Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

Криптографія

Комп’ютерний практикум №4

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

Виконали:

Студенти 3 курсу

Гончаров Д. К. та Сергеев А. А.

1. 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.

2. Постановка задачі

Необхідно реалізувати повноцінну криптосистему RSA на мові Python, яка включає:

1. Тест пробних ділень та тест Міллера-Рабіна для перевірки простоти
2. Генерацію великих простих чисел довжиною 256 біт
3. Створення ключових пар для двох абонентів (A та B)
4. Шифрування та розшифрування повідомлень
5. Створення та перевірку цифрового підпису з використанням геш-функції SHA-256
6. Реалізацію протоколу конфіденційного розсилання ключів з автентифікацією

3. Хід виконання роботи

Було реалізовано скрипти для виконання роботи:

- crypto_utils.py: реалізація функцій для генерації випадкових простих чисел, тест Міллера-Рабіна, тест пробних ділень, піднесення до степеня за модулем за схемою Горнера, алгоритм Евкліда для знаходження оберненого елемента за модулем, геш-фукнція
- rsa_core.py: процедури для алгоритму RSA: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey()
- run_lab.py: повноціння криптосистема RSA, в якій демонструються створення ключів, шифрування/розшифрування, цифровий підпис та протокол розсилання ключів.

4. Демонстрація роботи

Спочатку скрипт генерує прості числа p, q для абонента A та p1, q1 для абонента B. У виводі зазначені кандидати на числа, що не пройшли тест на просте число

A:

```
--- Генерація ключів (біт: 256) ---
Генерація ключів для Абонента A...
Число 648949938870860376533175224476053513716980015982859722617767668759170923218 53 не пройшло тест Міллера-Рабіна
Число 676354990100586428394695982694406743016701538912004771978068888202145676271 17 не пройшло тест Міллера-Рабіна
Число 597957477513534636859497619927345436522101562770268663769596454207254060458 51 не пройшло тест Міллера-Рабіна
Число 993364765421191945838927030134790243319611625210685049116766876218179276329 71 не пройшло тест Міллера-Рабіна
Число 711020119790713038828506140369752644574143166353955977215397421368837208237 13 не пройшло тест Міллера-Рабіна
Число 660191995345211070687227757178438333943418328417721743204900128454702921475 3 не пройшло тест Міллера-Рабіна
Число 783067848270975655839716729459678406343142906151243702091616464456823899012 81 не пройшло тест Міллера-Рабіна
Число 786078943895346512157308105675486293114933534354571709809165132191172949227 81 не пройшло тест Міллера-Рабіна
Число 103496691386788951993655598565371750065627661351573916379643479084587605589 103 не пройшло тест Міллера-Рабіна
```

```
p = 759208457987465100279136724890416022101270663533110628465572048865550414355 93
q = 101050470327928556724056001310785472271726081472265852704605116137509253791 779
```

B:

```
Число 937620010711668309232052534830866003353160562229443566487942354395815648133 29 не пройшло тест Міллера-Рабіна
Число 104366055241776925898911688364980854209967269354602070181721371355286491698 963 не пройшло тест Міллера-Рабіна
p = 74051244307940331949390168320366399966981700516322957043480190521429639828287
q = 10757798759439220484384132758443451388255672832524340401847858094135396113665 3
```

У разі якщо pq > p1q1, то ключі перегенеруються

```
n_B < n_A, перегенерація ключів B...
```

Далі програма виводить параметри криптосистеми RSA для A, B

```
Параметри RSA Абонента A:
  p_A = 659804194874479237203526929978218526502959524091861307286883966029643695323 63
  q_A = 102774468714093200628712873681826914468448450746719068692090217358255877100593
  n_A = 678110255835546196868061562072682313090905138280238759588535749168388465029973049316809254697737581182371847718737363323153958371507007882178127171991259
  e_A = 27480682416113932945694150284851965141535402297963727911721087190635010584193103791177644847709905549294376216929158372972771450448153315155531874783540 35
  d_A = 38503626802076943142367617506974045210860977456986307233461298924675800520053022541226667222089804082476163063185812545572037072659093498595500232621382 67
Параметри RSA Абонента B:
  p_B = 11480311961915478491005268370924723083584381423239333315333406340363509179458 9
  q_B = 10953777860996531589939538539740290232034955046391559205128226421198207985482 7
  n_B = 12575278700576342501848925262559135684569167642978077425160360325157477058296067953699032780039672456611539632139805162886528010047827202607991784024131103
  e_B = 10583650184770579122179045373480221368347043313859490017587051880903835603209072965671578508340821280272137504023614098024760167781943795195584885452799981
  d_B = 70136409787670563523296804145986943425776865243826436326077001945744174128256554406270171232724405891264857301476284356066350178852196660266864612701932053
```

Перевірка зашифрування/розшифрування

```
--- Тест шифрування/розшифрування ---
Відкритий текст M = 571083084912975623310241290891520865400514036976189036110944306953569888775662741093627352443432832516788496153236921953005801986492426307094999587117853 69
Шифротекст для B: C = 114215645390280218348895215003880459796713922814789339223409687049693320859333556674565501111076950166056188518515343070311650919317206230890375872825238 1
B розшифровує C: M' = 571083084912975623310241290891520865400514036976189036110944306953569888775662741093627352443432835167884961532369219530058019864924263070949995871178536 9
УСПІХ: M == M'
----------------------------------------
```

Перевірка цифрового підпису

```
--- Тест Цифрового Підпису ---
Відкритий текст: 'aboba'
Підпис Абонента A: S_A = 1489077682808311027269632327245749055910861860032193423560349427729852063330758129574604713595346277438122212847249465252665771988780071487734659059150513
Абонент B проверил підпис: True
----------------------------------------
```

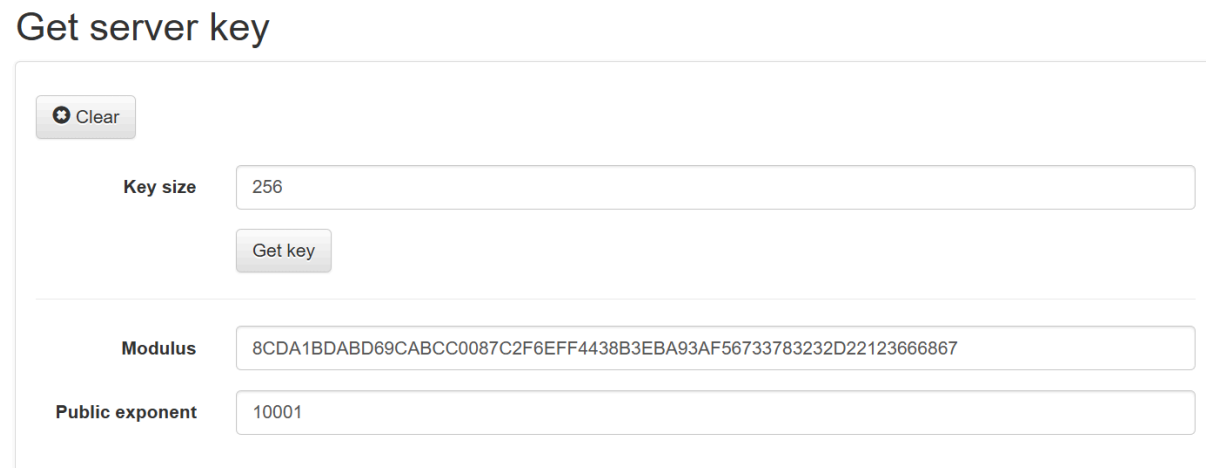
Перевірка розсилки ключів

```
--- Тест протокола розсилки ключів ---
Абонент A генерує секретне значення k = 523041136818057275863148794231646725681923591980932932791403700951431208501448775797479638500574669985737786897345819233256582055020594353191166004698369 8
A відправляє пару (k1, S1): (860238298735413116800218757035733027520222641020521491577460091498721274403257471921767398021789083484578543914262343638814951792819901380168595995888640, 533471225577257648831235142315606537256201159005829907212980826859047817263440489000885070673078888853652327176411464802626145022496460061161765631080378 94)
Абонент B отримав значення k' = 523041136818057275863148794231646725681923591980932932791403700951431208501448775797479638500574669985737786897345819233256582055020594353191166004698369 8
Статус автентифікації: True
УСПІХ: Протокол виконаний, ключі співпадають і автентифікація пройдена.
```

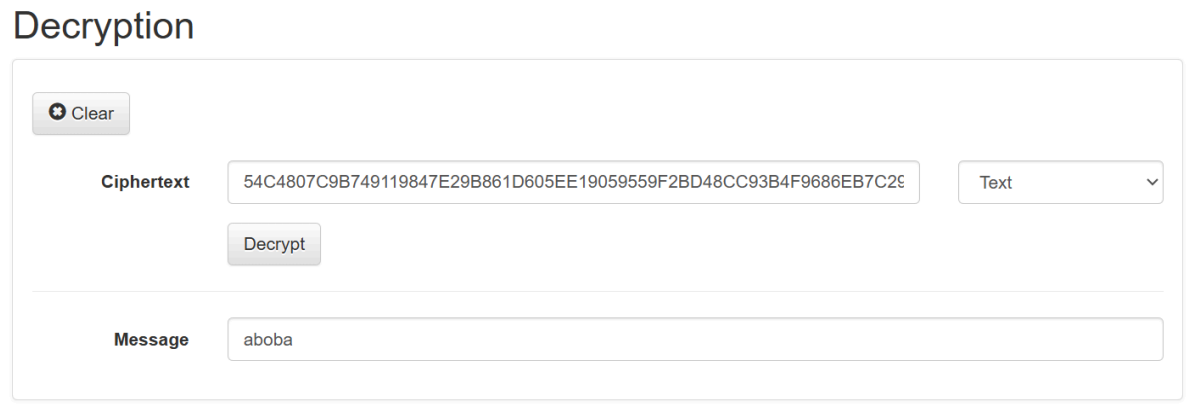
6. Перевірка із стороннім сервісом

1. Зашифрування

Сервер генерує ключі. Ми використаємо публічний ключ для зашифрування повідомлення, щоб сервер його розшифрував

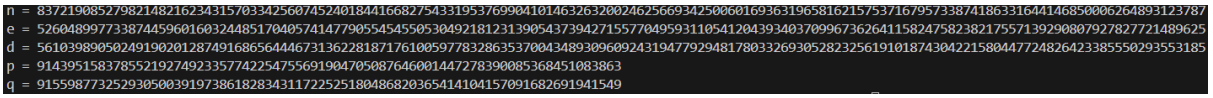



```
--- Зашифрування повідомлення для сервера ---
n = 637091637061462623464519727833931220661883768748003325332940853783410103030 79
e = 65537
M = aboba
Mbytes = 418263294561
C = 383414680886280738777830363993274429392895810413036862546861988489143532755 43
Cbytes = 54C4807C9B749119847E29B861D605EE19059559F2BD48CC93B4F9686EB7C297
```

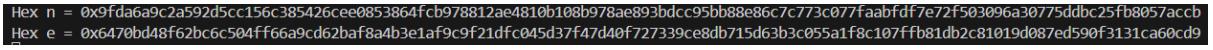


2. Розшифрування

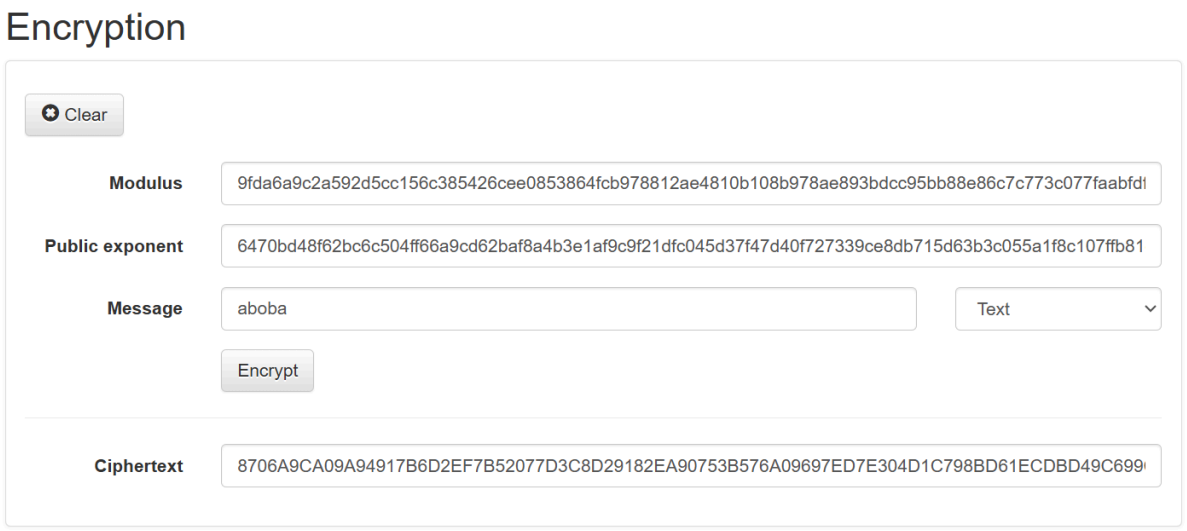
Згенеруємо ключі для нашої системи



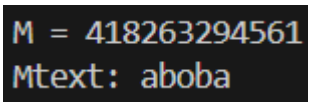
Перетворимо у hex формат



Зашифруємо повідомлення на сервері

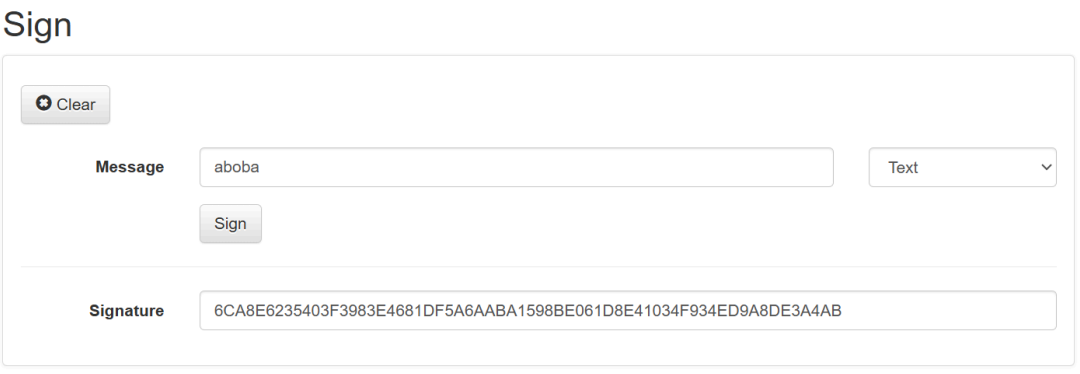


Після застосування decrypt, все правильно розшифрувало

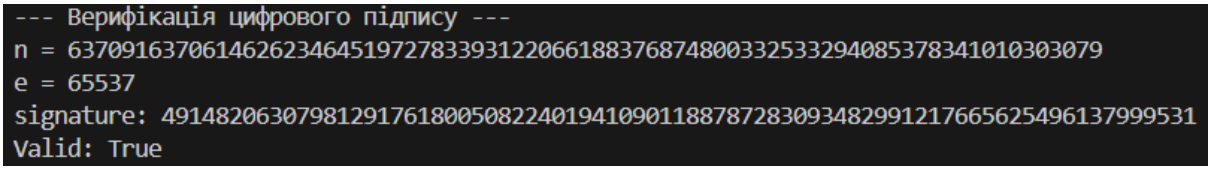


3. Перевірка цифрового підпису

На сервері зробимо підпис(публічні ключі з 1го пункту)



Перевірка підпису



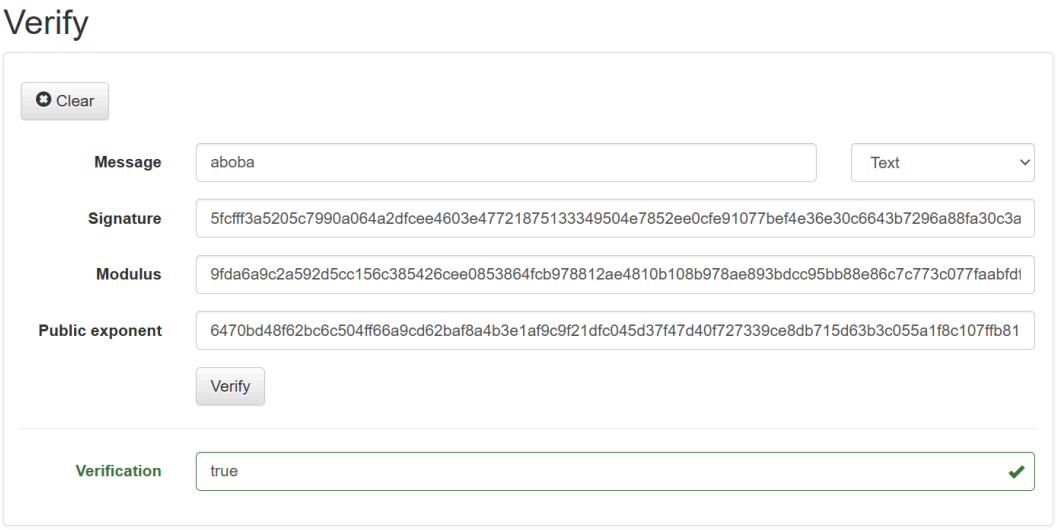
4. Підписування

Для правильності виконання скрипта було прибрано гешування повідомлень

Підпишемо повідомлення секретним ключем з 2го пункту

```
--- Підписування ---
n = 0x9fda6a9c2a592d5cc156c385426cee0853864fcb978812ae4810b108b978ae893bdcc95bb88e86c7c773c077faabfdf7e72f503096a30775ddbc25fb8057accb
e = 0x6470bd48f62bc6c504ff66a9cd62baf8a4b3e1af9c9f21dfc045d37f47d40f727339ce8db715d63b3c055a1f8c107ffb81db2c81019d087ed590f3131ca60cd9
d = 0x6b1f0f00cc26cfde9fd2b29d4cc82ebb87e6c7fba60fca2273dafefa99c5b7c84386c877f4783d76e1a1dd37377c53915ed73c59865799477a8e9350e46fd021
p = 0xca28ee919dd48e85ea578a3cd4bd44376defa9ab8cb68fc0a1f7ffefbb92ae57
q = 0xca6d0dd94fc6d1cbd1801917f09a0de2a62507454df851a3dcee5d7acc6584ad
message = aboba
signature = 0x5fcfff3a5205c7990a064a2dfcee4603e47721875133349504e7852ee0cfe91077bef4e36e30c6643b7296a88fa30c3a30cabd3ef126bf446a8eb0082bef1b1
```

Верифікація підпису

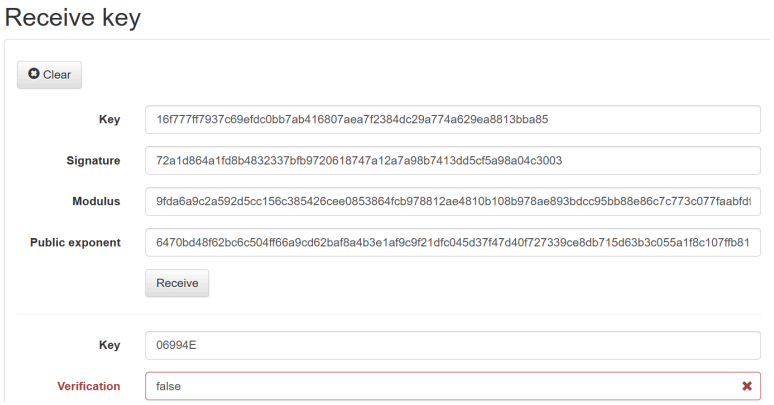


5. Надсилання ключа(протокол передачі ключів)

Генерується випадковий k_secret та надсилаються ключі(k1, s1 отримано)

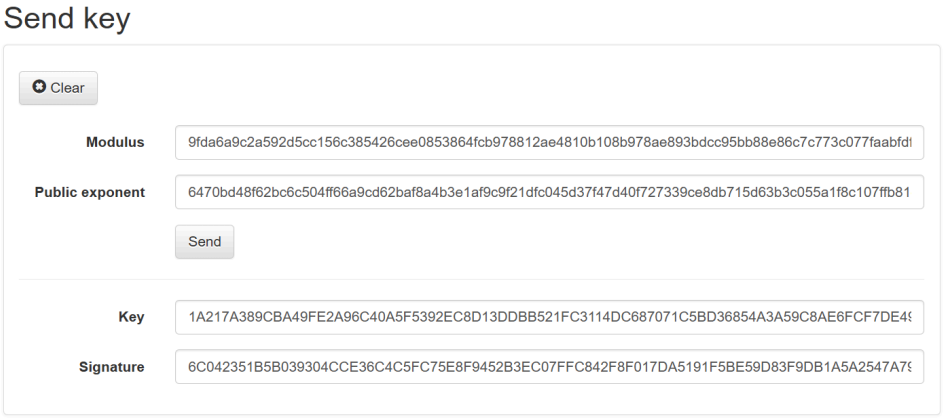
```
--- Надсилання ключа ---
k_secret = 0x6994e
k1 = 0x16f777ff7937c69efdc0bb7ab416807aea7f2384dc29a774a629ea8813bba85
S1 = 0x72a1d864a1fd8b4832337bfb9720618747a12a7a98b7413dd5cf5a98a04c3003
```

Ключ отримано вірний, але не пройшло верифікацію



6. Отримання ключа

Сервер генерує k, s з відкритих ключів скрипта



Після отримання автентифікація пройшла успішно

```
--- Отримання ключа ---
k = 0x1a217a389cba49fe2a96c40a5f5392ec8d13ddbb521fc3114dc687071c5bd36854a3a59c8ae6fcf7de4904c3217ff4f0c6384419e02e13222bb37b18a611d689
s = 0x6c042351b5b039304cce36c4c5fc75e8f9452b3ec07ffc842f8f017da5191f5be59d83f9db1a5a2547a7946c68ce35b28ec818058d4aa194f72f30547c8be5b9
Абонент B отримав значення k' = 13758389945329525239
Статус автентифікації: True
```

5. Висновки

1. **Щодо криптосистеми RSA:**
    - Шифрування та розшифрування працюють коректно
    - Схема Горнера ефективно реалізує піднесення до степеня за модулем
2. **Щодо цифрового підпису:**
    - Використання SHA-256 дозволяє підписувати повідомлення будь-якої довжини
    - Підпис забезпечує автентифікацію та цілісність повідомлення
3. **Щодо протоколу розсилання ключів:**
    - Протокол одночасно забезпечує конфіденційність (шифрування) та автентифікацію (підпис)
    - Умова $n_B \geq n_A$ є критичною для коректної роботи протоколу
    - Шифрування підпису S_1 необхідне для повної конфіденційності

Було ознайомлено з алгоритмом RSA та способами його використання. Асиметрична криптографія RSA є ефективним засобом захисту інформації. Реалізація високорівневих функцій дозволяє побудувати повноцінну криптосистему. Математичні основи RSA забезпечують високий рівень безпеки. Практична робота дозволила глибоко зрозуміти принципи роботи асиметричної криптографії.