Національний технічний університет України «КПІ» імені Ігоря Сікорського
Фізико-технічний інститут

Лабораторна робота 4

Криптографія

Виконали:

студенти ФБ-14

Кот Микита Сергійович

Чавалах Артем Дмитрович

Перевірила:

Селюх П. В.

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 p1 q1, довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq <p1q1; p і q – прості числа для побудови ключів абонента A, p1 і q1 – абонента B.

```
p: 219164896231649635106285341997536085204379323500484044286254778889977160686 39,
q: 104202587695544563947687992312378333120027636515785547309475916649238938369 07

p1: 583620063108924929568803746837948628110796301984014094530561540810522381491 09,
q1: 950375485661283394785412634021937719615953060815712040219425150795621598481 61
```

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (e, n) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n), (e1, n1) секретні d, d1.

```
A Public key:
(22837549319363395437763985310556478961805334391421349943570425726731139014005803696004335396397187405644663440466959452063734965113758764510653483459573, 65537),
A Private key:
(60103765912443328884531366805999381896214109218187503826342783913462298941254777574689185534316956759777551835336499679639886459354224252965709619833985, 219164896231649635106285341997536085204379323500484044286254778889977160686 39, 104202587695544563947687992312378333120027636515785547309475916649238938369 07)

B Public key:
(55465820091881339436508934192183239322242516678445399095018103402855921269004799292411642739463634793458717521766383052718472186451342295996233271174385 49, 65537),
B Private key:
(24228687574810854713953610449880555259345345022613996498607165821165081273309772310173801773371611827831376297156801389329175751010015786971830651346897 93, 583620063108924929568803746837948628110796301984014094530561540810522381491 09, 950375485661283394785412634021937719615953060815712040219425150795621598481 61)
```

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. 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 и 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 $0 < k < n$.

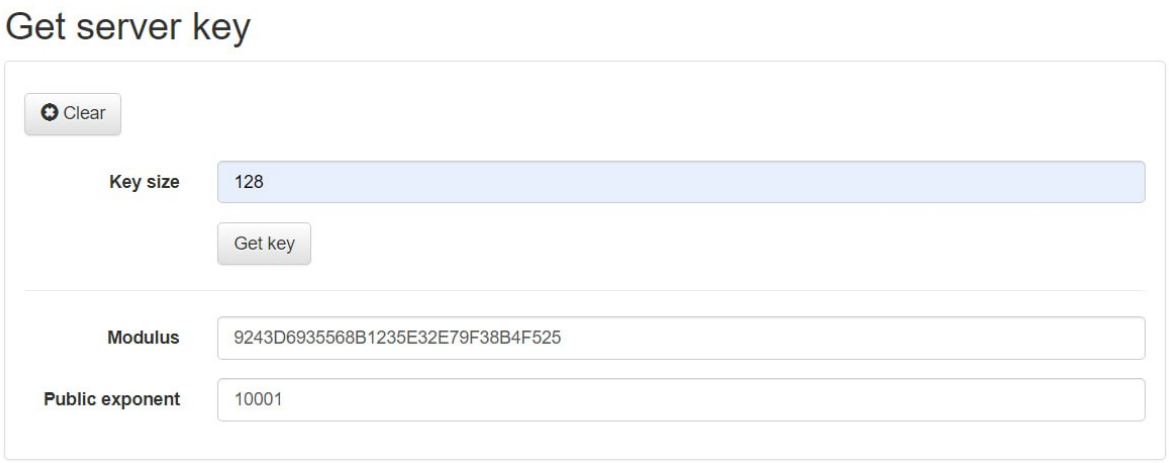
```
Message from A to B: 544
Encrypted message: 11711804417395685396327189228975856471900689389266492599890447172606317195220569369202831564343724205537444569674624478208217366019458412412751086076100 91
Sign: 17919767935426880239841104911164497007028169343731262147383572195168666728864597812924830434131117801238926562327176742438678370794144274534032327069440 7
Message Verified
Decrypted message: 544


Message from B to A: 322
Encrypted message: 20175081135237498557191762593161371086946130023185785258748725165247797670952145514796107022194146390949919500210524308070205234043689287199823068913540 8
Sign: 32049391456101593636355943671349168079043180296806029061453570855756146738624607261162106642092020216068104964322488486738343948509993928300015860361404 21
Message Verified
Decrypted message: 322


Message: Kot1337
Encrypted message: 37032578509857899851866589090783955629621299396138651398265609075094923107936025447364281263077121974236502186536216557646282093164488640740382587228898
Sign: 17318243423662823585490385408735870047409282802504605771731207568048106432901717320508406353953993930598267581703620873813226792151335749170685927089831 4
Message Verified
Decrypted message: Kot1337
```

Для випадкового ключа:

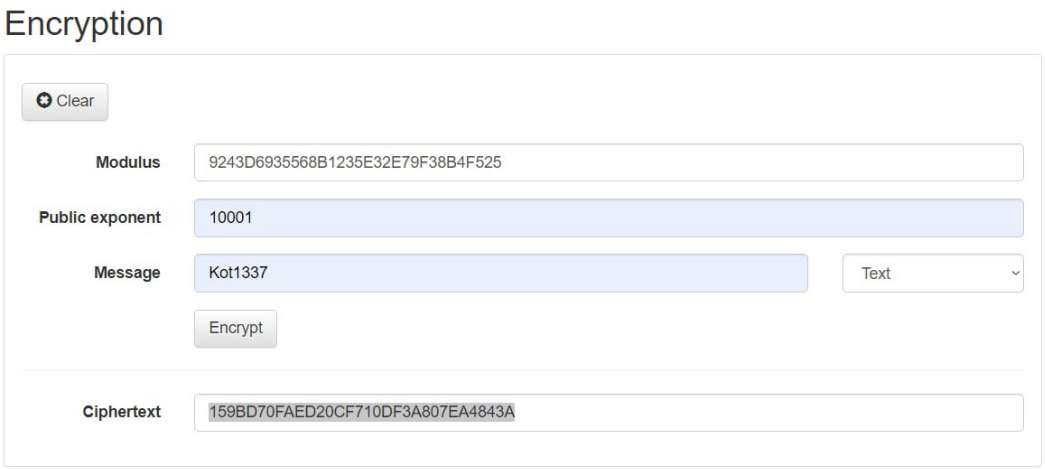
```
Ключ 12972828247229571767130773247552454887302769451474658958542491294456903860736061423475778176221267364380758093277603839468532189396152674596762180571936 7:
('Verified', 12972828247229571767130773247552454887302769451474658958542491294456903860736061423475778176221267364380758093277603839468532189396152674596762180571936 7)
```

Перевірка на сайті:

Отримуємо публічний ключ сервера:



Шифруємо повідомлення на сервері:



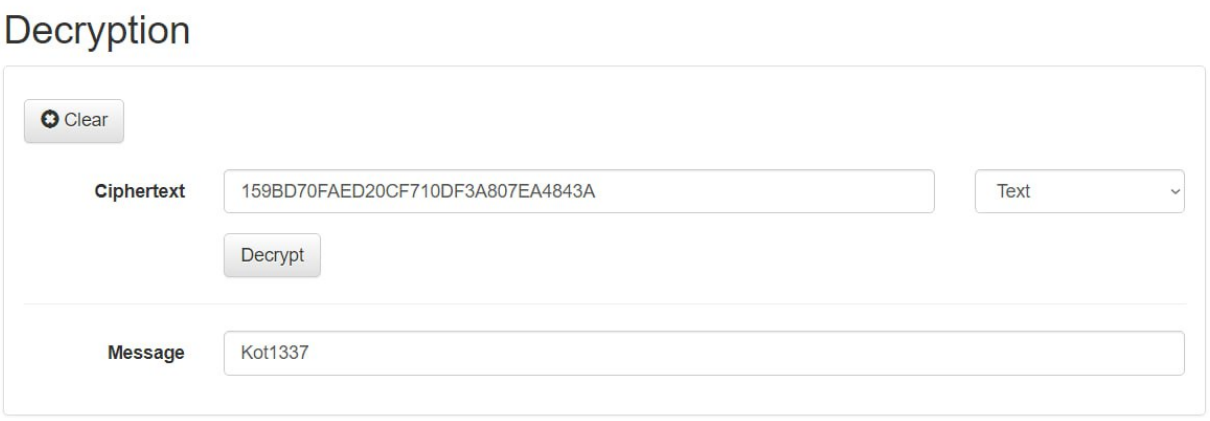
Шифруємо локально:

```
Public_key_Server = hex_to_decimal('9243D6935568B1235E32E79F38B4F525'), hex_to_decimal('10001')
message = 'Kot1337'
print(f"Encrypted by us: {decimal_to_hex(Encrypt_text(message, Public_key_Server))}")
print('Encrypted by site: 159BD70FAED20CF710DF3A807EA4843A')
```

```
Encrypted by us: 159BD70FAED20CF710DF3A807EA4843A
Encrypted by site: 159BD70FAED20CF710DF3A807EA4843A
```

Співпадає

Розшифровуємо на сайті.

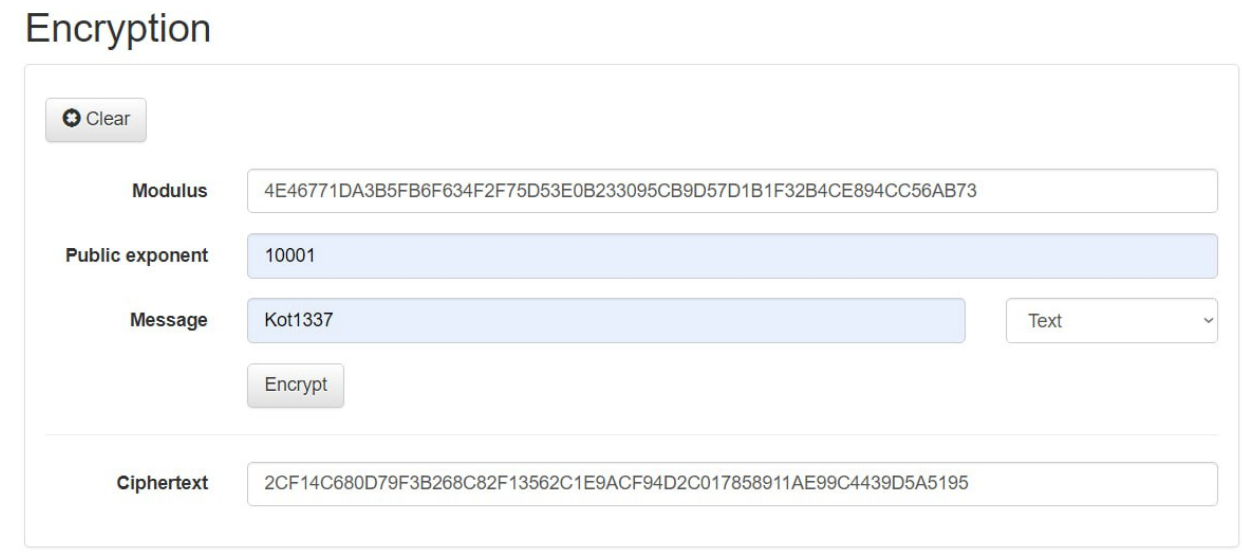


Створюємо власні ключі.

```
p = 160974941335832298448894538799832754593
q = 219940465870775757011659172596898530451
print(f'p = {p}')
print(f'q = {q}')
Public_key, Private_key = GenerateKeyPair(p, q)
print(f'Шифруємо на сервері з нашим Public Key: {decimal_to_hex(Public_key[0]), decimal_to_hex(Public_key[1])}')
```

```
p = 160974941335832298448894538799832754593
q = 219940465870775757011659172596898530451
Шифруємо на сервері з нашим Public Key: ('4E46771DA3B5FB6F634F2F75D53E0B233095CB9D57D1B1F32B4CE894CC56AB73', '10001')
```

Шифруємо за допомогою нашого публічного ключа на сайті:



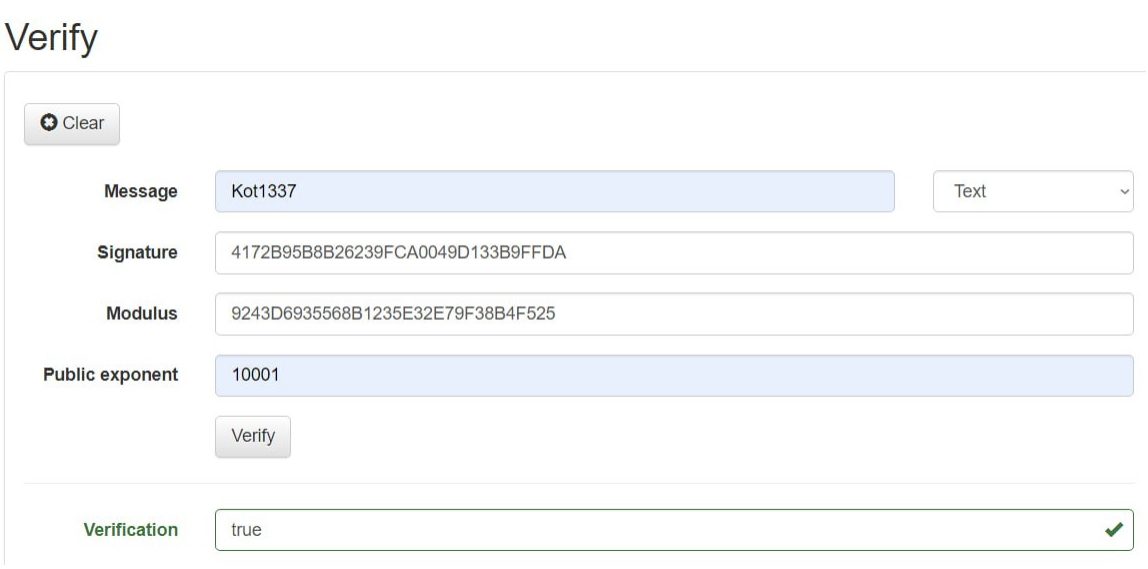
Тепер розшифровуємо:

```
encrypted_by_server_message = '2CF14C680D79F3B268C82F13562C1E9ACF94D2C017858911AE99C4439D5A5195'
print(f'Розшифровуємо локально: {Decrypt_text(hex_to_decimal(encrypted_by_server_message), Private_key)}')
```

```
Розшифровуємо локально: Kot1337
```

Все правильно

Тепер підписуємо повідомлення на сервері та перевіряємо його на сервері і у нас:



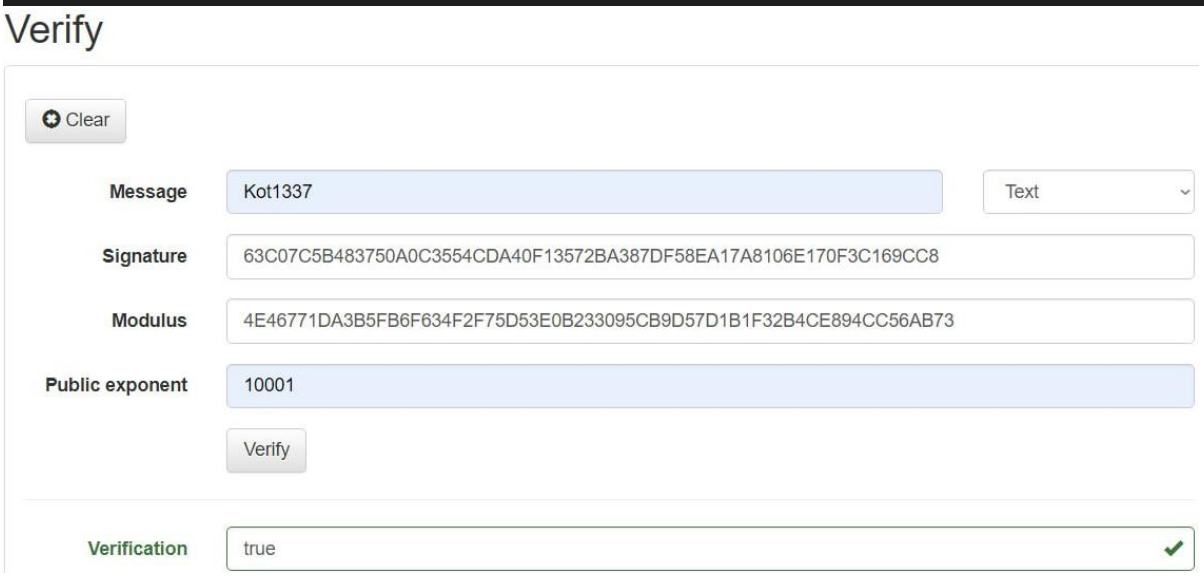

```
signed_by_server = (message, hex_to_decimal('4172B95B8B26239FCA0049D133B9FFDA'))
if Verify_text(signed_by_server, Public_key_Server):
    print('Verified\n')
else:
    print('Not verified\n')
```

```
Verified
```

Тепер ми підписуємо та перевіряємо на сервері:

```
signed_by_us = Sign_text(message, Private_key)
print(f'Signed by us: {decimal_to_hex(signed_by_us[1])}')
```

```
Signed by us: 63C07C5B483750A0C3554CDA40F13572BA387DF58EA17A8106E170F3C169CC8
```



Висновки

У ході виконання лабораторної роботи, ми набули практичних навичків захисту інформації на основі асиметричної криптосистеми RSA. Змогли створити алгоритм відправки та отримання, шифрування та дешифрування повідомлень на основі відкритого та закритого ключів. Реалізували алгоритм Міллера-Рабіна, який перевіряє числа на простоту задля створення тих самих ключів. Впевнились в правильності написаного коду на сайті https://asymcryptwebservice.appspot.com/?section=rsa.