

Грипас Владислав ФБ-12

Варіант 15

Лабораторна робота №4

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 криптосистеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів

Враховуючи те, що було перебрано велику кількість чисел, що не пройшли тест перевірки простоти, надаю останні декілька строк:

```
possible q: 174225771852004410218889248298371463216603650068110970185826092912812362846135
possible q: 174225771852004410218889248298371463216603650068110970185826092912812362846137
possible q: 174225771852004410218889248298371463216603650068110970185826092912812362846139
possible prime P: 159673358919913031369037809901034429131340185731113213002897142822118643277951
possible prime q 174225771852004410218889248298371463216603650068110970185826092912812362846139
```

Параметри криптосистеми RSA:

```
a stats:
d = 0x27835d79047b7809250bfb6f316fcaeade2cc689b3b1ef9b772f86dc72e8755834e3a677bb9cb7845a16b80e96509cf2c3301308a6545fa5374eac238801ed15
modulus = 0x1a0aa3d8dfb7d332494a98f83f836e16c2ba5bd4a70ee0da6910832e9213223b55f6cf54bab8f7b655eaf3113eb105d90d2e9bd5ad21cf0fb5870af274a062ab5
exponent = 0x10001
a private is 0x27835d79047b7809250bfb6f316fcaeade2cc689b3b1ef9b772f86dc72e8755834e3a677bb9cb7845a16b80e96509cf2c3301308a6545fa5374eac238801ed15

b stats:
d = 0xf8b6015293812e767068b68c642e07a40a5114bc7cab5227a6fe8dd3f07e2e26a6c4c9602f8fe12a0ce842e337bf6d9811484aeafbd7d9241272a573aa1c70f5
modulus = 0x213297c8c06234de7d08abd878b183757e7040e0733e99976f21859845624beb08fa91cd2d7fd10c395f93f397331e111d7860eba7b5f64d521c7978f1469edc5
exponent = 0x10001
b private is 0xf8b6015293812e767068b68c642e07a40a5114bc7cab5227a6fe8dd3f07e2e26a6c4c9602f8fe12a0ce842e337bf6d9811484aeafbd7d9241272a573aa1c70f5
```

Чисельні значення ВТ, ШТ, цифрового підпису для А та В

```
cleartext: 0x68656c6cf2066726f6d20737276
True
verifying for b:
signed message: 0xdf69f4d88502119a9b52bb2e9f3402d1b5259dc1cf3016d2acc015b408300f78a0071c2fde4040a15df218f903ac7f3c2f18848372fabce6f1f7e0ca9909376
verified successfully
verifying for a:
signed message: 0xd0dd19b90b650893deb7817e443315037abdbb9322c9f1d13cdbe880c9b9ae4b00e0fa3bdf4664e48504f38880f848319146cf6e3590143e094c96f1b24403ee
verified successfully
syphertext = 0x101da46433358c3a5870ef256b0ac7f312e6481370e76415ca42e5fa2237a47fe5b18f72b5ecb06e645f49408f28c2b49bec117cc44fb9b9b3673ace0fd5ac3
decryption result: 0x68656c6cf2066726f6d20737276
b'hello from srv'
```

Чисельні значення характеристик на кожному кроці для протоколу конфіденційного розсилання ключів

```
generated k: 0x11bd3282ce186415299e90a2d28b674d2093f89c67cf422778456f94a68de217c69e2ffdec25893ce3d6c2ece1b8ffa26b926f470062e717e395bc3e97fea399e
generated S: 0x260a5c15c04eb3b49cad9a83a49c4570791d34b741c05846ad8a001c99773c3533b89e294d814bc6e62647279f23596a918d5c8dd0666e9d607828bf6278e0fb5
sending S1: 0x22e4948601a9cbb23884d765fc1ca15d1759fce4e024b4b99359528d947b9d84ad23bc47cc5a80a85a166ae86bad9d60026eb9ff81d7eeb2a315f2e0b72dc84c0
sending k1: 0x11bd3282ce186415299e90a2d28b674d2093f89c67cf422778456f94a68de217c69e2ffdec25893ce3d6c2ece1b8ffa26b926f470062e717e395bc3e97fea399e
decrypted k: 0x11bd3282ce186415299e90a2d28b674d2093f89c67cf422778456f94a68de217c69e2ffdec25893ce3d6c2ece1b8ffa26b926f470062e717e395bc3e97fea399e
decrypted S: 0x260a5c15c04eb3b49cad9a83a49c4570791d34b741c05846ad8a001c99773c3533b89e294d814bc6e62647279f23596a918d5c8dd0666e9d607828bf6278e0fb5
True
```

Код протоколу конфіденційного розсилання ключів має такий вигляд;

```
def SendKey(self):
    k = random.randint(1, self.public_key.n-1)
    print("generated k: ", hex(k))
    k1 = pow(k, self.friend_public_key.e, self.friend_public_key.n)
    S = pow(k, self.d, self.public_key.n)
    print("generated S: ", hex(S))
    S1 = pow(S, self.friend_public_key.e, self.friend_public_key.n)
    print("sending S1: ", hex(S1))
    print("sending k1: ", hex(k))
    return k1, S1

def RecieveKey(self, k, S1):
    k = pow(k, self.d, self.public_key.n)
    S = pow(S1, self.d, self.public_key.n)
    print("decrypted k: ", hex(k))
    print("decrypted S: ", hex(S))
    return k == pow(S, self.friend_public_key.e, self.friend_public_key.n)
```

Відповідно, в моєму випадку абонент А відпавляє ключ, та цифровий підпис до В, в результаті вдалої перевірки повертається значення True

Результати перевірки SendKey та RecieveKey в RSA testing env:

Receive key

Key

5c0cc6f1ed014e8fbae5517dc087e5ee94812b32811558cddda3fd056d9ee403c71a8fa6046682e918ffc48bae7b6a

Signature

36f19ff277fb33215a3f5a6ae6bc9f9f91a64bf967962f2a6458bfdc84f0d6373db5811dab256db5170b37b7f9979d40f6

Modulus

836c984134102c8503ab5fd1364c48db9eac4536f15451a178a890345ef89b079a86972b1d16ff810b1a4bf822e202f

Public exponent

10001

Key

19DAEEF1E2A33CB027EE453D4C5AC5D8FA07083E2C0E26761169148F3A685095A77563C57D82BC1B48029

Verification

true

✓

```
In [28]: a.ReceiveKey(0x02780B2B11E834BC6415D82B8252C739CA86C4EBAA1084D151B86038B2F2DFC00B766F
879834CE5C662916DFA27BF4AC8F49B2964D8AFC322D8B86F7A54D3746, 0x227E29BD8A5164BEB36CA1A35F337A7C
F9
....: 3DC758E11E9F907F5F9F52F0D2F9444D75F7F4D5620E542E6190793C209F4AEDA5232D6782BFD1EEB704C
B7E233C36)
decrypted k: 0x177b23074531295a
decrypted S: 0x7c97ea9a093488513f3a991a104529423fa3faae25a61ee847b9627a154a8aa909d9b46fd2e5dd
a8d7b355ebb8b0e0e5c7feb1cf6369ef62eb95eff26f782c1b
Out[28]: True
```

В результаті виконання даної лабораторної роботи я ознайомився з тестами перевірки чисел на простоту для асиметричної криптосистеми RSA, та її імплементації на мові програмування Python