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 групи ФБ-11
Комар Анастасія та Сергеєв Максим

Київ - 2023

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.

На наступному зображенні наведено лише перші 5 чисел, які не пройшли перевірку (з рештою можна ознайомитись у файлі not_prime.txt:

```
1   718749971682567270257929947433303634522161362340900629083033358260253162688 28
2   802304435280027654831268157686133769027461287450516150058175184599136673258 97
3   657556947779180902617898083879528965910176209562192517508751696126774146618 42
4   899794861583454318237525129764714371120563068939034111073867655136542930781 25
5   106128478343114239808285222598653013291787848086835922167574476173002070215751
```

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 $pq \leq p_1q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 – абонента B.

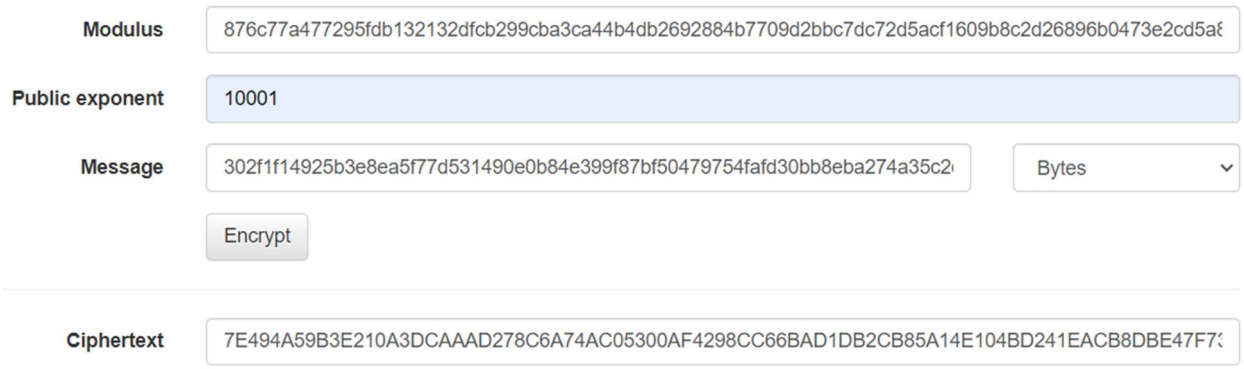
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (n, e) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n), (e_1, n_1) та секретні d і d_1.

```
Користувач A
----- p = 864886003251590266776733968200658749998429482958137577196239496099591151802 09
----- q = 590218941659968247528056274154339991712455578107863609639606884300865476459 73
----- публічний ключ: (5104721014956734637003682149222970797280566996676986794703022913143762895734772114902647538179590609217564682528852346708278611893465884217491058628148357, 65537)
----- секретний ключ: 463270353855961832255165789079961905656127505472263215570672587706259953537098717595226997195068404204508353749412510786411622052167061098443639315407276 9

Користувач B
----- p = 603449328790389409428130424882010548084158996030338975199280333311471960753 79
----- q = 871476113243562108026576594589542033498484789193286490292879486381084149990 73
----- публічний ключ: (525891675593684944892202618400996991697233728332088929791734122806112562514573830315318032864705699154301627991110701400487993197703836474834076352312366 7, 65537)
----- секретний ключ: 105985584497938427943546579548047183519798939283309132012286560172469516848380248449766341665938438694582969922968929131734639634157652435885430942066841 7
```

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.

```
Повідомлення: 26488443401651187439477503267549521455137107384402688205335574090502366411674366147502012894949997875415531927410672498918356807503058247633263715665930 99

Зашифроване повідомлення за допомогою відкритого ключа A: 28260445307008544467066466158409995759583896451959061240081175348252258664160912144280425388371734147246476579768393455480654974772953211375823044888999809

Розшифроване повідомлення за допомогою секретного ключа A: 26488443401651187439477503267549521455137107384402688205335574090502366411674366147502012894949997875415531927410672498918356807503058247633263715665930 99

Повідомлення з цифровим підписом: (26488443401651187439477503267549521455137107384402688205335574090502366411674366147502012894949997875415531927410672498918356807503058247633263715665930 99, 3846531178309155682317751650181733277112528647007264445106900715972147181959579082644296540739750542958098576289046192803297653374906408315716608397223 17)

Цифровий підпис підтверджено, повідомлення не спотворено.
```

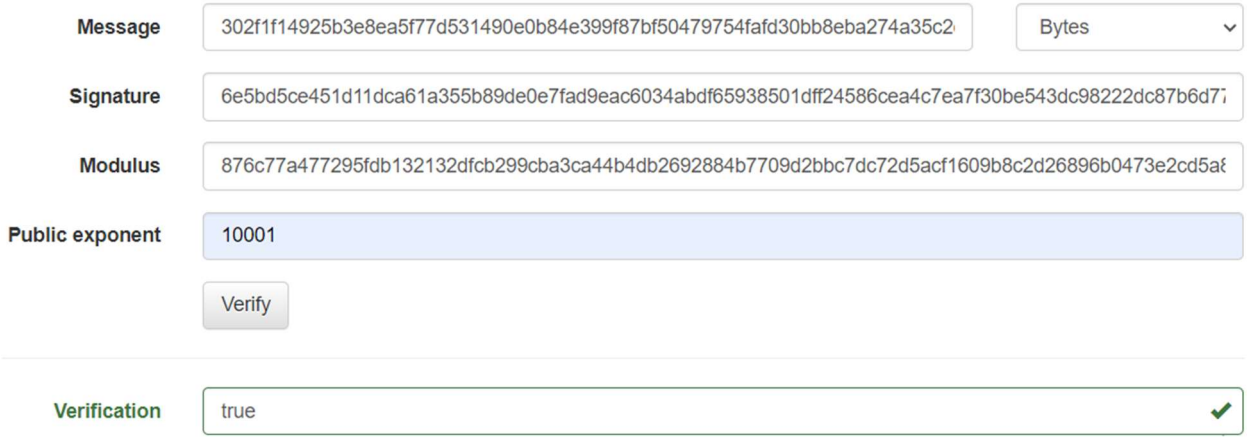
Перевіримо, чи правильно працює наша функція шифрування:



Переведемо наш ШТ з hex і отримаємо такий самий ШТ, як і в реалізованій нами функції:

```
In [16]: int("0x7E494A59B3E210A3DCAAAD278C6A74AC05300AF4298CC66BAD1DB2CB85A14E104BD241EACB8DBE47
    ... : F739ABC83A9292BD9EB0773876940889A125E2F3F393638A", 16)
Out[16]: 661414972860085984410318114426996342192872083702219414911342913152360114891021453476894
61468074583794122421879605349263127761135583457975761265070369268 58
```

Також перевіримо чи правильно працює функція цифрового підпису:



Усе працює.

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.

Абонент A формує повідомлення (k_1, S_1) і відправляє його B, де

$$k_1 = k^{e_1} \bmod n_1, \quad S_1 = S^{e_1} \bmod n_1, \quad S = k^d \bmod n$$

```
k = 419339466159155862743644832601325713723196182076956436424414811083359474535277773015649482985973530135290449971216177520441933994409760775950144667955382 9
S = 278408679467332685532949257150967486197198145095588028226262059262757563923515970339284618424040765738263049234440535685663599964515169733830099163772040 9
S1 = 134288939950689556390570135817424910346531126148574776825553539728041587314097850874861669481349060072424261985902186050171217514191928708259256369663 64
```

Абонент B за допомогою свого секретного ключа d_1 знаходить (конфіденційність):

$$k = k_1^{d_1} \bmod n_1, \quad S = S_1^{d_1} \bmod n_1,$$

і за допомогою відкритого ключа e абонента A перевіряє підпис A (автентифікація):

$$k = S^e \bmod n.$$

```
k1 = 257236334814514052201603625047507758005540518314452779871982333842651136333139347118543473935755234692214353448324557719859518199138194194446071824772989 1
k = 419339466159155862743644832601325713723196182076956436424414811083359474535277773015649482985973530135290449971216177520441933994409760775950144667955382 9
S = 278408679467332685532949257150967486197198145095588028226262059262757563923515970339284618424040765738263049234440535685663599964515169733830099163772040 9
Автентифікація пройдена.
```

Висновки

Під час лабораторної роботи ми ознайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA. Крім того, ми на практиці ознайомились з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи засекречений зв'язок й електронний підпис, вивчивши протокол розсилання ключів.