**УДК 004.91:004.056.55.347.135.224**

# ПРОЕКТУВАННЯ ЗАХИСНИХ СИСТЕМ НА БАЗІ ФРАКТАЛЬНИХ АЛГОРИТМІВ

**Плотніков В.М.[1], Борцова Ю.В.[2]**

[1,2] Одеська національна академія харчових технологій, Одеса, Україна
ORCID: [1]http://orcid.org/0000-0001-6712-8357, [2]http://orcid.org/0000-0001-9000-2568,
E-mail: [2]bortsova.07@gmail.com



**DOI:**

***Анотація.*** *Для захисту конфіденційних даних від комп'ютерних злочинів користувач має подбати про безпеку своєї інформації власноруч, використовуючи існуючі сучасні програмні засоби. Одним з таких засобів є реалізація шифрування повідомлень за допомогою прикріплення цифрового підпису до даних. Для роботи криптосистем шифрування з відкритим ключем необхідно три алгоритми: алгоритм шифрування, алгоритм розшифрування та алгоритм генерації ключів. Одним з перспективних шляхів розвитку шифрування з відкритими ключами є використання моделі піднесення до великої степені дискретних логарифмів для генерування ключів, так званий алгоритм Діффі-Хеллмана. Рекурентні відношення, що становлять основу множини Мандельброта, забезпечують хаотичну поведінку та суттєву залежність процесу від початкових умов. Ці властивості дозволяють створити криптографічну систему, що здатна використовувати їх для вирішення поставлених задач.*

*Спроектована криптографічна система повінна поєднувати в собі засоби створення ключів, шифрування текстових повідомлень та генерації цифрового підпису. Протокол обміну ключами передбачає встановлення між учасниками спільного секретного ключа, який у подальшому можна використовувати для шифрування повідомлень тексту або зображень цифровим підписом.*

*Проаналізовано інструментальні засоби, за допомогою яких можна вирішити і реалізувати систему фрактальних алгоритмів для захисту інформації. В ході дослідження реалізовано програмний продукт мовою програмування C# у середовищі Visual Studio 2010. Система спроектована у рамках об'єктно-орієнтованого підходу до розробки програмних продуктів, тому вона використовує програмні класи для розподілення функціональності. Реалізований алгоритм має більшу кількість можливих ключів у порівнянні з поширеною на сьогодні схемою обміну ключами Діффі-Хеллмана. Великий розмір простору ключів робить важкими для реалізації атаки перебором, також відомі як метод «грубої сили».*

*Хаотичні властивості фрактального алгоритму не вимагають використання чисел великої розрядності, проте забезпечують високу якість шифрування. Економія часу на розрахунках дозволяє зменшити затрати ресурсів та підвищити продуктивність системи в цілому.*

***Abstract.*** *To protect confidential data from computer crimes, the user must take care of the security of their information themselves, using existing modern software. One such tool is the implementation of message encryption by attaching a digital signature to the data. Three algorithms are required for cryptosystems to operate with a public key: an encryption algorithm, a decryption algorithm, and a key generation algorithm. One of the promising ways to develop public key encryption is to use the model of raising highly discrete logarithms to generate keys, the so-called Diffie-Hellman algorithm. The recurrent relations that form the basis of the Mandelbrot set provide chaotic behavior and a significant dependence of the process on the initial conditions. These properties allow you to create a cryptographic system that can use them to solve problems.*

*The designed cryptographic system must combine the means of generating keys, encrypting text messages and generating a digital signature. The key exchange protocol provides for the establishment of a shared secret key between the participants, which can then be used to encrypt text or image messages with a digital signature.*

*The tools with which it is possible to solve and implement a system of fractal algorithms for information protection are analyzed. The study implemented a software product in the C # programming language in Visual Studio 2010. The system is designed as part of an object-oriented approach to software development, so it uses software classes to distribute functionality. The implemented algorithm has a larger number of possible keys in comparison with the Diffie-Hellman key exchange scheme common today. The large size of the key space makes it difficult to implement a bust, also known as the method of "brute*

*force". The chaotic properties of the fractal algorithm do not require the use of large numbers, but provide high quality encryption. Saving time on calculations allows you to reduce resource costs and increase system performance as a whole.*

**Ключові слова:** криптографія, стратегія безпеки, шифрування з відкритими ключами, алгоритм Діффі-Хеллмана
**Keywords:** cryptography, security strategy, public key encryption, Diffie-Hellman algorithm

**Вступ**

Відомо декілька класифікацій криптографічних алгоритмів, одна з них поділяє алгоритми в залежності від кількості ключів, що застосовуються в конкретному алгоритмі на безключові, які не використовують в обчисленнях ніяких ключів; одноключові, які працюють з одним секретним ключем; двохключові, в яких на різних стадіях роботи застосовуються два ключових параметри: секретний і відкритий ключі [1].

Ідея криптографії з відкритим ключем дуже тісно пов'язана з ідеєю односторонніх функцій, тобто таких функцій f(x), що за відомим x досить просто знайти значення f(x), тоді як визначення x з f(x) складно в сенсі теорії. Але сама одностороння функція марна в застосуванні: нею можна зашифрувати повідомлення, але розшифрувати не можна. Тому криптографія з відкритим ключем використовує односторонні функції з лазівкою. Лазівка — це якийсь секрет, який допомагає розшифрувати. Тобто існує такий *y*, що знаючи *f(x)*, можна обчислити *x*.

Перевага асиметричних шифрів перед симетричними шифрами полягає у відсутності необхідності попередньої передачі особистого ключа по надійному каналу, а число ключів у великих мережах в асиметричній криптосистемі значно менше, ніж у симетричній. Найбільш відомими з асиметричних криптосистем є RSA, DSA, схеми Рабина та Ель-Гамаля.

Схема обміну ключами Діффі-Хеллмана, винайдена в 1976 році при співпраці Уітфілда Діффі і Мартіна Хеллмана, під сильним впливом роботи Ральфа Меркля про систему розповсюдження публічних ключів, стала першим практичним методом для отримання спільного секретного ключа при спілкуванні через незахищений канал зв'язку. Однак цей алгоритм не вирішує проблему аутентифікації. Без додаткових засобів, один з користувачів не може бути впевнений, що він обмінюється ключами саме з тим користувачем, який йому потрібен.

Алгоритм передбачає, що обом абонентам відомі деякі два числа g і p, які не є секретними і можуть бути відомі також іншим зацікавленим особам. Для того, щоб створити невідомий більш нікому секретний ключ, обидва абонента генерують великі випадкові числа: перший абонент – число *a*, другий абонент – число *b*. Потім перший абонент обчислює значення $A = g^a mod p$ і пересилає його другому, а другий обчислює B = $g^b mod p$ і передає першому.

Передбачається, що зловмисник може отримати обидва цих значення, але не модифікувати їх (тобто у нього немає можливості втрутитися в процес передачі). На другому етапі, перший абонент на основі наявного в нього *a* і отриманого *b* обчислює значення $u^a mod p = g^{ab} mod p$, а другий абонент на основі наявного в нього *b* і отриманого *a* обчислює значення $a^b mod p = g^{ab} mod p$.

Отримане значення K = $g^{ab} mod p$ можна використовувати в якості секретного ключа, оскільки тут зловмисник зустрінеться з практично нерозв'язною (за розумний час) проблемою обчислення $g^{ab} mod p$, якщо числа *p, a, b* обрано досить великими.

$$K = a^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = b^a \bmod p.$$

Криптографічні системи з відкритим ключем використовують так звані необоротні функції, які мають наступну властивість:

– якщо відомо *x*, то *f(x)* обчислити відносно просто;

– якщо відомо *f(x)*, то для обчислення *x* немає простого (ефективного) шляху.

Під односпрямованістю розуміється не теоретична односпрямованість, а практична неможливість обчислити зворотне значення, використовуючи сучасні обчислювальні засоби, за доступний для огляду інтервал часу.

У криптографічній системі з відкритим ключем кожен учасник має як відкритий ключ (англ. public key), так і закритий ключ (англ. private key). Кожен ключ – це частина інформації. Кожен учасник створює свій відкритий і закритий ключ самостійно.

Відкритий ключ необхідний для шифрування документів і може бути опублікований для загального огляду. Закритий ключ має бути відомий тільки одержувачам зашифрованих повідомлень.

В основу криптографічної системи з відкритим ключем RSA та схеми Рабіна покладено завдання множення і розкладання складених чисел на прості множники, яка є обчислювально односпрямованим завданням. У цих схемах кожен ключ складається з пари цілих чисел.

Безпека схеми Рабіна спирається на складність пошуку квадратних коренів по модулю складеного числа. Складність цього алгоритму аналогічна проблемі розкладання на множники. Велика перевага криптосистеми Рабіна полягає в тому, що випадковий текст може бути відновлений повністю від зашифрованого тексту тільки за умови, що дешифрувальник здатний до ефективної факторизації відкритого ключа.

Головною незручністю практичного застосування криптосистеми Рабіна є те, що при розшифровці тексту виходить чотири різних повідомлення і потрібно застосувати додаткові зусилля для знаходження істинного вихідного тексту.

**Методи дослідження**

Рекурентні відношення, що становлять основу множини Мандельброта, забезпечують хаотичну поведінку та суттєву залежність процесу від початкових умов. Ці властивості дозволяють створити криптографічну систему, що здатна використовувати їх для вирішення поставлених задач.

Спроектована криптографічна система повінна поєднувати в собі засоби створення ключів, шифрування текстових повідомлень та генерації цифрового підпису. Протокол обміну ключами передбачає встановлення між учасниками спільного секретного ключа, який у подальшому можна використовувати для шифрування повідомлень тексту або зображень цифровим підписом.

**Опис експериментальної (аналітичної) складової дослідження**

Криптографічна система являє собою комплекс засобів, які дозволяють вирішувати одразу декілька завдань щодо захисту інформації. Після аналізу літературних джерел, можна спроектувати систему, що поєднує в собі засоби створення ключів, шифрування текстових повідомлень та генерації цифрового підпису.

Процедура генерації ключів вимагає завдання початкового значення, на підставі якого будуть проводитися обчислення. Визначимо величину m, як точку на комплексній площині. Обмежимо область визначення цієї точки її належністю до множини Мандельброта.

Формально множиною Мандельброта називають сукупність елементів $c$ поля комплексних чисел, для яких послідовність $z_n$, що визначена ітераційно не уходить в нескінченість.

$$z_{n+1} = z^2 + c, \quad (1)$$

де $z_0 = 0$.

Таким чином, зазначена вище послідовність може бути розкрита для кожної точки на комплексній площині наступним чином:

$$c = x + iy,\ z_0 = 0 \qquad z_1 = z^2 + c = x + iy$$
$$z_2 = z^2 + c = (x + iy)^2 + x + iy = x^2 + 2ixy - y^2 + x + iy = \quad (2)$$
$$= x^2 - y^2 + x + (2xy + y)I$$

Якщо переформулювати ці вирази у вигляді ітеративної послідовності значень координат комплексної площини $x$ і $y$, тобто замінивши на $x_n + i \cdot y_n$, і на $p + i \cdot q$, отримаємо:

$$x_{n+1} = x^2 - y^2 + pn$$
$$y_{n+1} = 2x_n y_n + q$$

Було доведено, що як тільки модуль $z_n$ виявиться більше 2 (або в термінах дійсної та уявної частин $(x^2 + y^2)^{1/2} > 2$), послідовність стане прагнути до нескінченності. Порівняння з цим числом дозволяє виділяти точки, які не потрапляють всередину множини. Для точок, що лежать всередині множини, послідовність не буде мати тенденції до нескінченності, тому після певного числа ітерацій розрахунок необхідно примусово завершити.

Візуально, всередині множини Мандельброта можна виділити нескінченну кількість елементарних фігур, причому найбільша в центрі являє собою кардіоїду (рисунок 1).

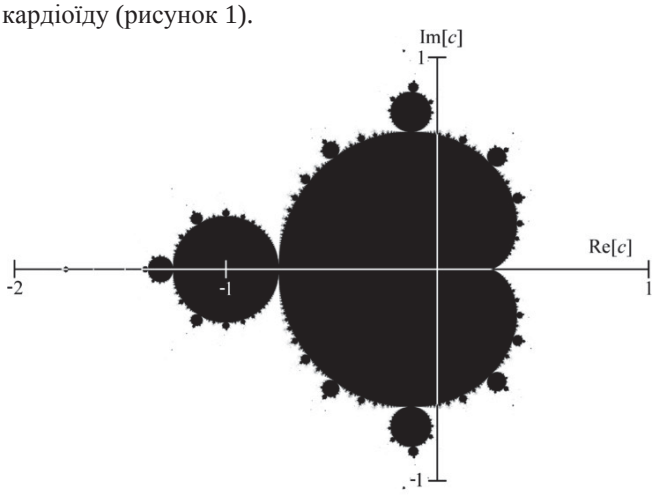


Рисунок 1 – Множина Мандельброта

Також є набір овалів, що торкаються кардіоїди, розмір яких поступово зменшується, прагнучи до нуля. Кожен з цих овалів має свій набір менших овалів, діаметр яких також прагне до нуля і т.д. Цей процес триває нескінченно, утворюючи фрактал.

Оберемо на комплексній площині довільні точки c і d, які задовольняють умовам (1). Параметр *c* є відкритим і може бути відомий громадськості, параметр *d* є закритим і повинен триматися в секреті.

Для генерації ключа використовується рекурсивна функція:

$$\begin{aligned} Z_0 &= c \\ Z_n &= c \cdot f(Z_{n-1}) \\ f(Z_{n-1}) &= c \cdot d \cdot Z_{n-1} \end{aligned} \qquad (3)$$

де n – кількість ітерацій, яка вибирається за наступним принципом:

– більше значення n забезпечує більшу видалення точки від початкових значень і підвищує криптостійкість системи;

– надвеликі значення n призведуть до тривалого обчислювального процесу, не підвищуючи при цьому надійності;

– рекомендується вибирати значення n в межах 15-30 ітерацій.

Обчислення рекурсивної функції зупиняється при досягненні заданого числа ітерацій. Результатом обчислення є значення відкритого фрактального ключа A, який необхідно передати другому учаснику.

Розраховане значення відкритого фрактального ключа A разом із параметром c передаються іншому учаснику по відкритому каналу зв'язку.

Другий учасник на підставі отриманого значення c та власних секретних параметрів d і n виконує свою частину обчислень фрактального ключа B, та передає його назад по каналу зв'язку. Таким чином, обидва учасники мають у розпорядженні фрактальний ключ напарника.

Вихідними даними для обчислення закритого ключа є параметрі c, d і n, обрані на попередньому етапі розрахунків, а також фрактальний ключ B, отриманий від другої сторони. Використовуючи їх, обидва учасники здатні згенерувати однаковий спільний секретний ключ.

Для генерації ключа використовується рекурсивна функція:

$$\begin{aligned} Y_0 &= B \\ Y_n &= c \cdot f(Y_{n-1}) \\ f(Y_{n-1}) &= d \cdot Y_{n-1} \\ Y_{n+1} &= c^n \cdot Y_n \end{aligned} \qquad (4)$$

Отриманий у результаті цих операцій ключ утворюється однаковим з обох сторон завдяки властивостям фракталів Мандельброта. У подальшому його можна використовувати у якості секретного ключа для шифрування тексту та цифрового підпису повідомлень.

Описана вище схема обміну ключами може також використовуватися в якості алгоритму шифрування з відкритим ключем. У цьому випадку загальна схема залишається аналогічною наведеній вище, але з невеликими відмінностями. Користувач A не передає значення c і A користувачу B безпосередньо, а публікує їх заздалегідь в якості свого відкритого ключа. Користувач B виконує свою частину обчислень, після чого шифрує повідомлення симетричним алгоритмом, використовуючи K як ключ, і передає шифротекст Користувачу A разом зі значенням B.

Шифрування тексту здійснюється шляхом його складання з ключем, отриманим на попередньому етапі. Отриманий у разі перетворення шифртекст може бути розшифрований другим учасником у разі наявності у нього того ж самого ключа.

Для того, щоб текст можна було скласти з ключем, його необхідно попередньо привести до форми комплексного числа. Для цього використовується наступний алгоритм:

а) строка розбивається на послідовність текстових символів;

б) кожен символ перетворюється в його двійкове подання;

в) отримане двійкове число ділиться на праву і ліву частини;

г) кожна частина перетворюється в десяткову форму;

д) виконується нормування обох частин з метою отримати значення, що лежать у діапазоні [0,1];

е) з двох частин формується комплексне число, приймаючи отримані величини за дійсну і уявну частини відповідно.

**Обґрунтування та опис моделі та методу дослідження**

На відміну від асиметричних алгоритмів шифрування, в яких зашифрование проводиться за допомогою відкритого ключа, а розшифрування – за допомогою закритого, в схемах цифрового підпису підписування проводиться із застосуванням закритого ключа, а перевірка – із застосуванням відкритого.

Загальновизнана схема цифрового підпису охоплює три процеси:

- генерація ключової пари. За допомогою алгоритму генерації ключів, описаного раніше, з набору можливих закритих ключів вибирається закритий ключ та обчислюється відповідний йому відкритий ключ;

- формування підпису. Для заданого електронного документа за допомогою закритого ключа обчислюється підпис;

- перевірка (верифікація) підпису. Для даних документа та підпису за допомогою відкритого ключа визначається дійсність підпису.

Для того, щоб використання цифрового підпису мало сенс, необхідно виконання двох умов:

- верифікація підпису повинна проводитися відкритим ключем, що відповідає саме того закритому ключу, який використовувався при підписанні;
- без володіння закритим ключем має бути обчислювально складно створити легітимний цифровий підпис.

У більшості ранніх систем ЕЦП використовувалися функції з секретом, які за своїм призначенням близькі до односторонніх функцій. Такі системи уразливі до атак з використанням відкритого ключа, оскільки, вибравши довільну цифровий підпис і застосувавши до неї алгоритм верифікації, можна отримати вихідний текст.

Щоб уникнути цього, у схемі цифрового підпису використовується хеш-функція, тобто обчислення підпису здійснюється не щодо самого документа, а щодо його хешу. У цьому випадку в результаті верифікації можна отримати тільки хеш вихідного тексту, отже, якщо використовується хеш-функція криптографічно стійка, то отримати вихідний текст буде обчислювально складно, а значить атака такого типу стає неможливою.

Цифровий підпис являє собою зашифровану за допомогою фрактального секретного ключа користувача хеш-суму повідомлення. У подальшому підпис може бути переданий разом із даними по відкритому каналу зв'язку.

Схема перевірки ЕЦП повідомлення, що здійснюється одержувачем, складається з наступних етапів.

На першому з них проводиться розшифрування блоку ЕЦП за допомогою фрактального відкритого ключа відправника. Потім обчислюється хеш-функція електронного документа. Результат обчислення порівнюється з результатом розшифрування блоку ЕЦП.

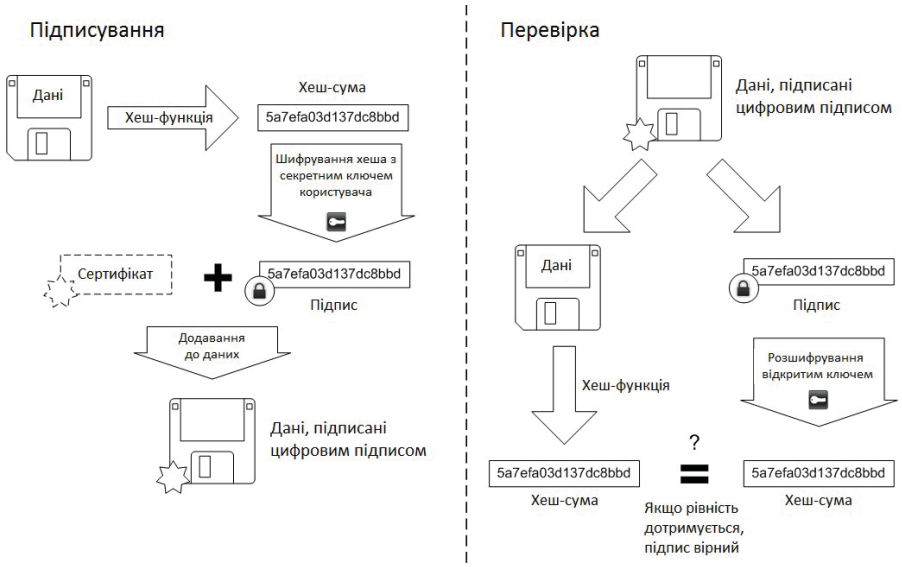


Рисунок 2 – Схема роботи алгоритму цифрового підпису

У разі збігу приймається рішення про відповідність ЕЦП повідомлення заявленим даними. Розбіжність результатів розшифрування з результатом обчислення хеш-функції повідомлення може пояснюватися такими причинами:

- у процесі передачі по каналу зв'язку була втрачена цілісність електронного документа;
- при формуванні ЕЦП був використаний не той (підроблений) секретний ключ;
- при перевірці ЕЦП був використаний не той відкритий ключ (в процесі передачі по каналу зв'язку або при подальшому його зберіганні ключ був модифікований або підмінений).

**Програмна підтримка дослідження**

Реалізацію даної системи вирішено виконувати мовою програмування C# у середовищі Visual Studio. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники, атрибути, події, властивості, винятки. Вона успадкувала від Java концепції віртуальної машини, байт-коду і більшої безпеки вихідного коду програм, а також врахувала досвід використання програм на Java. Технологія об'єктно-орієнтованого програмування використовує механізм пересилки повідомлень та класи, що організовані у ієрархію успадкування. Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET.

Головний файл програми – Program.cs, що є стандартом для програм, які розроблені на C#. Розташований в ньому метод Main називається точкою входу і викликається першим при запуску програми. Код у файлі включає візуальні стилі і створює об'єкт класу formMain, який представляє головне вікно програми.

Арифметичні дії виконуються аналогічно до дій з многочленами, але з урахуванням рівності i2=–1. Нехай $z_1 = a + bi$ та $z_2 = c + di$ – комплексні числа. Тоді можна визначити наступні операції:

– складання

$$z_1 + z_2 = (a + bi) + (c = di) = (a + c) + (b + d)i$$

— віднімання

$$z_1 - z_2 = (a + bi) - (c + di) = (a - c) + (b - d)i$$

— множення

$$z_1 z_2 = (a + bi)(c + di) = ac + adi + bci + bdi^2 =$$
$$= ac + adi + bci - bd = (ac - bd) + (ad - bc)i$$

— ділення

$$\frac{z_1}{z_2} = \frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{ac+bd+(dc-ad)i}{c^2+d^2}$$

Для комплексних чисел певним чином визначають також інші операції, наприклад, зведення до довільного комплексного степеня, логарифмування, знаходження синуса, косинуса тощо.

Перевантаження операторів – один із засобів реалізації поліморфізму, що полягає в можливості одночасного існування в одній зоні видимості декількох різних варіантів застосування операторів, що мають одне й те саме ім'я, але різні типи аргументів, до яких вони застосовуються. Можливість перевантаження операторів в мові C# дозволяє використовувати коротку і виразну форму запису типових операцій, значення якої важко переоцінити:

– складання

```
public static ComplexNum operator +(ComplexNum op1, ComplexNum op2)
{
      return new ComplexNum(op1.real + op2.real, op1.imag + op2.imag);
}
```

– віднімання

```
public static ComplexNum operator -(ComplexNum op1, ComplexNum op2)
{
      return op1 + (-op2);
}
```

– множення

```
public static ComplexNum operator *(ComplexNum op1, ComplexNum op2)
{
      double real = op1.real * op2.real - op1.imag *
      op2.imag; double imag = op1.imag * op2.real +
      op1.real * op2.imag;

      return new ComplexNum(real, imag);
}
```

– ділення

```
public static ComplexNum operator /(ComplexNum op1, ComplexNum op2)
{
      double den = Math.Pow(op2.real, 2) + Math.Pow(op2.imag,
      2); double real = (op1.real * op2.real + op1.imag *
      op2.imag) / den; double imag = (op1.imag * op2.real -
      op1.real * op2.imag) / den;

      return new ComplexNum(real, imag);
}
```

– зведення до довільного степеня

```
public static ComplexNum operator ^(ComplexNum c, int n)
{
      ComplexNum x = c;

      if (n == 0)
            return new ComplexNum(1);

      for (int i = 1; i < n;
            i++) c = c * x;

      return c;
   }
```

Модулем (абсолютною величиною) комплексного числа називається довжина радіус-вектора відповідної точки комплексній площині (або, що те ж, відстань між точкою комплексної площини, відповідної цього числа, і початком координат).

Модуль комплексного числа z позначається |z| і визначається виразом:

$$z = (x^2 + y^2)^{½},$$

де $z = x+iy$

Кут $\varphi$ (в радіанах) радіус-вектора точки, що відповідає числу $z$, називається аргументом числа $z$ і позначається *Arg(z)*.

У програмі для отримання модулю та аргументу комплексного числа передбачені такі аксесори:

```
public double Modulus
{
      get { return Math.Sqrt(real * real + imag * imag); }
}

public double Argument
{
      get { return Math.Atan2(real, imag); }
}
```

Бібліотека System.Math містить потрібні методи для здобуття квадратного кореню та арктангенсу.

Звертання до класу Fractal відбувається для перевірки належності точки до множини Мандельброта, а також для розрахунку значень рекурсивних функцій Mandelfn та Juliafn, які використовуються у процесі генерації ключів.

Визначити точки, які потрапляють всередину множини Мандельброта дозволяє порівняння, чи не перевищує модуль координат точки значення 2. Для точок, що лежать всередині множини, послідовність не буде мати тенденції до нескінченності і ніколи не досягне цього числа, тому після певного числа ітерацій розрахунок необхідно примусово завершити. Максимальне число ітерацій, після яких число вважається потрапившим всередину множини, задається в програмі.

```
public bool Check(int n, ComplexNum k)
{
   bool b = false;
   ComplexNum    m    =    new
 ComplexNum(0); for (int i = 0;
 i < n; i++)
   {
      m = m * m + k;
   }

 if (m.Modulus <
    2)    b    =
    true;

   return b;
}
```

У класі Encoder описані методи, за допомогою яких можна представити текстові дани у вигляді комплексного числа та навпаки:

- public ComplexNum encodeString(string str) приймає у якості параметру строку тексту та перетворює її на послідовність ASCII-кодів, яка поділяється на дві рівні частини, одна з яких приймається за дійсну частину комплексного числа, друга – за уявну;
- public string decodeString(ComplexNum cnum) приймає комплексне число у якості параметра та перетворює його на набір символів, вважаючи поєднання дійсної та уявної частин послідовністю ASCII-кодів. Повертає строку тексту, що відповідає кодам.

У разі введення даних користувачем вручну, необхідно перетворити їх з текстового вигляду в комплексне число. Для цього у введеному тексті за допомогою регулярного виразу виконується пошук двох дробових чисел за заданим шаблоном. Для роботи з регулярними виразами в C# передбачений клас System.Text.RegularExpressions.

```
public ComplexNum parseString(string str)
{
    MatchCollection matches = Regex.Matches(str, @"-
    ?\s?\d+.\d+E[\+-]\d+"); string[] values = new
```

```
    string[matches.Count];
    for (int i = 0; i < matches.Count; i++)
    {
        values[i] = matches[i].Value;
        values[i] = values[i].Replace(" ", string.Empty);
    }
    double real =
    double.Parse(values[0]); double imag
    = double.Parse(values[1]);
    return new ComplexNum(real, imag);
}
```

Для пошуку збігів використовується метод Regex.Matches(). Цей метод повертає масив класу MatchCollection, що містить знайдені збіги і ряд іншої інформації. Доступ до знайденого тексту виробляється за допомогою конструкції matches[i].Value. Отримати кількість знайдених збігів можна з властивості matches.Count.

**Висновки**

Проаналізовано інструментальні засоби, за допомогою яких можна вирішити і реалізувати систему фрактальних алгоритмів для захисту інформації. В ході дослідження реалізовано програмний продукт мовою програмування C# у середовищі Visual Studio 2010. Система спроектована у рамках об'єктно-орієнтованого підходу до розробки програмних продуктів, тому вона використовує програмні класи для розподілення функціональності. Реалізований алгоритм має більшу кількість можливих ключів у порівнянні з поширеною на сьогодні схемою обміну ключами Діффі-Хеллмана. Великий розмір простору ключів робить важкими для реалізації атаки перебором, також відомі як метод «грубої сили».

Хаотичні властивості фрактального алгоритму не вимагають використання чисел великої розрядності, проте забезпечують високу якість шифрування. Економія часу на розрахунках дозволяє зменшити затрати ресурсів та підвищити продуктивність системи в цілому.

**Перелік використаних джерел**

[1] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке С / Б. Шнайер. – М.: Издательство ТРИУМФ, 2002.– 816 с.
[2] Мандельброт Б. Фрактальная геометрия природы / Б. Мандельброт. – М.: Институт компьютерных исследований, 2002. – 656 с.
[3] Кроновер Р. Фракталы и хаос в динамических системах. Основы теории / Р. Кроновер. – М.: Постмаркет, 2000. – 350 с.
[4] Al-Saidi N. Using IFS as an Encryption method / N. Al-Saidi, M. Rushdan // International Conference on Education Technology and Computer. – 2009. – p. 275-278.
[5] Al-Saidi N. A New Public Key Cryptosystem Based on IFS / N. Al-Saidi, M. Rushdan // International Journal of Cryptology Research. – 2010. – p. 1-13.
[6] Kumar S. Public key cryptography system using Mandelbrot sets / S. Kumar // Military Communications Conference. – 2006.
[7] Alia M. New Key Exchange Protocol Based on Mandelbrot and Julia Fractal Sets /M. Alia, A. Samsudin // IJCSNS International Journal of Computer Science and Network Security. – 2007. – p. 302-307.
[8] Alia M. A New Public-Key Cryptosystem Based on Mandelbrot and Julia Fractal Sets / M. Alia, A. Samsudin // Asian Journal of Information Technology. – 2007. – p. 567-575.
[9] Alia M. A New Digital Signature Scheme Based on Mandelbrot and Julia Fractal Sets / M. Alia, A. Samsudin // American Journal of Applied Sciences. – 2007. – p. 848-856.
[10] 10. Ojha D. B. An Approach for Embedding Elliptic Curve in Fractal Based Digital Signature Scheme / D. B. Ojha, Ms. Shree, A. Dwivedi, A. Mishra // Journal of Scientific Research. – 2011. – p. 75-79.

**References**

[1] Shnayer B. Prikladnaya kriptografiya. Protokolyi, algoritmyi, ishodnyie tekstyi na yazyike C. Izdatelstvo TRIUMF, 2002. 816 p.
[2] Mandelbrot B. Fraktalnaya geometriya prirodyi. Institut kompyuternyih issledovaniy, 2002. 656 p.
[3] Kronover R. Fraktalyi i haos v dinamicheskih sistemah. Osnovyi teorii. Postmarket, 2000. 350 p.
[4] Al-Saidi N., Rushdan M. Using IFS as an Encryption method. *International Conference on Education Technology and Computer*, 2009. Pp. 275-278.
[5] Al-Saidi N., Rushdan M. A New Public Key Cryptosystem Based on IFS. *International Journal of Cryptology Research*, 2010. Pp. 1-13.
[6] Kumar S. Public key cryptography system using Mandelbrot sets. *Military Communications Conference*, 2006.
[7] Alia M., Samsudin A. New Key Exchange Protocol Based on Mandelbrot and Julia Fractal Sets. *IJCSNS International Journal of Computer Science and Network Security*, 2007. Pp. 302-307.

[8] Alia M., Samsudin A. A New Public-Key Cryptosystem Based on Mandelbrot and Julia Fractal Sets. *Asian Journal of Information Technology*, 2007. Pp. 567-575.
[9] Alia M., Samsudin A. A New Digital Signature Scheme Based on Mandelbrot and Julia Fractal Sets. *American Journal of Applied Sciences*, 2007. Pp. 848-856.
[10] Ojha D. B., Shree Ms., Dwivedi A., Mishra A.An Approach for Embedding Elliptic Curve in Fractal Based Digital Signature Scheme. *Journal of Scientific Research*, 2011. Pp. 75-79.