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

Звіт з виконання комп'ютерного практикума

РЕАЛІЗАЦІЯ ОСНОВНИХ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ

Виконали студентки
групи ФІ-32мн
Зацаренко А. Ю.
Футурська О. В.

Перевірила:
Селюх П. В.

Київ — 2023

ЗВІТ

1.1 Мета роботи

Дослідження можливостей побудови загальних та спеціальних криптографічних протоколів за допомогою асиметричних криптосистем.

1.2 Завдання на лабораторну роботу

Розробити реалізацію асиметричної криптосистеми Ель-Гамаля (ElGamal encryption system) за допомогою бібліотеки OpenSSL під Windows платформу.

1.3 Необхідні теоретичні відомості

Криптосистема ElGamal - це алгоритм криптографії з відкритим ключем, який дозволяє двом сторонам безпечно спілкуватися через незахищений канал. Він був запропонований Тахером Ель-Гамалем у 1985 році і базується на математичних властивостях модулярного піднесення до степеня та складності проблеми дискретного логарифму (складність знаходження дискретного логарифму в циклічній групі, тобто навіть якщо ми знаємо g^a та g^k, надзвичайно складно обчислити g^{ak}). Він широко використовується для захищених комунікацій і цифрових підписів. Криптосистема забезпечує конфіденційність завдяки своїй схемі шифрування та автентичність завдяки схемі цифрового підпису.

1.3.1 Алгоритм шифрування Ель-Гамаля

Припустимо, що Аліса хоче поспілкуватися з Бобом. Позначимо повідомлення Аліси як m, $m \in \mathbb{Z}_p^*$.

1) Боб генерує відкритий та закритий ключі:

   а) Боб вибирає дуже велике число p;

   б) З циклічної групи $\mathbb{Z}_q$ він вибирає довільний елемент g – генератор

поля та елемент x такий, що $gcd(x, q) = 1 \Longrightarrow x \in \mathbb{Z}_q^*$;

в) Потім він обчислює $y = g^x \ mod \ p$;

г) Боб публікує (y, p, g) як свій відкритий ключ, а x зберігає як закритий.

2) Аліса зашифровує дані, використовуючи відкритий ключ Боба :

а) Аліса вибирає елемент k з циклічної групи $\mathbb{Z}_q$ такий, що $gcd(k, q) = 1$– одноразовий особистий ключ;

б) Потім вона обчислює $c_1 = g^k \ mod \ p$ і $c_2 = m * y^k = m * g^{ak}$;

в) Аліса відправляє (c_1, c_2) – шифротекст.

3) Боб розшифровує повідомлення: обчислює $c_2 * c_1^{-x} \ mod \ p$.

Коректність:

$$c_2 * c_1^{-x} \ mod \ p = m * g^{kx}(g^k)^{-x} \ mod \ p = m.$$

1.3.2 Схема підпису Ель-Гамаля

Припустимо, що Аліса хоче підписати своє повідомлення. Позначимо повідомлення Аліси як m, $m \in \mathbb{Z}_p^*$.

1) Аліса генерує відкритий та закритий ключі:

а) Аліса вибирає дуже велике число p;

б) З циклічної групи $\mathbb{Z}_q$ вона вибирає довільний елемент g – генератор поля та елемент x такий, що $gcd(x, q) = 1 \Longrightarrow x \in \mathbb{Z}_q^*$;

в) Потім вона обчислює $y = g^x \ mod \ p$;

г) Аліса публікує (y, p, g) як свій відкритий ключ, а x зберігає як закритий.

2) Аліса підписує дані, використовуючи свій відкритий ключ:

а) Аліса вибирає елемент k з циклічної групи $\mathbb{Z}_q$ такий, що $gcd(k, q) = 1$ – одноразовий особистий ключ;

б) Потім вона обчислює $r = g^k \ mod \ p$ і $s = (m - r * x) * k^{-1} \ mod \ q$;

в) Аліса відправляє (r, s) – підпис m.

3) Боб перевіряє підпис, використовуючи відкритий ключ Аліси: обчислює $y^r * r^s$.

Коректність:

$$y^r * r^s = (g^x)^r * (g^k)^{(m-x*r)k^{-1} \ mod \ q} \ mod \ p = g^{x*r+m-x*r} \ mod \ p = g^m \ mod \ p.$$

1.3.3 Стійкість

Криптосистема ElGamal в цілому вважається безпечною, але її безпека залежить від складності певних математичних задач. Ось деякі аспекти захисту ElGamal від різних типів атак:

❄ Цілочисельна факторизація: Безпека ElGamal базується на складності проблеми дискретного логарифмування, а не цілочисельної факторизації. Це робить ElGamal стійким до атак, заснованих на факторизації великих чисел.

❄ Дискретна логарифмічна задача: Безпека ElGamal базується на припущенні, що обчислення дискретних логарифмів у скінченному полі або групі є обчислювально важкою проблемою. Зокрема, злам шифрування ElGamal вимагає розв'язання задачі дискретного логарифмування, яка стає дедалі складнішою зі збільшенням розміру простого модуля.

❄ Атака обраного відкритого тексту (CPA): Криптосистема ElGamal захищена від атак з підбором відкритого тексту до тих пір, поки проблема дискретного логарифмування є складною. Випадковий вибір k в процесі шифрування додає додатковий рівень безпеки.

❄ Атака на вибраний шифрований текст (CCA): Алгоритм ElGamal у своїй базовій формі вразливий до атак з підбором шифрованого тексту. Атака на обраний шифротекст (CCA) - це атака, при якій зловмисник може отримати розшифровку обраного шифротексту та скористатися цим для отримання секретного ключа або несанкціонованого доступу до зашифрованих даних. Зловмисник може надсилати ретельно створені зашифровані тексти до оракула для дешифрування і аналізувати отримані відповіді, щоб отримати уявлення про секретний ключ. Цю вразливість можна усунути, використовуючи такі методи, як схеми заповнення або гібридне шифрування, щоб зробити систему захищеною від адаптивних

атак з підбором шифрованого тексту.

❄ Безпека генерації ключів: Безпека ElGamal також залежить від безпечної генерації ключів. Якщо противник може передбачити або маніпулювати процесом генерації ключів, це може поставити під загрозу безпеку системи.

❄ Атаки побічних каналів: Реалізації ElGamal повинні бути стійкими до побічних атак, таких як атаки на час або аналіз потужності.

❄ Квантові атаки: Хоча вважається, що класичні комп'ютери мають труднощі з ефективним вирішенням проблеми дискретного логарифма, поява великомасштабних квантових комп'ютерів може потенційно загрожувати безпеці ElGamal. Квантові алгоритми, такі як алгоритм Шора, могли б ефективно вирішити проблему дискретного логарифма, але на сьогоднішній день практичних квантових комп'ютерів, здатних зламати ElGamal, не існує.

1.4 Програмна реалізація

У роботі було реалізовано наступні функції:

1) $generate\_parameters(size)$ – генерування початкових параметрів криптосистеми, параметром виступає розмір модуля поля в бітах;

2) $generate\_key\_pair()$ – генерування пари ключів (приватного і публічного);

3) $sign(message, a)$ – підпис повідомлення, параметрами виступають саме повідомлення та приватний ключ;

4) $verify(message, r, s, b)$ – перевірка підпису, на вхід функції подаються повідомлення, публічний ключ та отримана пара (r, s) – сам підпис;

5) $encrypt(message, b)$ – шифрування повідомлення публічним ключем;

6) $decrypt(x, y, private_key)$ – розшифрування шифротексту приватним ключем;

7) $can\_modinv(k, p)$ – перевірка існування оберненого;

8) $split\_message(message, block\_size, state)$ – розбиття повідомлення на блоки.

1.5 Отримані результати

1) Генеруємо параметри криптосистеми:

```
Розмір модуля поля в бітах:  2048
Модуль поля:
0xaa851bf5c87a68ddb4003bb0fdcf70179cac8001a1db42a5171f0cdf29e19e49a0457d4bd95e3b3c14155eab8
c44e56f3efe6871055f718cd5840eb9ccbb3e91c2f850b742de7bbe10abf7de7b982c9b80d9b637ac4bd3936696
624032142693e43397efba9ea11f4c5f96fdecc5e8252d4e7ba5e640526ef523643705cd152988646365f8adefd
08c5a2971305b04b7da7c8031eb1699cbcef4b8777003cb129795cd8fb1906cd8605175ba1bce02f15e15ca19ee
efa25616bcc3ad08e4e2b60ffdbdd07776f403d16411d2c7270f8817482d993ccd2a5675861e21007275e47e20d
f83980d657d66937b436ef7342875e671be464477661e2004fdb073d297
Генератор поля:
0x30203e43f9b6fe08ab5f2fde2537520b08820e4dba7b5b0504362798acf0b93d7d5a1b6be96dced7a10c0d912
aed57aeb82992e0984c31f60c37ebb4dc929d235ec14e5c14a79dd74891027ab5764a95f270821764e22377d0b1
33ff3ad006861e7c632bdcc72b610a9ae07767ce44bae48c10aee2a7545becd2d7afc8f0552ad5c8691d3603a55
5258c336745a8830ffdedec9a79684464a71070deea856a1bddb8520eb4e40a40263ac0891077f7912de4279b29
2802d1c999ab56c2162d165ce38929f85799607f6c5c028956568c48eaa26a6ea80196f1297c1c2ebd090472be7
954ac4ea0b0140594d85a1bf987f9c40157f90de99f7bbbd15e7589d41a
```

2) Генеруємо ключову пару для алгоритму підписання:

```
Приватний ключ:
0x8b0f0299e46a8297750a1e055b834060732650513ef077be1d2e11047d6ecf6e644fc4935a90ebfcf87345bfe
7ce1df1e11c94ee619382ae47cba9f48f8b95cd96f5997f1b760cf3969324e7eba1ecbcdabe88337317f01fd7a9
4e3ae8f0144e2db31f4f3bcdb8b257257c8ac85c32309ea7d162fe82ede7dd630696a43ad37385fdc2c6d544105
4f6c9055610985c89aad248cae531664627cff3745eb23cf2932c3234db7f2a82295a57a806c9e5bac6c78d74df
ba0ef1a10d4a4504b3afa7a2a1af049cc8dfe473730db11425a4ba1de90eb802aafcc3d1469671881720a3f8f2a
259baa41301f38a52b56cd198d51059f95ef3adc9376a3c83cedbf0a758
Відкритий ключ:
0x266d7b5686b5bac6d556483df8dca1bcf47e6f6dc8e1cffc218f3aec2c494a61aa4cd0be59fb5e456a8e4c69e
c939aad063eb40d5d395d4aa04211680f0df18e96cd7d9630bac295d9498802be5724054b817053ed406fa5580b
0b0160e528dd778182ae98082dd676f93b0068bf52464852714a23da7296e9e41fdedf7d0e9c0dfb3bda125362c
dddf3fda085189b1bba3802a5162f13ba0449d4824c736399687b18fa93c1585bf0d7f484bfb51855ef91355f12
40062e50529e6d5bb0cce045ba76d3817b1305fdb920357ae70a82dfcc9be9f55ae5446b6c142784f4e9156577a
b68ff579121d47107fdbcff8cdc3f48724858b63847bde1166f8ef8a18b
```

3) Підписуємо повідомлення:

```
Повідомлення, яке буде підписано:  This message is to verify the corectness of the El Gamal
digital signature algorithm.
Підпис:
r:
0x996d7112f7ed0335300c9937a6bd4a4ba6f076ee6e17f2285aa93dd6dbe6ed842f3450ed38e7b6ca2734abe99
c87f5616e4da16a0334a071a316d37bfcb0fee4a245b5b27c99ea2c1a52bd5a023ac7caa005eb7b3322a8427cb0
cf3bc8dab85a85fd8a938b3616e0badcbd7ba3ef38324e35a84668dc3a2d09e87297381089856f7c418a9431c44
7b8d861c9df6b85400babe89b0448e55294ae99d12a0616ab6c0dc6c7c679a6ee4306c78406cb0c6b33ed3211af
2e9418c44c98540e12e3cb560c6eac2ca3fb910eb7e44b5e986c188febf5a89d3855f9edde3efe974091c804932
7440450085fcfd80505dfb6712a537a6f08350aa4198732a5ea0996a76d
s:
0x29ea311ef9e66bd2e9aecc6886cd0ac59f3e62d21aa0940aa77b4ffd3aac3ce613eea0397d771f9112db3928c
32a80f6d67ac83dafb3733e04ce49922421c9be3279bfb83957d4f1202503c1b99b1321ec07e1f6614d89232eb3
5c466c8e991d1ae5f974a7d23b655ff74ef56a8526bfa95e9de9e14752a94d615a435a94113f9cb5144359e03e1
fe5d22532f4e3bb07999f7c5b33a7f92d01271c57357376c054179cc8b8b3f4c1f2c7102a852e3810ed2f95d8fd
21bf8b4c72573137bae7445ed81e382759c894524b0fd5a207e5c1977470d78077043a53f914a9d673ffeb2f480
a797aa4ecbfeda06e9d86ca39db5d3365de8c724ffb7b195f381b5b9d
```

4) Перевіряємо корекстність підпису:

```
Результат:  True
```

5) Перевіряємо підпис для зміненого повідомлення:

```
Змінене повідомлення:  This another message is to verify the corectness of the El Gamal
digital signature algorithm.
Результат:  False
```

6) Генеруємо ключому пару для алгоритму шифрування:

```
Приватний ключ:
0x78f0aca5e1c25164413a1f806bc9bc10ac6d6a6fefca41155a87fb6b5f38c3f4987ce91aee4616ec38c2ff7de
eb9eb34655963297a458e187a8d34848a86a3fea54cce050123596d472e646f8d730599b6145f0cb68f04a0207c
7c422338deb32e45e5291ca3ce0688b7d7b07115648347788cae5345e0834a6a5c4e52f342dae3c5633c0c0bafb
7133ae3ecbbf0ed948f1b4226d1d7a5344884dbff2ba320712853a13cc6ff9a8514b9dd0a3725e5004bb89476d0
5a0b7fdc3cbc368cfbd9ebe0bdbfa914f5817a208e0869364bc6361ca737630f7cbb30653f0e67098cf439b4f64
5e24063f53383cbcf0ddeb240b1a97fd040cce1da945f202f2ede94eab2
Відкритий ключ:
0x1f72cf6dfba134a6a3226ba50cc58b70d8114b83756c8ae81b8d5d9c71b02a0b25d5bcaeb94cb92f69d811fe0
203babdc0c1faa6eb2d75503554ab414621989fcfc127d9940d16dd7554345d2041256bae8ca1bc78bc55fe4665
677ff3f7d99085788c40e95f37645162c969151214cd2cf5938631743b8f0218e0a85d958d3aa4de79e8bad6507
2e6faf56e8efbe1e61090867bd5ea0760875fe80936d710dfbd4f88953e75b46e9285f1dedfb7f151c71cb93baa
fd6e682e88413b49f58cafcfad8d3778b2d99bd2334ae83d785cadfa343bf0e09df5435fa9375820a47796d2748
090b4d54a61b7c949886693a5194f31eb73cd6264ea9cac63e332356e89
```

7) Шифруємо повідомлення:

```
Шифртекст:
x:
0x4a2190c85c93fe4d3ca8f08fd1f5e93be45242eab560eaea2c9a372e406c41be7c5b4c31cc3c83f956dfe0526
b7774ec1a8c077f05743f901579f84438f8adcce65e31fec98687a9fbf1281ce3dc999880faa3d72a3b23b8279a
8a9248fa355fe3e3825c58a8dbd6971c9968a589143b3d50f14af7301223e1c8b4cf1106bf730fabb39e30de06c
2eefbe68e92200f9016688467623a5dd62ace1a912580dc115a32db222be79420757193f6e8236cb767ae8c629e
e8907cd90d701dfd27337851d7634668e09d5fb19fde707a25774a75dd03d610df0243466ae500b651d6d988c06
994dc9e340e53797008836a9e191999bd7c08cf88a0103385dd9ce2acee
y:
142d7f0e9681f020f576554d2d4b6afce919997ef7c8677ddbdc1a9e5c161bf3877e7081d376c9ea2a7e8ed2b02
d29593f6da96ae2625a1b0c9442b9cab959010a6908ed6cdc23a8bba989fbb4269f8144f57c25dc4a9d7034fa83
0848aa82bbc89ffe7da12629c73cb574bded0f9f3e948730ec030b9e3f623103cf1e4a1ab190893f693fc666e14
8e8423d2e03b92229c8c4c747331f1cd46fc60591a9c08c77d8b0a51cd77993dfae3458a6f4d968f193184917c1
f1b34f76f58dda35ff640fa7f83db3f92c00c73a9d7337913ed5a00305562e8958137717658206 25f9020fd20c3
57766370f4989fc10ccfe94f11c4cddb7e3414cb05f13897cc867de9342bcb0dc346876856ec3fd3b0deb40dc46
804902321b43ab4a58779e8180efe636e375308eb97f6e4679301ef698d7ba86532deb9729f338881bf89c120a9
be8009fcce0e84ba588d70729624f4f195cb67e5e7aaaf4e9f474ec03ee56d0c74106959030c82824fc44e3388b
9bab374866e0dd05c1ad34349b1235252030184b2d429ade283ff209d0440f0de84416f2c9d36955e446fb18fd5
47290bf8a4de2c8eaf1b984fba3f08b656c0bd5198051abe3e675fc2994876acfcb54d99ab5aa1e07cf5238ce29
70a662fee5205a197ca479f7033a72cce0abe005a450f2449a1bb11d85f3e0f72a43854588416a71287a0b1f932
5c64cd046d352e112604823
```

8) Розшифрування повідомлення:

```
Розшифроване повідомлення:  This message is intended to test the functionality of the
encryption and decryption functions of the El-Gamal algorithm. Please note that this
message is very long. The length of this message is 263 bytes, while the length of the
module is 256 bytes or 2048 bits.
```

9) Перевіряємо коректність розшифрування:

```
True
```

1.6 Середній час роботи алгоритму для 10 ітерацій

Алгоритм шифрування працює в середньому 0.02 секунди для 1000 ітерацій. Отримані результати щодо схеми підпису для різних довжин ключа проілюстровано в таб. 1.1

Розмір ключа, біт / Схема	1024	2048	4096
Підписання	0.13	0.67	0.7

Таблиця 1.1 – Середній час, с

✏ Посилання на код програми

ВИСНОВКИ

У даній лабораторній роботі було досліджено криптосистему з відкритим ключем Ель-Гамаля. Сама криптосистема включає в собі схему цифрового підпису і алгоритм шифрування. У цьому практикумі було розроблено реалізацію криптосистеми за допомогою бібліотеки OpenSSL та PyCryptodome.

Python, будучи універсальною мовою програмування, має кілька бібліотек і модулів, які взаємодіють з OpenSSL і надають криптографічні можливості.

У Python модуль hashlib надає загальний інтерфейс для різних алгоритмів безпечного гешування і дайджесту повідомлень. Він є частиною стандартної бібліотеки Python і не має прямого зв'язку з OpenSSL. Однак, реалізація деяких алгоритмів гешування в модулі hashlib може використовувати OpenSSL.

Модуль hashlib підтримує різні алгоритми гешування, такі як MD5, SHA-1, SHA-224, SHA-256, SHA-384 і SHA-512. У нашій схемі було використано саме SHA-256. При використанні цих алгоритмів модуль hashlib взаємодіє з бібліотекою OpenSSL, якщо вона присутня в системі.

PyCryptodome - це самостійний модуль Python, який надає можливість використовувати криптографічні функції. З бібліотеки PyCryptodome було використано функцію генерації простого числа заданого розміру.