НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАКУОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт за темою:
«Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем»

Виконав студент
групи ФІ-94
Кріпака Ілля

Київ — 2022

1 Мета практикуму

Практично ознайомитися із тестами перевірки чисел на простоту, методами генерації ключів для асиметричної криптосистеми типу RSA, протоколом розсилання ключів та безпосередньо реалізувати їх.

1.1 Постановка задачі та варіант

Треба реалізувати	**Зроблено**
PRNG	DefaultPRNG ✓
Miller-Rabin test	✓
Extended Euclidian algorithm	✓
RSA Encrypt/Decrypt	✓
RSA Sign/Verify	✓
RSA Send/Receive key	✓

2 Хід роботи/Опис труднощів

На початку релазіції практикуму, треба було вибрати генератор псевдовипадкових чисел із попередньої лабораторної роботи та використати тут. Не довго думаючи, обрав влаштований генератор, що працює на алгоритмі ChaCha12 і дає найкращі результати на тестуванні послідовності. Труднощів із реалізацією самої криптосистеми RSA не було, одразу функціонал підпису та шифрування розбив на RsaEncryptor, RsaSigner, що спростило подальшу роботу. Також проблем із написанням тесту Міллера-Рабіна, розширеного алгоритму Евкліда не виникало, адже вони уже були написані і відлагоджені у курсі Теоретико-Числових Алгортимів. Хочу додати, що для генерації простого числа застосовую комбінацію алгоритмів Міллера-Рабіна та пробні ділення (перші 100 простих). Уже для 2048 бітних чисел треба куди більше часу чекати, ніж на менші, але дочекатися можна.

Основна проблема була із перевіркою вихідних даних із сервером та із протоколом обміну ключем між користувачами. Головне, що я упустив, що функції:

1. SendKey() — спочатку записуємо собі публічний ключ сервера, потім зашифровуємо ключ та надіслаємо підписану пару назад (кроки 0, 1);
2. ReceiveKey() — надаємо свій публічний ключ, де сервер у свою чергу зашифрує повідомлення та надсилає нам (кроки 2,3).

Зауваження. Нульовим кроком вважаю надсилання публічного ключа користувача $B > A$.

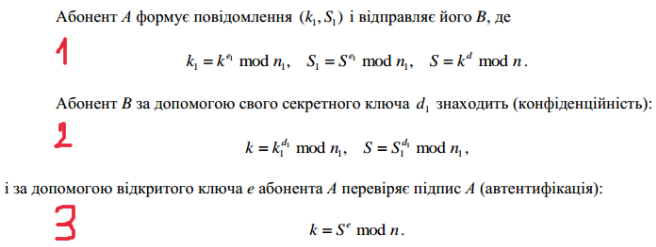

Рис. 1: Вигляд протоколу.

# біт	Число у HEX	Просте?
256	0xDA44D604C658EAA955379B1B7B94F321 B6A690462AA14483A0C387A1E0D362D9	*false*
256	0xDBF6CB5A81388144448A6387CC960642 5551B6A6833B30E1C736CFEE4FD98F33	*false*
256	0x3393DFF87DC8C9FAA2C8C31597BB3C66 076BF4CBC331E38F9679D368F6746239	*true*
256	0x17CCA6C243E91E490147765C97BD3F7D 42E5FE4C21F2C6F878337BA30C889007	*true*
128	0x4203E2FF7902D5725FE064183FC35681	*false*
128	0xD705D142F93F21868CBBA75E5B06388	*false*
128	0xE3856FDE94235F7457F7B023BD918D6D	*true*
128	0xFB23844228242601CEE61AEE41E66525	*true*

Табл. 1: Результати тестування чисел на простоту алгортимом Міллера-Рабіна

3 Результати дослідження

Наведу числені результати виконання програми для ключів 128 та 256 біт.

Зауваження. Повідомленням за замовчуванням вважаю — $ABCDEF123456789$.

Приклад .1.

1. (а) Шифруємо повідомлення публічним ключем №1: $66F4C31219CC070D2725B4EB9D34913D$,
   (б) Розшифровка сервера повідомлення: $0ABCDEF123456789$.
2. (а) Сервер шифрує повідомлення приватним ключем №2: 17D126D38F6FA437130F625C61FC593 7772BEC1B732E46DF0D6BE92E470CFFAD’
   (б) Розшифровуємо попереднє повідомлення приватним ключем №2: $ABCDEF123456789$.

# біт	n	e
128	883CEC2DC61C496CC99199CC1C9416E5	10001
256	8C9AE8DD1243B91B291034D99C4B2C6E 9141D8B2E0B3F255E1B60AA6473014B7	10001

Табл. 2: Публічні ключі 1,2.

# біт	Особистий ключ №1	# біт	Особистий ключ №2
128	n = 15df8f40346b5b3441655b57380fb7eb	256	n = 91fbf48e2e7324ef16f0faa145a41625 f2e06886b5c4f461bf6fd513e9fd799b
–	e = 10001	–	e = 10001
–	d = c6b700f68b43c88a2eb58b3d3edce01	–	d = 3bfb7a90b1253e97bea56555be779d61 7822a9d70be57c2d03b528fd02378681
–	primes = [c5c22fa2cc1c442b, 1c50aca1dec5bb41]	–	primes = [ed5198a17488705858bb68d817e5175f, 9d79cd5e9562c54492a4c974c151f345]

Табл. 3: Особисті ключі 1,2.

Напрям	Дані	
$A > B$	n=91fbf48e2e7324ef16f0faa145a41625 f2e06886b5c4f461bf6fd513e9fd799b	e=10001
$A < B$	key = 55FA139F08277E93FB22A5D9E730965C C0F2140644D0249AD7AC64DBEE18A9DA	s = 19B568288BB4C6442243BD535FFE8A72 85B65BB5016169DCA77AB2E049219B38
A	key = C189A5C812E05395	isok = true

Табл. 4: Приклад дешифрування по протоколу 1 (Публічний ключ №1, особистий ключ №2).

3. (а) Підписуємо повідомлення приватним ключем №2:
   i. s = 57B7156804546A4DD2EB54C9490655EA
6B87AFEFE754B0F5F810BF70D996B2C3 ,
   ii. $m = ABCDEF123456789$,
   iii. $is\text{-}ok = true$.

   (б) Підписуємо повідомлення приватним ключем №1:
   i. $s = 5D1BEC41053A2C59035AFA74E951B92$,
   ii. $m = 0xABCDEF123456789$,
   iii. $is\text{-}ok = true$.

Зауваження. Інші приклади шифрування, підпису на більших числах є у файлі додатку до звіту — «appendix_to_report.txt» .

Із вище зазначених результатів дослідження можна зробити висновок, що:

- Умову $n_1 \geq n$ у протоколі передачі ключа треба виконувати обов'язково, адже за іншої умови будемо отримувати неправильні повідомлення. Доприкладу, візьмемо (100 mod 10) mod 49, вийде неправильна відповідь, коли треба брати 100 mod 49;
- Криптосистема напряму залежна від простоти чисел, довжини n та інших критеріїв, що при невиконанні усіх умов буде давати ніяк незахищене з'єднання.

Напрям	**Дані**	
$A < B$	n=8C9AE8DD1243B91B291034D99C4B2C6E 9141D8B2E0B3F255E1B60AA6473014B7	e=10001
$A > B$	key = 5263AEF93A9B9DA25F66A66276CF9FCC FD2ADE9A72488ECA3567DB1B7B942D11	s = 14418308DAA7CEA590A693591BFF8FED 4B6CB0C3ECD7B6D492D1D72D9950BDDC
B	key = ABCDEF123456789	is$_o k = true$

Табл. 5: Приклад надсилання ключа (Публічний ключ №2, особистий ключ №1).

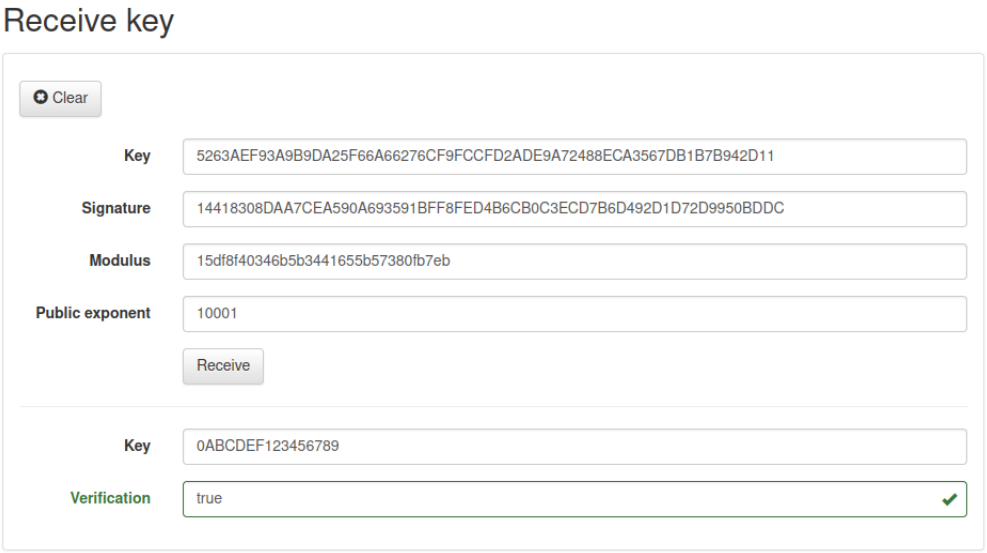

Рис. 2: Результат надсилання ключа №2.

4 Висновки

За допомогою реалізації практикуму "Вивчення криптосистеми RSA та алгоритму електронного підпису" дізнався на практиці, як генеруться параметри для асиметричних криптосистем та як реалізовувати їх.

Рекомендую для алгоритмів знаходження псевдопростих чисел одразу додавати алгоритм пробних ділень. Він значно скоротить час перебору чисел.