

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”
Фізико-технічний інститут

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4
з предмету «Криптографія»

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

Виконав:
Студент 3 курсу,
ФТІ, групи ФБ-05
Савченко Ярослав

Мета та основні завдання 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 і рекомендації щодо 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 $1 < p, q$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $p \cdot q \leq p_1 q_1$; p і q – прості числа для побудови ключів абонента А, $1 < p$ 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) , (d, n) і 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 А і В, перевірити правильність розшифрування. Скласти для А і В 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 $0 < k < n$.

Хід роботи

Створимо функцію що буде брати випадкове число заданої довжини і якщо воно не виявиться простим то візьме інше випадкове число цієї довжини. Звісно створимо функцію для визначення простоти числа використовуючи метод Міллера-Рабіна описаний в методичці.

Для наступного завдання створив функцію що використовуючи функції минулого завдання складає дві пари чисел.

```
p = 96461904010644145815056134002109100188233814667859705403760289267209968487171
q = 67836484867032579895555977474809008008402137806270827977469114918298828181999
p1 = 104599226489937365286746720599879809914658043025933087234726124778379279925851
q1 = 83855362685772713535351725362007282424132200413075315847016749708743567646243
=====
```

Власне сгенерував їх:

p = 96461904010644145815056134002109100188233814667859705403760289267209968487171
q = 67836484867032579895555977474809008008402137806270827977469114918298828181999
p1 = 104599226489937365286746720599879809914658043025933087234726124778379279925851
q1 = 83855362685772713535351725362007282424132200413075315847016749708743567646243

Звісно вони виконують умову $pq \leq p_1q_1$

Тепер створимо функцію що генерує наші RSA ключі

```
Dlya A
e = 12867031871303065474115840229373429285793761756677916527330578440725512439678575746402801089781481121786073850184428392177004863124181732
69599624675595571
n = 65436364916632109232629943259497241934204766546575705222017192427811307976949065033674333115079358090934043785332237540598771603388195145
23009390284634829
d = 16748325953525930783578700848258498866260052821556007000365608524601661619214889659649356646429527794967477032693792174948183888302402656
65261956441317191
p = 96461904010644145815056134002109100188233814667859705403760289267209968487171
q = 67836484867032579895555977474809008008402137806270827977469114918298828181999
Dlya B
e = 37299874432202442142474633988152332888211371238391204874570727007879540348855603368910959244835828978296009590185724259466962423894922721
46867698549240933
n = 8771206073964982507149570623899445520020906584868332149226622176972242559689799879472089066782191573537726471161173114652776771670054670
92294247638727793
d = 51704915992827472781831669007966865680938575726495856387983615132097431262611100044452322153695902220960308618969209555789020511018264461
17727789239499397
p = 104599226489937365286746720599879809914658043025933087234726124778379279925851
q = 83855362685772713535351725362007282424132200413075315847016749708743567646243
```

Для А

e = 128670318713030654741158402293734292857937617566779165273305784407255124396785757464 0280108978148112178607385018442839217700486312418173269599624675595571
n = 654363649166321092326299432594972419342047665465757052220171924278113079769490650336 7433311507935809093404378533223754059877160338819514523009390284634829
d = 167483259535259307835787008482584988662600528215560070003656085246016616192148896596 4935664642952779496747703269379217494818388830240265665261956441317191
p = 96461904010644145815056134002109100188233814667859705403760289267209968487171
q = 67836484867032579895555977474809008008402137806270827977469114918298828181999

І для В

e = 372998744322024421424746339881523328882113712383912048745707270078795403488556033689 1095924483582897829600959018572425946696242389492272146867698549240933
n = 877120607396498250714957062389944552002090658486833214922662217697224255968979987947 208906678219157353772647116117311465277677167005467092294247638727793
d = 517049159928274727818316690079668656809385757264958563879836151320974312626111000444 5232215369590222096030861896920955578902051101826446117727789239499397
p = 104599226489937365286746720599879809914658043025933087234726124778379279925851
q = 83855362685772713535351725362007282424132200413075315847016749708743567646243

Для наступного завдання реалізуємо необхідні функції та перевіримо їх користуючись запропонованим сервісом

Шифрування:

На цьому етапі зіткнувся з проблемою що на сайті усі ключі та тексти перевіряються з 16 системою HEX чисел, а я маю всі дані в 10 системі. Тому переведу свої значення в HEX значення і зашифрую повідомлення на своєму комп'ютері

```
_fb-05_cp4\lab4.py'
=====
m=29b409230426f1f97b6d145562f02a16fb651a4614344f312b946371d2cd9fbabec205b1dbec1acfb099a11b23a80dcb3d11c91b03cd51c094d77998b85c7aa
e=1891462fea5d78ec7e87c7f941e8eea722cc84de5498c090390e53f9d2dfff6f345fc0e767d996ab59e7a6046657fd54b729ea13aaa9d5b9b6063f996c321533
n=7cf0a100283ac99a1dc5ab1b95a120b79c4bfc2c773b2a02102f4e0265cedd402e912b996ed748fe4a362a695e5782322dbdb0f1732c7d330a683d1cc3facd
=====
197246175300151594361217816860036863971159303827854108240672060143748779559760152175206985567420497891651870589450140040346744369255375249705
5065837243517
=====
c=25a931350d7fb04aec63395caf2fdea0e2ee68faaf880038aab903251a0e0b4eb782b2ac945f18d49240fb6e6f42a9e6d2f12469f38d2d3e790ccd69f2698c7d
PS D:\savchenko_fb-05_cp3>
```

Encryption

Clear

Modulus: 7cf0a100283ac99a1dc5ab1b95a120b79c4bfc2c773b2a02102f4e0265cedd402e912b996ed748fe4a362a695e578

Public exponent: 1891462fea5d78ec7e87c7f941e8eea722cc84de5498c090390e53f9d2dfff6f345fc0e767d996ab59e7a6046657fd54b729ea13aaa9d5b9b6063f996c321533

Message: 29b409230426f1f97b6d145562f02a16fb651a4614344f312b946371d2cd9fbabec20 Bytes

Encrypt

Ciphertext: 25A931350D7FB04AEC63395CAF2FDEA0E2EE68FAAF880038AAB903251A0E0B4EB782B2AC945F18D49240F

Працює відмінно. Перевіримо наступні функції. Дешифрування. Передамо в неї наш зашифрований текст і секретні ключі.

```
_fb-05_cp4\lab4.py'
=====
218417718493919229324617355689891458511508677668911428523128425707452336411473324519702322458180419988665622094603597085100428699340326607768
7782759057322
=====
29b409230426f1f97b6d145562f02a16fb651a4614344f312b946371d2cd9fbabec205b1dbec1acfb099a11b23a80dcb3d11c91b03cd51c094d77998b85c7aa
PS D:\savchenko_fb-05_cp3>
```

Отримали початковий текст 😊

Підпис:

```
_fb-05_cp4\lab4.py'
=====
483992051946587938883691653758661764706961394882861685883863818172476169342058204407346466315583068760820243169454336912446106960693707636043
681269243958
PS D:\savchenko_fb-05_cp3>
```

Верифікація:

```
_fb-05_cp4\lab4.py'
=====
True
PS D:\savchenko_fb-05_cp3>
```

Спробуємо ввести ключ для користувача В

```
Kotik\.vscode\extensions\ms-python.python-2022.20
_fb-05_cp4\lab4.py'
=====
False
PS D:\savchenko_fb-05_cp3> 
```

Працює

І останнє завдання. Передача ключів.

Згенеруємо якесь випадкове k і запишемо для подальшої роботи з ним

```
_fb-05_cp4\lab4.py
=====
248165035553309697217797289561899736198490564387108779961421701210111192705978978821771933585647555346141850631499216248795099936575266065514
2630791743192
PS D:\savchenko_fb-05_cp3> 
```

Запишемо функції що використовують метод з методички, та виведемо повідомлення чи є підпис A істинним

```
Kotik\.vscode\extensions\ms-python.python-2022.20
_fb-05_cp4\lab4.py'
=====
True
PS D:\savchenko_fb-05_cp3> 
```



Висновки: на цій лабораторній роботі я отримав дуже корисні навички роботи з генеруванням простих чисел, ключів для RSA і реалізацією простої системи шифрування на основі RSA з підписами і перевіркою їх. А також з протоколом розсилання їх. В цій роботі в мене майже не виникало проблем і на мою думку це була найлегша з робіт.