МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4

Виконав студент групи ФБ-14
Шовкун Богдан

2023

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

Мета: 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.

Хід 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.

```
def prime_number(bits):
    def random_number(bits):...

    def miller_rabin(p):...

    num = random_number(bits)
    while not miller_rabin(num):
        num = random_number(bits)

    return num
```

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, 1 p і q1 – абонента B.

```python
def get_pair(bits):
    pair = (prime_number(bits), prime_number(bits))
    while pair[0] == pair[1]:
        pair = (prime_number(bits), prime_number(bits))

    return pair
```

```python
A_pair = get_pair(256)
B_pair = get_pair(256)

while A_pair[0] * A_pair[1] > B_pair[0] * B_pair[1]:
    A_pair = get_pair(256)
    B_pair = get_pair(256)
```

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) ,(e1, n1) та секретні d і d1 .

```python
def generate_keys(pair):
    n = pair[0] * pair[1]
    fi = (pair[0] - 1) * (pair[1] - 1)
    e = 2**16 + 1
    d = mod_inverse(e, fi)
    open_key = (n, e)
    secret_key = (d, pair[0], pair[1])
    return open_key, secret_key
```

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.

```
4 usages
def encrypt(message, key):
    encrypted_message = pow(message, key[0][1], key[0][0])
    return encrypted_message

3 usages
def sign(message, key):
    signed_message = (message, pow(message, key[1][0], key[0][0]))
    return signed_message

4 usages
def decrypt(encrypted, key):
    decrypted_message = pow(encrypted, key[1][0], key[0][0])
    return decrypted_message

3 usages
def verify(signed, message, key):
    if message == pow(signed, key[0][1], key[0][0]):
        return 'Verified'
    else:
        return 'Fake sign'
```

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.

```
print(f'Generated public key for A: {A_keys[0]}\nGenerated private key for A: {A_keys[1]}\n')
print(f'Generated public key for B: {B_keys[0]}\nGenerated private key for B: {B_keys[1]}\n')

print('======test======')
M = random.randint( a: 0, min(A_pair[0] * A_pair[1], B_pair[0] * B_pair[1]))
print(f'M: {M}')
# Знаходження криптограми для абонентів A і B
print('======crypto======')
encrypted_A = encrypt(M, A_keys)
encrypted_B = encrypt(M, B_keys)
print(f'encrypted A: {encrypted_A}\nencrypted A: {encrypted_B}')
print('======decrypt======')
# Перевірка правильності розшифрування
decrypted_A = decrypt(encrypted_A, A_keys)
decrypted_B = decrypt(encrypted_B, B_keys)
print(f'Check decrypt for A: {M == decrypted_A}')
print(f'Check decrypt for B: {M == decrypted_B}')

# Складання повідомлення з цифровим підписом для A і B
print('======sign======')
signed_message_A = sign(M, A_keys)
signed_message_B = sign(M, B_keys)

# Перевірка підпису
verification_A = verify(signed_message_A[1], M, A_keys)
verification_B = verify(signed_message_B[1], M, B_keys)
print(f'Check sing for A: {verification_A}')
print(f'Check sing for B: {verification_B}')
```

```
Generated public key for A: (212242384584706191878182756489411130481254017570087155849067686893004407418330952325361780200728887938571877519181115769063230189129909981019283564431392
Generated private key for A: (136610166912380201901176178601291327297415783803942604867192917530671298932318092139545758022193540660185942147846648920922990916631971100961823681120211

Generated public key for B: (307602824183154377404012711479865076730525049886350942411393373975481854414938327769507737797675611583628055962730065060283568361270989577697639838384791
Generated private key for B: (124023025561372526489214213164227150884651325483268036411167104291135275051653837513922440568006729722526562981412969998759764311995457232614412817522501

======test======
M: 152128196582453488000452047584477925619101112302089100226458730951390038207389426464659558641825678416740456552322915442867277586894919084462752280510703777
======crypto======
encrypted A: 19514252458534296130892306471239819810670883200820487133894478209455079721942291216254669308861926004375657857332867176635550964506116981850104905752968795
encrypted A: 10349550262106271902048502920561042017184703850661117237703011365747480854579128019309209436499706318126346206741050725813718724535236800946616341252114835
======decrypt======
Check decrypt for A: True
Check decrypt for B: True
======sign======
Check sing for A: Verified
Check sing for B: Verified
```

Generated public key for A:
(212242384584706191878182756489411130481254017570087155849067686893004407418330952

232536178020072888793857187751918111576906323018912990998101928356443139237,
65537)

Generated private key for A:
(136610166912380201901176178601291327297415783803942604867192917530671298932318092139545758022193540660185942147846648920922990916631971100961823681120218132,
163838454809782337700012775995493131484441660157380685003289339689842955787367,
129543692798568673011898814387881327874011166488859185934061454709322605774611)

Generated public key for B:
(307602824183154377404012711479865076730525049886350942411393373975481854414938327769507737797675611583628055962730065060283568361270989577697639838384791 59,
65537)

Generated private key for B:
(124023025561372526489214213164227150884651325483268036411167104291135275051653837513922440568006729722526562981412969998759764311995457232614412817522567 05,
185140066453561669465953834797494463957510812360768215465005488145288900315147,
166146005062772194912318419836144070395442352240869170919901971859488073281797)

======test======

M:
152128196582453488000452047584477925619101112302089100226458730951390038207389426464659558641825678416740456552322915442867277586894919084462752280510703 77

======crypto======

encrypted A:
195142524585342961308923064712398198106708832008204871338944782094550797219422912162546693088619260043756578573328671766355509645061169818501049057529687 95

encrypted A:
103495502621062719020485029205610420171847038506611172377030113657474808545791280193092094364997063181263462067410507258137187245352368009466163412521148 35

======decrypt======

Check decrypt for A: True

Check decrypt for B: True

======sign======

Check sing for A: Verified

Check sing for B: Verified

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 $0 < k < n$.

```
1 usage
def send_key(msg, B_keys, A_keys):
    encrypted = encrypt(msg, B_keys)
    signed = sign(msg, A_keys)
    s1 = encrypt(signed[1], B_keys)
    final_message = (encrypted, s1)
    return final_message

1 usage
def receive_key(final_message, B_keys, A_keys):
    decrypted = decrypt(final_message[0], B_keys)
    decrypted_sign = decrypt(final_message[1], B_keys)
    verification = verify(decrypted_sign, decrypted, A_keys)
    return decrypted, verification
```

```
print('======protocol======')

msg = random.randint( a: 0, B_pair[0] * B_pair[1])
final_message = send_key(msg, B_keys, A_keys)
decrypted, verification = receive_key(final_message, B_keys, A_keys)
print(f'Original message: {msg}')
print(f'Encrypted message: {final_message[0]}')
print(f'Signed message: {final_message}')
print(f'Decrypted message: {decrypted}')
print(f'Verification result: {verification}')
```

======protocol======

Original message:
14313778501314591941668111425323959617078517872756346154950489099337261603685210387721149553742452305583854667896943317933908504308035530431135324542506750

Encrypted message:
28268302632046612259973651425780936389895604461790513744793125543872062015911800316144937700765838143557596634263648729849223093316697246364292303519628639

Signed message:
(28268302632046612259973651425780936389895604461790513744793125543872062015911800316144937700765838143557596634263648729849223093316697246364292303519628639,

22585753692427325823116735627124674509082696077779761624144518764894577877475586
70597731245874769002157210967755575925911194140896683201764670135849502050 5)

Decrypted message:
14313778501314591941668111425323959617078517872756346154950489099337261603685210
38772114955374245230558385466789694331793390850430803553043113532454250675 0

Verification result: Verified

Висновок: В ході лабораторної роботи я провів тести перевірки чисел на простоту і методи генерації ключів для асиметричної криптосистеми типу RSA. Ознайомився з системою захисту інформації на основі криптосхеми RSA. Реалізував протокол конфіденційного розсилання ключів з підтвердженням справжності за допомогою системи RSA. Було вибране повідомлення розміром [0, n], проведена перевірка pq <= p1q1. За допомогою протоколу я зашифрував та підписав довільне повідомлення, після чого розшифрував та провів автентифікацію.