

**Ministry of Education, Culture and Research of the Republic of Moldova**

**Technical University of Moldova**

**Department of Software and Automation Engineering**

**REPORT**

Laboratory work no. 5

*Public key cryptography*

Elaborated:

st. gr. FAF-213 Botnari Ciprian

Verified:

asist. univ. Cătălin Mîțu

Chişinău – 2023

Table of Contents

[Topic: Public keys 3](#_Toc150408136)

[Tasks 3](#_Toc150408137)

[Theoretical notes 3](#_Toc150408138)

[Results 4](#_Toc150408139)

[Conclusion 7](#_Toc150408140)

[GitHub 7](#_Toc150408141)

## Topic: Public keys

## Tasks

1. Generate keys, encrypt and decrypt the message: <Last name> <First name> with RSA. The value of n must be at least 2048 bits (617 digits).
2. Generate keys, encrypt and decrypt the message: <Last name> <First name> with ElGamal. The values of prime and generator are given below.
3. Use the Diffie-Helman key exchange between Bob and Alice, which has to be prepared for AES 256 bits

## Theoretical notes

**RSA**, or Rivest-Shamir-Adleman, is a widely used asymmetric-key cryptosystem that offers secure communication and digital signatures. The RSA algorithm relies on the mathematical properties of large prime numbers. Here are the key phases of the RSA algorithm:

1. Key Generation: RSA begins with key generation, which involves selecting two large prime numbers, p and q. The product of these primes, n = p \* q, becomes the modulus used for encryption and decryption. The public key (e, n) and private key (d, n) are generated, with e as the encryption exponent and d as the decryption exponent.
2. Encryption: To encrypt a message M, the sender uses the recipient's public key (e, n). The message is raised to the power of e and then taken modulo n to obtain the ciphertext C = Me mod n.
3. Decryption: The recipient uses their private key (d, n) to decrypt the ciphertext C. The ciphertext is raised to the power of d and taken modulo n to recover the original message: M = Cd mod n.
4. Digital Signatures: RSA is also used for digital signatures. To sign a message, the sender applies their private key to a hash of the message, creating a signature. Recipients can verify the signature using the sender's public key.
5. Key Security: The security of RSA relies on the difficulty of factoring the modulus n, given that it is a product of two large prime numbers. The prime factorization problem forms the basis of RSA's security.

**ElGamal** is another asymmetric-key cryptosystem, primarily used for secure key exchange and digital

signatures. It involves the following phases:

1. Key Generation: ElGamal starts with key generation. The sender selects a large prime number, p, and a primitive root modulo p, g. The sender also generates a private key, x, which is a random number, and computes the corresponding public key, y = gx mod p.
2. Key Exchange: In the key exchange phase, two parties can securely exchange a shared secret by using their respective private keys and the other party's public key. This shared secret can be used as a symmetric key for encryption and decryption.
3. Encryption: ElGamal encryption is probabilistic and involves generating a random number, k. The message is encrypted as a pair of ciphertexts (c1, c2), where c1 = gk mod p and c2 = (M \* yk) mod p, with M being the plaintext message.
4. Decryption: The recipient uses their private key, x, to decrypt the ciphertext (c1, c2). They calculate the shared secret, S = c1^x mod p, and then compute the plaintext as M = (c2 / S) mod p.
5. Digital Signatures: ElGamal can also be used for digital signatures, where the sender signs a message using their private key, and the recipient verifies it using the sender's public key.

**Diffie-Hellman** is a key exchange protocol that allows two parties to securely exchange a shared secret over an untrusted network. Here are the key phases of the Diffie-Hellman protocol:

1. Parameter Selection: The parties agree on two public parameters, a prime number p and a primitive root modulo p, g.
2. Key Generation: Each party generates its private key, a random number (a for the first party, b for the second party). They then compute their corresponding public key: A = ga mod p for the first party and B = gb mod p for the second party.
3. Key Exchange: The parties exchange their public keys (A and B) over the insecure channel.
4. Shared Secret: Each party uses the received public key to compute the shared secret. The first party computes S = Ba mod p, and the second party computes S = Ab mod p. Both parties arrive at the same shared secret, which can be used as a symmetric encryption key.
5. Encryption: Once the shared secret is established, the parties can use it for secure communication, such as encrypting and decrypting messages using a symmetric encryption algorithm.

## Results

Here are the results for each one.

**RSA**

**Message to encrypt:** Botnari Ciprian

**Message (hex):** 0x426f746e617269204369707269616e

**Message (decimal):** 344952164757312787425800986779279726

**Cipher text:** 1756521601129284782837789973332517248617222747754575908974336176554248252575092682013364295704446473575048000226543224084592591314059079196787616920008360133045729201033476600367130180733678868948763388544785844192349593366729592721806604124056711568015446030892289806227470043899935795953180332705410729914033870690068260484684458540234055513417825068852742176924918375484585665866275870174230585963499077539623253083675653589372444591025479648911064582991015418933319706501295457219576457245634780345490120418891294756114577120680886969609986600158836225534124850575057187146985043311674876295649575559776736046604857447444115405511545359688118728824594525705298284527312288644799511570535626681722900401997138357123768447698566529696846385918015390102252057121913612492073798614320122133381443245513596812600014610574828804062044294937013821323500356824253756515579504540065394986559858884484438099889114108768457647634582902114461275907112208561426598227862487727167253395016596932988951724432020545344530930428759543330867804342713383976982400678883490663197418837944086424483615366970439283222530622343246350253656244520905801285264623922049106502762791455806709124662377299325979040815614527830200287929527267060529172295889786588735545339188661427082636522286541811214172792246227656788749152719988595060830646221795335251724456251854563626257654977163778912055924722226580931031571288877063244181258243754723682853954232650906272978703700604143661397907938361423924382354452461381263099997056927615053155104965616697883919042650195313001781727474138287985948916952636161955166991115921408988959015582100643112069355093551642565464515673472099233322126198359187716518222199780263288957517155294544950224540683084461220614567515756600714717068214729075466636167962663727647628028818928448103960337944383555278906012880673711894207998549044939071156243205512072618535973494520351704893377417905611894240924784992273070702512764765536246946999833022005223748395979581411313407395897905674637833118400887571552135124694414457613836638041768704495347487523251763138452203067609368713724191474963334329284568444802864748189795067269181919085752881960524426653680256634452823784769992266531303510451914469107375630439455033798663513874682694610484691159449434772845942507078185431658131057983657448429463164664569845828813286945010819338287958334819568254814965771084202497419866498993252702000397736408023374341813972412116111432017364405792091804567138683346782533016180482953916758386

**Decrypted cipher**: Botnari Ciprian

**ElGamal**

**Message to encrypt**: Botnari Ciprian

**Message (hex)**: 0x426f746e617269204369707269616e

**Message (decimal):** 344952164757312787425800986779279726

**Cipher text:** 3158738416902844493723886487486927618941226059126888391174977151870485200325821794717131144853162743095527301178663454526985880569937001184776112381003952328683889782440709529408601337898527780833564519697124469738286734885992312448382410580189583444893770542437631600704908483609542826604476005524962508831879177914123965074246334362108990847881033786521333609742907001144992496031725200638305086817482586417810084088202538076247228600106147230227182310476802487669641238481492446246972935538967076136561543438895895897713328508918903897514261011304017041469816385910942742328332382333843466056122980198382837154851

**Decrypted cipher: Botnari Ciprian**

**Diffie-Hellman**

**Prime:**

32317006071311007300153513477825163362488057133489075174588434139269806834136210002792056362640164685458556357935330816928829023080573472625273554742461245741026202527916572972862706300325263428213145766931414223654220941111348629991657478268034230553086349050635557712219187890332729569696129743856241741236237225197346402691855797767976823014625397933058015226858730761197532436467475855460715043896844940366130497697812854295958659597567051283852132784468522925504568272879113720098931873959143374175837826000278034973198552060607533234122603254684088120031105907484281003994966956119696956248629032338072839127039

**Generator: 2**

**Private key 1:** 17176772790485768693060458190302180857601220146769520749640620367512224444198230311158267176989059500817820936184254287969979207543705439182939170325516071900316875511202914843851871478154354181194982689075378079277914421509978403238450768233570537994723085364195479165004666801176075451051916818070854422518191598968412011218916584753431683305434825147070392574493534519708671061191954115264274717403882530586861432039576978757477862060616050443437242321603532937486354102195956679707003610566489409612942571033659439618243628918698014819173086085176707875147797110310297017697463795221606466961282429812407373256890

**Private key 2:** 6976378920129883368810290366141379881280032966166184166433427970648684343944330021663344388578410537833435976611199542127807982982084598388368504045593976641993133516736143871216608515263926634011226715284914455711402415353892834760421466599994456779544165463545776999800582771585732652412261928418748228644954236792342967790843376437608448081638932267087091351261481712785115281819966927952293373599847490342000520214580736095380133612672268490657328176148490030319945640792372975112117958950096011152903036886614820972245118969067900653786234578772938473823177949242914588195633809050453458562302798500873089485750

**Public key 1:** 23588914219017859532886582164563527546694425669329356741520324018340420227232590007326041707088975964234065453517620612914438740238081713544785800084678242082699360529846392884285682641423807439569780676297207389998052831831550241368435448651250787193981154910215721098788337685040922071127774497930978715962754087803872302692796264943670033458445966786761805200915096777258973398358631954243547294677254702796827108541708746327369226538164965789712004181461595894762981028577698584099284586653226119220339995923345125471142374046835141380692476032506915682254312280148756926272842614547490350150173844817472771731820

**Public key 2:** 28579068021347094198681757418256654730793728062517197023108114028614810292912562401124842322123617829973170827695536752112153381959280115453300068175303400862682755100134599393526724765079774511458672975073289866212086175876777817802318068060562939695163577098251259382633533998199834722059375434761618452260789063604999558104664172881339280366114143063744827975026765300865217781914884656576683744894825261754747265681769374968021909797042224593208403640995727101349779202730899296931451125709295653900121224455854110040002231613314194713058941771465276342459327481116922504185604174354726102202105227554540979061653

**Shared secret 1:** 24987627590650235790650448857643353111459168490155178937164262233607947737681525692463786904424162855042542077994074341603840396119449581470012820081983229303277057077293262907961088129053988809261261903770873602222616483137616253337169768241476059534505563012069674248285060054166995268670432966094129742135650601763135996396738386852257549027307107086136757956168175208628039939176226281006310091584378925343455748090917081376813397911341552912799367417884146014710471885737187823952128562484469535148597718449024366668350517544779163078137848233087601337164081094825020757065207293667496800634447716837440863581312

**Shared secret 2:** 24987627590650235790650448857643353111459168490155178937164262233607947737681525692463786904424162855042542077994074341603840396119449581470012820081983229303277057077293262907961088129053988809261261903770873602222616483137616253337169768241476059534505563012069674248285060054166995268670432966094129742135650601763135996396738386852257549027307107086136757956168175208628039939176226281006310091584378925343455748090917081376813397911341552912799367417884146014710471885737187823952128562484469535148597718449024366668350517544779163078137848233087601337164081094825020757065207293667496800634447716837440863581312

**Shared secrets match!**

**Shared secret (256 bits):** b'\xabQ\xc5\xd2\xe6\xc3\x88\x94\xe6\xa3\xb2\xdbj\x90\xca\x90F\n\x17\xd2\xd6\xff\xad\x95\x83\xc5Pd\xc0;\xb4\x80'

**Shared secret hex**: ab51c5d2e6c38894e6a3b2db6a90ca90460a17d2d6ffad9583c55064c03bb480

**Shared secret decimal:** 77489977049906984294413060619135534752564390230583346125826464717709866939520

**Shared secret length**: 32 bytes, 256 bits, 77 digits.

## Conclusion

In conclusion, the study and analysis of RSA, ElGamal, and Diffie-Hellman encryption algorithms have provided valuable insights into the field of modern cryptography. This exploration has equipped me with practical knowledge and experience in the implementation of these algorithms for secure communication and data protection.

RSA, as a widely used public-key cryptosystem, has shown its strength in ensuring data confidentiality and digital signature authenticity. Its reliance on the factorization problem makes it a robust choice for securing communications and digital transactions in an era where cyber threats are on the rise.

ElGamal, another public-key encryption method, has demonstrated its importance, particularly in the context of key exchange and digital signatures. Its security relies on the discrete logarithm problem, which makes it suitable for various cryptographic applications, further contributing to the diversity of cryptographic tools available.

The Diffie-Hellman key exchange protocol, a foundation for secure key establishment, has illustrated its critical role in securing communication channels. By enabling parties to exchange secret keys over an insecure medium, Diffie-Hellman addresses the issue of key distribution and has had a significant impact on the development of secure communication systems.

## GitHub

[Sufferal/cryptography-labs (github.com)](https://github.com/Sufferal/cryptography-labs)