

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

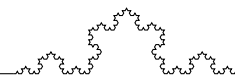
Hashing and Digital Signatures

Author: Timur CRAVTOV
std. gr. FAF-231

Verified by: Maia ZAICA
asist. univ



Chişinău
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.

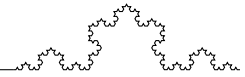
Digital signatures, on the other hand, are a cryptographic mechanism used to verify the authenticity and integrity of digital messages or documents. They provide a way to ensure that a message has not been altered in transit and confirm the identity of the sender.

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 [?]:

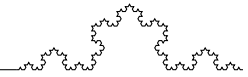
- Key Generation:
 - Choose a large prime number p and a generator g of the multiplicative group of integers modulo p .
 - Choose a private key x such that $1 < x < p - 2$.
 - Compute the public key $y \equiv g^x \pmod{p}$.
 - The public key is (p, g, y) and the private key is x .
- Signing:
 - Choose 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

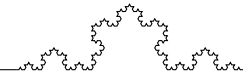
4.1 Task 1

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

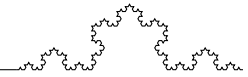
```
1 package lab6
2
3 import lab4.util.hexToBooleanArray
4 import lab4.util.toBitString
5 import lab5.algorithms.decryptRsa
```



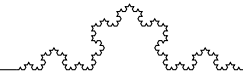
```
6 import lab5.algorithms.encryptRsa
7 import lab5.algorithms.keyGen
8 import lab5.algorithms.messageToBigInt
9 import java.math.BigInteger
10 import java.security.MessageDigest
11
12 fun main() {
13
14     // sender
15     val keys = keyGen(nMin = 3072)
16
17     val hash = hash(messageFromLab2, "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     val hashReceiver = hash(messageFromLab2, "SHA3-224")
27     val hashNum = BigInteger(hashReceiver)
28
29     println(hashNum == decryptedSignature)
30 }
31
32 fun hash(message: String, alg: String): ByteArray {
33
34     val digest = MessageDigest.getInstance(alg)
35     val hash = digest.digest(message.toByteArray())
36     return hash
37 }
38
39 val messageFromLab2 = "it ran with almost unbelievable
40     efficiency. the bags of mail for deliverythat morning to the\
41     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 a language\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 eexample , 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" +



59 "the court. all told, the ten-man cabinet handled an
average of between 80 and 100 letters a\n" +
60 "day. astonishingly, their nimble fingers hardly ever
stuffed letters into the wrong packet,\n" +
61 "despite the speed with which they worked. in one of
the few recorded blunders, an intercepted\n" +
62 "letter to the duke of modena was erroneously re-sealed
with the closely similar signet of parma.\n" +
63 "when the duke noticed the substitution, he sent it to
parma with the wry note, \"not just\n" +
64 "me you too.\" both states protested, but the viennese
greeted them with a blank stare, a\n" +
65 "shrug, and a bland profession of ignorance. despite this
, the existence of the black chamber was\n" +
66 "well known to the various delegates to the austrian
court, and was even tacitly acknowledged\n" +
67 "by the austrians. when the british 'ambassador complained
humorously that he was getting\n" +
68 "copies instead of his original correspondence, the
chancellor replied coolly, \"how clumsy these\n" +
69 "people are!\" enciphered correspondence was subjected to
the usual cryptanalytics sweating\n" +
70 "process. the viennese enjoyed remarkable success in
this work. the french ambassador, who was\n" +
71 "apprised of its successes from papers sold 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 the ability of the\n" +
73 "austrian decipherers.\" he added that though he
suggested new ways of ciphering and continual\n" +
74 "changes of ciphers, \"i still find myself without secure
means for the secrets i have to\n" +
75 "transmit to constantinople, stockholm, and st.
petersburg.\""



```

15:09:36: Executing '':lab6.Task1Kt.main()'...

Hash: 5271722392087894329846276671926754941851262193062542026547809585008
Signature:
336199498144283410136325591597103344767376696027965522017987478971177264278897878048179089500154797248446485819479613676945481216598868437782339474067392566896296526082103105297748817
638291929698190383252466747781671048460622767682206487773634654079785676800123102518216722501609348614420638545543398835506811435077543897209250927908349719635497812721261596838041293
94857610615136793213393259165529848814420376836337920062616909112872841324460738121295048584912577634431034378959703682521915410422707065774171153181820156367206699371435254655377117
58243987394321883556392005687969471422541815796431747487282612465869939235498474114402014089378096391899683847159109534026700632850506753749783353177151728240662357287928914492788326
850580761376607424609689234428293455077113938432436279054777982022148967521548104022101345242290913643880824723061958468912099148329836108074821223113528947075131284313435584964848853
844687723974882980656144440541640059092168472367934766781212055560750762554274606678008488236011192622941934033779948986604591565839815902846326034789276321579063989998278859839629735
4587612589508912359173757367285080642157746673942800213203737421153684317764771197847786411897497837167976939895134753225731067100870534712900029341142901019432600497320
Decrypted Signature: 5271722392087894329846276671926754941851262193062542026547809585008
Hash Receiver: 5271722392087894329846276671926754941851262193062542026547809585008
Signature valid: true
15:09:42: Execution finished '':lab6.Task1Kt.main()'...

```

Figure 1: RSA signature output

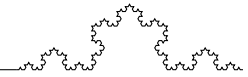
4.2 Task 2

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
18     val setup: ElGamalSetupData = ElGamalSetupData(p, g,
19         privateKeyElGamal)
20     val publicKeys: ElGamalPublicData = setup.toPublic()
21
22     val hash = hash(messageFromLab2, "SHA-512")
23     val hashNum = BigInteger(hash)
24
25     val k = randomInt(BigInteger.valueOf(1), p - 1);
26     val r = g.modPow(k, p)

```

```

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

```

```

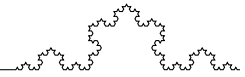
Hash: 2728066480467571013202530991176822832708733169929812785654371360793299456571756101857463634888964677525839154437378300168780396377352484005684471721597148
Signature: ElGamalSignature
(r
=28462976941303819207679698922154551380330482610333602035716831031345039657891849112828888433748376660356982976957739839613515607928462894250821074340472101887551833439404358708234882
787352143027441517342257368055054404754227359605974652546201161032317337131476738539952482163702484525438558712191552376141719578968024595245833427965597027687971603370238572678936186
8420645404843214131846817802084089818965237655880595430775913892530297802485358288236069167351373916734442012512584908288057001408126864668791625677274612894386959848057818455468459
0047578782963630120719665087472466525426419906958754701887655199491,
s=143610294120466982952570899938951931589728664032084261068874458757099422804284957009445364268167905461041099976518665689092799395094811842045346882408415389751359651576212569661575
776295725871006088621088690867830209629706524889029377105268244589535173336679263942307567668828949119150646308122922155350753646885977032145787845962746822208185749713826297951474168
385854894928218471721008349418407634932050039534364657472424150743500481951428121566366274631838617796356798714386070185021961860992670926910321528469919527894196058621930490076146417
4016841876803525992188421206558882331860254334442491284918342034499549)
v1:
257243797515518245525712032999123072185266744306572692529078678234182663556809908479977299583653192009495887642522426707358355325194584571149815393153318122534701858715223015705471317
372073112642373416888058795642606479698461277194088632031147744724744243193616163439473654231629841703509306257644725882807957776503797944393022753912954263879192458198019210713708283
251344023936817754232209252253326537737743736978956251199767960209799319860914194403715266283314688481973545078498099496180027867183631701419064242342207357782885546079662109983360829
24914129706623362730757120004795661986967819314903471833529836080046
v2:
257243797515518245525712032999123072185266744306572692529078678234182663556809908479977299583653192009495887642522426707358355325194584571149815393153318122534701858715223015705471317
372073112642373416888058795642606479698461277194088632031147744724744243193616163439473654231629841703509306257644725882807957776503797944393022753912954263879192458198019210713708283
251344023936817754232209252253326537737743736978956251199767960209799319860914194403715266283314688481973545078498099496180027867183631701419064242342207357782885546079662109983360829
24914129706623362730757120004795661986967819314903471833529836080046
Signature valid: true
15:10:23: Execution finished ':Lab6.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