

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-33 Ольшевський  
Богдан,  
ФБ-25 Степура Нікіта

## 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.

```
24 def miller_rabin_test(n, k=20):
25     """
26     Тест Міллера-Рабіна на простоту числа.
27
28     Args:
29         n (int): Число для перевірки
30         k (int): Кількість раундів тестування
31
32     Returns:
33         bool: True якщо число ймовірно просте, False якщо складене
34     """
35     # Перевірка тривіальних випадків
36     if n < 2:
37         return False
38     if n == 2 or n == 3:
39         return True
40     if n % 2 == 0:
41         return False
42
43     # Представляємо n-1 у вигляді d * 2^s
44     s = 0
45     d = n - 1
46     while d % 2 == 0:
47         d //= 2
48         s += 1
49
50     # Виконуємо k раундів тестування
51     for _ in range(k):
52         a = random.randint(2, n - 2)
53
54         # Перевіряємо НСД(a, n)
55         if math.gcd(a, n) != 1:
56             return False
57
58         # Обчислюємо a^d mod n
59         x = pow(a, d, n)
60         if x == 1 or x == n - 1:
61             continue
62
63         # Перевіряємо послідовність квадратів
64         for _ in range(s - 1):
65             x = pow(x, 2, n)
66             if x == n - 1:
67                 break
68         else:
69             return False
70
71     return True
```

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  $pq \leq p_1q_1$ ;  $p$  і  $q$  – прості числа для побудови ключів абонента А,  $p_1$  і  $q_1$  – абонента В.

$p = 48567968306380870604893231928375565109999867986982953555518657836663878595709$   
 $q = 41114984726175839504123089274205806665507535906835052448065721822642262646757$   
 $p_1 = 90204722267153738228133958885132331265679983340331553534428936669288766540631$   
 $q_1 = 70560061736961412940752647727865078560813287753268083770394763733321667743837$

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  $d_1$ .

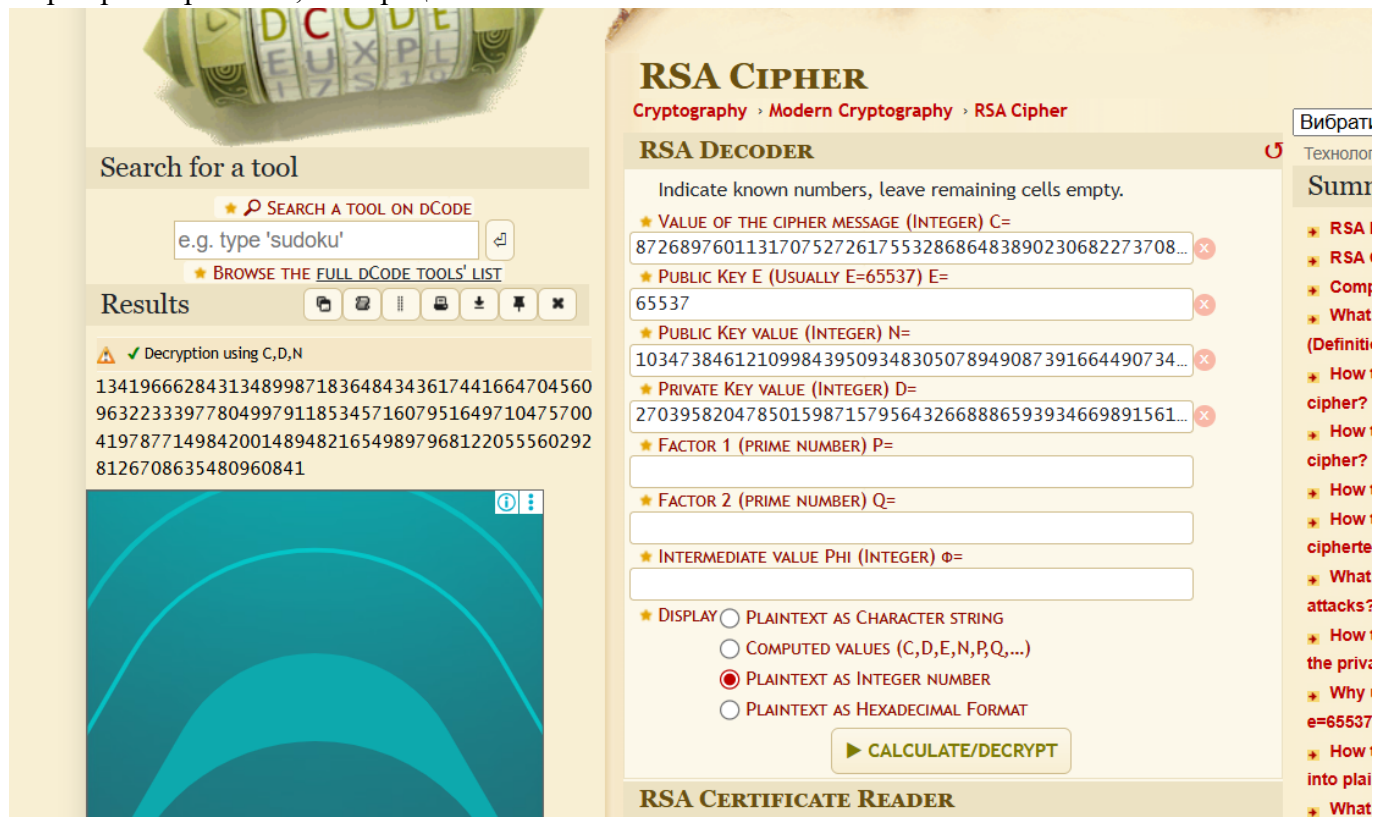
$$n = p * q$$

$$d = e^{-1} \bmod (\varphi(n))$$

```

=== КОРИСТУВАЧ А ===
Відкритий ключ:
  n = 66436711891088519247527262633836338083478581247353103021034073597237833057738071362170900683
  07777718583191776499212973075806096023737825425625874172176687
  e = 65537
Закритий ключ:
  d = 24846480742947947064358960696329228472863573651107382929422236963368773185301734514707125467
  62375217261363080854442454617008279157504897649037426406956753
  p = 93864189177838302834144587937819744749937335908201015107300034777975051520509
  q = 70779615179134241345670541237137906743935444717495533094071811697927091577243
  
```

Перевірка через сайт, все працює



The screenshot shows the RSA Cipher website interface. On the left, there's a search bar with the text "Search for a tool" and a search button. Below it, a list of results is shown, including "Decryption using C,D,N" with a list of numbers. On the right, the "RSA DECODER" tool is active. It has a title "RSA CIPHER" and a subtitle "Cryptography · Modern Cryptography · RSA Cipher". The tool includes a section "Indicate known numbers, leave remaining cells empty." with fields for "VALUE OF THE CIPHER MESSAGE (INTEGER) C=", "PUBLIC KEY E (USUALLY E=65537) E=", "PUBLIC KEY VALUE (INTEGER) N=", "PRIVATE KEY VALUE (INTEGER) D=", "FACTOR 1 (PRIME NUMBER) P=", "FACTOR 2 (PRIME NUMBER) Q=", and "INTERMEDIATE VALUE PHI (INTEGER) Φ=". There are also radio buttons for "DISPLAY" options: "PLAINTEXT AS CHARACTER STRING", "COMPUTED VALUES (C,D,E,N,P,Q,...)", "PLAINTEXT AS INTEGER NUMBER" (which is selected), and "PLAINTEXT AS HEXADECEMAL FORMAT". A "CALCULATE/DECRYPT" button is at the bottom of the form. On the far right, there's a sidebar with a "Вибрати" (Select) button and a list of links related to RSA and cryptography.

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 і В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.

**Зашифрувати  $C = M^e \bmod n$**

**Розшифрувати  $M = C^d \bmod n$**

**Цифровий підпис створити  $S = M^d \bmod n$**

**Перевірити цифровий підпис  $M = S^e \bmod n$**

```
=== ТЕСТУВАННЯ ШИФРУВАННЯ ===
Оригінальне повідомлення: 574046036020958603865163368389000862908380482448991428680908764010397621
118065480305794605620247977622609153347
Шифрування коректне: True
397621118065480305794605620247977622609153347
Шифрування коректне: True
397621118065480305794605620247977622609153347
Шифрування коректне: True
397621118065480305794605620247977622609153347
Шифрування коректне: True
397621118065480305794605620247977622609153347
Шифрування коректне: True

=== ТЕСТУВАННЯ ЦИФРОВИХ ПІДПИСІВ ===
Цифровий підпис: 464543826078590707490517961099750667084142096085336297890003770338534243021212781
9917152000379055015697594927619933492959927288387988103002787177756671046
Перевірка підпису: True
```

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  $0 < k < n$ .

### **Повідомлення для обміну**

Програма генерує випадкове число М, яке виконує роль повідомлення, що користувач А надсилає користувачу В. Це число знаходиться у межах від 0 до  $n_A$ .

### **Зашифроване повідомлення**

Повідомлення М шифрується відкритим ключем користувача В ( $n_B$ ,  $e_B$ ).

Завдяки цьому лише В може його розшифрувати, адже тільки він має відповідний закритий ключ  $d_B$ .

Цей етап забезпечує конфіденційність обміну.

### **Зашифрований підпис**

Користувач А створює цифровий підпис свого повідомлення за допомогою свого закритого ключа ( $d_A$ ).

Потім цей підпис додатково шифрується відкритим ключем користувача В.

Таким чином досягається автентичність і захист від підробки повідомлення.

## Перевірка автентичності

Після отримання повідомлення користувач В розшифровує його, отримує підпис і перевіряє, чи дійсний він для цього повідомлення.

Якщо перевірка успішна, програма виводить повідомлення:

```
=== БЕЗПЕЧНИЙ ОБМІН ПОВІДОМЛЕННЯМИ ===
```

```
Повідомлення для обміну: 5740460360209586038651633683890008629083804824489914286809087640105530377  
660340240978162457749841766169624422397621118065480305794605620247977622609153347
```

```
Зашифроване повідомлення: 511195412100469229195213318478006205195795092200982516130383455943298306  
8305509277144992627512950829073188252243725922002009478211089723833311108661408409
```

```
Зашифрований підпис: 23284753270024285708632448018348467550006122907801034071582160532671017147075  
0262865831538791967306570684361776803224655251780014355282071891601046740743
```

```
Повідомлення автентичне: True
```

Як можна побачити, повідомлення автентичні, тому програма вивела true.

**Висновки:**

У нашій роботі була створена функція для генерації випадкових простих чисел заданої довжини, що поєднує тест Міллера-Рабіна з попередніми пробними діленнями для перевірки простоти чисел. Такий підхід забезпечує отримання простих чисел високої якості, що відповідають вимогам криптографії. Генерація пар простих чисел завдовжки понад 256 біт дозволила забезпечити стійкість системи до атак.

Були реалізовані функції для роботи з RSA, включаючи генерацію ключових пар, шифрування, розшифрування, а також створення та перевірку цифрового підпису. Особлива увага була приділена коректному виконанню операцій піднесення до степеня за модулем, що здійснювалося за допомогою схеми Горнера для підвищення ефективності при роботі з великими числами. Ці функції забезпечують необхідні криптографічні операції в системі RSA, гарантуючи конфіденційність, автентифікацію та цілісність повідомлень.

На останньому етапі був розроблений і протестований протокол конфіденційної передачі ключів з використанням RSA. Протокол передбачає шифрування та підписування повідомлень відкритими та закритими ключами абонентів, що забезпечує підтвердження автентичності відправника та захист даних у відкритому каналі. Тестування з випадковими ключами та повідомленнями підтвердило правильність реалізації та рівень безпеки обміну, що демонструє повну функціональність системи для використання у реальних сценаріях.