data, signature, publicKey) {

const md = forge.md.sha256.create();

md.update(data, 'utf8');

return publicKey.verify(md.digest().bytes(), signature);

}

Листинг 1.2 – Функция верификации ЭЦП на основе алгоритма RSA

Сгенерированный открытый и закрытый ключ, а также результат верификации ЭЦП на основе алгоритма RSA, представлены на рисунке 1.1.

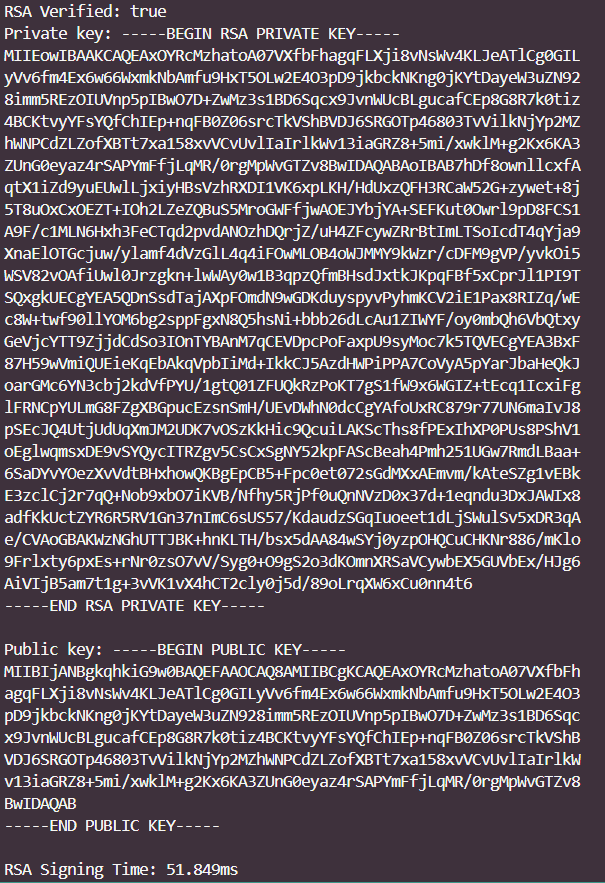


Рисунок 1.1 – Результат верификации ЭЦП на основе алгоритма RSA

# Задание 2.

Ключевая информация отправителя в ЭЦП Эль-Гамаля: открытый ключ: *y*, *g* и *р*; тайный ключ: *х*. Чтобы подписать сообщение *М*о, 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*}):

*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:

*y*a*a*b ≡ *g*h mod *p*

Если равенство выполняется, подпись верифицируется.

Алгоритм генерации ЭЦП на основе алгоритма Эль-Гамаля реализован в функции rsaSignData, код которой представлен в листинге 2.1. Алгоритм верификации ЭЦП на основе алгоритма RSA реализован в функции rsaVerifyData, код которой представлен в листинге 2.2.

function elGamalSignData(data, privateKey) {

    const ec = new EC('secp256k1');

    const key = ec.keyFromPrivate(privateKey);

    const hash = crypto.createHash('sha256').update(data).digest();

    const signature = key.sign(hash);

    return { r: signature.r.toString(16), s: signature.s.toString(16) };

}

Листинг 2.1 – Функция генерации ЭЦП на основе алгоритма Эль-Гамаля

function elGamalVerifyData(data, signature, publicKey) {

    const ec = new EC('secp256k1');

    const key = ec.keyFromPublic(publicKey);

    const hash = crypto.createHash('sha256').update(data).digest();

  return key.verify(hash, signature); }

Листинг 2.2 – Функция верификации ЭЦП на основе алгоритма Эль-Гамаля

Сгенерированный открытый и закрытый ключ, а также результат верификации ЭЦП на основе алгоритма Эль-Гамаля, представлены на рисунке 2.1.

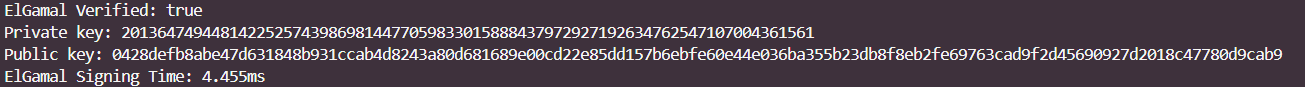


Рисунок 2.1 – Результат верификации ЭЦП на основе алгоритма Эль-Гамаля

# Задание 3.

Ключевая информация в ЭЦП Шнора: *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 Мо 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*(*Mo*||*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*:

*b* ≡ (*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.

Алгоритм генерации ЭЦП на основе алгоритма Шнора реализован в функции schnorrSignData, код которой представлен в листинге 3.1. Алгоритм верификации ЭЦП на основе алгоритма RSA реализован в функции schnorrVerifyData, код которой представлен в листинге 3.2.

function schnorrSignData(data, privateKey) {

  const hash = crypto.createHash('sha256').update(data).digest();

  const key = ec.keyFromPrivate(privateKey);

  const signature = key.sign(hash);

  return { r: signature.r.toString(16), s: signature.s.toString(16) };

}

Листинг 3.1 – Функция генерации ЭЦП на основе алгоритма Шнора

function schnorrVerifyData(data, signature, publicKey) {

  const hash = crypto.createHash('sha256').update(data).digest();

  const key = ec.keyFromPublic(publicKey, 'hex');

  return key.verify(hash, signature);

}

Листинг 3.2 – Функция верификации ЭЦП на основе алгоритма Шнора

Сгенерированный открытый и закрытый ключ, а также результат верификации ЭЦП на основе алгоритма Шнора, представлены на рисунке 2.1.

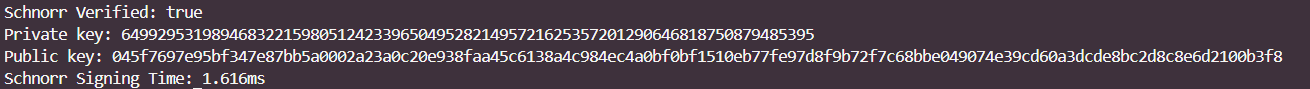


Рисунок 3.1 – Результат верификации ЭЦП на основе алгоритма Эль-Гамаля

Результаты измерения времени выполнения генерации ЭЦП предоставлены на рисунке 1.3

# Задание 4.

Результаты измерения времени выполнения генерации и верификации ЭЦП предоставлены на рисунке 4.1

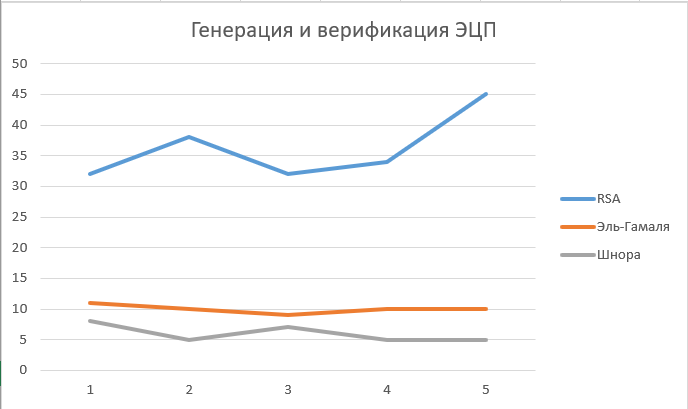


Рисунок 2.1 – Время генерации и верификации ЭЦП

**Вывод**

В ходе лабораторной работы изучены способы генерации и верификации ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнора. Для генерации и верификации ЭЦП были сгенерированы открытые и закрытые ключи. Так же было разработано приложение, реализующее данные алгоритмы.