

Ministry of Education, Culture and Reasearch of Republic of Moldova  
 Tehnical University of Moldova

FCIM, Software Engineering

Report  
for laboratory work No. 5

# For the course „Cryptographic Method of Data Protection”

Done by: Alexandra Konjevic, gr. FAF-213

Marked by: Cătălin MÎȚU

Chișinău – 2023

**Subject: Cryptography using public keys**

**Objectives:**

Task 2.1: Using the platform wolframalpha.com or the Wolfram Mathematica application, generate the keys and perform encryption and decryption of the message m = First Name Last Name using the RSA algorithm. The value of n must be at least 2048 bits.

Task 2.2: Using the platform wolframalpha.com or the Wolfram Mathematica application, generate the keys and perform encryption and decryption of the message m = First Name Last Name using the ElGamal algorithm (p and generator are provided below).

Task 2.3: Using the platform wolframalpha.com or the Wolfram Mathematica application, perform the Diffie-Hellman key exchange between Alice and Bob, who are using the AES algorithm with a 256-bit key. The secret numbers a and b must be randomly chosen in accordance with the algorithm's requirements (p and generator are provided below).

**Theory:**

Asymmetric encryption, also known as public-key encryption, is a cryptographic technique that uses a pair of mathematically related keys: a public key and a private key. The public key is freely shared and used for encryption, while the private key is kept secret and used for decryption. Messages encrypted with the public key can only be decrypted with the corresponding private key, providing a secure way for parties to communicate without the need for a shared secret key.

Asymmetric encryption offers several important advantages, including secure key exchange, digital signatures, and a means for secure communication between parties who have never met before. Common asymmetric encryption algorithms include RSA, ElGamal, and Diffie-Hellman. It plays a fundamental role in securing data transmission, online communication, and many other applications in modern cryptography.

The following algorithms are popular for this strategy:

1. **RSA (Rivest-Shamir-Adleman):**

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric encryption algorithm named after its inventors. It's based on the difficulty of factoring large integers into their prime factors. In RSA, a user generates a public and private key pair. The public key is used for encryption, while the private key is used for decryption. The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, which forms the modulus for both the public and private keys. The algorithm involves generating a public key (N, e) and a private key (N, d), where N is the product of two large prime numbers, e is the public exponent, and d is the private exponent. Encryption of a message M is done as C = M^e mod N, and decryption is done as M = C^d mod N.

1. **ElGamal:**

ElGamal is an asymmetric encryption algorithm based on the Diffie-Hellman key exchange and discrete logarithm problem. It provides both encryption and digital signature capabilities. Similar to RSA, ElGamal also uses public and private keys. However, it operates in a different mathematical group. ElGamal encryption involves generating a public key (p, g, y) and a private key x. Here, p is a large prime, g is a generator of the multiplicative group modulo p, y = g^x mod p is the public key, and x is the private key. To encrypt a message M, the sender chooses a random integer k, computes c1 = g^k mod p and c2 = M \* y^k mod p, and sends (c1, c2) as the ciphertext. The recipient can then decrypt the message using the private key.

1. **Diffie-Hellman:**

Diffie-Hellman is a key exchange algorithm used to securely exchange cryptographic keys over a public channel without needing prior communication or sharing of secret keys. It enables two parties to agree upon a shared secret key, which can then be used for symmetric encryption. The key exchange is based on modular exponentiation and the difficulty of the discrete logarithm problem. In the Diffie-Hellman key exchange, both parties agree on common public parameters: a large prime number (p) and a primitive root modulo p (g). Each party generates their private key (a and b, respectively) and computes their public key by calculating A = g^a mod p and B = g^b mod p. The shared secret key is then generated by each party using the received public key and their own private key. For instance, party A computes K = B^a mod p, and party B computes K = A^b mod p. Both parties will arrive at the same shared secret key K.

## Task Completion:

## The tasks will be solved using the website <https://www.wolframalpha.com>.

## Task 2.1:

## The Public key is generated using the following commands:

## P = RandomPrime[{2^512, 2^1024}]

## Q = RandomPrime[{2^1024, 2^1536}]

## Results:

## P = 66283446345071763693380316397691419239664342530904593926636764594592488454533912532582606941406364630097156371306793279374175028526948137436901205557801628098562620820296837342428232836192669285901197870913793660372422279722124898815831493410292368949958361665325841388882155006503562859888844306617794238013

## Q = 2199084176824905907827538245330504943176926250922701310367497714425536684330660366765028603064551790287065450223365632413512454733434585981930759477924906956864267681968569912769535308694313454450096742540123808275393958439849695809717912249109552435767755037083176978872892460359373434728305325948562774792455267678630481828985020262153169429840122581958229591779335782740004270064467318091308002311630769023215087304593813566339887136662716862043112919965465611

## n (public key) = P\*Q (https://www.dcode.fr/big-numbers-multiplication)

## n = 145762878042869957603743314832473441168635868504918634327706548401833475112644318811671694973807950467469989202805244273514032970541402398121313640067213531724104831662692927416028352952339419786905054622880976706693028011062002089097706211562214860930186893100344061546260493134755445849690177218553149800320074929437151586967977638906348106482863117690675583642974721993571741035188315381513827108794387800217326388268405967207481809810645185944551676003885538087474362588732579434300564015886880477375663367009707280305331913462014045269545509039641454194873058927677080593033226382588249936319981579585154225034953076060224548100352492781273145480688202215144697073341065809321128839950468381433880965268550179841194442479291751382318715339971062720741247422500470943

## Next, I computed the LCM of P-1 and Q-1 (<https://www.dcode.fr/lcm>). By Euler’s theorem, this should be equal to (P-1)\*(Q-1), since they are both prime.

## phi(P\*Q) = 24293813007144992933957219138745573528105978084153105721284424733638912518774053135278615828967991744578331533800874045585672161756900399686885606677868921954017471943782154569338058825389903297817509103813496117782171335177000348182951035260369143488364482183390676924376748855792574308281696203092191633386312640876721113510025016776836266923281031906625480388767870713191034058476275835791466417621639334988376823007506722465661559179535099993770152754326771844054441426948707638025118718250191297210204015594164924232195560071612649154319177367987493392123445632239588717894509091342307995773923136122564315406811618303162207662421688348481189976363132288664766094628011609026474141293517041251846592110758410143269614974809424822303970283290914328308971314123461220

## The exponent I chose is 65537 – the popular exponent used for TLS X.509 certificates and the last known Fermat Prime.

## Given that, the private key, via Euclid’s extended algorithm is

## d (private key) =

## 1989114555080947130378479910562106101161430617812313125416363628495451494205434626606421602121583894615367304124013765180168711109564483341010851357758893986683213367263302278393396457833624076416203882555857304545815819835509465925350798101944562979058605236676600582513780526422981731514100154504977345694049981398972877841445202557707911688211636437599406865833474132886508213036661674090315528586259924493761234604243115686570028023214143866313237402989417546045686142133554559800460610679929299492347143562846773325449156588557203951387410253087887598488402112739332484859269355999554827125484406494555443741290433553790506222691834836473290895420366630467905684785295486428064455836870965154911100802086296730530612538792245198841617781408045017100354609939217433

## The last step before encryption and decryption is transforming the message – „Konjevic Alexandra” into an integer. For that I chose to transform the the text into bytes according to ASCII using the tool <https://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.html>:

## m = 6571339555131150922286486951457441001534049

## 42 digits is way less than the 700+ digits of the private key, which means that there should be no issue encrypting this message using the key pair. For modular exponentialtion, I used the website <https://www.dcode.fr/modular-exponentiation> (c = m^e mod n):

## C =

## 10029654958850340334815218423719814138611757209368555630504412025863560485035836139426746111017471658474898300590904820165338944618351781692926801905214738322745150719975728614341863917444232959423054977428189107370169706794320261145639024877824612980320551631630620392997835627360696889313266502127028311822993461680028695836383745895828621910199216439500292906648016234846260143547904476913008891745332815308480631702393341931197184136310505846229919664539139902134815018442075844433214617050693654473881215249584767773918707937142109679849892237410086592929302966396375718878120404642273685371129757775908831457165411357436208831515061584580806676706655420008830142678602364620397851539673985638926895570299937965984133450988902844006623226356574403331975094855551144

## To do the reverse, the encrypted message will be exponetiated to the private key, which is the inverse operation. And the result is the same that we started with (d = c^d mod n):

## 6571339555131150922286486951457441001534049.

## Task 2.2:

## The given p is

## p =

## 32317006071311007300153513477825163362488057133489075174588434139269806834136210002792056362640164685458556357935330816928829023080573472625273554742461245741026202527916572972862706300325263428213145766931414223654220941111348629991657478268034230553086349050635557712219187890332729569696129743856241741236237225197346402691855797767976823014625397933058015226858730761197532436467475855460715043896844940366130497697812854295958659597567051283852132784468522925504568272879113720098931873959143374175837826000278034973198552060607533234122603254684088120031105907484281003994966956119696956248629032338072839127039

## And the generator g is 2.

## The plaintext message, converted to decimal, is as in the previous task:

## 6571339555131150922286486951457441001534049

## The private key, a, will be chosen via Wolfram Alpha with the following command:

## RandomInteger[2^1024]

## (given that p is 600+ digits, this number is within the limits of the constrains) which evaluates to:

## K pr B =

## 78055468146677511273371268691936351107490984499446201725275997747606268413367558007475888026344294734826261106601243040850305056129467245415448675898633433433011922208567026570552491192859109286343305366267752616120646159179397338193698749765556385722030398200556165022726750331453670872627164625972592283959

## The public key, is computed using the formula:

## K pub B = g^(K pr B) mod p

## And again, using the tool <https://www.dcode.fr/modular-exponentiation> that evaluates to:

## K pub B = 287147995938547044032911053915092951772630773664920247221222396948866870082603019545919908515294286649567284703183343012693400313852275897160652896297079018108506317216402724037165097562950768672780357480908204625370744957853568325432074065626252799231480009771841176609070042104413918681854308635155970318834166690378329715264867059888902177860225831513676792437386661574993631425834065027030773802460498501115929742006254312421582051795493304417970315654228197055534532897389873774162916516647736188096217715595325928518093763088907977836914437021285920855657034252764066908040484199724727943074642046239748582303598778643207490645509723635204972391627356650635871960474390189340851611580930030830291833613960818202839731226406174992807639202409507663314651471331485703231963169831296771967775059059669898804437799738818159269951746217622511437286698816985662826181093856770877651082859958780499376441245467241333152045853457599044728116896488391125131939349382533209938284682393818920129748730480468368323265038565921301540385057203055540496431457187201249675269555360266655373083752879346133269620954891205208541935168606105571772324098761451536064786978108615233

## The recipients can publish the public key K pub B, the generator and p.

## Now the process is done again in the same way for the sender of the message.

## K pr A =

## 83049133145191695783589617852592530230534931576127015286129883162068987279870873979859577336571392809936220016130163722687291558171280391271828811023859279402145331176005488892623553928767765077397129951765507698479399425570764531930922990170829538130973188275619784182349205188167091463239120810479717527699

## K pub A =

## 26497063199767620739194179903142363763663478048379554607711789159212482676655021936442619549583608496075945431318710975970034465757033661393072610118556517312877494472186317058846880902601057724346838842748694729546396437913249472132406003120464250135014056212487501559106822279305423897455391169853152229424019408585246345951373011657964214316943904378465509681239603804026869844551398496841427804125730145113657712796978225373491207231462818080623881753584053351989169972621016579324676138098408034757886403814997911026322850659821678736986233777672087551753082919769631937899744654157706588056994509808151699206302832317379094975227114268580263188226256984888655967587621401138377363626118754847818134545765894179508700265559538456785296402034554312019581060858945916696707485412792547037752175002213403301281858516442573390164123010398700318253610438157987504876569026743704135760673176205830984155136397005561366114984434249096429566522441719099596891161370117901251946268478899655709995720774194072448483601418061878699316886130684775158140025960182939840727332016388212063254309807450656740454653750346206569704944630162824287671192859353305971335334620095677

## Then the sender can compute the shared key KmA = KpubB^KprA mod p:

## K m A =

## 293981397981899522733306023452786807547206546781587868433967205647045831810459168884484713380333288461997408355559289284667628945460274130676858428275415563603803682183819617274841234276169529993753146980958001989904651030940938586034652164534224482164718936499704767364128480647238066177715196393543359874725689321840796105058021032296855345127793502397080725089493100877588802684841313183430553381687580195997057555086033679130540167233783275172894316266547208766298038100231009178448430349225621994257494884036907631614989414316830384386789315948215291908888863935237521203931076244196388425736770054812061444820052760429841444399565021407052340053324119292197961372445181914059436686714780549236853207567360700871162221695892943462644258803569423568896794721856903099593939926047327637769365819286080968269311660073566707466244430485708330237836524762119543756428074075829576189373448394935066458294395467732062803337063184432299779262211073312014863890703519056825974008865925831210386187896826841981612307949358616973831120974648731653875630692231851435304717069036713477295325576676875784739231154529029959381623032052642543034078911802287820508549003502082360

## The encrypted message will be E= (m\*KmA) mod p

## E = 1931851589031209439253392484045919761082557521880363609254100599768002969275858668209100895085862286616848491942588420408235196506132479070116540584561663441721258618429587986862767080210673215412402852761768979453374877097573085948674582008402830069673581390330072152804262813534085224612651802217328826349126064109769186101625276417907245458518563041599058448890425454264332545900450279959771122794597983723046504306384600853489480496790350518294456011435311374002925871568786317035548820340534401722621199137957639841747795084821804168622107712852272275636087967177965694593572307125896487179516973585614497764318775837790247589687715895902396014072215872001985469563584616383127903624525335416056946121217983728811740170589840658217306427220722247647460356743670674031150372681579248513140884255410816271658060111552035801699149589141185233725858281491669144185833494974876033198867033719569134924700958468673271117917481477502960152397517976147456555081519538885053839195874550566950734059447558151588308382087247925988782153823797012393934879942442752653708080834090815042272579892247836408175863717113582819379847830532800089139013923080804156317526856499145732341794667353974810941599507114125942275640

## The message transmitted via a public channel will be the tuple (K pub A, E)

## For the receiver to decrypt the message, he must also calculate Km, that will be KmB = KpubA^KprB mod p:

## K m B = 293981397981899522733306023452786807547206546781587868433967205647045831810459168884484713380333288461997408355559289284667628945460274130676858428275415563603803682183819617274841234276169529993753146980958001989904651030940938586034652164534224482164718936499704767364128480647238066177715196393543359874725689321840796105058021032296855345127793502397080725089493100877588802684841313183430553381687580195997057555086033679130540167233783275172894316266547208766298038100231009178448430349225621994257494884036907631614989414316830384386789315948215291908888863935237521203931076244196388425736770054812061444820052760429841444399565021407052340053324119292197961372445181914059436686714780549236853207567360700871162221695892943462644258803569423568896794721856903099593939926047327637769365819286080968269311660073566707466244430485708330237836524762119543756428074075829576189373448394935066458294395467732062803337063184432299779262211073312014863890703519056825974008865925831210386187896826841981612307949358616973831120974648731653875630692231851435304717069036713477295325576676875784739231154529029959381623032052642543034078911802287820508549003502082360

## (KmA = KmB)

## The result is as expected – the 2 communicators over a public channel got the same key without sharing their private keys. The decripted message will be calculated using the program below, by the formula: D = (E \* (Km)^(-1)) mod p

## 

## *Figure 1. Program to compute the value of decrypted message*

## D = 6571339555131150922286486951457441001534049

## Task 2.3:

## 256 bits are needed for the AES-256 private key. Since we can’t predict the size of the shared key, which will be up to 2048 bits like the modulo used, the result will be converted from decimal to binary and then the 256 bits will be taken from it.

## The same variables from task 2.2 can be reused. The recipients share g, p, and their public keys. Then each of them raises their peer’s public key by their own private key. Thus both actors calculate the shared key (KpubB^KprA mod p):

## 293981397981899522733306023452786807547206546781587868433967205647045831810459168884484713380333288461997408355559289284667628945460274130676858428275415563603803682183819617274841234276169529993753146980958001989904651030940938586034652164534224482164718936499704767364128480647238066177715196393543359874725689321840796105058021032296855345127793502397080725089493100877588802684841313183430553381687580195997057555086033679130540167233783275172894316266547208766298038100231009178448430349225621994257494884036907631614989414316830384386789315948215291908888863935237521203931076244196388425736770054812061444820052760429841444399565021407052340053324119292197961372445181914059436686714780549236853207567360700871162221695892943462644258803569423568896794721856903099593939926047327637769365819286080968269311660073566707466244430485708330237836524762119543756428074075829576189373448394935066458294395467732062803337063184432299779262211073312014863890703519056825974008865925831210386187896826841981612307949358616973831120974648731653875630692231851435304717069036713477295325576676875784739231154529029959381623032052642543034078911802287820508549003502082360

## Since this key is way larger than 256 bits, as per the convention that was mentioned earlier, only the first 256 bits are taken. The hex representation is the following:

## 4E3C2CFFFCD8FBF5B01E6A780D3153F064B7CF0FA33C322F8FA1020D3BFA44BAC33A6EF30F44EC9781C459B8B03E3

**Conclusion:** I used a variety of asymmetric encryption techniques to transmit data between two strangers over a public channel for this lab project. The drawbacks of asymmetric algorithms include long computation times and a maximum message length, but their main benefit is that they can be used even in the absence of prior agreement between the recipients on a symmetric private key. By starting with an asymmetric step, this procedure can also be utilised to obtain that symmetric key.