



*Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут ім. Ігоря
Сікорського»
Навчально-науковий Фізико-технічний інститут*

Криптографія

Комп'ютерний практикум №4.

*Вивчення криптосистеми RSA та алгоритму
електронного підпису. Ознайомлення з методами
генерації параметрів для асиметричних
криптосистем*

Варіант №3

Виконав:

Студент групи ФБ-31

Васалатій А. Ю., Яковчук О. С.

Перевіreno:

Байденко П. В.

12.11.2025

Виконання лабораторної роботи було почато з ознайомлення з необхідними відомостями з теорії чисел. Першочергово було розглянуто і програмно реалізовано піднесення до степеня за модулем з використанням схеми Горнера.

$$x^\alpha \pmod{m} = (...((x^{\alpha_{k-1}})^2 x^{\alpha_{k-2}})^2 ... x^{\alpha_1})^2 x^{\alpha_0} \pmod{m}, \text{ або}$$

$$x^\alpha \pmod{m} = x^{\alpha_0} (x^2)^{\alpha_1} (x^{2^2})^{\alpha_2} ... (x^{2^{k-1}})^{\alpha_{k-1}} \pmod{m}$$

(x, a – цілі невід'ємні числа, $x < m$;

a перед безпосереднім початком обчислень подається у двійковій формі)

Після цього ми перейшли до поняття псевдопростих чисел - складених чисел, що зберігають певні властивості простих. Серед пройдених – псевдопрості Ферма, Ойлера-Якобі та сильні псевдопрості числа.

Базуючись на даних відомостях, ми почали розгляд ймовірнісних тестів для перевірки числа на простоту - Ферма, Соловея-Штассена і Міллера-Рабіна. Після ознайомлення з постулатом Белтрана і алгоритмом генерації простих чисел, ми перейшли до програмної реалізації генерації “гарних” простих чисел з використанням вищезгаданих тестів (у користувача є можливість використати будь-який з трьох) на простоту для подальшого застосування у нашій імплементації криптосистеми RSA.

Серед реалізованого функціоналу:

- 1) Функція генерації пари відкритий/секретний ключ на основі пари двох великих простих чисел (довжина обиралася 256 біт+). Як експонента використовувалася одна з найбільш поширеніх у сучасних реалізаціях криптосистема RSA - 65537.
- 2) Функція зашифрування, що базується на перетворенні $C = M^e \pmod{n}$
- 3) Функція розшифрування, що базується на перетворенні - $M = C^d \pmod{n}$.

- 4) Функція генерації цифрового підпису - $S = M^d \text{ mod } n$.
- 5) Функція перевірки цифрового підпису - як перевірки чи є одинаковими M та $S^e \text{ mod } n$.

Також було імплементовано модель протоколу конфіденційного розсилання ключа по відкритим каналам з підтвердженням відправника у вигляді двох функції $((e, n), d)$ і (e_1, n_1) , d_1 є відповідно відкритими і секретними ключами абонентів А/В відповідно, при чому $n \leq n_1$):

- 1) Створення повідомлення виду (k_1, S_1) , де

$$k_1 = k^{e_1} \text{ mod } n_1; S_1 = S^{e_1} \text{ mod } n_1; S = k^d \text{ mod } n;$$

- 2) Отримання ключа з повідомлення вище з перевіркою підпису.

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

Тестування

Перевірка коректності реалізованого здійснюється двома способами - локальним тестування через створення двох абонентів, а також використанням RESTful API веб-ресурсу, що був запропонований в методичних рекомендаціях (<http://asymcryptwebservice.appspot.com/docs>).

При локальному тестуванні першочергово генеруються дві пари з відкритого й секретного ключів для абонентів А та В, при чому н з відкритого ключа абонента А не більший за В, що необхідно для коректної роботи протоколу розсилання ключа.

```

=====
[LOGGED-IN]
=====

A public key: (65537, 72979275126070078311923764672614257092940230153539158639240958708282209905957227254055706277316377499322170287205964713517684152533345:192136341315397201)
A private key: (63674381356849003585060542712433523579526302705487322125676141109576041696025737194185507738612472666121168095342800510803345577566914642880:770355007829, 112709771727442483769230260406343384128904608545477062779290707480698893284327, 64749731995325250311766390148380017215548227169373985255288:9794977049863)

B public key: (65537, 75401509733262878124859871502740926781912641598600628015161578376458387421771074499734090028915809433473872258104217791266260463375912:686000688054134529)
B private key: (5801602956817544123661723729186892860376164403697201990162103470118086949554223399261703655557965425998209501232273032334229457562767181546:294785738505, 65365786688528336567123681454597515868109514429729065475919115229705945309273, 115353174119291684466125770631283586943552370915240128791034327:8214045595323)
=====
```

Обрані числа p , q , p_1 , q_1 були додатково перевірені на простоту з використанням ресурсу <https://bigprimes.org/primality-test>.

1127097717274424837692302604063433841289046085454770627792907

07480698893284327 is a prime number!

[TEST THAT PRIME](#)

6474973199533252503117663901483800172155482227216937398525528

8439794977049863 is a prime number!

35332525031176639014838001721554822272169373985255288439794977049863

[TEST THAT PRIME](#)

6536578668852833656712368145459751586810951442972906547919115

2297059453092723 is a prime number!

38528336567123681454597515868109514429729065479191152297059453092723

[TEST THAT PRIME](#)

1153531741192916844661257706312835869435523709152401287910343
27428214045595323 is a prime number!

| 9291684466125770631283586943552370915240128791034327428214045595323

TEST THAT PRIME

Далі обирається випадкове повідомлення в межах [1, n-1], після чого воно зашифрується і розшифрується з використанням відкритих і секретних ключів обох абонентів.

Також відбувається підпис і його перевірка з використанням ключів обох абонентів.

```
=====
Message: 33383759404033248341318432832072007309730431619682675066954235954614101983155493128113168906080325133572900236935331266346246290978911058705083710312050
=====
Enc msg with A pub key: 68401444088327820310487594578828122414401802464572639511599005907351411118638816436526614537455115761941561946671956236243075341490196987073121609486629
Dec msg with A priv key: 33383759404033248341318432832072007309730431619682675066954235954614101983155493128113168906080325133572900236935331266346246290978105870508371031812050
=====
Enc msg with B pub key: 70169928042230920098593414006325836528651940596944012283522894056665009155821095957872402850962752186490130872559090731584455853422739180504771526783218
Dec msg with B priv key: 33383759404033248341318432832072007309730431619682675066954235954614101983155493128113168906080325133572900236935331266346246290978105870508371031812050
=====
```

```
=====
Signed message with A priv key: 2211669285648802954874649044044198706504482973987197453565146788192275355161344392002571464908815646276326027975293466405535367027556140676764691946419521
Verified: True
=====
Signed message with B priv key: 6195956866185635736308208380044575726940941448594291809310073988570647374064599128979259848461146182202275271258085743817328634517350509991944883057519784
Verified: True
=====
```

Далі обирається випадкове число - “ключ” - і здійснюється створення повідомлення з зашифрованим ключем та його підписом для абонента В від А.

Після чого відбувається отримання ключа і підпису, з подальшою перевіркою, з використанням секретного ключа абонента В і відкритого А.

```
=====
Key: 4413889461252188303
Prepared key for sending: (550188129020994697809413727635874572175767300699670585240632120558783242310096306061078488308130414993464925327046559, 1050503709956936110590470961345059697662669968912589807988754108601759753134764297076049774716516219499667280976492)
Retrieved key: 4413889461252188303
Verified: True
=====
```

Для тестування з використанням API (<http://asymcryptwebservice.appspot.com/rsa/>) спочатку здійснюється отримання відкритого ключа серверу через GET-запит за шляхом <http://asymcryptwebservice.appspot.com/rsa/serverKey>. Після цього відбувається генерація пари ключів локального абонента (модуль ключа локального абонента спочатку обирається як не більший за серверного).

Далі обране клієнтом повідомлення спочатку шифрується локально з використанням відкритого ключа серверу і відправляється йому на розшифрування (<http://asymcryptwebservice.appspot.com/rsa/decrypt>) для перевірки коректності локального шифрування. Потім інше повідомлення відправляється на сервер для шифрування відкритим ключем клієнта (<http://asymcryptwebservice.appspot.com/rsa/encrypt>), а результат локально розшифровується для перевірки коректності реалізації розшифрування.

Message: 503630226990865578911255689909851443749552324876141146583609946847781999336512623656235179654622540432915449765618322446833812212417041207604123095921790894465723795459769348189640122621903802858161545255304695785785452576
Local encryption with pub key: 6869373457148569058151006729259839186933468584570030258016043170649877962466140767094218284132734321890944204779456470801224623986903746403180897658395909485436985694591669369567668288553063393848718657445178855
Decrypted message from server: 503630226990865578911255689909851443749552324876141146583609946847781999336512623656235179654622540432915449765618322446831241704120760412761273095592179089446572379545976934818964012262190380285816154525530465978587452576

Message: 402349354370931864979249827805195514114997213231897790522514677603759065464251168147701560461382678129341176720307132109535137023228963920338996
85573
Encrypted message from server: 409494921153986740550283207456098046145727428635138195405889924371144588587845689042189593658396501640300518277841044371923184852389345242677315655
Locally decrypted message: 40234935437093186497924982780519551411499721323189779052251467760375906546425116814770156046138267812934117672030713210953513789639203389909907485573

Також відбувається перевірка коректності реалізації створення і перевірки цифрового підпису. Спочатку повідомлення підписується на клієнті і перевіряється на сервері (<http://asymcryptwebservice.appspot.com/rsa/verify>),

а потім інше повідомлення підписується на сервері (<http://asymcryptwebservice.appspot.com/rsa/sign>) і перевіряється на клієнті.

```
=====
Signed message from client: (3362675276011368389897720946236337983340163920553046516653274810620154401702912030930879124714303016944896855053819013160
289725029904226664315122201560164875783905139432806328773205936, 19395848254260973122042286258008484069047709066455512499247171010517084511686478048
462212827457752300178998275221053919706596241642787289916307321371597899249227943964531805029894943)
Verified by server: True
=====
Sent to server: 304553647804558634680679643238687585331482760910560852026254942045910159598135276219073040864
Signed message from server: 4174702156788010687410685608973843545314224937320740115400025955990825899550095815505375512942013509335510823921209009218
44905415351621460034347064595217865594387120736217318794172045628112403714406175907505588689472057662432
Verified locally: True
=====
```

Також з використанням імплементації протоколу конфіденційного надсилання ключів спочатку здійснюється отримання такого ключа від серверу (<http://asymcryptwebservice.appspot.com/rsa/sendKey>) і локальна перевірка його цілісності на основі підпису, а потім повернення назад з перевіркою цілісності з боку серверу (<http://asymcryptwebservice.appspot.com/rsa/recieveKey>).

```
=====
Received encrypted secret and signature: (3243203868342594456622708969235257875688272166608019639618113391583659743889146279067725952661751015098206432877
7631164614529535419204010351783159878073258217833324287762585302372252695562610422422717941727533570419773424190603218911994170753134108141037378162
5767629930551227649351011836519047, 5554180584295960795385948746792086293905161779864693846320374791435763981743286024917916861328292765874973383
6001349981372071146485383451937296781289899927822298151071984597319372961078643882983461869318820729201369705778024334365840067566314913190931857898594948
3664016159450430010025361233)
Retrieved key: (3609163105358882241, 19910888753645085491170015720996818110768668162039621360429764098495683885378178464856721694576490663374645002443857922
2794290868699710283455316291678253370735289241976566096179057868486554432890410286437237342483787854838693697)
Verified: True
=====
Sent encrypted secret and signature: (3047755456341324506782924779845352478326528843996152287072985870939955358449894947279998975946181160139536358859131038
69064781253610803918409329606614011677104818133883710543857343400244817154838830962093998009229057378827535561, 498418811575038087880705595284199036545701
20251526673286730606646171464259862651058574813590543074991583410914908950947456173147674817817073126116527386311029939875362540996481975710376244649477424
3555391074949915066706812869)
Server got: 3609163105358882241
Verified: True
```

Висновки

Першочергово в межах даного практикуму відбулося ознайомлення з необхідними відомостями з теорії чисел - піднесенням до степеня за модулем за схемою Горнера; поняттям псевдопростого числа і зокрема псевдопростими числами Ферма, Ойлера-Якобі й сильними псевдопростими; ймовірнісними тестами Ферма, Соловея-Штрассена та Міллера-Рабіна, а також алгоритмом генерації простих чисел загалом. Після цього було розглянуто криптосистему RSA, а також протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника

Базуючись на отриманих теоретичних знаннях, було реалізовано програмну імплементацію вищеописаного з перевіркою коректності роботи, базуючись на відкритому RESTful API веб-ресурсу, що був запропонований в методичних рекомендаціях (<http://asymcryptwebservice.appspot.com/docs>). Зокрема було використано всі запропоновані розробником запити, що стосуються криптосистеми RSA.