MINISTERUL EDUCAŢIEI ŞI CERCETĂRII
AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Inginerie Software și Automatică

Report

Laboratory work №5

Cryptography and Security

Subject: Public-Key Cryptography

Author:
Mihaela Untu,
std. gr. FAF-232

Verified:
Maia Zaica,
university assistant

Chișinău, 2025

Contents

1 Purpose of the laboratory work

The goal of this laboratory work is to study and implement public-key cryptographic algorithms: RSA, ElGamal, and the Diffie–Hellman key exchange. For each algorithm, we generate the required keys, perform encryption and decryption, and verify correctness. Additionally, we show how a Diffie–Hellman shared secret can be transformed into a 256-bit AES key.

2 Message Conversion

The initial plaintext message is:

$$m = \text{“Mihaela Untu”}$$

Converted to ASCII (hexadecimal representation):

$$\text{4D 69 68 61 65 6C 61 20 55 6E 74 75}$$

Converted to decimal:

$$m = 771051049710110897328511011611 7$$

This numeric form will be used for RSA and ElGamal encryption.

3 Task 2.1 — RSA Algorithm

The RSA cryptosystem operates on the principle that while it is easy to multiply two large prime numbers, it is computationally infeasible to factor their product. To generate an RSA key pair, two large primes p and q must be chosen.

1. Choosing the primes

3.1 Key Generation

Two large prime numbers were generated:

```
p = 143603244606771102415217435439314933770509539885692644840103633883
8436578917817797529472103169802803849611170165866797259883043594487351
1537461688703473445791965671944205718431908704113933915232897698252435
5692150833478309619347531057186057501514593645658569441221581447242270
39500501131618774192930337867940 7
```

q = 174635039450951519630058466274370873312573890818231673489376406952
3483437668776125118156925363210226645161582104259140263474383312691113
6417124622602001355348148649844695376564903268599581162583561721416539
0626338861972580088633640099503795800561912047594875788396049477921322
272417449550343082114103215505399

RSA operates in the ring Z_n, therefore:

The RSA modulus is:

$$n = p \cdot q$$

n = 250781582871881125135849173998371384530785986620507714723691401851
2527991932342799114383073000725208684121999320608299608000933571539741
0216417344282494668844820277897998673230005276287503285410553223498013
7441790667761537433729321949497083529069413634055224567297853524926040
1205021282067101880521283937727418302366375074806011481303441813264027
8735462619818024714916764627507373568206434191482983141449220349059644
6861636007937686744170533320425881463649661318250553482729282604421650
1215922329746282187778959638345738654477799140953962937741240607681969
1018257742872112942798714626031920094725113384791274698618393

The totient measures the number of integers that are relatively prime to n.

Euler's totient:

$$\varphi(n) = (p-1)(q-1)$$

phi = 2507815828718811251358491739983713845307859866205077147236914018
5125279919323427991143830730007252086841219993206082996080009335715397
4102164173442824946688448202778979986732300052762875032854105532234980
1374417906677615374337293219494970835290694136340552245672978535249260
4012050212820671018805212839377273864785379693025438069537132704446833
1956520319158937063854367237913153719095470419267200802881479172995823
6945915100704143293172633548556304228325191138381365420505503924949219
6892432088146684657283592906637138896849327101701558458220722552774743
185325004297642495311873551033364497633858582560747868104433588

The public exponent is:

$$e = 65537$$

The private exponent:

$$d \equiv e^{-1} \pmod{\varphi(n)}$$

d = 159931279370893871020230041461493996009341292869739535481890872947

```
6953589923436643254736111449491281214472419573749544259218441775218631
8568063275772263983465359468922087592469110128971973081064651601020789
5454165447901085753737842911320789875909122218522942929798644858237085
1097301745334612861381922611308361961494948304008703657940493739956655
6483767550006522397719271111860872010612609935262000940585351839968950
8452510236858063289668765982438523616966543522970486519107582805518465
7703585950059791035854197679019821160287511200721173934319695886629524
43758267027780962137381183978434957139801861257985130144010530
```

3.2 Encryption

Using the RSA formula:

$$c = m^e \bmod n$$

Ciphertext:

```
c = 8138005031606560900729620903249792731991459445360898259233817033311
7727999791438381025784780151650178634938284613669101155802594677360293
7381859503264521266331300517976149526652727798282410676519623307452052
6345819206437883301523058394433904537639741833759518553359710080439522
8354183863815750976194464391915926676459699790408280337800155266062348
6065707292631749649158022056526823547953193012983636402451520173344036
9694772930357808687344081783626408261392181629690949480299663301836299
1335657693561095057634030264957751728000485108363691705677423271679530
040500870038215554589925504203714256935119649521863991617958
```

3.3 Decryption

Using the RSA formula:

$$m_{\text{dec}} = c^d \bmod n = 7710510497101108973285110116117$$

The decrypted message matches the original value.

4 Task 2.2 — ElGamal Algorithm

ElGamal encryption is based on the difficulty of the discrete logarithm problem in the multiplicative group Z_p^*. The algorithm requires a large prime p and a generator g of the group.

We use the large prime p provided in the laboratory instructions and generator $g = 2$:

$$p = 32317006071311007300153513477825\ldots27039$$

Private key x: Alice chooses a random private key:

$$x \in \{2, 3, ..., p-2\}$$

```
x = 2330707416183353311035903024506250211940308423531853792895865218624
7623153676019765211901252824179981162640520201671530069639393615955263
1094150118917970223739997908299911780516591743830363976240370829695682
6740675063237051721604351881970161675706422106975132126257895228944557
6910880086034159096013262197850940170321341135857601943812951529317850
4130873442498634628705003721363148541845616357327686635300378737951306
7886366990843329332583366615203744120402261422510174114208390791869368
5013979022322130104839041557461982966750404097660985205320243412326576
008632943986350789990628689811400177990393482268505727484617
```

Public key:

$$y = g^x \bmod p$$

Random ephemeral exponent. Security requires a fresh random k for each encryption:

$$k \in \{2, ..., p-2\}$$

```
k = 1738650931527475337036117286912652412022799144182506653370266394175
5065350323155093330624163240421094067636090235628192637365047208043985
7837311576431600280811514072730800896372620243864324709484821338376159
4690240297557563842260460667063867204862239023368939183690615406989676
3203019931780316584411171862363176255894705962180981463087783750900702
4952772443780834405709815007885618882972574085858176192683249183578432
9951353969588654371878150578675030863327797226280281058553156777597906
0689352660923603220968545822953411785023537842343468942764900285048688
696761581049680431207280820495018134438162204924933531276420
```

4.1 Encryption

ElGamal produces two ciphertext components:

$$c_1 = g^k \bmod p$$

$$c_2 = m \cdot y^k \bmod p$$

```
c1 = 2123183007020186782280270313521698553114560477602912613432747539586315...
c2 = 2305393079633984685138869219935491072427909472408733409498063093986217...
```

4.2 Decryption

The receiver computes:

$$s = c_1^x \bmod p$$

$$m = c_2 \cdot s^{-1} \bmod p$$

Recovered value:

$$m_{\text{dec}} = c_2 \cdot s^{-1} \bmod p = 7710510497101108973285110116117$$

The ElGamal decryption successfully recovers the original message.

ElGamal's semantic security relies entirely on the randomness of k. If k repeats, the private key x can be computed — making k the most sensitive value in the scheme.

5 Task 3 — Diffie–Hellman Key Exchange with AES-256 Key Derivation

In this task, we implement a full Diffie–Hellman key exchange between Alice and Bob using the large prime p and generator g provided in the laboratory instructions. Both parties independently generate secret exponents, compute public values, exchange them, and then compute the shared secret. Finally, the shared Diffie–Hellman key is converted into a 256-bit AES key using the SHA-256 hash function.

5.1 Given Parameters

The Diffie–Hellman parameters are:

$$g = 2,$$

$$p = \begin{array}{r} 3231700607131100730015351347782516336248805713348907517458843413926980683413621\,0 \\ 0027920563626401646854585563579353308169288290230805734726252735547424612457410\,26 \\ 2025279165729728627063003252634282131457669314142236542209411113486299916574782\,68 \\ 0342305530863490506355577122191878903327295696961297438562417412362372251973464\,02 \\ 6918557977679768230146253979330580152268587307611975324364674758554607150438968\,44 \\ 9403661304976978128542959586595975670512838521327844685229255045682728791137200\,98 \\ 9318739591433741758378260002780349731985520606075332341226032546840881200311059\,07 \\ 48428100399496695611969695624862903233807283912703\,9 \end{array}$$

5.2 Secret Exponents

Alice chooses a random secret integer a, and Bob chooses a random secret integer b, both in the interval $[2, p - 2]$:

Alice's secret a:

2381405874158175272783155910796640603224087296488621319886437134703948
7339386805554396061709078390148833173014504088959120339333084879898697
6635303887137516201203834639244985159590192438540550241434072556202059
6961539994167475110407629687760121901363774907166450444498081652283069
0821050168876488715506534661938558043403654902438589738061762397610645
5963791893384669185927834920242513274375273445032953843072738752049518
8490293252908559921427125358594164388908920467531050001498975218621763
5740743728163496226147357041570727278328843250839068291680698476298432
3973904272276483922571279565356172889555934434554776691 70

Bob's secret b:

1505154382686251732241492861272640816629933959430341920274641014703368
0499350496154433057933618476295483685928936834629566149146645793969095
5996343329846776996507025453017578832595557671862111294337192974108912 8
3233874941341471822235337073737029646859791789018970598540699225655342
1346680057567828778003563786028607261789341424712246555947107276149049
6799162779736120726323844882908207814389783168846177170787138195957250
7912774194408921496865493698531631994377993672259213379471255325197547
2790861444150273272497223667791724474629748761114265121069048826204731
3383436899992324167249895046474187695266423692140516443 93

5.3 Public Values

Alice computes her public value:

$$A = g^a \bmod p$$

A = 93661183225100852516760008910634369034975767399284723932461285213 2
8280364612873569747047368893200080375227637375719069360781337152273610
0448818436692452468809053996268454450116384872471937184761707276074535
9123657797717611063825532338789894360704882973985162674150368381138425
6608838647506456176534662538832381504577985708207900566001726613607182
1873274941165888203076137237842790571533126120053696543202926597541937
7639460077459871370675485385318873984157235575648460375095289726614833
1078008741622845115265246299122302395584185554682397674993193974362269
9901541170395493300865064772139107313647903725273904403967 88

Bob computes his public value:

$$B = g^b \bmod p$$

```
B = 1461498566050763595315618503309393971873525644120722165806582677651
1223629669324007650480568385095926652287961917900289503830043886220492
9306054479127172247970098360447424281771174282002940271319003297565580
2150975085531335538465143167936476007287560817319824842100800737843773
5402839334852079307890298214623722144811321248248910562243866028213164
2061191798967574072353306541704890610211014416789715330122913908612543
9969366906254039806669086599689127543958584990810722495729866320447143
2036876627871608324405588077386596233533679655788225993099644327695022
5632763410553663856505195843254915517869881171216150241875845
```

5.4 Shared Secret Computation

Alice computes:

$$K_A = B^a \bmod p$$

Bob computes:

$$K_B = A^b \bmod p$$

The results are identical:

$$K_A = K_B =$$

```
4890945194009989533134990696540696597300696082620683388934586774187648
9882308272671683666483933348908892806254754204589519982728560679560975
9829549687398565500656579906936802423448498243805625439003817850621743
3398108682177256421223880168634010265107052433464660332467254822042688
0336096416004221375312257524016300293885468258872056810472701151464425
8215004998289259102425700887700618493289230690180046903272866549533446
9499705550213423735808766203861757093449626965896302459524924233533821
3961236402643961116614374112381674317844968601414480744388352057069597
1124651306301104858417970580424167846631651311227636327 9
```

Thus, the Diffie–Hellman key exchange was successful.

5.5 AES-256 Key Derivation

The shared Diffie–Hellman secret K obtained by both Alice and Bob is a large integer of approximately 2048 bits. However, AES-256 requires a symmetric key of exactly 256 bits. To derive such a key from the DH shared value, we apply the SHA-256 cryptographic hash function to K.

$$K_{\text{AES}} = \text{SHA256}(K)$$

In Mathematica/WolframAlpha, this is performed as follows:

```
K = KA
AESkey = Hash[K, "SHA256"]
```

The resulting 256-bit integer is:

```
AESkey = 30792610846910912726992900756246742062371910635590222000618986413562022008989
```

To obtain the AES key in hexadecimal form (64 hex = 256 bits), we compute:

```
AESkeyHex = IntegerString[Hash[K, "SHA256"], 16]
```

Resulting in:

```
AESkeyHex = 441400077da91bf6889fdd0208438c18c80271daa2a758eb72e992d3b35b189d
```

This 256-bit hexadecimal value is the final AES-256 key derived from the Diffie–Hellman shared secret and is suitable for symmetric encryption.

Conclusion

All algorithms (RSA, ElGamal, Diffie–Hellman) were successfully implemented. For each algorithm the encryption/decryption cycle was correct. The DH shared secret was transformed into a valid AES-256 key using SHA-256.

Bibliography

1. Aureliu Zgureanu — *Cryptography & Security*.
2. William Stallings — *Cryptography and Network Security*, Pearson, 2020.
3. Github Repository, *https://github.com*