Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 7

ОТЧЕТ

по лабораторной работе №3

Тема: «Алгоритм Эль-Гамаля»

Выполнил: студент группы АСУ-17-1б

Хохряков Денис

Проверил: доцент кафедры ИТАС

Шереметьев В.Г.\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_

# ЦЕЛЬ РАБОТЫ

Получить практические навыки по использованию несимметричных методов шифрования. Научиться делать электронную подпись.

# ЗАДАНИЕ

Создать электронную подпись текстового сообщения длиной не меньшей 256 символов, методом Эль-Гамаля.

# КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Цифровая подпись позволяет проверить, была ли информация подделана. Используется, например, в аккаунтах в криптовалютной системе. Подпись связана как с автором, так и с самим документом с помощью криптографических методов, и не может быть подделана с помощью обычного копирования.

Алгоритм Эль-Гамаля может использоваться для формирования электронной подписи или для шифрования данных. Он базируется на трудности вычисления дискретного логарифма. Для генерации пары ключей сначала берется простое число p и два случайных простых числа g и x, каждое из которых меньше p. Затем вычисляется y = gx mod p.

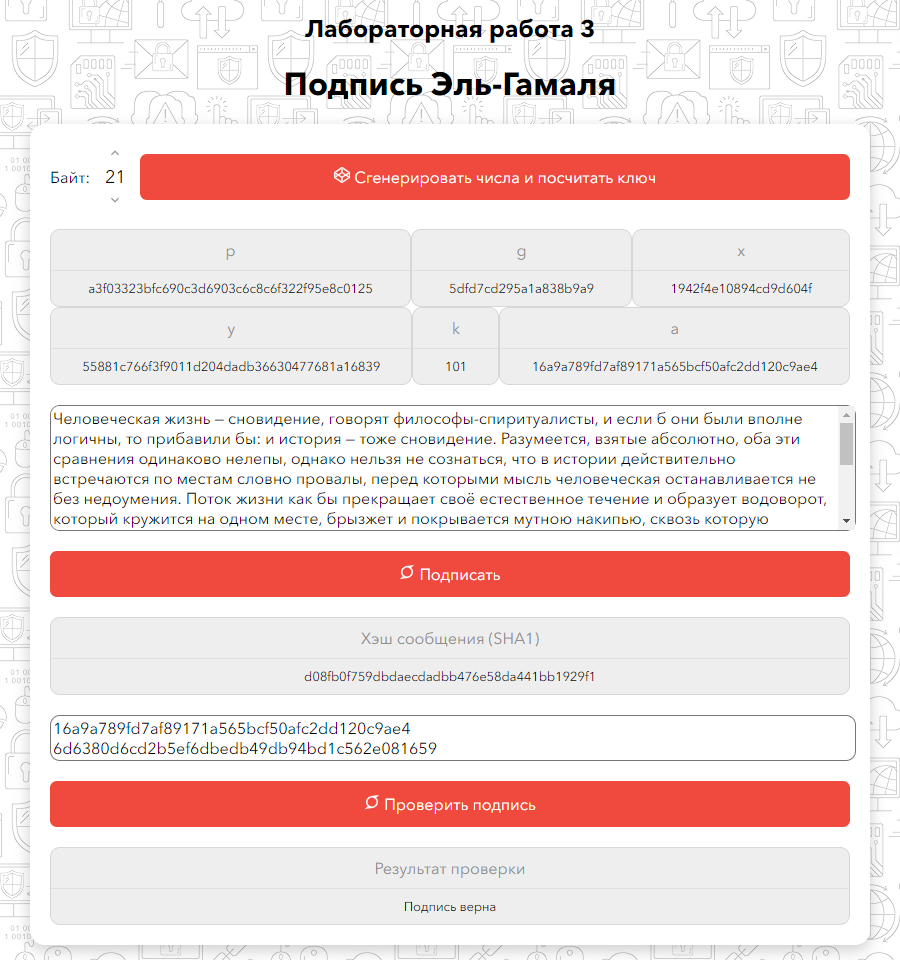
Допустим мы хотим зашифровать сообщение Mm. Тогда нам необходимо получить его дайджест (хэш-сумму) M по выбранному алгоритму, например SHA1.

Общедоступными ключами являются y, g и p, а секретным ключом является x. Для подписи хэша сообщения M выбирается случайное число k, которое является простым по отношению к p-1. После этого вычисляется a = gk mod p. Далее из уравнения *M = (xa + kb) mod (p-1)* находим b. Электронной подписью для сообщения Mm будет служить пара a и b. Случайное число k следует хранить в секрете. Для верификации подписи необходимо также посчитать хэш-функцию SHA1 от всего сообщения и проверить равенство yaab mod p = gM mod p.

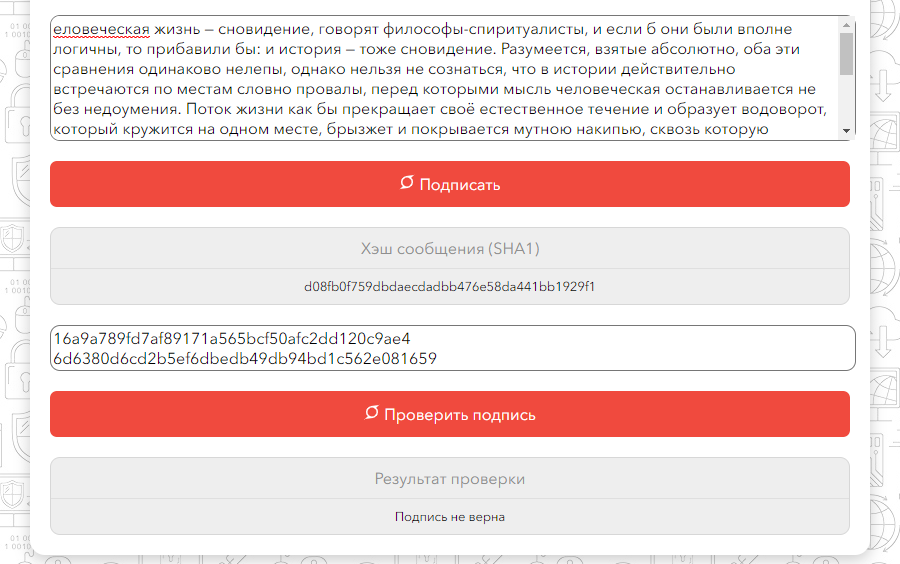
# ХОД РАБОТЫ

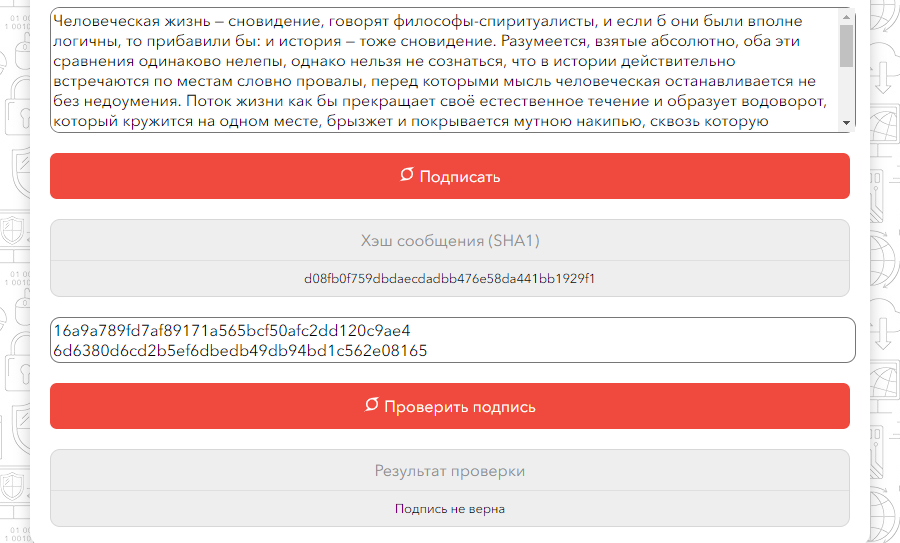
Методы генерации простых чисел и быстрого возведения в степень возьмем из лабораторной работы №2 (RSA).

Число k, которое должно быть взаимно простым с p-1, но при этом, нам необходимо решить уравнение M = (xa+xb) mod (p-1), где a = gk mod p. Поэтому возьмем число k не сильно большим, например 17. Тогда метод формирования ключа не будет столь затратным.



Также можно проверить работу алгоритма, изменив один символ в подписи или в тексте. Тогда вся подпись сразу будет неверна





# ЛИСТИНГ ПРОГРАММЫ (Язык – JavaScript)

const generate = () => {

const size = values.size || 3;

const p = generatePrime(size);

const g = generatePrime(size/2);

const x = generatePrime(size/2);

const y = fastModularExponentiation(g, x, p);

const k = 257n;

const a = fastModularExponentiation(g, k, p);

setPq({

p, g, x, y, k, a

});

}

const onSign = () => {

const message = values.text;

const { p, g, x, y, k, a } = pq;

const \_\_M = sha1(message);

const M = BigInt("0x"+\_\_M);

const \_M = (x\*a) % (p-1n);

let delta = M-\_M;

if(delta < 0) delta = delta + (p-1n);

for(let i = 0; i < 2000; i++){

if(delta % k == 0)

break;

delta += (p-1n);

}

const b = delta/k;

console.log(a);

setValues(Object.assign({}, values, {

hash: \_\_M,

sign: a.toString(16)+"\n"+b.toString(16)

}))

}

const checkSign = () => {

const message = values.text;

const { y, g, p } = pq;

const sign = values.sign.split("\n");

const a = BigInt("0x"+sign[0]);

const b = BigInt("0x"+sign[1]);

const \_\_M = sha1(message);

const M = BigInt("0x"+\_\_M);

const x1 = fastModularExponentiation(y, a, p);

const x2 = fastModularExponentiation(a, b, p);

const x3 = fastModularExponentiation(g, M, p);

setValues(Object.assign({}, values, {check: (x3 === (x1\*x2) % p)}));

}