Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы Защиты Информации

ОТЧЁТ

к лабораторной работе №5

на тему

**Цифровая подпись**

Выполнил: студент гр. 253505

Сенько Н. С.

Проверил: ассистент кафедры

информатики

Герчик А. В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc207722280)

[2 Ход работы 4](#_Toc207722281)

[Заключение 5](#_Toc207722282)

[Приложение А (обязательное) исходный код программного продукта 6](#_Toc207722283)

# 1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение алгоритма подписи документов с использованием цифровой подписи на примере ГОСТ 34.10.

В рамках работы требуется разработать программные средство для шифрования и дешифрования данных, закрепить навыки работы с асимметричными алгоритмами шифрования, освоить процесс преобразования данных с использованием открытых и закрытых ключей.

Результатом выполнения работы должен быть скрипт, который позволяет:

– подписывается документ с использованием подписи ГОСТ 34.10

– проверка подписанного документа на валидоность

# 2 ХОД РАБОТЫ

Алгоритм был реализован на языке Python. Для выполнения задачи был созданы объект, содержащий все методы, необходимые для реализации ГОСТ 34.10 и проверки подписи. Результат полного выполнения скрипта можно увидеть на рисунке 1. На рисунке изображен расчет значения SHA1 системной утилитой MacOS.

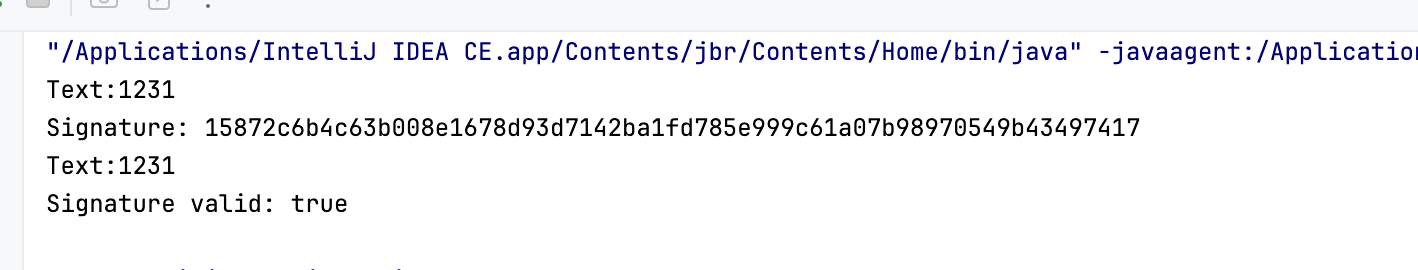


Рисунок 1 – Вывод работы программы

Немного изменим текст, на вход в проверку, и убедимся, что подпись вернет false. Результат видим на рисунке 2

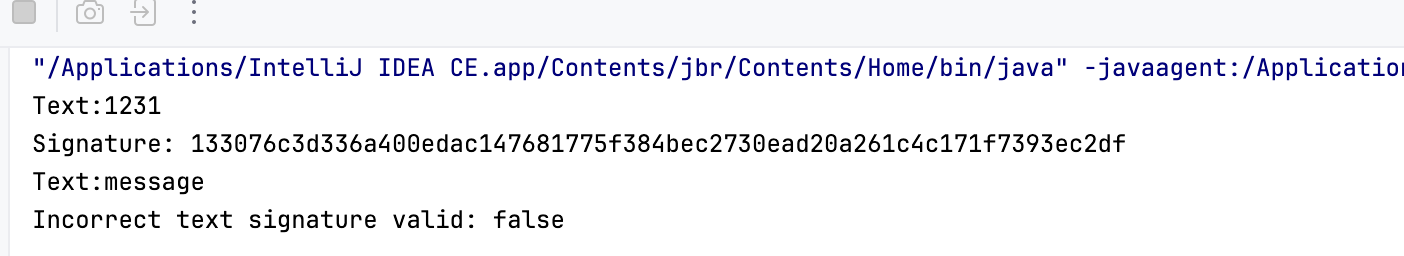


Рисунок 2 – Значение SHA1 встроенной утилитой

Как видно на рисунке 1 и рисунке 2, алгоритм работает корректно.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены теоретические основы алгоритмом цифровая подписи ГОСТ 34.10. Также на Kotlin был реализована его реализация, а также был разработан функционал проверки подписанного документа. Таким образом, поставленные цели были достигнуты: алгоритм был реализован, проведены эксперименты с его использованием.

# ПРИЛОЖЕНИЕ А

# (ОБЯЗАТЕЛЬНОЕ)

# ИСХОДНЫЙ КОД ПРОГРАММНОГО ПРОДУКТА

import java.io.File

import java.math.BigInteger

import java.security.MessageDigest

import java.util.Random

object Sign {

private fun isPrime(n: BigInteger, k: Int = 5): Boolean {

if (n.mod(BigInteger.TWO) == BigInteger.ZERO) return false

var d = n - BigInteger.ONE

var r = 0

while (d.mod(BigInteger.TWO) == BigInteger.ZERO) {

d /= BigInteger.TWO

r++

}

repeat(k) {

val i: Int = n.bitLength()

val rnd = Random()

val a = BigInteger(i, rnd).mod(n - BigInteger.TWO) + BigInteger.TWO

var x = a.modPow(d, n)

if (x == BigInteger.ONE || x == n - BigInteger.ONE) return@repeat

for (i in 0 until r - 1) {

x = x.modPow(BigInteger.TWO, n)

if (x == n - BigInteger.ONE) return@repeat

}

return false

}

return true

}

fun generateQ(): BigInteger {

while (true) {

val q = BigInteger(256, Random()).mod(BigInteger.ONE.shiftLeft(256)) +

BigInteger.ONE.shiftLeft(254)

if (isPrime(q)) return q

}

}

private fun generateP(q: BigInteger): BigInteger = q \* BigInteger.TWO + BigInteger.ONE

private fun generateG(p: BigInteger, q: BigInteger): BigInteger {

var g = 1.toBigInteger()

while (g <= p) {

if (g.modPow(q, p) == BigInteger.ONE) return g

g++

}

return BigInteger.valueOf(-1)

}

@OptIn(ExperimentalStdlibApi::class)

private fun hashMessage(message: String): ByteArray {

val digest = MessageDigest.getInstance("SHA-1")

return digest.digest(message.toByteArray())

}

private fun generateK(q: BigInteger): BigInteger {

return BigInteger(q.bitLength(), Random()).mod(q - BigInteger.ONE) + BigInteger.ONE

}

private fun generateSignature(

message: String,

p: BigInteger,

q: BigInteger,

g: BigInteger,

d: BigInteger

): Pair<BigInteger, BigInteger> {

msg = message

File(".idea/1.txt").writeText(msg)

val h = hashMessage(message)

val e = BigInteger(1, h).mod(q).takeIf { it != BigInteger.ZERO } ?: BigInteger.ONE

while (true) {

val k = generateK(q)

val r = g.modPow(k, p).mod(q)

if (r == BigInteger.ZERO) continue

val s = (r \* d + k \* e).mod(q)

if (s == BigInteger.ZERO) continue

val v = e.modInverse(q).mod(q)

val z1 = (s \* v).mod(q)

val z2 = (-r \* v).mod(q)

val c = (g.modPow(z1, p) \* g.modPow(d, p).modPow(z2, p)).mod(p)

return Pair(r, s)

}

}

private fun verifySignature(

message: String,

p: BigInteger,

q: BigInteger,

g: BigInteger,

q1: BigInteger,

r: BigInteger,

s: BigInteger

): Boolean {

if (r <= BigInteger.ZERO || r >= q || s <= BigInteger.ZERO || s >= q || message != a()) return false

val h = hashMessage(message)

val e = BigInteger(1, h).mod(q).takeIf { it != BigInteger.ZERO } ?: BigInteger.ONE

val v = try {

e.modInverse(q).mod(q)

} catch (e: Exception) {

return false

}

val z1 = (s \* v).mod(q)

val z2 = (-r \* v).mod(q)

val c = (g.modPow(z1, p) \* q1.modPow(z2, p)).mod(p)

val r1 = c.mod(q)

return r1 == r

}

fun generateSignature(message: String, q: BigInteger): Triple<String, BigInteger, BigInteger> {

val p = generateP(q)

val g = generateG(p, q)

val d = BigInteger(q.bitLength(), Random()).mod(q - BigInteger.ONE) + BigInteger.ONE

val (r, s) = generateSignature(message, p, q, g, d)

return Triple(r.toString(16) + s.toString(16), r, s)

}

fun verify(message: String, q: BigInteger, r: BigInteger, s: BigInteger): Boolean {

val p = generateP(q)

val g = generateG(p, q)

val d = BigInteger(q.bitLength(), Random()).mod(q - BigInteger.ONE) + BigInteger.ONE

val q1 = g.modPow(d, p)

return verifySignature(message, p, q, g, q1, r, s)

}

}

fun main() {

val message = File("in.put").readText()

val q = Sign.generateQ()

val (signature, r, s) = Sign.generateSignature(message, q)

println("Signature: $signature")

File("out.put").writeText("$r $s $q")

val a = File("out.put").readText().split(" ").map { BigInteger(it) }

val isValid = Sign.verify(message, a[2], a[0], a[1])

println("Signature valid: $isValid")

val isValid1 = Sign.verify("message", q, r, s)

println("Incorrect text signature valid: $isValid1")

}