



Міністерство освіти і науки України
Національний технічний університет України
Київський політехнічний інститут імені Ігоря Сікорського
Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №4

з дисципліни

Криптографія

Виконали:

студенти 4 курсу ФТІ

групи ФБ-72 та ФБ-73

Морозов Артур

Сницін Максим

Перевірили:

Чорний О.

Криптоаналіз афінної біграмної підстановки

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 $pq \leq p_1 q_1$; p і q – прості числа для побудови ключів абонента А, p_1, 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 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 $0 \leq k \leq n$. 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. Відповідно, програмний код повинен містити сім високорівневих процедур: `GenerateKeyPair()`, `Encrypt()`, `Decrypt()`, `Sign()`, `Verify()`, `SendKey()`, `ReceiveKey()`.

Хід роботи:

Абонент А:

$q = 300444781422354751194320408009007945479$

$p = 327529937742101032799769513941198783731$

публічний ключ (exponent):

10001

modulus:

d98f1069d6ee7e2ef4e9e61e449f26cbc3b329261da5bc62781f54235851f7a5

секретний ключ (private exponent):

5a6fd127de13b7da772ebfdcf7339d61346e218696b45117764d4ac21ac808e5

Абонент В:

$q = 230413693420247645171238366177493326707$

$p = 265841159451787212900698952933303292463$

публічний ключ (exponent):

10001

modulus:

876c386a8f93d5c1f1d3db50256931df0af9d1967ad72e0973bc45bccfd1a01d

секретний ключ (private exponent):

4fc93ee3e97b9c310c1d59d53a7563dfbd4d74f25fe73ff97e0adab8483076b1

1. Перевірка зашифрованого повідомлення:

The screenshot shows a web application titled "RSA Testing Environment" with tabs for "asymcryptwebservice.appspot.com/?section=onment", "RSA", "Rabin", and "Zero Knowledge". The "RSA" tab is active. Below the tabs, there's a "g Environment" section and an "Encryption" section. The "Encryption" section has a "Clear" button and input fields for "Modulus" (d37e6bbf50089f183b1e4...), "Public exponent" (10001), and "Message" (c44042a230bdbab4ad52393ac8fbdebad46ff41089630c426953596). A "Bytes" dropdown menu is set to "Bytes". An "Encrypt" button is present. The "Ciphertext" field shows the result: 53C0209A0B39D1D969651729726754A1.

Overlaid on the right is a terminal window titled "server@server: ~/Рабочий стол/4". It displays the following output:

```
2 - 301612489405901163307784329487587839671
3 - 190420527949449594666151964881127023891
4 - 18002230825739343517757396447672209723

first
module:      d37e6bbf50089f183b1e4d64fb5b54830a3828e06d205f2140fe09e6ae2749b5
public exp:  10001
priv exp:    a8357844e4c6b6a273029ad3a206f5a3011203a9583e859478658ac149921b51

second:
module:      4bc9c299815b59ecaddc8cc88a8f7bd36634396b69f22785a0620a7f07b4ce61
public exp:  10001
priv exp:    49f73256109a704bfe6ee6737e4f84bfffcd5566f16edcefb46291834d215cf5

start: 53c0209a0b39d1d969651729726754a1
after encoding a753492b4b4f1807519b8f2db0dcd156b8f2d44063cc1e426027bce77771c214
after decoding 53c0209a0b39d1d969651729726754a1

all fine
signed msg is c44042a230bdbab4ad52393ac8fbdebad46ff41089630c426953596a52601afb
```

At the bottom right of the terminal, there's a snippet of code: `ule1, data->pub1, data-module2, data->pub2, da`.

2. Відправка та отримання ключа:

RSA Testing Environment

[Server Key](#)
[Encryption](#)
[Decryption](#)
[Signature](#)
[Verification](#)
[Send Key](#)
[Receive Key](#)

Send key

✖ Clear

Modulus

d98f1069d6ee7e2ef4e9e61e449f26cbc3b329261da5bc62781f54235851f7a5

Public exponent

10001

Send

Key

04D1F84CD6AFCD6A7F87A2D7EE211796AA5EC0F68E0DFD4B0C26BE0E8DD52A40

Signature

C3332603980A9C0C476545C7DA5B62A1BBB150ABF5A36E2D3B19ED2EB999602A

```
server@server: ~/Рабочий стол/4
server@server:~/Рабочий стол/4$ ./work_site
Enter modulus from site
A8D1A169235012311C7D966D315A1E1F7AE5E1CFCB793945E6B7F322E91B1D6D

Enter Key from site
04D1F84CD6AFCD6A7F87A2D7EE211796AA5EC0F68E0DFD4B0C26BE0E8DD52A40

Enter Signature from site
C3332603980A9C0C476545C7DA5B62A1BBB150ABF5A36E2D3B19ED2EB999602A

Enter your modulus
d98f1069d6ee7e2ef4e9e61e449f26cbc3b329261da5bc62781f54235851f7a5

Enter your private exponent
5a6fd127de13b7da772ebfdcf7339d61346e218696b45117764d4ac21ac808e5

Verificated | Key is 18370b261ca38ffe

Enter your Key
18370b261ca38ffe

Key:          90c1796ab591f56597ed511ba5e4a1527623f3daf92ed28411f7cfb531b9c7b6
Signature:    227145f7a5733b697aaabdfcab3e23b5e1a69e0d5c6bf5ef8c3c3681c65beab8
```

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Receive key

Clear

Key

90c1796ab591f56597ed511ba5e4a1527623f3daf92ed28411f7cfb531b9c7b6

Signature

227145f7a5733b697aaabdfcab3e23b5e1a69e0d5c6bf5ef8c3c3681c65beab8

Modulus

d98f1069d6ee7e2ef4e9e61e449f26cbc3b329261da5bc62781f54235851f7a5

Public exponent

10001

Receive

Key

18370B261CA38FFE

Verification

true

✓

Enter your Key
18370b261ca38ffe

Висновок: Під час виконання лабораторної роботи №4 ми ознайомились з асиметричною криптосистемою RSA, генерацією ключів для цієї криптосистеми, а також з тестами перевірки чисел на простоту. У ході роботи реалізували функції шифрування, розшифрування, підпису, перевірки підпису для RSA. RSA алгоритм з відкритим ключем, що часто використовується в криптографічних застосунках. Складність задачі цього алгоритму полягає у великих цілих простих числах.