

# КРИПТОГРАФІЯ

## КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Вивчення криптосистеми RSA та алгоритму електронного підпису;  
ознайомлення з методами генерації параметрів для асиметричних  
криптосистем

*ФБ-12 Приходько Юрій*

### **Мета роботи:**

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

### **Порядок виконання роботи**

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.
2. За допомогою цієї функції згенерувати дві пари простих чисел  $p, q$  і  $p_1, q_1$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p \nmid q_1$  і  $p_1 \nmid q$  – прості числа для побудови ключів абонента А,  $p_1$  і  $q_1$  – абонента В.
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ  $(d, p, q)$  та відкритий ключ  $(n, e)$ . За допомогою цієї функції побудувати схеми RSA для абонентів А і В – тобто, створити та зберегти для подальшого використання відкриті ключі  $(e, n)$ ,  $(e_1, n_1)$  та секретні  $d$  і  $d_1$ .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її

виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення  $M$  і знайти криптограму для абонентів  $A$  і  $B$ , перевірити правильність розшифрування. Скласти для  $A$  і  $B$  повідомлення з цифровим підписом і перевірити його.

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа. Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція `Encrypt()`, яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою

<http://asymcryptwebservice.appspot.com/?section=rsa>.

Наприклад, для перевірки коректності операції шифрування необхідно

- а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері,
- б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

### **Хід роботи:**

Напишемо консольний застосунок мовою python що імплементує в собі класи для приватного і публічного ключа. Всі додаткові математичні операції, наприклад `xgcd` та `mod_pow` напишемо самостійно. Клас приватного ключа має мати статичний метод генерації пари ключів використовуючи генерацію параметрів `rsa`, а саме генерацію простих чисел заданої довжини. Їх перевірку опишемо за допомогою тесту Мюллера. Клас публічного ключа зберігатиме публічні дані (публічну експоненту та модуль), та матиме методи доступні лише зі знанням цих даних, а саме шифрування та перевірка підпису. Клас приватного ключа має наслідуватись від публічного, і додатково зберігати приватну експоненту

та функції підписування, розшифрування, а також функції необхідні для передачі інформації відкритими каналами з підтвердженням відправника.

Так як в реальних системах теперішній стандарт довжини 2048 біт, а рекомендований 4096, напишемо дешифрування за допомогою Китайської теореми про лишки (CRT-RSA) що дозволить нам пришвидшити виконання коду.

Результат виконання програми:

- Створення ключів для Alice та Bob.
- Передача зашифрованого повідомлення відкритим каналом з можливістю підтвердження особистості відправника.
- Шифрування-розшифрування тексту.
- Цифровий підпис та його валідація.

```
python3rsa.py --keysize 2048 --pubkey a --privkey b --encrypt c --decrypt d --verify e --main.py
[*] Alice -----
PrivateKey:
e = 88c9d0d7
d = 63cfa4a50ba5b3cd33bcb3f7f506094fd9e1d89238f5b4ea7e83951ac12b12731f8a02aafa84615de8ef9e47d25f4c0f20e82b158f051dc1bd436c9540e62b1efb752c8a58a6ae10ad9c2e980b03cf7d423dc2010974dc8c7160b2b28bec8ec4fd078676b6cc498ec2da5771cfb2b3ebc1cdcc8ac1081e18d1ad17e1fae276b753b26772a2649cb2b076dc071b8422f9c936880e8bdc7f962b643e43a4b686c9629a88bc5ef3ed462e5f9782edada9d6566b250738ce86ab85456801d1046ca7f6cab6b5b41bd0ba421d3e8ae6f11ac9684b8d8443cbf4ff3c840b9f31e12aab4c0590ac2eb8fdaefabc2a492128a883b90167b46f3502a0f1260c986e57
n = 60bb8796515338545331c9d685a74dd201c2d82b2085f93198de9c2908f0a174da0b97cde927e0d89cd94fa3e8b7a3122ef9ba38d6a7bcbaf08a8d01443920c20fbaa354f22bc8a54b3cd5f9a9c8b571874025b4c3c0940bd41fbc27870893d238662f0bb5976a7b0cb20c77560aa2c0b7b79ff980aa45b15f233e11546ab4d0ebf27f678829ed08843baef752ae58f64c8977cfce0e2130137ad44667b2bfc2e37baa95a667e5114724dfe814d60cb869cfbeb33e11cb7b5855cb07d26a6ab9cde32d757a9bf3fed07d47d0442c6ca2a15e275004cea834f922850ab29640b7ea3ba159ac79dadd224e66c0d4ce25280b4f591ecbc427cd0b

[*] Bob -----
PrivateKey:
e = 8832fe65
d = c37ab2c2e8d7612f4910ac8ad4979492075689d96f1978bf2f38cbab8b17662e8714645e6e6ee44d37d5699ff4da9ac3fc991541b7120b3eccc909a4ede431fa6fb28c19be6ef119574cb9e764c52f9722fbedf9429aa5ace7e62fb973f90b98214ce45887e0092b74cd4d082b1d510ed4f005eb2185f116a4eb7b4f22b1bbdb7eb115f4cfe14315f08f4ea76a3f1a1c161fe3c4f84230f722c7e718f4de74c1aac6e5cf9e38cba935ce808d00b45f92d5717dca5364d71d40aa2d0bf858b1a1e6e8ae26f16c14647a30cfda9f05b09260063835b109f54e571f4bb3f4a08572dbdbfb074b3f0b1b9bf4f0c10c5ee2581d09a21f2ac0d
n = 7346a620b9339a985ac1773b75b9a75f51e5b4bb2c41017daad9f5698b252770dbdcaef71a77d395c1a61c911257a88cd6dc8aaecf1a0c4aca8f435508a9bd2a17ad01833843d12abc6d75db56b6ccf9e8aa722263a56941a255b349c8fab2c487e446c5d37832b3c5bae508e8738bae4089974f0d7aa63a39e5ee1b20a15403bb116c624fe88508925967f851e2c98f0a5e3203ee4e23775f61e86862ae8ba82b9f4c1a6cef65696a3ca9225a9f1f61227feefc773b6a2d1507b4535e5b1474eaccac84d65f40aad98912f6faac101405ec2ae3f7c6bb604eef2202c849642c2d91b462ac4cc87b8baed535152c67a930dd415016fd636f49e5

[*] Alice signed&encrypted
st = 454525924139588302577902085648303208700979172754511619413542306790349148314136881287752159026237038016808951478654060574047393387605178322718754319834991417944388416983778125724619579594695270745511344049708747644609752483017758190879390973768545235851976274294091789398266797285413833289820562643641409514696791530808759384685801180440976364957234225874121334567162861763482540308760677902794497117647785015234250624017083440019874487001667892248690478467591862924102637370914913707481447840453015030570961315693421479048113636489994725847334603698599352362509241849409135774073336870192102588319801913
s = 195903116065094959469872838898447539996447711929260304443191006949296768721286959213997958686422232115299188826376583854417178599937835992052274502571460193687561705268980688151659414312698167622868927947187855922138632472261206795722458593954496804944321046062613862880302124685669498716379037066565661430425783774721727559251894746569724167326924113989462441162802552303566629135577696164801079861186759880540123716320887970631634908034836665728556338279901512217895792999034105730630645807109927958323043717379804425309792086808228437727124353477453877381031443056419194635812486822126379603806613
true Crypto is fun!
[!] Bob verified&decrypted message Crypto is fun!
[*] Message: Crypto is fun!
[*] Message encrypted: 4926177343945362434646498291662567007057042746144844661904640491101312274394932437272720978421678774315058013437990641979426103850908193151150942246808027821692427809240545007371241424713708692852240010414932688246207763311069706395629353425134707456359281834815641750275599357456825644596771685161931435326264891246719740801795737931539443282668385385581636034479083475975210052345968056985621228867759981667300326861099730779789645803273691910553087599982364723530802685843532306924114686231777822915899177623168478926837526293221144099721562982904859438693239423360415965248984558248980745788097794867603805989
[!] Message decrypted: Crypto is fun!
[*] Alice signed message: s = 2205549373950685490524650775834305121739010851544772589472684114264357611640819864712976879496114444205029007162485631477101671512444280473784727947108495276485321652100309249711258633957655282089221045263448648036130672910268073483097373693364904658097012840400957807249543388671122315835059998443977463402092921946073750954996947955683943841152919923136546080657149895532636744503213470177329748016624397270780762253934507195093953408944470515831847918142585488637711508474367937552241252084920010742938663109130450271638523593922898093782587281304178196548562077698641550938132499935242266590
[!] Alice public key verified message, result: True
```

Також описаний оракл для взаємодії з функціями шифрування-підпису для тестування з сервером лабораторної роботи. Візьмемо собі ключ Аліси, та перевіримо правильність функцій свого коду разом з сервером.

1. Зишифруємо повідомлення у себе використовуючи публічний ключ сервера, перевіримо дешифрування на сайті.

```

In [1]: from rsa import *

In [2]: privateKey,publicKey = PrivKey.gen_rsa()

In [3]: message = "Hello from client!"

In [4]: serverPub = PubKey(0x10001,0x8280831f0727161b4c98103a67f3bf9c3b10998ee16c9090d824f8e2c
....: F255C400EE4E81E2E3156718F9AFD4C023F14B50A4C44BA35DF0A92A5FF1E9B1382375B9679EE67420B031
....: 09F43B32EF36A806508086C8864646C533FC39D37AE8EA60072C6BC9D192C8132E05C910D2B76779766C8

In [5]: serverPub.encrypt(message)
Out[5]: 20009435053796486884134220553632718806900572662489433782909422671706714546341138119773
6245322153408581225897123320364864339889475867634160295596056868855488928046445152356748845314
7219991565369123937190822004625904786335721385310352691014937317450949374365866937051778194200
782480814335548325023749726656794200906043265393654

In [6]: hex(_)
Out[6]: '0xfd9bc0e7893c051de961c2b01a59f6320a3421cadf59014752cbbc8ad01adf69b811159505992df6dfe
5fc8a656fb61224c6c10fedcb054c6ea46e4c2f002277389ba2c6505fe16ef7463d9b1087a2fc0aaa6c7f6ce6ddef
7f98c648003c311fbfd82c1869b255aeade833b94307514f343bac65b5072027eff31e45b1e0f304ae6c852382f18a

In [7]: 

```

## RSA Testing Environment

Server Key  
Encryption  
Decryption  
Signature  
Verification  
Send Key  
Receive Key

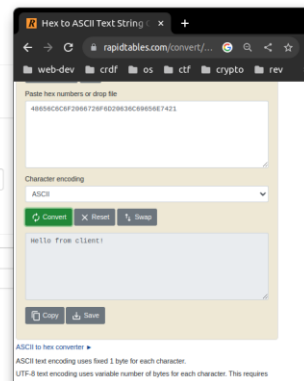
### Decryption

Clear

Ciphertext

Decrypt

Message



- Тепер нехай сервер зашифрує повідомлення нашим ключем, а ми його розшифруємо

## RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

### Encryption

Clear

Modulus

972cf864a7836ff83b9b8d4abbc1ab7b36fc9358ad57439d74c5edfad5b05cc490ce2972da5f9a74755e03006784dc

Public exponent

b0d822bf

Message

Hello from server

Text

Encrypt

Ciphertext

3327880b983984202bd51ba11f43195df73c44fce71ebae9edc565c51525d17ca2876281d0199082ab5d

```
In [7]: publicKey.export()
Out[7]:
[2966954687,
 19084156967671590971011249280029926000925824514096505155128574303180027388044849372
334286529578092339596476098671656707203783114432559761647914583524206272059587106896
089429564488783499540110892219702209415412118466551729103846334848210792793535351603
265715359643941288845242020373324416289740801]

In [8]: for i in _:
...:     print(hex(i)[2:])
...:
b0d822bf
972cf864a7836ff83b9b8d4abbc1ab7b36fc9358ad57439d74c5edfad5b05cc490ce2972da5f9a74755
eeeca5253b9c797d022a33f5d8ed11d303ef1761c5ce3a57703cb6c7ad08b5e18684813a6d5713f42cc6
cb83f5413f947f33ab037f71e9a93a6abe8567b24eda2ae02eaa35ac62041eadd679e311fb73836046b9

In [9]: privateKey.decrypt(0x3327880b983984202bd51ba11f43195df73c44fce71ebae9edc565
...: 836389c8519994031e9ca5d06e58a463df4e4e2f67eea18cd0e08505d46e8c20df5cb8f40646c9
...: b3678dbca644fe6fcd940d05cf403d050e6a8d5a4d502b4507b35a91af4c5d456378fac707b6d3
Out[9]: 'Hello from server'

In [10]:
```

3. Перевіримо також цифровий підпис, підписання повідомлення та верифікацію підпису, у себе на у сервера.

## Sign

Clear

Message

aaabbbccddddd

Bytes

Sign

Signature

2f61da698e61666340cb564ec7004861eb992d03ad88b08e226826746929b3a10ec24c1a3469754451c9a1

```

In [10]: serverPub.verify(0xaaabbbccdd, 0x2f610a698e61666340c8564ec7004861eb992d03ad88b08e226826746929b3a
...: 1e31e840e67ff33ad83881400c85e334fc8c5e122a9f2e4a2ffe72f785b97a896f4646e50594c02e0230f882e475f6038
...: 144d48492425706485a95faf30ac1a71704fe6e48d064e2bc309fafb3ab8b72f3e0649ae9121ce45623050e05a87ef73
...: )
Out[10]: True

In [11]: publicKey.sign(0xeeefffff)
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-11-8978e166d3b8> in <module>
----> 1 publicKey.sign(0xeeefffff)

AttributeError: 'PubKey' object has no attribute 'sign'

In [12]: privateKey.sign(0xeeefffff)
Out[12]: 2053501704653173492995766066995440876508728557815791667618884114152261287646996321762247563428510
7736367073567967361791368858853143038843828516755894119949872288071167804245427688903452826426917756381103
3466135685272101843650039815868330723881702017802840651017602237314489906964568823399334486692552266660246
0944204854790800375980902708320965986066583130958922

In [13]: hex(_)
Out[13]: '0x10445163a2e6ff889ef6f2958a86ae51e64a981a7bde4e5dde58ce64c80ff58f756f4b8f86298bc6751ec01a421f1688
099796fcfaba4777ceb3f125360a7db00236f3c103e92836ae0474b3a0f06028e2830468b750990d1e96817f1d7ce6aa6592b3c7e5
14db337a8d12ae44ba6280e2a3fce5073d2140ff24417ab892a23a53fe4e8b75933aa663db9687e9562c305fbfff0c0be2b90ce555

```

## Verify

**Message**


Bytes ▾

**Signature**

**Modulus**

**Public exponent**


Verify

**Verification**

☒

## 4. Тепер обмінємось повідомленням через відкритий канал, з верифікацією підпису.

## Send key

**Modulus**

**Public exponent**


Send

**Key**

**Signature**

```
[j] Please enter here your Private Key [d,n]:
*] d = 351e6e2f213351f2ad7233becf2d3f3a280cde1598052d7365679348dedc194acf010987e840bb05049db755b643394c1016f92168aad5c3da87a58ac0e8232a33c87390b18d3aef
677c073ad52d664b5bcd99be24c87fe1410130547668da01cebb0caa6b6de78f16db6b034e0e481a58f1cf54f941b8116b339e07d13fc0de0f27a0000530af19ce6fcebe3e3c04e9ea9eaa5f
022b48f8cd42284cc0b0265241b4678f6e9ab07712e7edec35fa5e13eeb136c55f459d51e7a0a2a04922e1df44758e2c94f90e0ff6fb5a70d8ed4981040b3b09b0aa1f
*] n = 972cf6f64a7836f83b9b0d4abb1ab736cf9358ad5739d74c5edfad5b05cc490ce2972da5f9a74755e03006784dc450b651fba22a7cc59d89a3370a498ae6b83a482645db1fd6
a7dfcdeeca5253b9c797d022a33f5d8ed11d303e1f761c5ca3a57703cb6cf7a080b5e186a813a6d5713f42cc60996c18154fcfc2c0151bfa232e46375205d4d151fa0f732f573011d0b352
24debd8cb3f5413f947f33ab03771e9a93aabe856724eda2ae02ea35ac62041eadd679e311fb73836d46b9cf1d1933baad101d99905910cb7628dbd7e7a656fe1801
[j] Please enter here sender Public Key [e,n]:
*] e = 10801
*] n = 82b0b31f0727161b4c9810386f73bf9c3b1099bee16c909dd824f2b2c9b5c6214bd096b33e627658f81a362c9d618732867f9ae55df6a52bea79f2c2a6b12c7028376d0d705e3b97f
cd23f14b5d4c44ba35f0a92a5ff1e9b1382375b9679ee67420b03113fb26ba9049c6577aa56438e717d6fcd2aect6faee56b24a1145fb6aee24c8a29e5757aac42b5bd337a76bb3e2bd03d
9d37aebea6d072c6bc9d192c8132e05c91d02b767779766cb372a55e1e1b3604dba6b2571697abe0f18eb3f8ee588e21266577bd54f8fb461475032a8bde2a07dfb99037
[j] Please enter here decimal value you want to verify
*] pt =
*] ct = 37658f0a35002546cd0745842f86200c7d806f8a50cc2f8e7a9af65c702d064ee7a946083ff29a403984ecbaad33245683058ef228ff0f02c8c4c2e766f64c7c812f8692c2f
23A8BE3040d08BE350EDD5329285578502EE4001217FD295F4A2532021030c849674740325E0FF1C021471A08B7FE959E9FD00B87A0B1303340A00307C39C9409AE7C19631C1F7EE2072E
C8779BE7570AE525C5CF9BA8ADA12F8693F899ED04280C38789C3D0AC91C72A1E20D282C278AE87FBFE082311304E0709FE22481E44875283D0EF312038722A98B41C60EA
[j] Please enter here signature value
*] s = 3C662CE43A131A705F72A538C2A828330BFA35C528EEE6F708346A9FD09D0E9D37057EDB8FEFC08F5C0EE1F22CC0AF8325854EEA01F02E18BEF5EE7E75AF41B8E37A648D733964FFAF
aA1B3C1562430A19AAE51574068DA3708921A5810780668077CA7B56500DAF94B5D3940FE5770C68CA1274E24A91560AC289EF5AF20C8D40FE9ACBF34C1CA4EE4B583C8898AA352534589
08F93861725A724751C00303d02c810330d12CE106457C0421C92102153A9fA6F6A5D09378E05998ED04971328651AECFD6A332F56F36453908758E3D16A497661A3EASAB1873
[j] Signature verified! Received message: ba97302ba8f77fe
*] #
```

```
*] # 1
[j] Please enter here your Private Key [d,n]:
*] d = 351e6e2f213351f2ad7233becf2d3f3a280cde1598052d7365679348dedc194acf010987e840bb05049db755b643394c1016f92168aad5c3da87a58ac0e8232a33c87390b18d3aef
677c073ad52d664b5bcd99be24c87fe1410130547668da01cebb0caa6b6de78f16db6b034e0e481a58f1cf54f941b8116b339e07d13fc0de0f27a0000530af19ce6fcebe3e3c04e9ea9eaa5f
022b48f8cd42284cc0b0265241b4678f6e9ab07712e7edec35fa5e13eeb136c55f459d51e7a0a2a04922e1df44758e2c94f90e0ff6fb5a70d8ed4981040b3b09b0aa1f
*] n = 972cf6f64a7836f83b9b0d4abb1ab736cf9358ad57439d74c5edfad5b05cc490ce2972da5f9a74755e03006784dc450b651fba22a7cc59d89a3370a498ae6b83a482645db1fd6
a7dfcdeeca5253b9c797d022a33f5d8ed11d303e1f761c5ca3a57703cb6cf7a080b5e186a813a6d5713f42cc60996c18154fcfc2c0151bfa232e46375205d4d151fa0f732f573011d0b352
24debd8cb3f5413f947f33ab03771e9a93aabe856724eda2ae02ea35ac62041eadd679e311fb73836d46b9cf1d1933baad101d99905910cb7628dbd7e7a656fe1801
[j] Please enter here receiver Public Key [e,n]:
*] e = 10801
*] n = 82b0b31f0727161b4c9810386f73bf9c3b1099bee16c909dd824f2b2c9b5c6214bd096b33e627658f81a362c9d618732867f9ae55df6a52bea79f2c2a6b12c7028376d0d705e3b97f
cd23f14b5d4c44ba35f0a92a5ff1e9b1382375b9679ee67420b03113fb26ba9049c6577aa56438e717d6fcd2aect6faee56b24a1145fb6aee24c8a29e5757aac42b5bd337a76bb3e2bd03d
9d37aebea6d072c6bc9d192c8132e05c91d02b767779766cb372a55e1e1b3604dba6b2571697abe0f18eb3f8ee588e21266577bd54f8fb461475032a8bde2a07dfb99037
[j] Please enter here decimal value you want to sign
*] pt = deadbeef
[j] Signed message:
t = 821851f128acc6cd42d01e42e7de403540987fec05e51bc6d128a8285bd20288f7777182cb73c8b912d46249b059ea8bdea4af9d0bf532ba37c6b3ed59291809a7eccaf723447c4c34a7
7717c074e4a3e40e55e94d0925b6c1c2645b138d5ae9b4fc39c7f68cf1ee4d293db3e48f198fa1013d70d4e604ea5e2974d9c10a428215645f49e30d36ba6bde7ea3777d3
6c2f522ba2857b5d5d6cf9c08ff33ff369ee138110747e73ddef8a30931869ab2f0dbcc95c68421968191538e7e832e4d16b66b5b34350d8d782b0dc5cd96ba7474
*] s = 59becca1a08719c4ba4c115754d9655c2711f40b5d3f5c188a5c17b68a3da924279d7131255e20df66996d1edd23696cab2ded4f4bc49a0f818343defee7d51c22e1e84168326a8eb0e1
ac40b74a3342d08cd27379cf0b3b6a376e132182dc747ec69e941ed21d30c7fea3a024302cb170a15b3107732692592307dc6ce8674d33d8974238947d1404bd23e0bf1052b4d973b8ddb017
18cb4c55dc2f28729bcbcf5dd54au5656ecd31f84abbc8c6ba9b2aa73f7c6dcdb05e6cc444f5581e19370b64073025c7010fe4b1b82cfb88142083c0b50858af86b
*] #
```

## RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Receive key

Clear

Key821051f128acc6cd42d01e42e7de403540987fec05e51bc6d128a8205bd20288f777102cb73c8b912d46240b059ea8bdea4af9d0bf532ba37c6b3ed59291809a7eccaf723447c4c34a77717c074e4a3e40e55e94d0925b6c1c2645b138d5ae9b4fc39c7f68cf1ee4d293db3e48f198fa1013d70d4e604ea5e2974d9c10a428215645f49e30d36ba6bde7ea3777d36c2f522ba2857b5d5d6cf9c08ff33ff369ee138110747e73ddef8a30931869ab2f0dbcc95c68421968191538e7e832e4d16b66b5b34350d8d782b0dc5cd96ba7474

Signature59becca1a08719c4ba4c115754d9655c2711f40b5d3f5c188a5c17b68a3da924279d7131255e20df66996d1edd236

Modulus972cf6f64a7836f83b9b0d4abb1ab736cf9358ad57439d74c5edfad5b05cc490ce2972da5f9a74755e03006784dc

Public exponentb0d822bf

Receive

KeyDEADBEEF

Verificationtrue

## Висновки:

В ході виконання лабораторної роботи я оновив свої знання роботи з rsa криптиосистемою. В результаті виконання роботи було отримано програму що дозволяє генерувати стійкі rsa ключі, шифрувати та дешифровувати повідомлення, робити цифровий підпис.