Ministry of Education of Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering Department

Cryptography and Security

Laboratory work #6

Hashing and Digital Signatures

Author: Timur Cravtov
std. gr. FAF-231

Verified by: Maia Zaica
asist. univ

Chișinău
2025

Contents

1 Introduction

Hashing is a fundamental concept in computer science and cryptography, playing a crucial role in data integrity, authentication, and digital signatures. A hash function takes an input (or 'message') and returns a fixed-size string of bytes. The output, typically a 'hash code' or 'digest', is unique to the input data. Even a small change in the input will produce a significantly different hash code.

Digital signatures, on the other hand, are a cryptographic mechanism used to verify the authenticity and integrity of digital messages or documents. They provide a way to ensure that a message has not been altered in transit and confirm the identity of the sender.

2 RSA Signature

For the RSA signature scheme, we will use the same key generation as in the algorithm itself: we generate two large prime numbers p and q, compute $n = p \cdot q$, and choose an encryption exponent e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, where $\phi(n) = (p-1)(q-1)$. The decryption exponent d is computed as the modular inverse of e modulo $\phi(n)$.

The steps of RSA signature algorithm are as follows [**?**]:

- Key Generation:
- Same as RSA encryption.
- The public key is (e, n) and the private key is (d, n).
- Signing:
- Compute the hash of the message $H(M)$.
- Compute the signature s using the formula $s \equiv H(M)^d \mod n$.
- Verification:
- Compute the hash of the received message $H(M')$.
- Compute the original hash h using the formula $h \equiv s^e \mod n$.
- Check if $H(M') == h$.

3 ElGamal Signature

ElGamal signature scheme is a digital signature scheme which is based on the difficulty of computing discrete logarithms. The steps of ElGamal signature algorithm are as follows [?]:

- Key Generation:
- Choose a large prime number p and a generator g of the multiplicative group of integers modulo p.
- Choose a private key x such that $1 < x < p - 2$.
- Compute the public key $y \equiv g^x \mod p$.
- The public key is (p, g, y) and the private key is x.
- Signing:
- Choose a random integer k such that $1 < k < p - 1$ and $gcd(k, p - 1) = 1$.
- Compute $r \equiv g^k \mod p$.
- Compute $s \equiv (H(M) - x \cdot r) \cdot k^{-1} \mod (p - 1)$.
- The signature is the pair (r, s).
- Verification:
- Compute $v_1 \equiv y^r \cdot r^s \mod p$.
- Compute $v_2 \equiv g^{H(M)} \mod p$.
- Check if $v_1 == v_2$.

4 Implementation

4.1 Task 1

Implement RSA digital signature. Use SHA-256 for hashing.

```
package lab6

import lab4.util.hexToBooleanArray
import lab4.util.toBitString
import lab5.algorithms.decryptRsa
```

```kotlin
import lab5.algorithms.encryptRsa
import lab5.algorithms.keyGen
import lab5.algorithms.messageToBigInt
import java.math.BigInteger
import java.security.MessageDigest

fun main() {

    // sender
    val keys = keyGen(nMin = 3072)

    val hash = hash(messageFromLab2, "SHA3-224")
    val signature = encryptRsa(BigInteger(hash), keys.privateKey, keys.n)

    // receiver

    val decryptedSignature = decryptRsa(signature, keys.publicKey, keys.n)

    val hashReceiver = hash(messageFromLab2, "SHA3-224")
    val hashNum = BigInteger(hashReceiver)


    println(hashNum == decryptedSignature)
}

fun hash(message: String, alg: String): ByteArray {

    val digest = MessageDigest.getInstance(alg)
    val hash = digest.digest(message.toByteArray())
    return hash

}

val messageFromLab2 = "it ran with almost unbelievable efficiency. the bags of mail for deliverythat morning to the\n" +
```

```
"embassies in vienna were brought to the blackchamber each day at 7 a.m. there the letters were\n" +
"opened by meltingtheir seals with a candle. the order of the letters in an envelope wasnoted\n" +
"and the letters given to a subdirector. he read them and orderedthe important parts copied.\n" +
"all the employees could write rapidly, andsome knew shorthand. long letters were dictated to\n" +
"save time,sometimes using four stenographers to a single letter. if a letter was in alanguage\n" +
"that he did not know, the subdirector gave it to a cabinetemployee familiar with it. two\n" +
"translators were always on hand. alleuropean languages could be read, and when a new one was needed,\n" +
"anofficial learned it. armenian, for ejample, took one cabinet polyglot onlya few months to learn,\n" +
"and he was paid the usual 500 florins for his newknowledge. after copying, the letters were\n" +
"replaced in their envelopes intheir original order and the envelopes re-sealed, using forged\n" +
"seals toimpress the original waj. the letters were returned to the post office by9:30 a.m.at\n" +
"10 a.m., the mail that was passing through this crossroads of thecontinent arrived and was\n" +
"handled in the same way, though with lesshurry because it was in transit. usually it would be\n" +
"back in the post by 2p.m., though sometimes it was kept as late as 7 p.m. at 11\n" +
"a.m.,interceptions made by the police for purposes of political surveillancearrived. and at 4\n" +
"p.m., the couriers brought the letters that theembassies were sending out that day. these\n" +
"were back in the stream ofcommunications by 6:30 p.m. copied material was handed to thedirector\n" +
"of the cabinet, who ejcerpted information of special interest androuted it to the proper\n" +
"agencies, as police, army, or railwayadministration, and sent the mass of diplomatic material to\n" +
```

```
        "the court. all told, the ten-man cabinet handled an average ofbetween 80 and 100 letters a\n" +
        "day.astonishingly, their nimble fingers hardly ever stuffed letters into thewrong packet,\n" +
        "despite the speed with which they worked. in one of thefew recorded blunders, an intercepted\n" +
        "letter to the duke of modena waserroneously re-sealed with the closely similar signet of parma.\n" +
        "when theduke noticed the substitution, he sent it to parma with the wry note, \"notjust\n" +
        " m e you  too.\" both states protested, but the viennese greeted themwith a blank stare, a\n" +
        "shrug, and a bland profession of ignorance. despitethis, the ejistence of the black chamber was\n" +
        "well known to the variousdelegates to the austrian court, and was even tacitly acknowledged\n" +
        "bythe austrians. when the british 'ambassador complained humorously that he was getting\n" +
        "copiesinstead of his original correspondence, the chancellor replied coolly,\"how clumsy these\n" +
        "people are!\"enciphered correspondence was subjected to the usual cryptanalyticsweating\n" +
        "process. the viennese enjoyded remarkable success in this work.the french ambassador, who was\n" +
        "apprised of its successes from paperssold him by a masked man on a bridge, remarked in astonishment\n" +
        "that\"our ciphers of 1200 [groups] hold out only a little while against theability of the\n" +
        "austrian decipherers.\" he added that though he suggestednew ways of ciphering and continual\n" +
        "changes of ciphers, \"i still findmyself without secure means for the secrets i have to\n" +
        "transmit toconstantinople, stockholm, and st. petersburg.\""
```

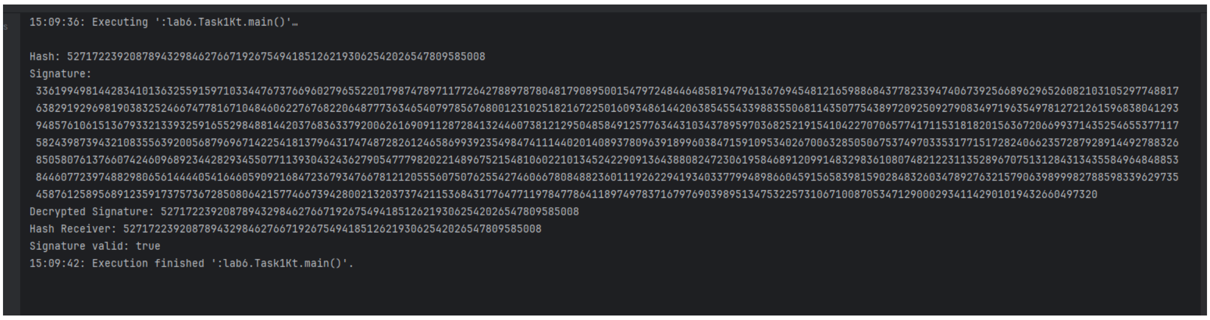

Figure 1: RSA signature output

4.2 Task 2

Implement ElGamal digital signature. Use SHA-256 for hashing.

```
package lab6

import lab5.algorithms.ElGamalPublicData
import lab5.algorithms.ElGamalSetupData
import lab5.algorithms.randomInt
import java.math.BigInteger
import util.minus

fun main() {

    val p = BigInteger("
        32317006071311007300153513477825163362488057133489075174588434139269
        ")
    val g = BigInteger.valueOf(2)

    val privateKeyElGamal = randomInt(BigInteger.ONE, p - 1);

    val setup: ElGamalSetupData = ElGamalSetupData(p, g,
        privateKeyElGamal)
    val publicKeys: ElGamalPublicData = setup.toPublic()

    val hash = hash(messageFromLab2, "SHA-512")
    val hashNum = BigInteger(hash)

    val k = randomInt(BigInteger.valueOf(1), p - 1);
    val r = g.modPow(k, p)
```

```
        val s = ((hashNum - privateKeyElGamal * r) * k.modInverse(p
            - 1)).mod(p - 1)

        val signature = ElGamalSignature(r, s);

        // receiver

        val v1 = (publicKeys.beta.modPow(signature.r, p) * r.modPow(
            signature.s, p)).mod(p)
        val v2 = g.modPow(hashNum, p)

        println(v1 == v2)


}

data class ElGamalSignature(val r: BigInteger, val s: BigInteger
    )

val p = 1;
```

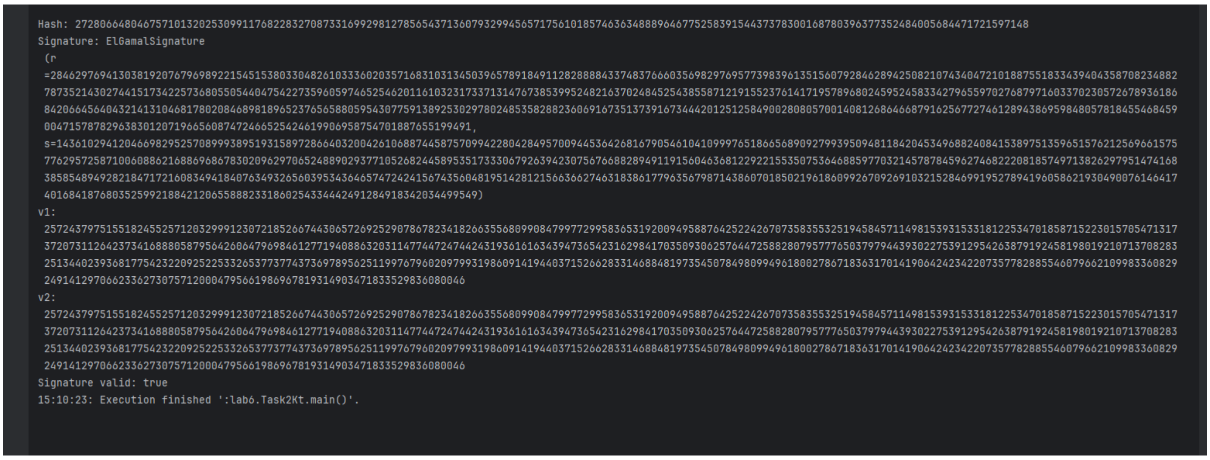

Figure 2: ElGamal signature output

5 Conclusion

During this lab, we explored the concepts of hashing and digital signatures, focusing on the RSA and ElGamal signature schemes. We implemented both algorithms in

Kotlin, utilizing SHA-256 for hashing. The implementations successfully demonstrated the signing and verification processes, confirming the authenticity and integrity of messages. This exercise reinforced our understanding of cryptographic principles and their practical applications in ensuring secure communications.

References

[1] GitHub repository `https://github.com/TimurCravtov/CryptographyAndSecurityLabs`

[2] Lecture Notes CS