МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий Фізико-технічний інститут Кафедра інформаційної безпеки

КРИПТОГРАФІЯ

Комп’ютерний практикум №4

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

Виконали:
Студенти ФБ-33
Дохоян Юлія
Терещенко Микола

Київ – 2025

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 1 1 p , q 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.

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) , (,) 1 n1 e та секретні d і d1 .

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. 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.

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 $0 < k < n$.

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().

Для кожного A та B генеруємо два простих числа pта qза допомогою функції generate_prime().

Просте число перевіряється функцією is_prime(),:

Якщо число не проходить перевірку, воно додається до лічильника невдалих кандидатів.

У результаті для кожного абонента отримано два простих числа p, qта кількість кандидатів, які не пройшли перевірку простоти.

```
Кількість кандидатів, що не пройшли перевірку простоти: 78
Кількість кандидатів, що не пройшли перевірку простоти: 47
Кількість кандидатів, що не пройшли перевірку простоти: 17
Кількість кандидатів, що не пройшли перевірку простоти: 114

=== ВИБРАНІ ПРОСТІ ЧИСЛА ===
A: p = 769701305975999584164803870685175077318214670917775945251865096867488934954 33, q = 662727385891446048707671751410110587952040604329073212934178201586320854226 19
B: p1 = 816551705860109404393826848868100380116525566519382045908626526562847632740 73, q1 = 111566347500417928767256990051243196904096277379146451962361621309448646372609
```

Генерація ключів RSA

1. Обчислюємо модуль $n = p \cdot q$.
2. Обчислюємо функцію Ейлера $\varphi(n) = (p-1)(q-1)$.
3. Генеруємо відкритий ключ eта перевіряємо, щоб $\gcd(e, \varphi(n)) = 1$, використовуючи розширений алгоритм Евкліда extended_euclid().
4. Обчислюємо приватний ключ dяк обернений елемент до eза модулем $\varphi(n)$.
5. У результаті для кожного абонента отримуємо: відкритий ключ: приватний ключ:

```
=== ПАРАМЕТРИ RSA ===
A:
  nA = 376645762652223163899208088937695509525082201985846048332259304907394826835919459324255210332133562307238404601128911560838117427856753021672268966012866 7 (модуль, modulus)
  eA = 311926673193288142397751350926561560263631571993510181831639816309503035639544071806542537973766503636303654952778023687608625498419414575561699139877989 3 (відкритий ключ, public exponent)
  dA = 238334407632498886396020504497085555098595042012058868091587618658926988732614908724281732048461297373138784820067766398861432960492320060609852897633455 7 (приватний ключ, private exponent)
B:
  nB = 917690457240241459623317424038570029778335749965423959070548030711336960906813201573357639887717741677497694005847179043241515430211912481704399329476830 3 (модуль, modulus)
  eB = 633436946852250944437773605735968736159222323837787175428386217852889198424675827422098409163341252397550713003014432854021217991615597966978152847499166 17 (відкритий ключ, public exponent)
  dB = 772791107973972168939503200142943226329517952850609643837468408330110251429010832040960612710071847896268730106004870349596134044481262301326438584452575 3 (приватний ключ, private exponent)
```

Далі обираємо відкритий текст випадково

```
=== ПОЧАТКОВЕ ПОВІДОМЛЕННЯ ===
M = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184 (випадкове повідомлення, 0 < M < min(nA, nB))
```

Шифрування та підписування повідомлення

1. Абонент A формує цифровий підпис:

$$\text{Signature} = M^d \bmod n_A$$

за допомогою функції Sign().

2. A шифрує повідомлення та підпис для B за допомогою відкритого ключа B.

```
=== ШИФРУВАННЯ ТА ПІДПИС ===
Encrypted M = 877971891732434725880241875256809931999956397109265117222953193470290716950387255438038663718341508180631331477578228503757288150746902114530160585922373
4 (шифротекст, ciphertext)
Encrypted Signature = 193479703894448858580651501979755273127471071678841379649205233084202653184751644751654135113969534430577892613854175236295890121336429382693398824025631
3 (шифрування підпису, encrypted signature)
Signature (A) = 231237274139810787345622306106064079332762354121769627246884881097965200018583457146528085840776483775200792553822074736744908194407932498873727081237302 (цифровий підпис, signature)
```

Розшифрування та перевірка підпису

```
Decrypted M = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184
(розшифроване повідомлення, plaintext)
Signature valid = True (перевірка підпису, True/False)
```

A обирає ключ K.

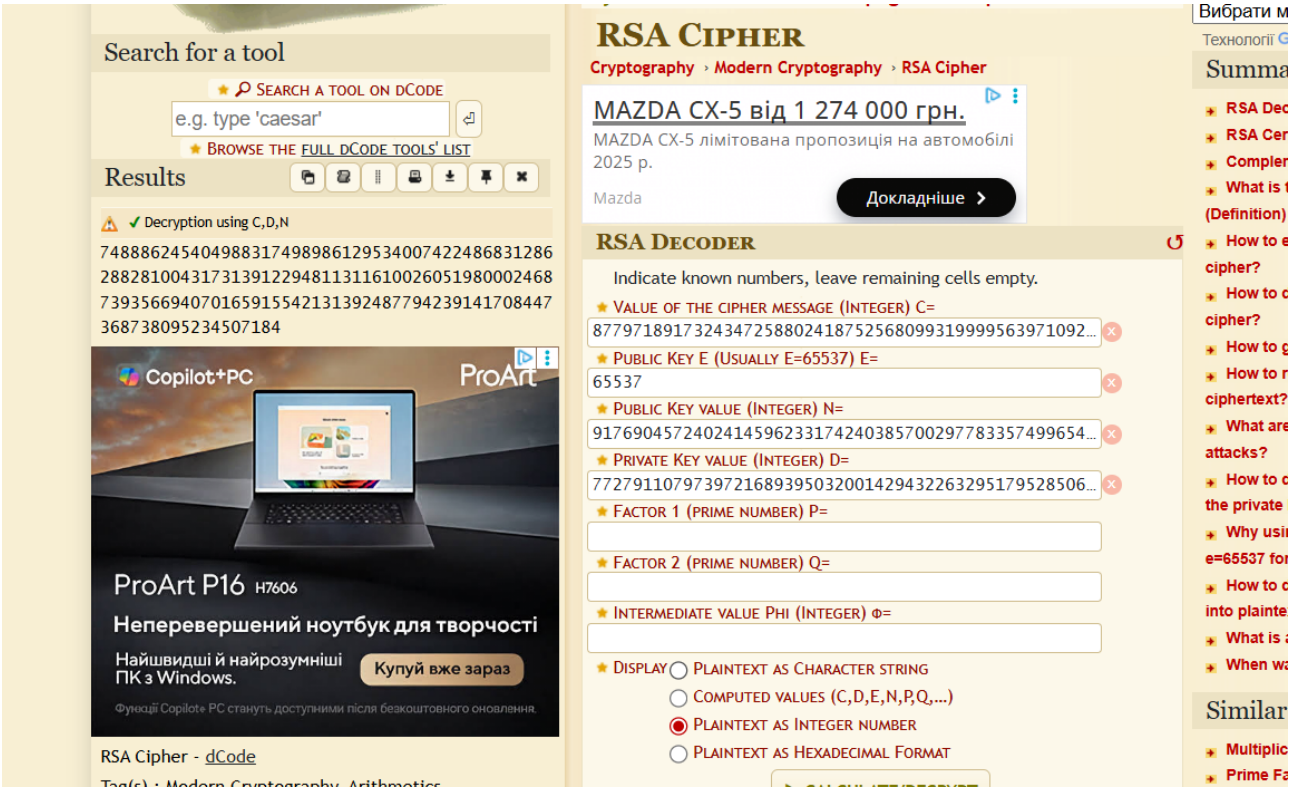
A шифрує та підписує його для B, відправляє зашифроване повідомлення і підпис.

B розшифровує отримане повідомлення і перевіряє підпис.

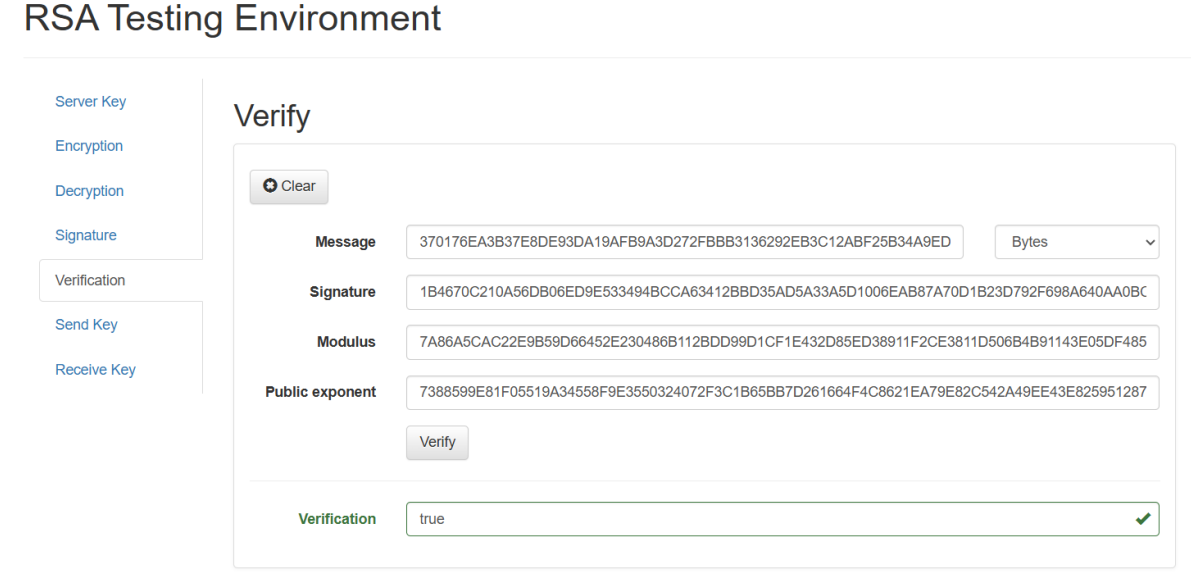
Результат: B отримав ключ K

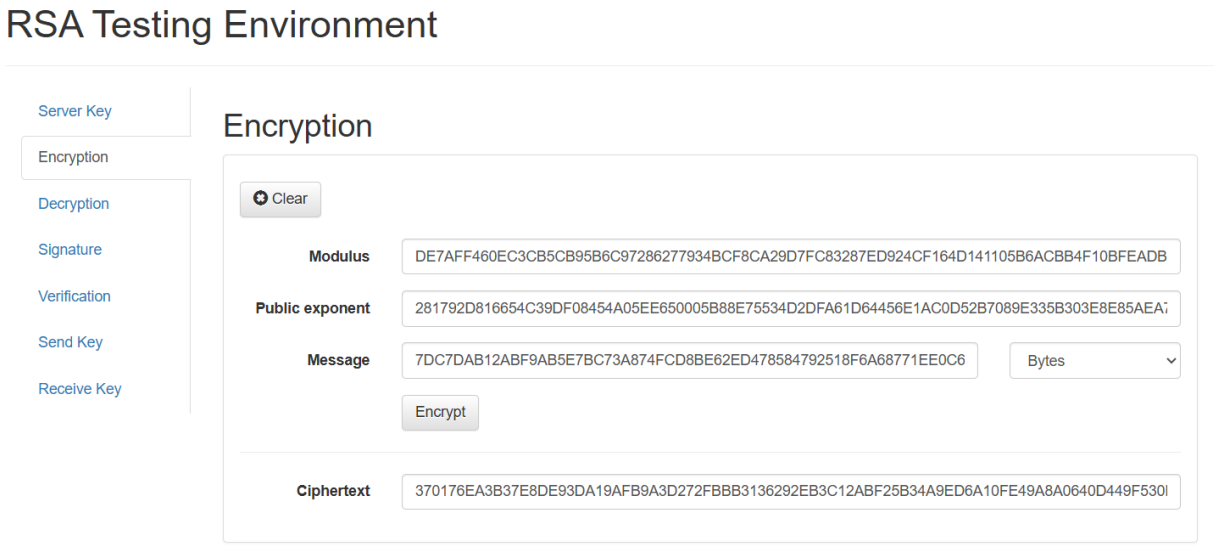
```
A обирає K = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184
A шифрує та підписує K для B:
  Encrypted message = 877971891732434725880241875256809931999956397109265117222953193470290716950387255438038663718341508180631331477578228503757288150746902114530160585922373
4
  Encrypted signature = 193479703894448858580651501979755273127471071678841379649205233084202653184751644751654135113969534430577892613854175236295890121336429382693398824025631
3
B розшифровує та перевіряє підпис:
  Decrypted message = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184
  Signature valid = True
```

Перевіримо на сторонньому сервері



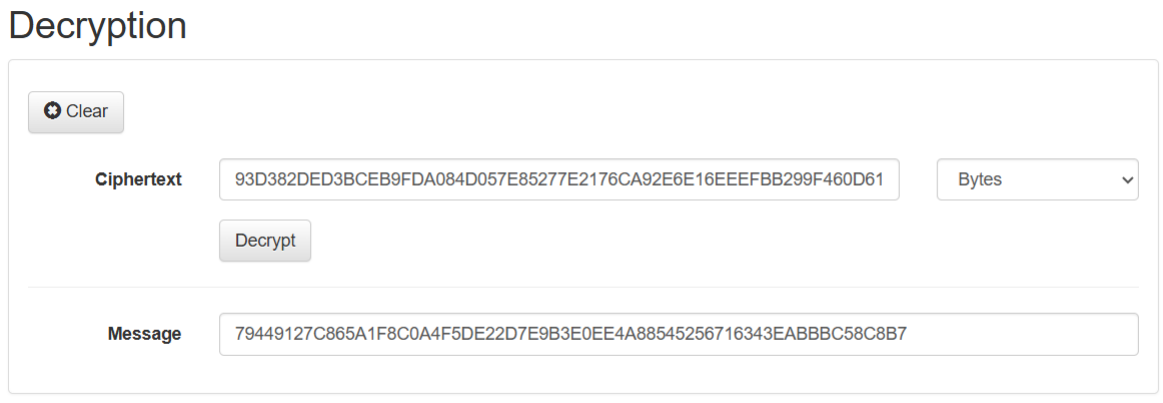
Тепер протестуємо на сайті





Зашифруємо локально, а розшифрую на сервері

```
Модуль (HEX): AE69FEE0621279E4158FC5DB95DB1DFF6C09370D680BD843AE9612DA7DC0C3D5
Початкове повідомлення M (HEX): 79449127C865A1F8C0A4F5DE22D7E9B3E0EE4A88545256716343EABBBC58C8B7
Зашифрований текст C (HEX): 93D382DED3BCEB9FDA084D057E85277E2176CA92E6E16EEEFBB299F460D6104A
```



Висновок: лабораторна робота успішно продемонструвала теоретичні та практичні аспекти роботи алгоритму RSA. Було перевірено всі ключові етапи: генерацію ключів, шифрування, розшифрування, створення цифрового підпису та його перевірку. Реалізований протокол конфіденційної розсилки ключів показав ефективність асиметричного шифрування для забезпечення безпеки переданої інформації.