

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

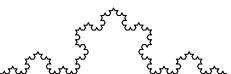
Hashing and Digital Signatures

Author: Timur CRAVTOV
std. gr. FAF-231

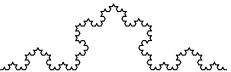
Verified by: Maia ZAICA
asist. univ



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.

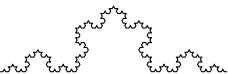
Digital signatures, on the other hand, are a cryptographic mechanism used to verify the authenticity and integrity of digital messages or documents. They provide a way to ensure that a message has not been altered in transit and confirm the identity of the sender.

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 $s \equiv H(M)^d \pmod{n}$.
- Verification:
 - Compute the hash of the received message $H(M')$.
 - Compute the original hash h using the formula $h \equiv s^e \pmod{n}$.
 - 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 $y \equiv g^x \pmod{p}$.
- 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 $r \equiv g^k \pmod{p}$.
- Compute $s \equiv (H(M) - x \cdot r) \cdot k^{-1} \pmod{p - 1}$.
- The signature is the pair (r, s) .
- Verification:
- Compute $v_1 \equiv y^r \cdot r^s \pmod{p}$.
- Compute $v_2 \equiv g^{H(M)} \pmod{p}$.
- Check if $v_1 == v_2$.

4 Implementation

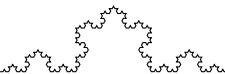
4.1 Task 1

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

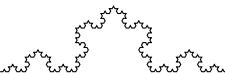
```

1 package lab6
2
3 import lab4.util.hexToBooleanArray
4 import lab4.util.toBitString
5 import lab5.algorithms.decryptRsa

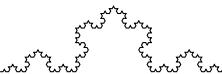
```



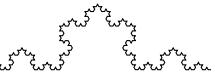
```
6 import lab5.algorithms.encryptRsa
7 import lab5.algorithms.keyGen
8 import lab5.algorithms.messageToInt
9 import java.math.BigInteger
10 import java.security.MessageDigest
11
12 fun main() {
13
14     // sender
15     val keys = keyGen(nMin = 3072)
16
17     val hash = hash(messageFromLab2, "SHA3-224")
18     val signature = encryptRsa(BigInteger(hash), keys.privateKey
19         , keys.n)
20
21     // receiver
22
23     val decryptedSignature = decryptRsa(signature, keys.
24         publicKey, keys.n)
25
26
27     val hashReceiver = hash(messageFromLab2, "SHA3-224")
28     val hashNum = BigInteger(hashReceiver)
29
30     println(hashNum == decryptedSignature)
31 }
32
33 fun hash(message: String, alg: String): ByteArray {
34
35     val digest = MessageDigest.getInstance(alg)
36     val hash = digest.digest(message.toByteArray())
37     return hash
38 }
39
40 val messageFromLab2 = "it ran with almost unbelievable
41 efficiency. the bags of mail for delivery that morning to the\
42 n" +
```



40 "embassies in vienna were brought to the blackchamber
 each day at 7 a.m. there the letters were\n" +
41 "opened by meltingtheir seals with a candle. the order
 of the letters in an envelope wasnoted\n" +
42 "and the letters given to a subdirector. he read them
 and orderedthe important parts copied.\n" +
43 "all the employees could write rapidly , andsome knew
 shorthand. long letters were dictated to\n" +
44 "save time,sometimes using four stenographers to a
 single letter. if a letter was in alanguage\n" +
45 "that he did not know, the subdirector gave it to a
 cabinetemployee familiar with it. two\n" +
46 "translators were always on hand. alleuropean languages
 could be read , and when a new one was needed,\n" +
47 "anofficial learned it. armenian, for ejample, took one
 cabinet polyglot onlya few months to learn,\n" +
48 "and he was paid the usual 500 florins for his
 newknowledge. after copying , the letters were\n" +
49 "replaced in their envelopes intheir original order and
 the envelopes re-sealed , using forged\n" +
50 "seals toimpress the original waj. the letters were
 returned to the post office by9:30 a.m.at\n" +
51 "10 a.m. , the mail that was passing through this
 crossroads of thecontinent arrived and was\n" +
52 "handled in the same way, though with lesshurry because
 it was in transit. usually it would be\n" +
53 "back in the post by 2p.m. , though sometimes it was kept
 as late as 7 p.m. at 11\n" +
54 " a.m. ,interceptions made by the police for purposes of
 political surveillancearrived. and at 4\n" +
55 "p.m. , the couriers brought the letters that
 theembassies were sending out that day. these\n" +
56 "were back in the stream ofcommunications by 6:30 p.m.
 copied material was handed to thedirector\n" +
57 "of the cabinet , who ejcerpted information of special
 interest androuted it to the proper\n" +
58 "agencies , as police , army, or railwayadministration ,
 and sent the mass of diplomatic material to\n" +



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



```

15:09:36: Executing ':lab0.Task1Kt.main()'-_
Hash: 5271722392087894329846276671926754941851262193062542026547809585008
Signature:
33619948144283410136325591597103347673766960279655220179874789711772642788978780481790895001547972484464858194796136769454681216598886437782339474067392566896296526082103105297748817
63829192969819038325246674778167104840622767682206487773634654079785678001231025182167250109348614420638545543398835506811435077543897209250927908349719635477812721261596838041293
948576106151367933213393259165529848814420376836337920062616909112872841324460738121295048584912577634431034378959703682521915410422707065774171153181820156367206099371435254655377117
582639873943210835503920058798967142254181379643174748723201246586993923549847411144020140893780963918994683847159109534026790063285050675374978353177151728240662357287928914492788326
850580761376607424609689234428293455077113930432462362790547779820221489675215461060221013452422909136438808472306195846871209914832983610807482123113528967075151284313435584964848853
8446077239748829805614444054516460509921168472367934768781210555067507625542746066780848823601119262294193403377994698660459156583919628483260347892763215790639899827885983339629735
45876125895689123591737573672850806421577466739428802132037374211536843177647711978477864118974978371679769039895134753225731067100870534712900029341142901019432660497320
Decrypted Signature: 5271722392087894329846276671926754941851262193062542026547809585008
Hash Receiver: 5271722392087894329846276671926754941851262193062542026547809585008
Signature valid: true
15:09:42: Execution finished ':lab0.Task1Kt.main()'.

```

Figure 1: RSA signature output

4.2 Task 2

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

```

1 package lab6

2

3 import lab5.algorithms.ElGamalPublicData
4 import lab5.algorithms.ElGamalSetupData
5 import lab5.algorithms.randomInt
6 import java.math.BigInteger
7 import util.minus

8

9 fun main() {
10

11     val p = BigInteger("
12         32317006071311007300153513477825163362488057133489075174588434139269
13             ")
14     val g = BigInteger.valueOf(2)

15

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

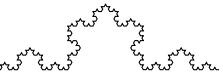
17     val setup: ElGamalSetupData = ElGamalSetupData(p, g,
18         privateKeyElGamal)
19     val publicKeys: ElGamalPublicData = setup.toPublic()

20

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

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

```



```

24
25     val s = ((hashNum - privateKeyElGamal * r) * k.modInverse(p
26         - 1)).mod(p - 1)
27
28
29     // receiver
30
31     val v1 = (publicKeys.beta.modPow(signature.r, p) * r.modPow(
32         signature.s, p)).mod(p)
33
34     v2 = g.modPow(hashNum, p)
35
36
37 }
38
39 data class ElGamalSignature(val r: BigInteger, val s: BigInteger)
40
41 val p = 1;

```

```

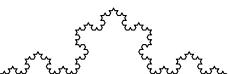
Hash: 2728066480467571013202530991176822832708733169929812785653713607932994565717561018574636348889646775258391543737830016878839637735248400568471721597148
Signature: ElGamalSignature
(r
=284629769413038192076790989221545153803394826183336020357168310313450394578918491128288843374837666035698297695773983961351506792846289425082107340472101887551833439404358708234882
7873521436274415175422573580550544047542273596059746524620116103231733713147673853952492165702484254385387121915523761417195789680249524583342796559702768797103370230572678936186
84206456460432143104081780208468918169652370553805934077519389253029780248535828823606107351373910734442012512584990280805700140812686466871625077274612894368659848057818455468459
00471578782963831207196656087472466525424619906958754701887655199491,
s=14361029412046698295257089938951915897281864032084261068874458757094228042849570942280440334426816790561041099975186056899279959509481184204334968824084153897513595157621256961575
77629572587100008862108869686783029962977052488902937710526824458953517333067926394230756768828949119150046368122922155350753646885977032145787845982746822208185749713826297951474168
38585489492812847172160834918407634932650309534364657724241567435604819514281215663662746318386177963567987143660701850219186099267092619103215284699195278941960586219304900761464417
40168418768035259921884212065588823318602543344249128491834203449549)
v1:
2572437975155182455257120329991230721852674430657269252907867823418263556809908479977299583653192009495887642522426707358355325194584571149815393153318122534701858715223015705471317
372073112642373416888058795642606479698461277194088632031147744724744243193616103439473654231629841703509306257644725882807957765053797944395022755912954263879192458198019210713708283
25134402393681775432209252253326537737743736978956251997679602097931986091419440371526628331468848197354507849809946180027867183631701419064242342207357782885546079662109983560829
249141297866233627307571200847956619869678193149034718335298360808046
v2:
2572437975155182455257120329991230721852674430657269252907867823418263556809908479977299583653192009495887642522426707358355325194584571149815393153318122534701858715223015705471317
372073112642373416888058795642606479698461277194088632031147744724744243193616103439473654231629841703509306257644725882807957765053797944395022755912954263879192458198019210713708283
25134402393681775432209252253326537737743736978956251997679602097931986091419440371526628331468848197354507849809946180027867183631701419064242342207357782885546079662109983560829
249141297866233627307571200847956619869678193149034718335298360808046
Signature valid: true
15:10:23: Execution finished ':lab0.Task2Kt.main()'.

```

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