## 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: ФБ-21 Худоба Арсен, ФБ-21 Шабанов Кирило

#### 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.

```
# Тест Міллера-Рабіна
def miller_rabin(p, k=5):
   if p == 2 or p == 3:
       return True
    if p < 2 or p % 2 == 0:
       return False
   # Крок 0: розклад p-1 = d * 2^s
    s, d = 0, p - 1
    while d % 2 == 0:
       s += 1
        d //= 2
    # Крок 1: k раундів перевірок
    for _ in range(k):
       x = random.randint(2, p - 2) # вибір випадкової основи
        g = gcd(x, p)
        # Якщо х і р не взаємно прості, то р складене
        if g != 1:
           return False
        # Крок 2: перевірка сильної псевдопростоти
        x_power = pow(x, d, p) # x^d mod p
        if x_power == 1 or x_power == p - 1:
            continue
        for r in range(s - 1):
            x_power = pow(x_power, 2, p) # x^(2^r * d) mod p
            if x_power == p - 1:
               break
            return False # Якщо не знайдено псевдопростоти, р складене
   return True # Якщо р пройшло всі k раундів, воно, ймовірно, просте
```

# C:\Users\Arsen\AppData\Local\Programs\Py Випадкове просте число з 16 біт: 263 Process finished with exit code 0

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q i p1 , q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq  $\leq$  p1q1 ; p i q — прості числа для побудови ключів абонента A, p1 і q1 — абонента B.

**p**=36646164541686624240937833154972327964008617161515222144838114 780214993351409

**q**=31539962308096287988814609326825756936315439947047156766760061 841746726548927

**p1**=7343161666160487504968692855841485317242165548933188804982656 5905174368641603

**q1**=8297658364564081889509217850336282512337406931943949412411121 9775406714256373

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.

$$n = p * q$$
$$d = e^{-1} mod(\varphi(n))$$

USER A	e	10292583905073801259730970
Public key		68221163096944145951063717
		90540194300170707603436633
		44329770786165553887635295
		14534051688161223792719935
		968293626984425867521933
	n	11558186483810908087884266
		22881741142391198428698886
		21595664031330980599111618
		37551180318912177765098878
		19775411606981763699200602
		691337568368892342888143

USER A	d	68728193437129102715116561
Private key		26101696554382650736791126
		91651321489876975239832565
		00211713398465456779789388
		993825647741163111084352384
		9502748138788608293381
	р	36646164541686624240937833
	P	15497232796400861716151522
		2144838114780214993351409
	q	31539962308096287988814609
	1	32682575693631543994704715
		6766760061841746726548927
USER B	e1	59436485859858122632415132
Public key		77426104524909531827080437
		31452114011054640690549381
		67875970589671950507961262
		36271132265225950722175245
		792764113021343803053007
	n1	60931046821562889419540430
		00069782210860972797811892
		57696473139667325401576985
		69343270820951145360223874
		38202033783377223528841980
		601410274294334295685919
USER B	d1	26044697145954966846244318
Private key		17223957824901859666667370
		31057018318417205985875014
		71776888359759950526687457
		37482622807834945549214587
		332372933752703806192447
	<b>p</b> 1	73431616661604875049686928
		55841485317242165548933188
		8049826565905174368641603
	q1	82976583645640818895092178
		50336282512337406931943949
		4124111219775406714256373

 $Kopektho: e * d = 1 \mod \mathfrak{q}(n) \mod \mathfrak{$ 



Бачимо що Повідомлення для A, M = 414883970301082156863761214430378048022940790945404465792261466485707245004725150654940372749419411867391748650585791266187489113959404637438472653088140

Це те ж саме що на сайті, отже алгоритм працює коректно.

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 М і знайти криптограму для абонентів А и В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.

Зашифрувати C = Memodn

Цифровий підпис створити S=M<sup>d</sup>modn

Перевірити цифровий підпис M=Semodn

Повідомлення для А: М = 9599312779368407460523833228257026962232415263570141310196299597767655815343314367957307973632856543732576315893126440723759376714460973860253103488263
Зашифроване повідомлення для А: С = 93048455217039977697467289204371541585149574087969463475302975489436191818508932244090131895276757597240212757136924627317272646338853904981
Розшифроване повідомлення для А: М = 95993127793684074605238332282257026962232415265570141310196299597767655815343331436795730797363285653732576315893126440723759376714460973869225299
Перевірка підпису для А: М = 959931277936840746052383322822570269622324152655701413101962995977676558153433314367957307973632856543732576315893126440723759376714460973860225399
Перевірка підпису для А: М = 9599312779368407460523833228225702696223241526557014131019629959776765581534333143679573079736328565437325763158931264407237593767144609738602531034
Повідомлення для В: М1 = 2032504936180657460718496224135171567725074589638475318459396188202391628492265518184453178578776069633890733874053799590569839703319725015102045068378
Зашифроване повідомлення для В: М1 = 2032504936180657460718496224135171567725074589638475318459396188202391628492265518184453178578776069633890733874053799590569839703319725015102045068378
Зашифроване повідомлення для В: М1 = 20325049361806574607184962241351715677250745896384753184593961882023916284922655181844531785787760696338907338740537999590569839703319725015102045068378
Пифоровий підпис для В: М1 = 203250493618065746071849622413517156772507458963847531845939618820239162849226551818445317857877606963389073387405379995905698397033197250151020450698378
Перевірка підпис для В: М1 = 2032504936180657460718496224135171567725074589638475318459396188202391628492265518184453178578770606963389073387405379995905698397033197250151020450698648678780780698389073387405379995905698397033197250151020450698648678780780698378907838740637999590569839703319725015102045069864867807807808987808897838740637999590569897033197250151020450698668678078078078078078078078078078078078078

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

Абонент A формує повідомлення k1, s1 і відправляє його B, де

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

Абонент В за допомогою свого секретного ключа d1 знаходить (конфіденційність):

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

і за допомогою відкритого ключа е абонента А перевіряє підпис А (автентифікація):

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

Повідомлення, k = 1138871245103717467790293341541266222372696957810768120515208988767366165699249917157999604271883662770956240389270967991142452518801509339383355463954446
Цифровий підпис для A: S = 408492525629938831641893319265390407353997359984141633100334192001286072430993761126752830222350183097824807385505462365237786259748181190056256503
Абонент А формук повідомлення k1, s1 і відправляє його В
k1 = 1040465534689167761106740303041542647656771485383653755889696015984183953216708897105881958006819740373543971571922675496012226281472158631323787261748818
S1 = 4393055890600098998184863330508126577895791316353436686417714968560893497546104293267775872705413676336120174676293216395550438708733031304665358088730210

Абонент В за допомогою свого секретного ключа d1 знаходить (конфіденційність):
k = 11388712451037374677790293341541266222372696957810768120515208988767366165699249917157999604271883662770956240389270967991142452518801509339383355463954446
S = 40849252562993883164189331926539040735399735998414163310033419200128607243099376112675283022235018309782480738650546236523778625974818119005625650337833
I за допомогою відкритого ключа е абонента A перевіряє підпис A (автентифікація): k = 11388712451037174677902933415412662223726969578107681205152089887673661656992499171579996

#### Висновки:

У роботі була розроблена функція генерації випадкових простих чисел заданої довжини, яка використовує тест Міллера-Рабіна в комбінації з попередніми пробними діленнями для перевірки чисел на простоту. Такий підхід гарантує отримання простих чисел високої якості, необхідних для криптографічних цілей. Генерація двох пар простих чисел довжиною понад 256 біт дозволила забезпечити належну стійкість системи до атак. Використання власної реалізації тесту забезпечило розуміння алгоритму його роботи та дозволило адаптувати його до специфічних потреб завдання.

Було реалізовано функції для роботи з RSA, зокрема генерацію ключових пар, шифрування, розшифрування, створення та перевірку цифрового підпису. Особливу увагу приділено правильній реалізації операцій піднесення до степеня за модулем, яка виконувалася за допомогою схеми Горнера, що забезпечує ефективність навіть для великих чисел. Ці функції дозволяють виконувати всі необхідні криптографічні операції в системі RSA, забезпечуючи конфіденційність, автентифікацію та цілісність повідомлень між абонентами.

На завершальному етапі було розроблено та протестовано протокол конфіденційної передачі ключів за допомогою алгоритму RSA. Протокол передбачає використання відкритих та закритих ключів абонентів для шифрування та підпису повідомлень, що гарантує підтвердження автентичності відправника і захист переданих даних у відкритому каналі. Проведене тестування для випадково згенерованих ключів та повідомлень підтвердило коректність реалізації та забезпечення безпеки обміну. Це демонструє повну функціональність розробленої системи, яка може бути використана у реальних сценаріях.