Міністерство освіти і науки України
Національний університет «Чернігівський колегіум» імені Т.Г.Шевченка

Кваліфікаційна робота

Освітнього ступеня «магістр»
на тему

Використання чисел Фібоначчі для кодування інформації

Виконав:
студент 2 курсу, групи 62
спеціальності 111 Математика
Семирод О. В.
Науковий керівник:
к. п. н., доцент Нак М.М.

Чернігів − 2020

Роботу подано до розгляду «_____» _____ 20__ року.

Студент _____ _____
(підпис) (прізвище та ініціали)

Науковий керівник _____ _____
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Кваліфікаційна робота розглянута на засіданні кафедри математики та економіки протокол №____ від «_____» _____ 20__ року
Студент допускається до захисту даної роботи в екзаменаційній комісії

Завідувач кафедри _____ _____
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

З поняттям шифрування даних ми знайомі з дитинства. Саме в дитячому віці ми вперше в формі гри придумуємо свої методи перетворення тексту, щоб дорослі не здогадалися що ми написали. Ми прагнули зберегти таємницю. Але, виявляється, хоча ми і шифруємо свої повідомлення, але такий погляд на речі досить поверхневий. Наука яка займається дослідженнями різних кодів називається криптографією. І вона древніша чим можна уявити. Проблемою методів шифрування займалося і займається багато науковців від античних Цезаря і Полібія до сучасних Леонарда Адлемана, Рональда Рівеста та Аді Шаміра. А головна мета криптології знаходити нові методи шифрування, так щоб їх ніхто не міг розшифрувати, зрозуміло крім адресата та власника повідомлень. Як бачимо це поняття далеко від дитячого, коли наші батьки наперед знали всі наші хитрощі.

Криптографія один з найцікавіших, найактуальніших та найскладніших розділів як математики так і інформатики. Взагалі, кажучи науковою мовою, криптографія це наука – яка розробляє та удосконалює нові методи шифрування будь якої інформації. Останнім часом ця наука повністю перебралася в цифрову сферу, адже весь документообіг, фінансові операції і навіть звичайні листи тепер доступні в комп’ютері завдяки мережі інтернет. Але постає питання, як захистити інформацію від чужого втручання, як захистити фінансові дані, як зберегти в таємниці свої листи, як зберегти від розголошення свої дані? Вся ця інформація вразлива при передачі по каналах зв’язку. Для цього і розробляються нові методи захисту даних, щоб коли зловмисник якимось чином перехопив повідомлення, він не зміг отримати з нього ніякої інформації.

Числа Фібоначчі широко використовують в криптології, наприклад на їх основі побудований всім відомий алгоритм RSA. Це зумовлено тим, що їх можна легко отримати та вони мають багато властивостей, які можна використовувати в різних алгоритмах не тільки шифрування.

Основи досліджуваного алгоритму шифрування в своїх працях привів В. Хогат, Стахов А. Але запропонував використовувати дані алгоритми в криптології Ю. Грицюк разом зі своїми студентами. В своїх працях він запропонував використовувати матриці, як вихідні дані та матриці з числами Фібоначчі, як ключі. Але в своїй роботі він брав лише числову інформацію.

Розробляючи наш метод шифрування, на основі робіт Ю. Грицюка, велика увага була приділена можливості кодуванню за допомогою чисел Фібоначчі будь-якої текстовоїї інформації, а не тільки чисел. А також виправленню помилок при пошкодженому прийнятому повідомленні.

Актуальність дослідженої проблеми полягає в тому, що оптимальних методів в криптографії не існує. В залежності від цілей які ставить задача, потрібно підбирати свій метод шифрування. Розроблений метод шифрування досить універсальний, але він особливо буде корисним для обміну зашифрованими повідомленнями в реальному часі.

Об’єкт дослідження – методи, засоби та алгоритми криптології.

Предмет дослідження – матричний метод шифрування за допомогою чисел Фібоначчі.

Мета дослідження – розробити ефективний метод шифрування використовуючи числа Фібоначчі.

Відповідно до мети були поставлені такі **завдання:**

1) Вивчити сучасні тенденції розвитку криптології та використання в ній чисел Фібоначчі.

2) Розробити алгоритм шифрування за допомогою чисел Фібоначчі.

3) Написати програму, яка реалізує розроблений алгоритм шифрування, щоб переконатися в його можливостях.

Наукова новизна – розроблений метод шифрування, який буде корисним для обміну зашифрованими повідомленнями в реальному часі.

Практичне значення одержаних результатів полягає в тому, що розроблений криптографічний метод можливо використовувати для

передавання інформації в таких сферах як банківська справа чи інші де потрібна конфіденційність повідомлень.

Апробація результатів дослідження представлена на І Всеукраїнській науково-методичній інтернет – конференції студентів, аспірантів та молодих вчених «Розвиток інтелектуальних умінь і творчих здібностей учнів та студентів у процесі навчання дисциплін природничо-математичного циклу “ІТМ*плюс-2020 Форум молодих дослідників”».

Структура роботи – логіка дослідження зумовила структуру дипломної роботи: вступ, три розділи, висновки, список використаних джерел із 36 найменувань. Загальний обсяг 67 сторінок.

Перший розділ присвячений теоретичним основам проблеми дослідження і в свою чергу містить три пункти. В першому описується появу чисел Фібоначчі. В другому описуються особливості та властивості чисел Фібоначчі. В третьому пункті дається перелік базових понять криптографії та їх визначення.

Другий розділ присвячений самому методу шифрування за допомогою чисел Фібоначчі та містить чотири пункти. В першому описується суть методу шифрування. В другому описується алгоритми вибору ключів для шифрування. Третій пункт присвячений кодуванню не числової (символьної) інформації та більш детальному розбору самого алгоритму. В четвертому пункті розглядаються особливості виявлення та виправлення помилок при прийомі пошкодженого повідомлення.

Третій розділ присвячений алгоритмам та програмній реалізації матричного методу шифрування на мові Java Script.

РОЗДІЛ 1. Теоретичні основи проблеми дослідження

1.1. Історія виникнення чисел Фібоначчі

Європа в середньовіччі була дуже далекою від розвитку науки. Нескінченні війни, Хрестові походи, церковні заборони, інквізиція та взагалі культура того часу не сприяла розвитку будь-яких галузей знань. Але в ці темні часи був період який був репетицією майбутньої епохи «Відродження». Фрідріх II імператор «Священної Римської Імперії» (Германія), який правив починаючи з 1220 року, був прихильником не рицарів, а вчених. Він відмовився від набридливих лицарських турнірів і запровадив нові змагання – математичні де замість грубого насилля, ударів та кровопролиття учасники вигадували нові задачі, обмінювалися ними та намагалися їх розв'язувати.

Італія в той час входила до складу «Священної Римської Імперії», а сам Фрідріх навчався в Неаполі, тому не дивно, чому він обожнював італійських вчених і хто перемагав на цих «математичних турнірах». Але найбільше перемог в таких змаганнях отримував Леонардо Пізанський, він навіть мав неофіційний титул «Лицар чисел» [35].

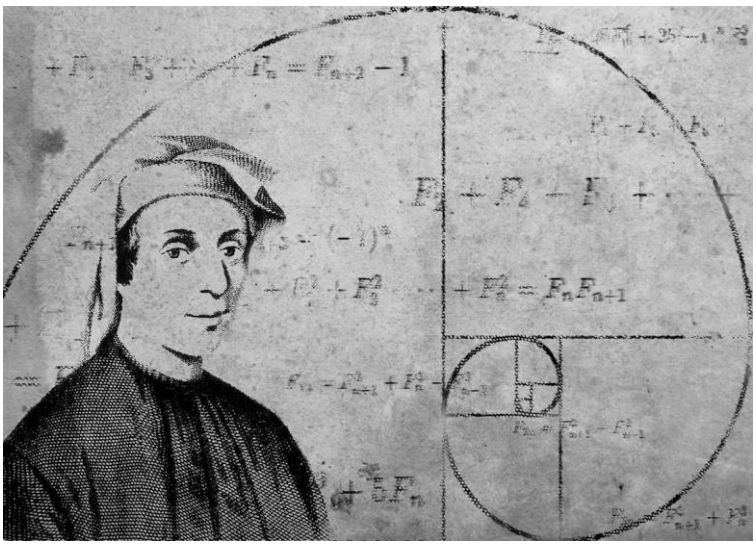

Рис 1.1.1 – Леонардо Пізанський (Фібоначчі)

Точна дата народження Леонардо Пізанського невідома, є декілька варіантів — 1170 та 1180, але точно відомо, що народився він в місті Піза. Його батько Гільєрмо був торговцем і багато мандрував. Сам Леонардо також об'їздив все Середземномор'я. Він побував в північній Африці та на

Близькому сході, де навчався в арабських вчених. І він скрізь збирав знання з математики.

Цікаво те, що прізвище Фібоначчі з'явилося приблизно через 250 років після смерті видатного математика. Вперше воно згадано в записках нотаріуса в 1506 році, де він записаний як «Леонардо Фібоначчі». Походження самого прізвища точно невідоме, є декілька версій. Але найбільш відома полягає в тому, що Фібоначчі так прозвали через його батька, який мав прізвисько Bonaccі (Добромисний), а його назвали filius Bonaccі — син добромисного [35].

Фібоначчі користуючись заступництвом Фрідріха II, видає декілька наукових трактатів: велика «Книга абака», написана в 1202 році, але в першому варіанті вона не дійшла, а вивчаємо її у варіанті, який був написаний в 1228 р.; «Практики геометрії» (1220 р.); «Книги квадратів» (1225 р.). За допомогою цих книг, вивчали математику аж до 17 століття.

У «Практиці геометрії» Фібоначчі застосував до розв’язування геометричних задач методи алгебри. В «Книзі квадратів» він розв’язав деякі завдання на квадратні рівняння.

Найбільший інтерес представляє «Книга абака». Цей трактат являє собою об'ємну роботу, що містить майже всі арифметичні і алгебраїчні відомості того часу і зіграла значну роль в розвитку математики в Західній Європі протягом кількох наступних століть. Зокрема, саме по цій книзі європейці познайомилися з арабськими цифрами. У ній Фібоначчі вперше в Європі привів від'ємні числа, які розглядав, як «борг», дав прийоми добутку кубічних коренів, привів «числа Фібоначчі» [3].

Матеріал в «Книзі абака» пояснюється на прикладі великого числа задач, які становлять значну частину цього тракту. Розглянемо одну з них: «На певній території, яка з усіх сторін обгороджена парканом, фермер, випустив пару кролів. Скільки пар кролів буде у вольєрі через один рік, якщо відомо, що кролики почнуть розмножуватися на другий місяць проживання в цьому

вольєрі, в тому числі і після свого народження, а за один місяць дають в потомство одну пару дітей? (вважається що кролі не помирають)» [3].

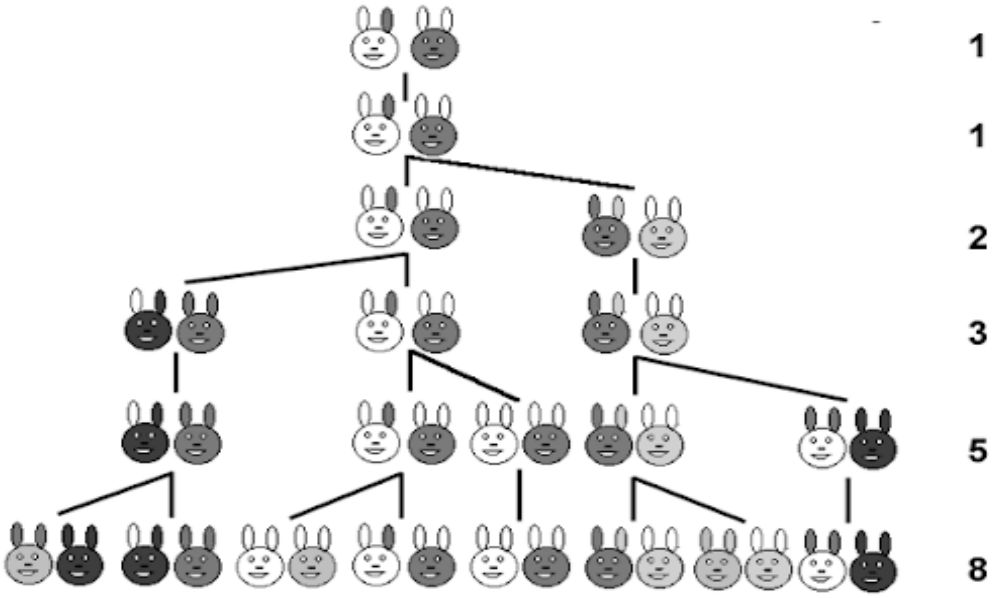

Рис 1.1.2 – Схема народження кроликів в задачі Фібоначчі

Розглянемо поетапне розв'язання даної задачі:

1) Після першого місяця, за умовою, в нас залишиться одна пара кроликів.

2) Після другого місяця перша пара кроликів дасть потомство у вигляді другої пари кроликів.

3) Після третього місяця тільки перша пара дасть потомство, а друга, за умовою, ще ні. Тобто буде три пари кроликів.

4) Після четвертого місяця, перші дві пари кроликів дадуть, потомство. Загалом п'ять пар.

5) Перші три пари дадуть потомство, загалом вісім пар. І так далі.

(Відповідь 144 кролика)

Тобто кількість пар кроликів утворюють послідовність чисел, яку називають послідовністю Фібоначчі. Її суть полягає в тому, що перші два члени це 1 та 1, а будь-яке наступне число є сумою двох попередніх.

Числа Фібоначчі тісно пов'язано із золотим перетином, відомою математичною константою (φ =1,6180339887...). Якщо якийсь член послідовності Фібоначчі, поділити на попередній то отримаємо число, яке

наближається до φ. Наприклад: $\frac{10946}{6765} \approx 1{,}6180339985$. Тобто чим більше F_n тим відношення $\frac{F_n}{F_{n-1}}$ ближче до золотого перетину [34].

Завдання, пов'язані з числами Фібоначчі, наводяться в багатьох популярних виданнях з математики, розглядаються на заняттях шкільних і студентських математичних гуртків, пропонуються на математичних олімпіадах. Була встановлена досить велика кількість раніше невідомих властивостей чисел Фібоначчі, а до самих чисел сьогодні істотно зріс інтерес. Значне число пов'язаних з математикою людей в різних країнах долучилося до цікавого хоббі «фібоначчізма». Найбільш переконливим свідченням цьому може служити журнал «The Fibonacci Quarterly», що видається в США з 1963 р. Пропорції Фібоначчі завдяки зусиллям багатьох ентузіастів виявляються в найнесподіваніших областях знання, через золотий перетин вдається зв'язати між собою абсолютно різні теорії і явища, що свідчить про фундаментальну роль теорії чисел Фібоначчі в природознавстві і в гуманітарних науках [34].

1.2. Числа Фібоначчі

Послідовність Фібоначчі [3] – це числова послідовність, яка задана рекурентним співвідношенням:

$F_0 = 0, F_n = 1, ..., F_n = F_{n-1} + F_{n-2}; n > 1$.

А самі члени ряду називаються числами Фібоначчі. Тобто послідовність матиме вигляд:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711 …

Скориставшись формулою яка задає числову послідовність, її можна розширити на від'ємні індекси:

$$F_{n+2} = F_{n+1} + F_n \Rightarrow F_n = F_{n+1} + F_{n+2}$$

...-21, 13, -8, 5, -3, 2, -1, 1, 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

Тобто, як бачимо $F_{-n} = (-1)^n \cdot F_n$.

Розглянемо деякі основні властивості чисел Фібоначчі [3; 34]:

1) НСД двох довільних чисел Фібоначчі дорівнює числу Фібоначчі під номером НСД номерів цих чисел: $\left(F_m, F_n\right) = F_{(m,n)}$.

Наслідки:

- Число F_n може бути простим тільки для простого n (крім $n = 4$). Навпаки ні.
- Кожен третій член послідовності Фібоначчі ділиться на 2.
- Кожен четвертий член послідовності Фібоначчі ділиться на 3.
- Кожен п’ятнадцятий член послідовності Фібоначчі ділиться на 10.
- Два сусідні члени послідовності Фібоначчі взаємно прості.
- $F_m \vdots F_n \Leftrightarrow m \vdots n$ (крім $n=2$).

2) $F_1 + F_2 + \ldots + F_n = F_{n+2} - 1$.

3) $F_1 + F_3 + F_5 + \ldots + F_{2n-1} = F_{2n}$.

4) $F_2 + F_4 + F_6 + \ldots + F_{2n} = F_{2n+1} - 1$.

5) $F_{n+1} \cdot F_{n+2} - F_n \cdot F_{n+3} = (-1)^n$.

6) $F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$.

7) $F_1^2 + F_2^2 + \ldots + F_n^2 = F_n \cdot F_{n+1}$.

8) $F_n^2 + F_{n+1}^2 = F_{2n+1}$.

9) $F_{2n} = F_{n+1}^2 - F_{n-1}^2$.

10) $F_{3n} = F_{n+1}^3 + F_n^3 - F_{n-1}^3$.

11) $F_{n+1} = C_n^0 + C_{n-1}^1 + C_{n-2}^2 + \ldots$, де C_n^k біноміальні коефіцієнти (комбінації).

12) Твірна функція для послідовності Фібоначчі:

$$x + x^2 + 2x^3 + 3x^4 + 5x^5 + \ldots = \sum_{n=0}^{\infty} F_n x^n = \frac{x}{1 - x - x^2}.$$

13) $F_{n+1} = \det \begin{pmatrix} 1 & 1 & 0 & \dots & 0 \\ -1 & 1 & 1 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix}$, де матриця $n \times n$.

Тобто числа Фібоначчі легко можна отримати з рекурентної формули, а також вони мають багато властивостей, які можуть бути використані в алгоритмах.

1.3. Криптологія, кодування, шифрування та інші базові поняття

Для подальшого розкриття теми розглянемо базові поняття на які надалі будемо спиратись. Спочатку визначимося з такими поняттями як криптологія.

Криптологія (від грец. κρυπτος — прихований та λογος — наука) – наука яка вивчає та розробляє методи та засоби математичного перетворення даних для його захисту [1; 2].

В свою чергу криптологія поділяється на три галузі:

- Криптографія – наука яка вивчає методи перетворення тексту для його захисту.
- Стенографія – наука яка вивчає символи та їх запис завдяки яким шифрується текст.
- Криптоаналіз – наука яка вивчає та розробляє алгоритми та методи розшифрування захищених текстів без знання ключів.

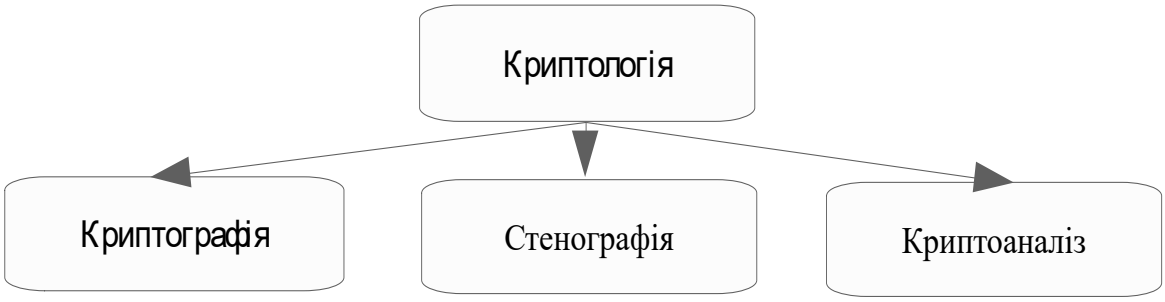

Схема 1.3.1 – Галузі криптології

Шифрування даних це процес перетворення тексту в зашифровані або перетворення зашифрованих даних в звичайні. Тобто дане поняття включає в

себе процеси шифрування та розшифрування даних. Сам же процес шифрування повинен забезпечити конфіденційність даних – неможливість отримання закодованих даних без попереднього розшифрування.

Всі операції з даними відбувається за допомогою шифру – сукупності математичних моделей, алгоритмів та правил за допомогою яких здійснюється шифрування. Головна характеристика всіх шифрів це криптостійкість – отримання розшифрованого тексту без знань про ключ. Ключем називають певну послідовність символів, яка використовується для того щоб зашифрувати або розшифрувати текст.

Алгоритми шифрування, зазвичай, спираються на такі математичні галузі як [12]:

- теорія ймовірностей;
- математична статистика;
- алгебра;
- теорія чисел;
- теорія алгоритмів.

Розглянемо, які бувають алгоритми шифрування [1]. Перш за все зазначимо, що алгоритмами шифрування називають певні математичні правила, за допомогою яких створюються або зламуються зашифровані дані.

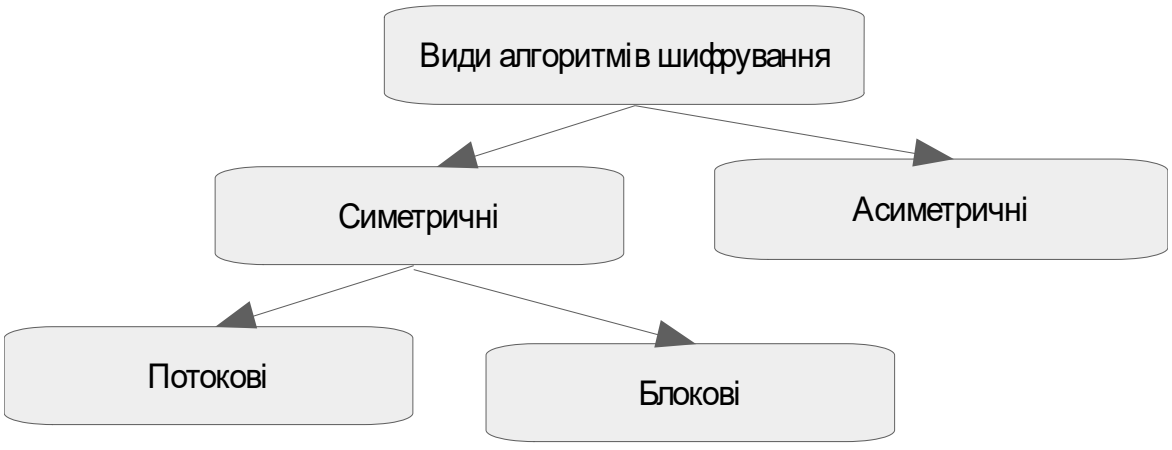

Схема 1.3.2 — Види алгоритмів шифрування

Всі алгоритми поділяються на дві великі групи — симетричні (для яких ключ шифрування і розшифрування співпадає) та асиметричні (для шифрування і розшифрування використовують два різних ключі).

В свою чергу симетричні алгоритми поділяються на потокові (виконується шифрування по-символьно, тобто символи залишаються на своїх місцях в тексті, але замінюються на інші позначки відповідно до алгоритмів шифрування) та блокові (все повідомлення розбивається на блоки, та ці блоки кодуються).

Переваги потокових шифрів:

- простота в алгоритмах шифрування;
- висока швидкість обробки даних;
- невелика кількість помилок в кодуванні.

Недоліки потокових шифрів:

- при передачі даних повинна бути повна синхронізації без втрати даних;
- невисока криптостійкість.

Приклади потокових шифрів: шифри Цезаря, Трітеміуса, Полібія, Плейфера (перехідний тим між блоковими і потоковими) [12].

Переваги блокових шифрів: висока криптистійкість через варіативність блоків.

Недоліки блокових шифрів: якщо помилитися в розшифруванні символу, невірно розшифрується весь блок.

Приклади блокових шифрів: Шифри Палиця, “Стандартна і вертикальна перестановка”, квадрат і прямокутник Кардано, Віженера [12].

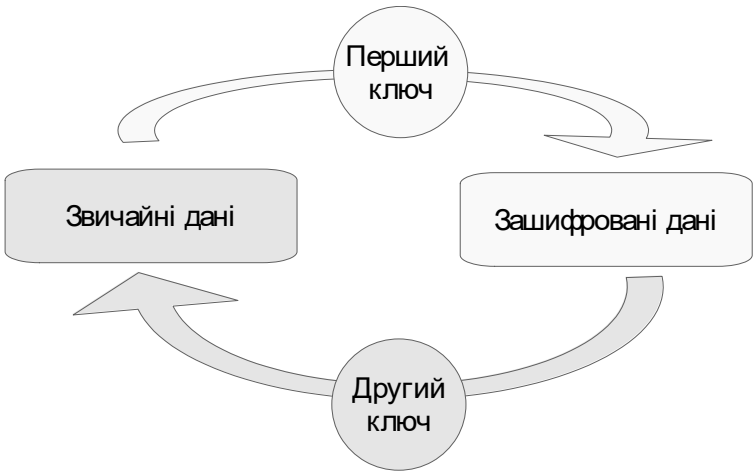

Схема 1.1.3 — принцип дії асиметричних алгоритмів шифрування

Особливість асиметричних алгоритмів шифрування [1] полягає в тому, що вони мають два ключі. Перший використовують для шифрування даних, іноді, він може бути відкритим, тобто вільно доступним. Другий ключ використовується для дешифрування даних, він відомий тільки власнику зашифрованого тексту.

Переваги асиметричних алгоритмів шифрування:

- не потрібна секретність ключа за допомогою якого шифруються дані, достатньо секретного ключа, за допомогою якого здійснюється розшифрування;
- потрібна менша кількість ключів;
- більша криптостійкість.

Недоліки асиметричних алгоритмів шифрування:

- невелика швидкість обробки даних;
- досить складно реалізовані шифратори;
- потрібно перевіряти відкриті ключі на підміну.

Приклади асиметричних алгоритмів шифрування: алгоритм шифрування Ель Гамаля, алгоритм RSA і т.д [12].

Отже, як висновок випливають такі факти:

1. Криптографія надзвичайно актуальна галузь математики / інформатики якою займається багато вчених.

2. Числа Фібоначчі вивчаються починаючи з середніх віків і донині в результаті чого вони стали об’єктом хобі багатьох людей, які цікавляться математикою

3. Числа Фібоначчі мають багато різних властивостей тому вони широко використовуються в різних алгоритмах.

РОЗДІЛ 2. Вимоги до використання матричного шифрування

2.1. Матричний метод шифрування

Матричний метод шифрування даних це асиметричний метод шифрування з двома ключами [7]. Суть матричного методу шифрування полягає в тому, що існує повідомлення: $m_1...m_n$ яке потрібно зашифрувати. Для цього записуємо його у вигляді квадратної матриці *M*, і вибираємо ключ *Q* у такому ж вигляді квадратної матриці. Перемноживши матриці *M* та *Q*, отримаємо матрицю *T*, яка і є нашим зашифрованим повідомленням. Для розшифрування повідомлення *T* помножимо на матрицю Q^{-1}, яка є оберненою матрицею до *Q*.

Шифрування:

$$M \cdot Q = T$$

Дешифрування:

$$T \cdot Q^{-1} = M'$$

Розглянемо детальніше механізм шифрування і розшифрування. Нехай початкове повідомлення і ключ мають вигляд:

$$M = \begin{pmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{pmatrix}, Q = \begin{pmatrix} q_1 & q_2 & q_3 \\ q_4 & q_5 & q_6 \\ q_7 & q_8 & q_9 \end{pmatrix}$$

Тоді:

$$T = \begin{pmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{pmatrix} \times \begin{pmatrix} q_1 & q_2 & q_3 \\ q_4 & q_5 & q_6 \\ q_7 & q_8 & q_9 \end{pmatrix}$$

Тобто матриця набуває вигляду:

$$T = \begin{pmatrix} m_1 \cdot q_1 + m_2 \cdot q_4 + m_3 \cdot q_7 & m_1 \cdot q_2 + m_2 \cdot q_5 + m_8 \cdot q_7 & m_1 \cdot q_3 + m_2 \cdot q_6 + m_3 \cdot q_9 \\ m_4 \cdot q_1 + m_5 \cdot q_4 + m_6 \cdot q_7 & m_4 \cdot q_2 + m_5 \cdot q_5 + m_6 \cdot q_7 & m_4 \cdot q_3 + m_5 \cdot q_6 + m_6 \cdot q_9 \\ m_7 \cdot q_1 + m_8 \cdot q_4 + m_9 \cdot q_7 & m_7 \cdot q_2 + m_8 \cdot q_5 + m_9 \cdot q_7 & m_4 \cdot q_3 + m_8 \cdot q_6 + m_9 \cdot q_9 \end{pmatrix} \Rightarrow$$

$$\Rightarrow \begin{pmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 \\ t_7 & t_8 & t_9 \end{pmatrix}$$

Аналогічний процес розшифрування:

$$T=\begin{pmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 \\ t_7 & t_8 & t_9 \end{pmatrix}, Q^{-1}=\begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 & \tilde{q}_3 \\ \tilde{q}_4 & \tilde{q}_5 & \tilde{q}_6 \\ \tilde{q}_7 & \tilde{q}_8 & \tilde{q}_9 \end{pmatrix}$$

$$M'=\begin{pmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 \\ t_7 & t_8 & t_9 \end{pmatrix}\times\begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 & \tilde{q}_3 \\ \tilde{q}_4 & \tilde{q}_5 & \tilde{q}_6 \\ \tilde{q}_7 & \tilde{q}_8 & \tilde{q}_9 \end{pmatrix}=\begin{pmatrix} \tilde{m}_1 & \tilde{m}_2 & \tilde{m}_3 \\ \tilde{m}_4 & \tilde{m}_5 & \tilde{m}_6 \\ \tilde{m}_7 & \tilde{m}_8 & \tilde{m}_9 \end{pmatrix}$$

Очевидно, що кодована інформація повинна бути числовою, як і сам ключ (що спростовується пізніше). Але постає питання, якщо виникла помилка при передачі даних, як про це дізнатися, адже ключ Q^{-1} та пошкоджене повідомлення $\tilde{T}$ при дешифруванні дадуть викривлене повідомлення $\tilde{M}$.

Подивимося більш детально на визначники цих матриць.

$$|M|=Det(M),|Q|=Det(Q),|T|=Det(T)$$

Очевидно, що:

$$Det(M)\times Det(Q)=Det(T)$$

Очевидно, якщо:

$$Det(Q)=(-1)^n, n=1,2,3...\Rightarrow Det(M)=\pm Det(T)$$

Тобто передаючи визначник матриці *M*, як окремий елемент шифрограми окремим повідомленням, ми зможемо перевірити його на пошкодження (момент коли пошкоджений сам елемент з $Det(M)$ не розглядаємо). Такий визначник назвемо контрольним детермінантом. Чіткого алгоритму як підібрати елементи матриці *Q* для всіх чисел немає, але в цьому випадку в нагоді стають числа Фібоначчі, для яких ми можемо розробити чіткий алгоритм формування *Q* – матриць.

Розглянемо приклад:

Нехай наше повідомлення має вигляд:

222 100 111 1 350 71 84 40 120

Тоді матриця *M* має вигляд:

$$M = \begin{pmatrix} 222 & 100 & 111 \\ 1 & 350 & 71 \\ 84 & 40 & 120 \end{pmatrix}, Det(M) = 6018960$$

Виберемо матрицю Q, і обчислимо обернену Q^{-1}:

$$Q = \begin{pmatrix} 144 & 21 & 89 \\ 21 & 2 & 13 \\ 89 & 13 & 55 \end{pmatrix}$$

$$Q^{-1} = \begin{pmatrix} -59 & 2 & 95 \\ 2 & -1 & -3 \\ 95 & -3 & -153 \end{pmatrix}$$

Тоді шифрограма матиме вигляд:

$$T = \begin{pmatrix} 222 & 100 & 111 \\ 1 & 350 & 71 \\ 84 & 40 & 120 \end{pmatrix} \times \begin{pmatrix} 144 & 21 & 89 \\ 21 & 2 & 13 \\ 89 & 13 & 55 \end{pmatrix} = \begin{pmatrix} 43947 & 6305 & 34378 \\ 13813 & 1644 & 13159 \\ 23613 & 3404 & 22396 \end{pmatrix}$$

Тобто:

Контрольний детермінант: 6018960.

Шифрограма: 43947 6305 34378 13813 1644 13459 23613 3404 22396.

Ключ дешифрування: -59 2 95 2 -1 -3 95 -3 -153

Контрольний детермінант порівнюємо $Det(M)$ з вирахуваним $Det(T)$ і якщо все збігається для розшифрування виконуємо обернену операцію:

$$Det(T) = 6018960 \Rightarrow Det(T) = Det(M) \Rightarrow$$

$$\Rightarrow M = \begin{pmatrix} 43947 & 6305 & 34378 \\ 13813 & 1644 & 13159 \\ 23613 & 3404 & 22396 \end{pmatrix} \times \begin{pmatrix} -59 & 2 & 95 \\ 2 & -1 & -3 \\ 95 & -3 & -153 \end{pmatrix} = \begin{pmatrix} 222 & 100 & 111 \\ 1 & 350 & 71 \\ 84 & 40 & 120 \end{pmatrix}.$$

Отримуємо аналогічний результат.

Зашифруємо це ж повідомлення за допомогою іншого ключа:

$$Q = \begin{pmatrix} 55 & 8 & 34 \\ 8 & 2 & 5 \\ 34 & 5 & 21 \end{pmatrix}$$

$$T = \begin{pmatrix} 222 & 100 & 111 \\ 1 & 350 & 71 \\ 84 & 40 & 120 \end{pmatrix} \times \begin{pmatrix} 55 & 8 & 34 \\ 8 & 2 & 5 \\ 34 & 5 & 21 \end{pmatrix} = \begin{pmatrix} 16784 & 2531 & 10379 \\ 5269 & 1063 & 3275 \\ 9020 & 1352 & 5576 \end{pmatrix}$$

Шифрограма: 16784 2531 10379 5269 1063 3275 9020 1352 5576

Як бачимо вихідне повідомлення в обох випадках однакове, але при різних ключах отримаємо два несподіваних варіанти, які не можуть бути пов'язані між собою. Тобто даний шифр не вразливий до різних видів частотного аналізу.

В розглянутому матричному методі шифрування є як свої переваги так і недоліки.

Переваги:

- Висока крипостійкість. Це пов'язано зі складністю відшукання ключа, а отже з неможливістю частотного аналізу.
- Простий алгоритм шифрування / дешифрування – матриці досить просто перемножуються.
- Перевірка помилок при передачі повідомлень в реальному часі за рахунок контрольних детермінантів.

Недоліки:

- Важкість знаходження *Q* матриць ключів.
- Надмірність (збільшення) тексту, що передається, у відношенні до вихідного, через множення матриць.

2.2. Алгоритми формування Q матриць

Розглянемо алгоритми формування *Q* матриць. Спочатку розглянемо матриці 2×2 . Нагадаємо головна особливість *Q* матриць це те, що їх визначник дорівнює $(-1)^n$, де *n* номер числа в послідовності Фібоначчі. Щоб ця властивість виконувалася нам в нагоді стануть дві властивості чисел Фібоначчі:

$$1) F_{n+1} \cdot F_{n+2} - F_n \cdot F_{n+3} = (-1)^n$$

$$2) F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$$

Скориставшись першою, отримаємо матрицю:

$$Q=\begin{pmatrix} F_{n+1} & F_n \\ F_{n+3} & F_{n+2} \end{pmatrix}, Det(Q)=F_{n+1}\cdot F_{n+2}-F_n\cdot F_{n+3}=(-1)^n$$

Зрозуміло помінявши місцями члени матриці ми також отримаємо правильний ключ: $Q=\begin{pmatrix} F_{n+2} & F_{n+3} \\ F_n & F_{n+1} \end{pmatrix}$. Назвемо таку матрицю ключовою матрицею другого порядку першого типу.

Аналогічно, використовуючи другу властивість, отримаємо:

$$Q=\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}, Det(Q)=F_{n+1}\cdot F_{n-1}-F_n^2=(-1)^n$$

Перестановки, як з першим типом також працюють. Назвемо таку матрицю ключовою матрицею другого порядку другого типу.

Деякі Q матриці другого порядку першого типу:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\begin{pmatrix} 13 & 21 \\ 5 & 8 \end{pmatrix},\begin{pmatrix} 8 & 5 \\ 21 & 13 \end{pmatrix},\begin{pmatrix} 5 & 8 \\ 2 & 3 \end{pmatrix},\begin{pmatrix} 144 & 233 \\ 55 & 89 \end{pmatrix},\ldots$$

Деякі Q матриці другого порядку другого типу:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix},\begin{pmatrix} 8 & 5 \\ 5 & 3 \end{pmatrix},\begin{pmatrix} 21 & 13 \\ 13 & 8 \end{pmatrix},\begin{pmatrix} 144 & 89 \\ 89 & 55 \end{pmatrix},\ldots$$

Для Q матриць $n\times n, n=3,4,\ldots$ ситуація дещо складніша. Щоб в ній розібратися згадаємо, що таке трикутник Паскаля. Трикутник Паскаля – це геометрично, на зразок трикутника, розміщені біноміальні коефіцієнти [2]. Якщо записати ці коефіцієнти на зразок таблиці то, такий запис називають прямокутним трикутником Паскаля [14].

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	1	1	1	1	1	1	1	1	1	1
1		1	2	3	4	5	6	7	8	9	10
2			1	3	6	10	15	21	28	36	45
3				1	4	10	20	35	56	84	120

4					1	5	15	35	70	126	210
5						1	6	21	56	126	252
6							1	7	28	84	210
7								1	8	36	120
8									1	9	45
9										1	10
10											1
F_n	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**

Таблиця 2.2.1 – Прямокутний трикутник Паскаля

Змістимо даний трикутник на одну комірку вправо починаючи з рядка №1. Причому кожний наступний рядок зміщуємо і на попереднє число зміщень. Тобто рядок №1 зміщено на 1 коміру, рядок №2 на 2 комірки і т.д. Отримаємо зміщений прямокутний трикутник [14]. Сума біноміальних коефіцієнтів у відповідних стовпцях дасть послідовність Фібоначчі. Тобто через трикутник Паскаля ми отримали послідовність Фібоначчі.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	1	1	1	1	1	1	1	1	1	1
1			1	2	3	4	5	6	7	8	9
2					1	3	6	10	15	21	28
3							1	4	10	20	35
4									1	5	15
5											1
F_n	**1**	**1**	**2**	**3**	**5**	**8**	**13**	**21**	**34**	**55**	**89**

Таблиця 2.2.2 – Прямокутний трикутник Паскаля з числами Фібоначчі

Виконаємо таку саму операцію змістивши початковий прямокутний трикутник Паскаля вже на дві комірки вправо. Просумуємо біноміальні коефіцієнти і отримуємо новий ряд чисел. Це одні із так званих *p* чисел

Фібоначчі. В нашому випадку *p=2.* Число *p* показує зміщення прямокутного трикутника Паскаля на відповідне число комірок.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	1	1	1	1	1	1	1	1	1	1
1				1	2	3	4	5	6	7	8
2							1	3	6	10	15
3										1	4
F_n	**1**	**1**	**1**	**2**	**3**	**4**	**6**	**9**	**13**	**19**	**28**

Таблиця 2.2.3 – Прямокутний трикутник Паскаля з p=2 числами Фібоначчі

Провівши подібні обчислення можна дістати таблицю з *p* числами Фібоначчі [14].

p \ n	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	2	4	8	16	32	64	128	256	512	1024
1	1	1	2	3	5	8	13	21	34	55	89
2	1	1	1	2	3	4	6	9	13	19	28
3	1	1	1	1	2	3	4	5	7	10	14
4	1	1	1	1	1	2	3	4	5	6	8
5	1	1	1	1	1	1	2	3	4	5	6

Таблиця 2.2.4 – p числа Фібоначчі

Легко помітити залежність притаманну звичайним числам Фібоначчі, але дещо модифіковану.

$$F_0(p)...F_p(p)=1, F(p)_{n+1}=F(p)_n+F(p)_{n-p}$$

Поєднаємо *p* числа Фібоначчі з спеціалізованою Q_p матрицею виду [19]:

$$Q_p=\begin{pmatrix} 1 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 \\ 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix},\ p=1,2,3,4...Det(Q)=(-1)^p$$

Розглянемо формування квадратних матриць на прикладі матриць 3×3. Відповідно:

$$Q_2=\begin{pmatrix}1&1&0\\0&0&1\\1&0&0\end{pmatrix}$$

Якщо підносити матрицю Q до степеня $n=2,3,4\ldots$ отримаємо наступні результати:

$$n=2,3,4,5,\ldots,9,10,11,\ldots$$

$$\begin{pmatrix}1&1&1\\1&0&0\\1&1&0\end{pmatrix},\begin{pmatrix}2&1&1\\1&1&0\\1&1&1\end{pmatrix},\begin{pmatrix}3&2&1\\1&1&1\\2&1&1\end{pmatrix},\begin{pmatrix}4&3&2\\2&1&1\\3&2&1\end{pmatrix},...$$

$$\ldots,\begin{pmatrix}19&13&9\\9&6&4\\13&9&6\end{pmatrix},\begin{pmatrix}28&19&13\\13&9&6\\19&13&9\end{pmatrix},\begin{pmatrix}41&28&19\\19&13&9\\28&19&13\end{pmatrix},\ldots$$

Легко помітити, що при такому записі використовуються p числа Фібоначчі (p = *2*) та вирисовується алгоритм запису:

$$Q=\begin{pmatrix}F_2^{n+1}&F_2^{n}&F_2^{n-1}\\F_2^{n-1}&F_2^{n-2}&F_2^{n-3}\\F_2^{n}&F_2^{n-1}&F_2^{n-2}\end{pmatrix}$$

Формування Q матриць працює за даним алгоритмом і для квадратних матриць $k\times k,k>3$. Єдина різниця, що використовуються p числа Фібоначі, причому $k=p+1$. Рекурентна формула матиме вигляд [19]:

$$Q=\begin{pmatrix}F_p^{n+1}&F_p^{n}&\ldots&F_p^{n-p+2}&F_p^{n-p+1}\\F_p^{n-p+1}&F_p^{n-p}&\ldots&F_p^{n-2p+2}&F_p^{n-2p+1}\\\vdots&\vdots&\ddots&\vdots&\vdots\\F_p^{n-1}&F_p^{n-2}&\ldots&F_p^{n-p}&F_p^{n-p-1}\\F_p^{n}&F_p^{n-1}&\ldots&F_p^{n-p+1}&F_p^{n-p}\end{pmatrix},Det(Q)=(-1)^{n\times p}$$

$$p=1,2,3,\ldots;n=\pm2,\pm3,\pm4,\ldots$$

Отже використовуючи наведені формули ми легко зможемо формувати знайти ключі для даного методу шифрування.

2.3. Розширення поняття матричного шифрування на текстову інформацію

Введемо поняття словника. Словник – спеціальним чином упорядкований набір символів, в нашому випадку цифр і літер великого і маленького регістру. Кожен символ має свій порядковий номер. Наприклад в запропонованому словнику літера *«б»* має порядковий номер 12, а велика літера *«П»* має порядковий номер 64. Користуючись порядковими номерами цих символів, кожну фразу можна перетворити у набір чисел. Наприклад слово *«Алгебра»* перетвориться в *«44 26 14 17 12 31 11»*, а слово *«Шифр»* перетворюється *«71 23 35 31»*. По такому самому принципу перетворюються цілі фрази, наприклад «Я люблю математику» перетворюється в *«76 77 26 43 12 16 43 77 27 11 33 17 27 11 33 21 25 34»*.

№ символу	Символ	№ символу	Символ	№ символу	Символ
1	0	26	л	51	Є
2	1	27	м	52	Ж
3	2	28	н	53	З
4	3	29	о	54	И
5	4	30	п	55	І
6	5	31	р	56	Ї
7	6	32	с	57	Й
8	7	33	т	58	Й
9	8	34	у	59	К
10	9	35	ф	60	Л
11	а	36	х	61	М
12	б	37	ц	62	Н
13	в	38	ч	63	О
14	г	39	ш	64	П

15	ґ	40	щ	65	Р
16	д	41	ь	66	С
17	е	42	ю	67	Т
18	є	43	я	68	У
19	ж	44	А	69	Ф
20	з	45	Б	70	Х
21	и	46	В	71	Ц
22	і	47	Г	72	Ч
23	ї	48	Ґ	73	Ш
24	й	49	Д	74	Щ
25	к	50	Е	75	Ю
		76	Я	77	Пробіл

Таблиця 2.3.1 – Словник для матричного шифрування Українською мовою

Розглянемо приклад. Нехай у нас є слово *«Алгебра»* воно має 7 символів і як вище визначили в числовому еквіваленті це «44 26 14 17 12 31 11». Нехай ключові матриці мають вигляд (їх ми вже використовували раніше при шифруванні):

$$Q=\begin{pmatrix}144 & 21 & 89\\ 21 & 2 & 13\\ 89 & 13 & 55\end{pmatrix}, Q^{-1}=\begin{pmatrix}-59 & 2 & 95\\ 2 & -1 & -3\\ 95 & -3 & -153\end{pmatrix}$$

Але в слові «Алгебра» не вистачає 2 символів для утворення матриці M 3×3, тому додамо ще два символи *«Пробіл»*, тобто виходить послідовність «44 26 14 17 12 31 11 77 77».

$$M=\begin{pmatrix}44 & 26 & 14\\ 17 & 12 & 31\\ 11 & 77 & 77\end{pmatrix}$$

Шифруємо повідомлення:

$$\begin{pmatrix} 44 & 26 & 14 \\ 17 & 12 & 31 \\ 11 & 77 & 77 \end{pmatrix} \times \begin{pmatrix} 144 & 21 & 89 \\ 21 & 2 & 13 \\ 89 & 13 & 55 \end{pmatrix} = \begin{pmatrix} 8128 & 1158 & 5024 \\ 5459 & 784 & 3374 \\ 10054 & 1386 & 6215 \end{pmatrix}$$

Порахуємо контрольний детермінант:

$Det(M) = -73062$

Тобто:

Контрольний детермінант: -73062

Ключ дешифрування: -59 2 95 2 -1 -3 95 -3 -153

Шифрограма: 8128 1158 5024 5459 784 3374 10054 1386 6215

Виконаємо процедуру дешифрування:

1) Перевіримо контрольний детермінант:

$$8128 \times 784 \times 6215 + 1158 \times 3374 \times 10054 + 5024 \cdot 5459 \cdot 1386 - 5024 \times 784 \times 10054 -$$
$$-8128 \times 3374 \times 1386 - 1158 \times 5459 \times 6215 =$$
$$= 39604167680 + 39281902968 + 38012458176 - 39600856064 - 38009486592 -$$
$$-39288259230 = (-73062)$$

Отже контрольний детермінант сходиться.

2) Проведемо процедуру розшифрування:

$$\begin{pmatrix} 8128 & 1158 & 5024 \\ 5459 & 784 & 3374 \\ 10054 & 1386 & 6215 \end{pmatrix} \times \begin{pmatrix} -59 & 2 & 95 \\ 2 & -1 & -3 \\ 95 & -3 & -153 \end{pmatrix} = \begin{pmatrix} 44 & 26 & 14 \\ 17 & 12 & 31 \\ 11 & 77 & 77 \end{pmatrix}$$

Тобто ми отримуємо числову послідовність: 44 26 14 17 31 12 11 77 77. Шукаємо в словнику відповідні позиції і отримуємо слово: *«Алгебра»*. Відкидаємо зайві відступи і отримуємо початкове повідомлення *«Алгебра».*

Розглянемо, ще один приклад, зашифруємо фразу **«Я люблю математику»**. Кодувати будемо за допомогою матриць 2×2. Також розберемо особливості матричного алгоритму шифрування для програмної реалізації.

Спочатку, за допомогою словника, переведемо кодовану фразу в числовий еквівалент:

«76 77 26 43 12 16 43 77 27 11 33 17 27 11 33 21 25 34»

Поділимо фразу на блоки по 4 елементи, в разі необхідності додамо відступи (код 77). В результаті отримуємо матриці:

$$\begin{pmatrix}76 & 77\\ 26 & 43\end{pmatrix},\begin{pmatrix}12 & 16\\ 43 & 77\end{pmatrix},\begin{pmatrix}27 & 11\\ 33 & 17\end{pmatrix},\begin{pmatrix}27 & 11\\ 33 & 21\end{pmatrix},\begin{pmatrix}25 & 34\\ 77 & 77\end{pmatrix}$$

Нехай ключ буде *«144 233 55 89»*.

$$Q=\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}$$

Обернена матриця:

$$Q^{-1}=\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}$$

Виконаємо шифрування:

Контрольні детермінанти: 1266 236 96 204 -693

Обрахуємо шифрограми:

$$\begin{pmatrix}76 & 77\\ 26 & 43\end{pmatrix}\times\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}=\begin{pmatrix}15179 & 24561\\ 6109 & 9885\end{pmatrix}$$

$$\begin{pmatrix}12 & 16\\ 43 & 77\end{pmatrix}\times\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}=\begin{pmatrix}2608 & 4220\\ 10427 & 16872\end{pmatrix}$$

$$\begin{pmatrix}27 & 11\\ 33 & 17\end{pmatrix}\times\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}=\begin{pmatrix}4493 & 7270\\ 5687 & 9202\end{pmatrix}$$

$$\begin{pmatrix}27 & 11\\ 33 & 21\end{pmatrix}\times\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}=\begin{pmatrix}4493 & 7270\\ 5907 & 9558\end{pmatrix}$$

$$\begin{pmatrix}25 & 34\\ 77 & 77\end{pmatrix}\times\begin{pmatrix}144 & 233\\ 55 & 89\end{pmatrix}=\begin{pmatrix}5470 & 8851\\ 15323 & 24794\end{pmatrix}$$

Тобто шифрограма матиме вигляд:

15179 24561 6109 9885 2608 4220 10427 16872 4493 7270 5687 9202 4493 7270 5907 9558 5470 8851 15323 24794

Ключ розшифрування: 89 -233 -55 144

Аналогічно виконуємо дешифрування:

$$\begin{pmatrix} 15179 & 24561 \\ 6109 & 9885 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 76 & 77 \\ 26 & 43 \end{pmatrix}$$

$$\begin{pmatrix} 2608 & 4220 \\ 10427 & 16872 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 12 & 16 \\ 43 & 77 \end{pmatrix}$$

$$\begin{pmatrix} 4493 & 7270 \\ 5687 & 9202 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 27 & 11 \\ 33 & 17 \end{pmatrix}$$

$$\begin{pmatrix} 4493 & 7270 \\ 5907 & 9558 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 27 & 11 \\ 33 & 21 \end{pmatrix}$$

$$\begin{pmatrix} 5470 & 8851 \\ 15323 & 24794 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 25 & 34 \\ 77 & 77 \end{pmatrix}$$

Оскільки матриці при шифруванні і розшифруванні співпадають то очевидно, що процедура дешифрування відбулася правильно.

Чому саме квадратна матриця *2x2*:

1) Робота з числами Фібоначчі, а не з *p* числами Фібоначчі, через алгоритми формування *Q* матриць.

2) Зменшення надлишковість повідомлення мінімум в 2 рази. Наприклад довжина шифрограми 15 символів. Для матриці 2 на 2 зайвий символ 1 (1 відступ, щоб довжина повідомлення була 16 символів), для 3 на 3 зайві 3 символи (3 відступи, щоб довжина повідомлення була 18 символів) і т.д.

3) Швидкість обчислення. Наприклад для визначення визначників для матриці для кодування тих же 15 символів потрібно 12 операцій. Якщо ж брати матриці 3 на 3 то в таких же умовах потрібно 34 операцій.

4) Більший шанс виправлення помилки.

Чому саме квадратна матриця *3x3* і більше:

1) Робота з *p* числами Фібоначчі, які працюють за тими ж алгоритмами що й оригінальні числа Фібоначчі.

2) З кожним збільшенням розмірності матриць з якими ми працюємо виростає криптостійкість через велику громіздкість обчислень, але падає швидкість обчислень.

Тобто спираючись на ці фактори нам потрібно тримати баланс в прагненні до криптостійкості та швидкості шифрування та дешифрування.

2.4. Виявлення та виправлення помилок в шифрограмі

При передачі даних через мережу інтернет іноді виникають різні проблеми. Наприклад помилки з'єднання при передачі може привести до втрати або зміні (викривленню) прийнятої інформації. В більшості випадків відновити інформацію неможливо, потрібно заново передавати повідомлення.

Нехай контрольний визначник кодованої матриці M дорівнює $Det(M)$, а закодована матриця яку приймаємо має вигляд:

$$T = \begin{pmatrix} t_1 & t_2 \\ t_3 & t_4 \end{pmatrix}$$

Тоді визначник

$$Det(T) = t_1 \cdot t_4 - t_2 \cdot t_3$$

Ми знаємо, що для матричного методу шифрування справедлива рівність:

$$Det(M) \times (-1)^n = Det(T)$$

Тоді знаючи ключ Q, а отже і коефіцієнт $(-1)^n = -1$ чи $(-1)^n = 1$ можна сказати правильно ми прийняли повідомлення чи ні, а саме, якщо:

$$Det(M) \neq \pm Det(T)$$

Якщо дійсно виявили помилку то можливі чотири випадки:

1) Помилка в 1 з 4 прийнятих значень;

2) Помилка в 2 з 4 прийнятих значень;

3) Помилка в 3 з 4 прийнятих значень;

4) Помилка в 4 з 4 прийнятих значень;

В 3 - 4 випадку відновити значення неможливо, виправити такі помилки можливо тільки при повторному прийомі повідомлення.

В першому випадку маємо 4 випадки:

$$\begin{pmatrix} x & t_2 \\ t_3 & t_4 \end{pmatrix}, \begin{pmatrix} t_1 & x \\ t_3 & t_4 \end{pmatrix}, \begin{pmatrix} t_1 & t_2 \\ x & t_4 \end{pmatrix}, \begin{pmatrix} t_1 & t_2 \\ t_3 & x \end{pmatrix}$$

Де x – пошкоджений елемент повідомлення (ціле число).

Тобто:

$$Det(T) = x \cdot t_4 - t_2 \cdot t_3$$
$$Det(T) = t_1 \cdot t_4 - x \cdot t_3$$
$$Det(T) = t_1 \cdot t_4 - t_2 \cdot x$$
$$Det(T) = t_1 \cdot x - t_2 \cdot t_3$$

А оскільки $\pm Det(M) = Det(T)$ то маємо:

$$\pm Det(M) = x \cdot t_4 - t_2 \cdot t_3$$
$$\pm Det(M) = t_1 \cdot t_4 - x \cdot t_3$$
$$\pm Det(M) = t_1 \cdot t_4 - t_2 \cdot x$$
$$\pm Det(M) = t_1 \cdot x - t_2 \cdot t_3$$

$$x = \frac{\pm Det(M) + t_2 \cdot t_3}{t_4}$$
$$x = \frac{\mp Det(M) + t_1 \cdot t_4}{t_3}$$
$$x = \frac{\mp Det(M) + t_1 \cdot t_4}{t_2}$$
$$x = \frac{\pm Det(M) + t_2 \cdot t_3}{t_1}$$

Знайшовши x – відновимо повідомлення [7].

Якщо в нас 2 випадок, то маємо наступні випадки:

$$\begin{pmatrix} x & y \\ t_3 & t_4 \end{pmatrix}, \begin{pmatrix} t_1 & x \\ y & t_4 \end{pmatrix}, \begin{pmatrix} x & t_2 \\ y & t_4 \end{pmatrix}, \begin{pmatrix} t_1 & x \\ t_3 & y \end{pmatrix}, \begin{pmatrix} x & t_2 \\ t_3 & y \end{pmatrix}, \begin{pmatrix} t_1 & t_2 \\ x & y \end{pmatrix}$$

Тобто:

$$\begin{array}{ll} \pm Det(M) = x \cdot t_4 - y \cdot t_3 & x \cdot t_4 - y \cdot t_3 = \pm Det(M) \\ \pm Det(M) = t_1 \cdot t_4 - x \cdot y & x \cdot y = t_1 \cdot t_4 \mp Det(M) \\ \pm Det(M) = x \cdot t_4 - t_2 \cdot y & x \cdot t_4 - y \cdot t_2 = \pm Det(M) \\ \pm Det(M) = t_1 \cdot y - x \cdot t_3 \Rightarrow & y \cdot t_1 - x \cdot t_3 = \pm Det(M) \\ \pm Det(M) = x \cdot y - t_2 \cdot t_3 & x \cdot y = t_2 \cdot t_3 \pm Det(M) \\ \pm Det(M) = t_1 \cdot y - x \cdot t_2 & y \cdot t_1 - x \cdot t_2 = \pm Det(M) \end{array}$$

$$y = -\frac{\pm Det(M) - x \cdot t_4}{t_3}$$

$$y = \frac{t_1 \cdot t_4 \mp Det(M)}{x}$$

$$y = -\frac{\pm Det(M) - x \cdot t_4}{t_2}$$

$$y = \frac{\pm Det(M) + x \cdot t_3}{t_1}$$

$$y = \frac{t_2 \cdot t_3 \pm Det(M)}{x}$$

$$y = \frac{\pm Det(M) + x \cdot t_2}{t_1}$$

Оскільки довжина словника $n = 77$ та $Q^{-1} = \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix}$ то маємо:

$$1)\begin{pmatrix} x & y \\ t_3 & t_4 \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} x \times \tilde{q}_1 - y \times \tilde{q}_3 & x \times \tilde{q}_2 - y \times \tilde{q}_4 \\ t_3 \times \tilde{q}_1 + t_4 \times \tilde{q}_3 & t_3 \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le x \times \tilde{q}_1 - \dfrac{\pm Det(M) - x \cdot t_4}{t_3} \times \tilde{q}_3 \le 77 \\ 1 \le x \times \tilde{q}_2 - \dfrac{\pm Det(M) - x \cdot t_4}{t_3} \times \tilde{q}_4 \le 77 \\ 1 \le t_3 \times \tilde{q}_1 + t_4 \times \tilde{q}_3 \le 77 \\ 1 \le t_3 \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \le 77 \end{cases}$$

$$2)\begin{pmatrix} t_1 & x \\ y & t_4 \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} t_1 \times \tilde{q}_1 + x \times \tilde{q}_3 & t_1 \times \tilde{q}_2 + x \times \tilde{q}_4 \\ y \times \tilde{q}_1 + t_4 \times \tilde{q}_3 & y \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le t_1 \times \tilde{q}_1 + x \times \tilde{q}_3 \le 77 \\ 1 \le t_1 \times \tilde{q}_2 + x \times \tilde{q}_4 \le 77 \\ 1 \le y \times \tilde{q}_1 + t_4 \times \tilde{q}_3 \le 77 \\ 1 \le y \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \le 77 \end{cases}$$

$$3)\begin{pmatrix} x & t_2 \\ y & t_4 \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} x \times \tilde{q}_1 + t_4 \times \tilde{q}_3 & x \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \\ y \times \tilde{q}_1 + t_4 \times \tilde{q}_3 & y \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le x \times \tilde{q}_1 + t_4 \times \tilde{q}_3 \le 77 \\ 1 \le x \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \le 77 \\ 1 \le y \times \tilde{q}_1 + t_4 \times \tilde{q}_3 \le 77 \\ 1 \le y \times \tilde{q}_2 + t_4 \times \tilde{q}_4 \le 77 \end{cases}$$

$$4) \begin{pmatrix} t_1 & x \\ t_3 & y \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} t_1 \times \tilde{q}_1 + x \times \tilde{q}_3 & t_1 \times \tilde{q}_2 + x \times \tilde{q}_4 \\ t_3 \times \tilde{q}_1 + y \times \tilde{q}_3 & t_3 \times \tilde{q}_2 + y \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le t_1 \times \tilde{q}_1 + x \times \tilde{q}_3 \le 77 \\ 1 \le t_1 \times \tilde{q}_2 + x \times \tilde{q}_4 \le 77 \\ 1 \le t_3 \times \tilde{q}_1 + y \times \tilde{q}_3 \le 77 \\ 1 \le t_3 \times \tilde{q}_2 + y \times \tilde{q}_4 \le 77 \end{cases}$$

$$5) \begin{pmatrix} x & t_2 \\ t_3 & y \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} x \times \tilde{q}_1 + t_2 \times \tilde{q}_3 & x \times \tilde{q}_2 + t_2 \times \tilde{q}_4 \\ t_3 \times \tilde{q}_1 + y \times \tilde{q}_3 & t_3 \times \tilde{q}_2 + y \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le x \times \tilde{q}_1 + t_2 \times \tilde{q}_3 \le 77 \\ 1 \le x \times \tilde{q}_2 + t_2 \times \tilde{q}_4 \le 77 \\ 1 \le t_3 \times \tilde{q}_1 + y \times \tilde{q}_3 \le 77 \\ 1 \le t_3 \times \tilde{q}_2 + y \times \tilde{q}_4 \le 77 \end{cases}$$

$$6) \begin{pmatrix} t_1 & t_2 \\ x & \dfrac{\pm Det(M) + x \cdot t_2}{t_1} \end{pmatrix} \times \begin{pmatrix} \tilde{q}_1 & \tilde{q}_2 \\ \tilde{q}_3 & \tilde{q}_4 \end{pmatrix} = \begin{pmatrix} t_1 \times \tilde{q}_1 + t_2 \times \tilde{q}_3 & t_1 \times \tilde{q}_2 + t_2 \times \tilde{q}_4 \\ x \times \tilde{q}_1 + y \times \tilde{q}_3 & x \times \tilde{q}_2 + y \times \tilde{q}_4 \end{pmatrix}$$

$$\begin{cases} 1 \le t_1 \times \tilde{q}_1 + t_2 \times \tilde{q}_3 \le 77 \\ 1 \le t_1 \times \tilde{q}_2 + t_2 \times \tilde{q}_4 \le 77 \\ 1 \le x \times \tilde{q}_1 + \dfrac{\pm Det(M) + x \cdot t_2}{t_1} \times \tilde{q}_3 \le 77 \\ 1 \le x \times \tilde{q}_2 + \dfrac{\pm Det(M) + x \cdot t_2}{t_1} \times \tilde{q}_4 \le 77 \end{cases}$$

Розв’язавши нерівності отримаємо набір числових значень. Якщо, отримуємо не цілі значення, то округлюємо до меншого цілого значення зліва і до більшого цілого значення справа.

Отримавши всі можливі набори x', y' можна провести дешифрування і отримати набір можливих розшифрованих повідомлень. Але такий варіант можливий, якщо ми пересилаємо не числову інформацію, а осмислену фразу

переведену в числовий еквівалент за допомогою словника. А отримані переведені набори повинна оцінити людина яка працює з дешифратором, або щоб програма яка виявляє помилки була підключена до спеціально навченої нейронної мережі, яка перевіряє осмисленість фрази.

Очевидно що варіанти з неправильною передачею контрольного детермінанта не розглядаються. Користувачі повинні якимось способом точно обмінятися контрольними детермінантами. Наприклад двічі відправити повідомлення, і якщо вони збігаються то приймати його за істинне, якщо ні то повторити процедуру.

Також даний варіант можна розглядати коли ми шифруємо матрицями $k \times k$ де $k = 3,4,...$, але в такому випадку отримаємо більший шанс на отримання 2, 3, 4 і т.д. помилок в повідомленні, що ускладнює або робить неможливим його відновлення.

Розглянемо приклад. Нехай ми приймаємо числову інформацію і отримуємо такі дані:

$$T = \begin{pmatrix} 15179 & 24561 \\ 4852 & 9885 \end{pmatrix}, Det(M) = 1266, Det(Q) = 1$$

$$Det(T) = 15179 \cdot 9885 - 4852 \cdot 24561 = 30874443 \neq Det(M)$$

Отже в даному повідомленні виявилася помилка, знайдемо її.

$$x_1 = \frac{1266 + 24561 \cdot 4852}{9885} \approx 12055,77$$

$$x_2 = \frac{-1266 + 15179 \cdot 9885}{4852} \approx 30923,98$$

$$x_3 = \frac{-1266 + 15179 \cdot 9885}{24561} = 6109$$

$$x_1 = \frac{1266 + 24561 \cdot 4852}{15179} \approx 7851,06$$

Тобто цілим виявилося лише одне число $x_3 = 6109$. Тобто повідомлення, яке ми були повинні отримати:

$$T = \begin{pmatrix} 15179 & 24561 \\ 6109 & 9885 \end{pmatrix}$$

Розглянемо інший приклад. Нехай ми приймаємо числову інформацію і отримуємо такі дані:

$$T=\begin{pmatrix}15179 & 18423\\ 4852 & 9885\end{pmatrix}, Det(M)=1266, Det(Q)=1, Q^{-1}=\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}$$

А повідомлення далі розшифровується як: «блю».

$$Det(T)=15179\cdot 9885-4852\cdot 18423=62173919$$

$$Det(M)\neq Det(T)$$

$$x_1=\frac{1266+18423\cdot 4852}{9885}\approx 9042,96$$
$$x_2=\frac{-1266+15179\cdot 9885}{4852}\approx 30923,98$$
$$x_3=\frac{-1266+15179\cdot 9885}{18423}\approx 8144,34$$
$$x_1=\frac{1266+18423\cdot 4852}{15179}\approx 5889,04$$

Не одного цілого значення не має, тобто в прийнятому повідомленні більше однієї помилки.

$$T=\begin{pmatrix}15179 & 18423\\ 4852 & 9885\end{pmatrix}, Det(M)=1266, Det(Q)=1, Q^{-1}=\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}$$

$$1)\begin{pmatrix}x & -\frac{1266-x\cdot 9885}{18423}\\ 4852 & 9885\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}$$

$$t_3\times\tilde{q}_1+t_4\times\tilde{q}_3=-111847$$
$$t_3\times\tilde{q}_1+t_4\times\tilde{q}_3<1$$

Отже даний варіант не підходить.

$$6)\begin{pmatrix}15179 & 18423\\ x & \frac{1266+x\cdot 18423}{15179}\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}$$

$$t_1\times\tilde{q}_1+t_2\times\tilde{q}_3=337666$$
$$t_1\times\tilde{q}_1+t_2\times\tilde{q}_3>77$$

Отже даний варіант не підходить.

$$2)\begin{pmatrix}15179 & x\\ y & 9885\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}=\begin{pmatrix}1350931-55x & -3536707+144x\\ 89y-543675 & -233y+1423440\end{pmatrix}$$

$$\begin{cases}24561\le x\le 24563\\ 24560\le x\le 24561\\ 6108\le y\le 6109\\ 6109\le y\le 6110\end{cases}$$

Отримаємо: $x=24561; y=6109$

$$3)\begin{pmatrix}x & 18423\\ y & 9885\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}=\begin{pmatrix}89x-1013265 & -233x+2652912\\ 89y-543675 & -233y+1423440\end{pmatrix}$$

$$\begin{cases}11385\le x\le 11386\\ 11385\le x\le 11386\\ 6108\le y\le 6109\\ 6108\le y\le 6110\end{cases}$$

Отримаємо: $\begin{matrix}x=11385; x=11386\\ y=6108; y=6109\end{matrix}$

$$4)\begin{pmatrix}15179 & x\\ 4852 & y\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}=\begin{pmatrix}1350931-55x & -3536707+144x\\ 431828-55y & -1130516+144y\end{pmatrix}$$

$$\begin{cases}24561\le x\le 24563\\ 24560\le x\le 24561\\ 7850\le y\le 7852\\ 7850\le y\le 7852\end{cases}$$

Отже $x=24561$ і три значення *y*. Щоб не брати 3 значення

скористаємося рівністю: $y=\dfrac{\pm Det(M)+x\cdot t_3}{t_1}$

$$y=\frac{1266+24561\times 4852}{15179}\approx 7851$$

$$5)\begin{pmatrix}x & 18423\\ 4852 & y\end{pmatrix}\times\begin{pmatrix}89 & -233\\ -55 & 144\end{pmatrix}=\begin{pmatrix}89x-1013265 & -233x+2652912\\ 431828-55y & -1130516+144y\end{pmatrix}$$

$$\begin{cases} 11385 \le x \le 11386 \\ 11385 \le x \le 11386 \\ 7850 \le y \le 7852 \\ 7851 \le y \le 7852 \end{cases}$$

Отримаємо: $\begin{matrix} x = 11385; x = 11386 \\ y = 7851; y = 7852 \end{matrix}$

Виконаємо процедуру розшифрування:

$$\begin{pmatrix} 15179 & 24561 \\ 6109 & 9885 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 76 & 77 \\ 26 & 43 \end{pmatrix}$$

$$\begin{pmatrix} 15179 & 24561 \\ 4852 & 7851 \end{pmatrix} \times \begin{pmatrix} 89 & -233 \\ -55 & 144 \end{pmatrix} = \begin{pmatrix} 76 & 77 \\ 23 & 28 \end{pmatrix}$$

А варіант 3-5 дасть 0 і від’ємні значення в результаті.

Тобто є можливі два варіанти за таких умов: «Я лю» та «Я їн». Виходячи з контексту отриманого надалі повідомлення робимо висновки що правильний перший варіант.

Таким чином ми зможемо виправляти помилки при прийомі інформації.

Для обчислень використовувався ресурс [32].

Отже, як висновок випливають такі факти:

1. Матричний метод шифрування даних це асиметричний метод шифрування з двома ключами.

2. Матричний метод шифрування простий, як при шифруванні так і при розшифруванні.

3. Матричний метод шифрування підходить для шифрування символьної інформації.

4. Матричний метод шифрування дозволяє знаходити та виправляти помилки при прийомі даних.

РОЗДІЛ 3. Алгоритми та реалізація матричного методу шифрування на мові Java Script

Представлена програма є тестовим веб-додатком призначення якої ознайомити користувачів з можливостями матричного методу шифрування даних за допомогою чисел Фібоначчі. Інтерфейс написаний на мові інтернет розмітки HTML, а сама програма написана на мові програмування Java Script.

Вибір мов програмування обґрунтований декількома факторами:

1. HTML та Java Script заточені під роботу з інтернетом, та містить широкий спектр фреймворків та можливість роботи з нейромережами. Тобто є багато можливостей для модернізації та створення зручного клієнта.
2. Java Script один з найшвидших мов програмування (швидший за Phyton 3.0+).
3. Багато функцій для обчислення вже написано і містяться у вільному доступі і не потребують встановлення бібліотек.

Із мінусів можна виділити лише не точне обчислення, але ця проблема легко вирішується операцією *.toFixed(0).*

Файл шифратор може бути, як серверним так і клієнтським. Проблема розміщення на сервері js файлу з шифратором також не сильно критична. Справа в тому, що ключ підбирається довільно і може містити дуже великі значення чисел Фібоначчі, наприклад, $F_n, n = 1000000$ чи ще більше. А оскільки ключі ніде не зберігаються то їх підбір є майже неможливим, але якщо ключ все таки підбирати то на це знадобиться дуже багато часу. За таким зразком працює всесвітньо відомий алгоритм шифрування RSA (відомий алгоритм шифрування та невідомі ключі), та веб-додатки такі як Приват24, де ми можемо подивитися їх алгоритми роботи. Інший варіант полягає в тому, щоб зробити підбір ключа клієнтським, а всі обчислення серверними. Тобто алгоритми формування ключів буде закритий. Файл з дешифратором для більшої безпеки можна зробити клієнтським.

Зовнішній вигляд програми представлено на рис 3.1 . Сам додаток поділено на дві частини: верхню – шифратор та нижню – дешифратор.

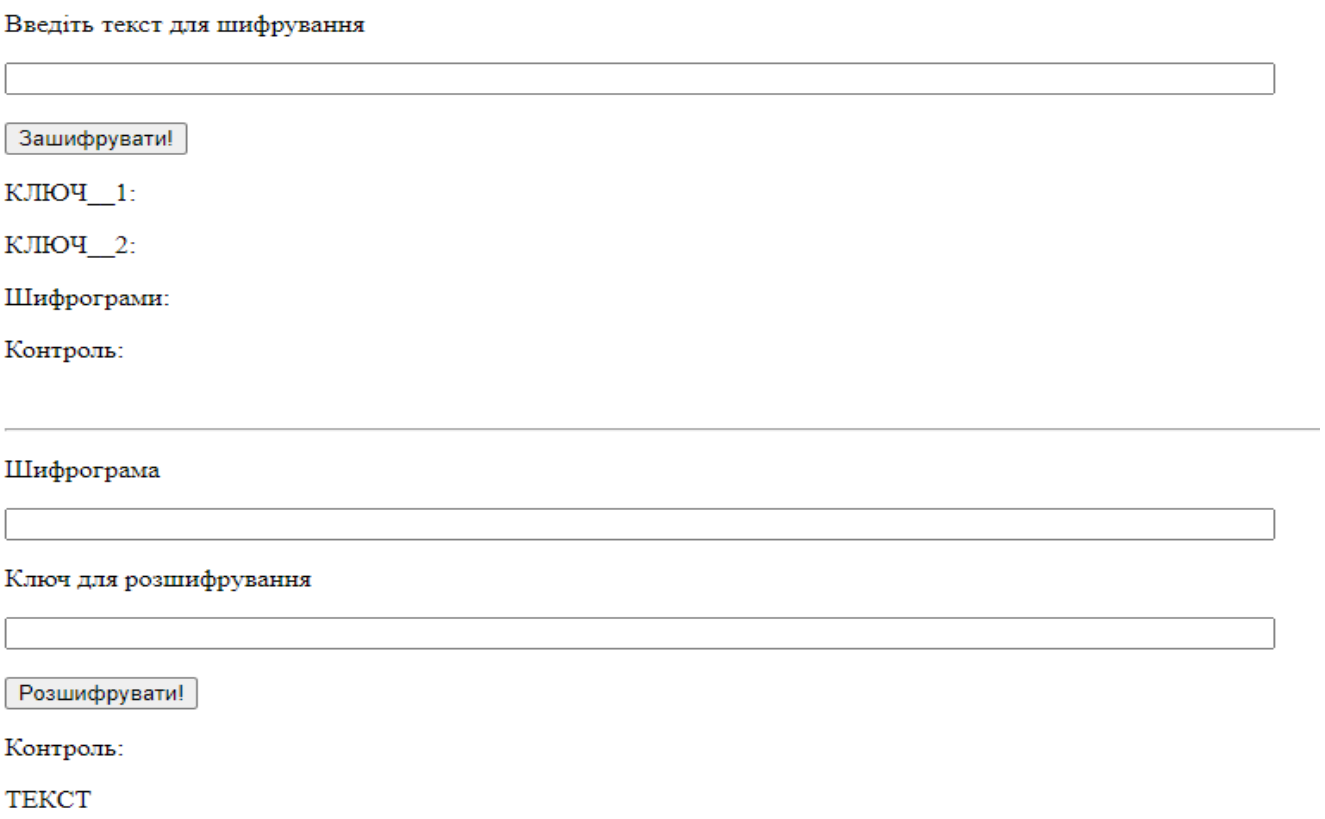

Рис 3.1 – Зовнішній вигляд шифратора

Шифратор містить (По-порядку):

1) Поле для введення тексту;

2) Кнопку дії;

3) Поле «КЛЮЧ_1» де відображається ключ за допомогою якого відбувається шифрування;

4) Поле «КЛЮЧ_2» де відображається ключ за допомогою якого відбувається дешифрування;

5) Поле «Шифрограми» в яких відображається інформація в зашифрованому вигляді.

6) Поле «Контроль» де відображається контрольні детермінанти.

Дешифратор містить (По-порядку):

1) Поле для введення одержаної шифрограми;

2) Поле для введення ключа для дешифрування;

3) Кнопка дії;

4) Поле «Контроль» відображаються контрольні детермінанти (їх можна порівняти з отриманими);

5) Поле «Текст» де відображається розшифрований текст.

Програмний код інтерфейсу веб-додатку:

```
<!DOCTYPE HTML5>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
<body>
    <p>Введіть текст для шифрування</p>
    <input type="text" class="input" id="1" size="100">
    <br/><br/>
    <button onclick="ReadList()"> Зашифрувати! </button>
    <p>КЛЮЧ__1:</p>
    <p id="2"></p>
    <p>КЛЮЧ__2:</p>
    <p id="3"></p>
    <p>Шифрограми:</p>
    <p id="4"></p>
    <p>Контроль:</p>
    <p id="8"></p>
    <br/> <hr/>
    <p>Шифрограма</p>
    <input type="text" class="input" id="5" size="100">
    <p>Ключ для розшифрування</p>
    <input type="text" class="input" id="6" size="100"><br/><br/>
    <button onclick="obr_read()"> Розшифрувати! </button>
    <p>Контроль:</p>
```

```
    <p id="9"></p>
    <p>ТЕКСТ</p>
    <p id="7"></p>
<script src="1.js" type="text/javascript"></script>
</body>
</html>
```

Програмна частина є блоковою. Тобто окремо описуються функції, які виконують окрему дію. Наприклад: одна функція знаходить числа Фібоначчі, інша множить матриці тощо.

Розглянемо функції та звернемо увагу на особливості функцій.

Функція вибору довільного числа[23]:

```
function getRandomInt(max) {
    return Math.floor(Math.random() * Math.floor(max));
}
```

Функція знаходження *n* –го числа Фібоначчі [29]

```
function fibonacci(Num) {
    var A = 0;
    var B = 1;
    var c;
    for (var i = 2; i <= Num; i++) {
       c = B;
       B = A + B;
       A = c;
    }
    return B;
}
```

Якщо ми хочемо працювати з квадратними матрицями 3×3 і більше то вище описану функцію легко замінити для знаходження *p* числа Фібоначчі, наприклад нижче представлена функція для *p=2* чисел Фібоначчі.

```
function fibonacci(Num) {
```

```
    var A = 1;
    var B = 1;
    var C = 1;
    var d;
    for (var i = 4; i <= Num; i++) {
      d = A + C;
      A = B;
      B = C;
      C = d;
    }
    return C;
}
```

Функція множення матриці на число [30]:

```
function multMatrixNumber(a, A) {
 let m = A.length;
 let n = A[0].length;
 let B = new Array();
   for (let i = 0; i < m; i++){
     B[ i ] = new Array();
     for (let j = 0; j < n; j++) {
        B[i][j] = a*A[i][j];
    }
   return B;
}
```

Функція множення матриці на матрицю [30]:

```
function MultiplyMatrix(A,B) {
   let rowsA = A.length;
   let colsA = A[0].length;
   let rowsB = B.length
   let colsB = B[0].length,
```

```
    let C = new Array();
    if (colsA != rowsB) { return false };
    for (let i = 0; i < rowsA; i++) {
        C[i] = new Array(); }
    for (let k = 0; k < colsB; k++) {
     for (let i = 0; i < rowsA; i++) {
        let t = 0;
        for (let j = 0; j < rowsB; j++) {
            t = t + A[i][j]*B[j][k] };
        C[i][k] = t;
      }
     }
    return C;
}
```

Для швидкодії вище описана функція можна замінити частинними випадками для матриць 2×2 чи 3×3, але записана функція для універсальності у загальному вигляді.

Аналогічним чином визначаються функції для знаходження визначника матриць, оскільки вони визиваються декілька разів походу виконання програми, вони спрощенні для швидкодії.

Функція знаходження визначника для квадратної матриці 2×2

```
function Det(A) {
    let det = (A[0][0]*A[1][1]) - (A[0][1]*A[1][0]);
    return det;
}
```

Функція знаходження визначника для квадратної матриці 3×3

```
function Det(A) {
    let det = (A[0][0]*A[1][1]*A[2][2]) + (A[1][0]*A[2][1]*A[0][2]) +
(A[0][1]*A[1][2]*A[2][0]) - (A[0][2]*A[1][1]*A[2][0]) -
(A[0][1]*A[1][0]*A[2][2]) - (A[0][0]*A[2][1]*A[1][2]);
```

```
    return det;
}
```

Загальна функція для знаходження визначника матриці [30]:

```
function Det (A) {
   let N = A.length;
   let B = new Array();
   let d = 1;
   let ex = 0;
   let maxN;
   let maxValue;
   let value;
   for (let i = 0; i < N; ++i) {
     B[i] = new Array();
     for (let j = 0; j < N; ++j) {
        B[i][j] = A[i][j];
        }
    }
   for (i = 0; i < N-1; ++i) {
     maxN = i;
     maxValue = Math.abs(B[i][i]);
     for (j = i+1; j < N; ++j) {
       value = Math.abs(B[j][i]);
       if (value > maxValue){
          maxN = j;
          maxValue = value;
       }
      }
     if (maxN > i) {
       let temp = B[i];
       B[i] = B[maxN];
```

```
            B[maxN] = temp;
             ++ex;
           } else {
              if (maxValue == 0) {
              return maxValue;
              }
         }
          let value1 = B[i][i];
          for ( j = i+1; j < N; ++j) {
            let value2 = B[j][i];
             B[j][i] = 0;
             for (let k = i+1; k < N; ++k) {
                 B[j][k] = (B[j][k]*value1-B[ i ][k]*value2)/d;
                 }
            }
           d = value1;
         }
        if (ex%2) {
           return -B[N-1][N-1];
        } else {
        return B[N-1][N-1];
        }
    }
```

Функція отримання ключа (Q матриці) індивідуальні для кожної розмірності матриці, але принцип дії в них однаковий, наприклад для матриць 2×2 маємо наступну функцію:

```
    function getKEY() {
        let Arr = new Array();
        Arr[0] = new Array();
        Arr[1] = new Array();
```

```
    let ch = getRandomInt(30);
    Arr[0][0] = fibonacci(ch+1);
    Arr[0][1] = fibonacci(ch);
    Arr[1][0] = fibonacci(ch);
    Arr[1][1] = fibonacci(ch-1);
    let S = '';
    for(let i = 0; i<2; i++) {
       for(let j = 0; j<2; j++) {
            S = S + String(Arr[i][j]) + ' '
       }
    }
    let Z = document.getElementById("2");
    Z.innerHTML = S;
    return Arr;
}
```

В рядку *let ch = getRandomInt(30);* число 30 означає, що при виборі F_n числа Фібоначчі ми вибираємо *n* в проміжку від 1 до 30 включно. Замінивши його на інше число ми замінимо проміжок вибору *n*.

Рядки:

```
Arr[0][0] = fibonacci(ch+1);
Arr[0][1] = fibonacci(ch);
Arr[1][0] = fibonacci(ch);
Arr[1][1] = fibonacci(ch-1);
```

говорять, що ми працюємо з ключовою матрицею 2го типу. Замінивши їх на:

```
Arr[0][0] = fibonacci(ch+2);
Arr[0][1] = fibonacci(ch+3);
Arr[1][0] = fibonacci(ch);
Arr[1][1] = fibonacci(ch+1);
```

Перейдемо до ключової матриці першого типу.

Аналогічно функція отримання ключа (Q матриці) для матриць 3×3:

```
function getKEY() {
    let Arr = new Array();
    Arr[0] = new Array();
    Arr[1] = new Array();
    Arr[2] = new Array();
    let ch = getRandomInt(30);
    Arr[0][0] = fibonacci(ch+1);
    Arr[0][1] = fibonacci(ch);
    Arr[0][2] = fibonacci(ch-1);
    Arr[1][0] = fibonacci(ch-1);
    Arr[1][1] = fibonacci(ch-2);
    Arr[1][2] = fibonacci(ch-3);
    Arr[2][0] = fibonacci(ch);
    Arr[2][1] = fibonacci(ch-1);
    Arr[2][2] = fibonacci(ch-2);
    let S = ''
    for(let i = 0; i<3; i++) {
       for(let j = 0; j<3; j++) {
            S = S + String(Arr[i][j]) + ' '
       }
    }
    let Z = document.getElementById("2");
    Z.innerHTML = S;
    return Arr;
}
```

Функція отримання Q^{-1} матриць також індивідуальні особливості, а саме для обернені матриці записані прямими формулами. Для квадратних матриць 2×2 функція має вигляд:

```
function Obern(A) {
```

```
    let B = new Array();
    B[0] = new Array();
    B[1] = new Array();
    B[0][0] = A[1][1];
    B[0][1] = A[1][0] * (-1);
    B[1][0] = A[0][1] * (-1);
    B[1][1] = A[0][0];
    let S = String(Det(A)) + ' ';
    for(let i = 0; i<2; i++) {
      for(let j = 0; j<2; j++) {
          S = S + String(B[i][j]) + ' '
      }
    }
    let Z = document.getElementById("3");
    Z.innerHTML = S;
    return B;
}
```

Функція отримання Q^{-1} матриць 3×3

```
function Obern(A) {
    let B = new Array();
    B[0] = new Array();
    B[1] = new Array();
    B[2] = new Array();
    B[0][0] = A[1][1]*A[2][2] - A[1][2]*A[2][1];
    B[1][0] = (A[1][0]*A[2][2] - A[1][2]*A[2][0]) * (-1);
    B[2][0] = A[1][0]*A[2][1] - A[1][1]*A[2][0];
    B[0][1] = (A[0][1]*A[2][2] - A[0][2]*A[2][1]) * (-1);
    B[1][1] = A[0][0]*A[2][2] - A[0][2]*A[2][0];
    B[2][1] = (A[0][0]*A[2][1] - A[0][1]*A[2][0]) * (-1);
    B[0][2] = A[0][1]*A[1][2] - A[0][2]*A[1][1];
```

```
        B[1][2] = (A[0][0]*A[1][2] - A[0][2]*A[1][0]) * (-1);
        B[2][2] = A[0][0]*A[1][1] - A[0][1]*A[1][0];
        let S = String(Det(A)) + ' ';
        for(let i = 0; i<3; i++) {
          for(let j = 0; j<3; j++) {
              S = S + String(B[i][j]) + ' '
          }
        }
        let Z = document.getElementById("3");
        Z.innerHTML = S;
        return B;
    }
```

Словник для переведення символів в числа:

```
let Slovnik =
['0','1','2','3','4','5','6','7','8','9','а','б','в','г','ґ','д','е','є','ж','з','и','і','ї','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ь','ю','я','А','Б','В','Г','Ґ','Д','Е','Є','Ж','З','И','І','Ї','Й','К','Л','М','Н','О','П','Р','С','Т','У','Ф','Х','Ц','Ч','Ш','Щ','Ю','Я',' '];
```

Функція шифратори і дешифратори будуть дещо відрізнятися в залежності від того яку розмірність матриць ми виберемо. Хоча алгоритми будуть однаковими.

Функція шифратор для матриць 2×2:

```
function ReadList() {
   let elem = document.getElementById("1");
   let St = elem.value;
   let St_length = St.length;
   let Arr_St = new Array();
   let Shif = '';
   let KEY = getKEY();
   let T_matrix = new Array();
   let KEY_2 = Obern(KEY);
```

```
let eee = 4 - (St.length % 4);
if (eee != 4) {
   for(let ij = St_length; ij < (St_length + eee); ij++) {
      St = St + ' ';
   }
}
let Num_Str = '';
let kt_cikl = ((St.length + 1) / 4).toFixed(0);
for(let i = 0; i < kt_cikl; i++ ) {
 Arr_St[i] = new Array();
 for (let ii = 0; ii<2; ii++) {
    Arr_St[i][ii] = new Array();
    for(let iii = 0; iii<2; iii++){
     Arr_St[i][ii][iii] = 0;
    }
 }
}
let y = 0, u = 0, t = 0;
for(let j = 0; j < St.length; j++) {
 for(let k = 0; k<Slovnik.length; k++) {
  if(St[j] == Slovnik[k]) {
    Arr_St[y][u][t] = k;
    t = t + 1;
    if (t == 2) {
      u = u + 1;
      if (t == 2 && u == 2) {
       y = y + 1;
       u = 0;
      }
      t = 0;
```

```
        }
      }
    }
  }
  let Shifr = '';
  let kontrol = '';
  let B = new Array();
  let C = new Array();
  for(let e = 0; e < kt_cikl; e++ ) {
    B = Arr_St[e];
    C = MultiplyMatrix(B, KEY);
    kontrol = kontrol + Number(Det(B)) + ' ';
    for(i = 0; i < 2; i ++) {
      for(j = 0; j < 2; j ++) {
        Shifr = Shifr + C[i][j] + ' ';
      }
    }
  }
  let elem_2 = document.getElementById("4");
  elem_2.innerHTML = Shifr;
  let elem_3 = document.getElementById("8");
  elem_3.innerHTML = kontrol;
}
```

Фунція дешифратор для матриць 2×2:

```
function obr_read() {
  let elem = document.getElementById("5");
  let str = elem.value;
  let elem_2 = document.getElementById("6");
  let KEY_back = elem_2.value;
  console.log(KEY_back);
```

```
let target = ' ', kontrol = '';
let c = 0;
let pos = -1;
while ((pos = str.indexOf(target, pos + 1)) != -1) {
   c = c + 1;
}
let kt_cikl = ((c+1) / 4).toFixed(0);
let Arr_St = new Array();
for(let i = 0; i < kt_cikl; i++ ) {
 Arr_St[i] = new Array();
  for (let ii = 0; ii < 2; ii++) {
    Arr_St[i][ii] = new Array();
    for(let iii = 0; iii < 2; iii++) {
      Arr_St[i][ii][iii] = 0;
    }
 }
}
let Sss = '';
let y = 0, u = 0, t = 0;
for(let j = 0; j < str.length; j++) {
 if (str[j]!= ' ') {
   Sss = Sss + str[j];
 }
 if (str[j] == ' ' || j == (str.length - 1)) {
 Arr_St[y][u][t] = Number(Sss);
   Sss = '';
   t = t + 1;
   if (t == 2) {
     u = u + 1;
     if (t == 2 && u == 2) {
```

```
      y = y + 1;
      u = 0;
     }
     t = 0;
     Sss = '';
   }
  }
}
let KEY_back_num = new Array();
KEY_back_num[0] = new Array();
KEY_back_num[1] = new Array();
let det_t = ' ',det_c,stt = '';
let r = 0, q = 0;
let posz = 0;
let zzz = KEY_back.indexOf(' ', posz);
let jj;
for(jj = 0; jj < zzz; jj++) {
  det_t = det_t + KEY_back[jj];
}
det_c = Number(det_t);
for(jj = (zzz+1); jj < KEY_back.length; jj++) {
  if (KEY_back[jj] != ' ') {
    stt = stt + KEY_back[jj];
  }
  if (KEY_back[jj] == ' ' || jj == (KEY_back.length - 1)) {
    KEY_back_num[r][q] = Number(stt);
    console.log(stt, Number(stt));
    stt = '';
    q = q + 1;
    if(q == 2) {
```

```
        q  = 0;
        r = r + 1;
       }
      }
     }
     let B = new Array();
     let C = new Array();
     let E_m = new Array();
     let Strok = '';
     for(let er = 0; er < kt_cikl; er++ ) {
      B = Arr_St[er];
      kontrol = kontrol + String(Det(B) * det_c) + ' ';
      E_m = MultiplyMatrix(B, KEY_back_num);
      C = multMatrixNumber((1/det_c), E_m);
      for(let y_q = 0; y_q < 2; y_q++) {
       for(let y_r = 0; y_r < 2; y_r++) {
        Strok = Strok + Slovnik[C[y_q][y_r].toFixed(0)];
       }
      }
     }
     let elem_3 = document.getElementById("7");
     elem_3.innerHTML = Strok;
     let elem_4 = document.getElementById("9");
     elem_4.innerHTML = kontrol;
}
```

Результати роботи програми приведено нижче.

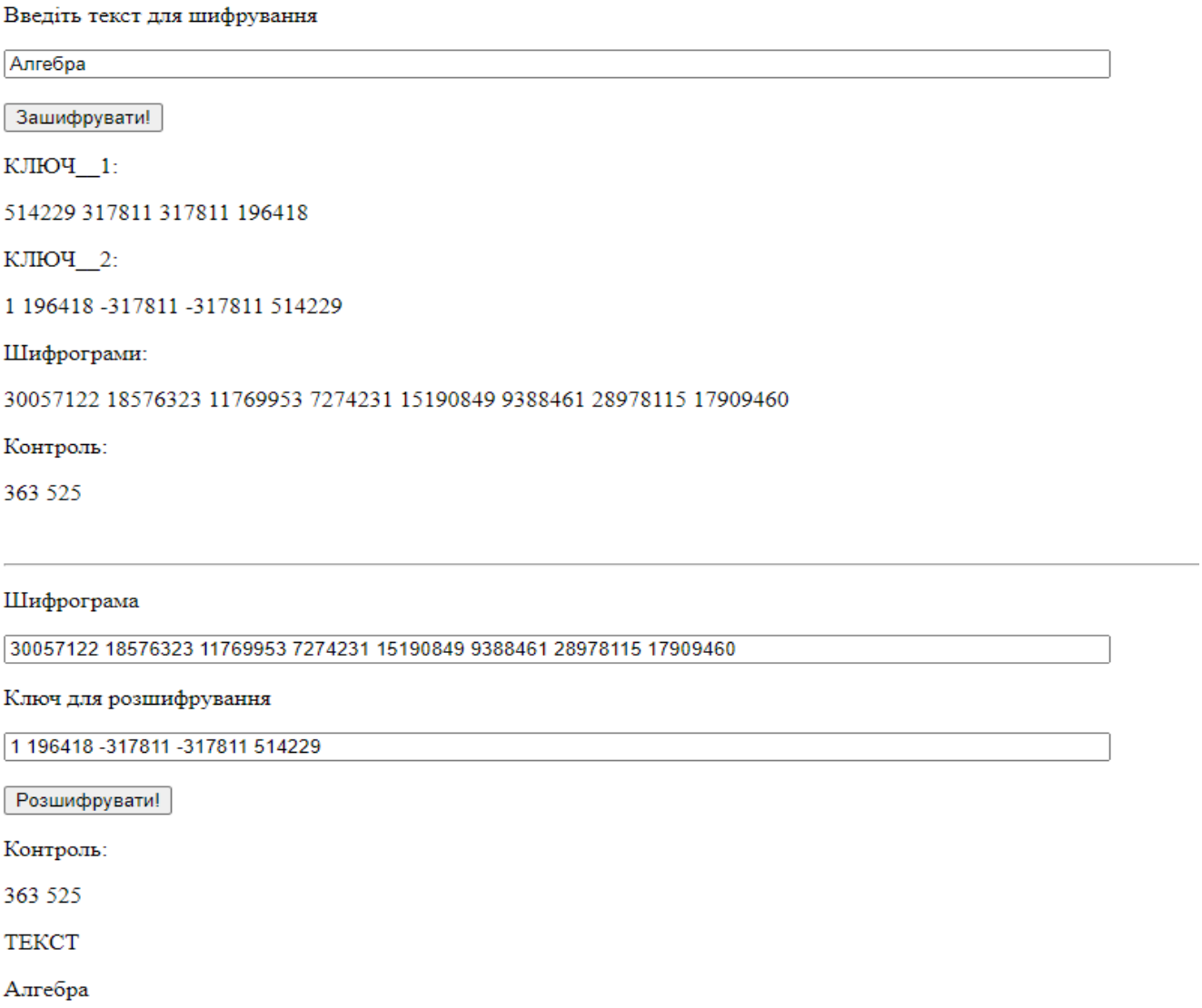

Рис 3.2 – Результати шифрування і дешифрування слова «Алгебра» за допомогою матриць 2×2

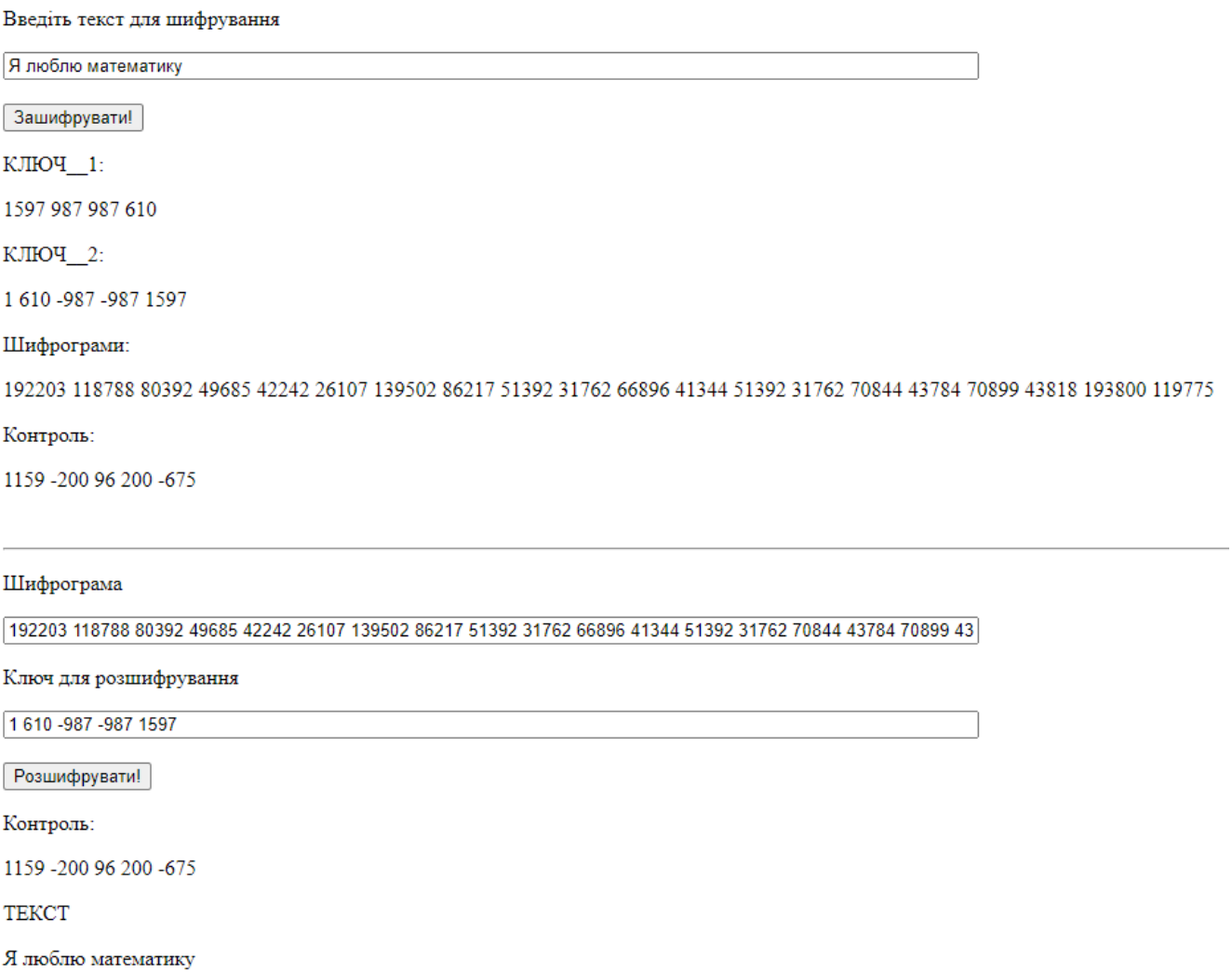

Рис 3.3 – Результати шифрування і дешифрування фрази «Я люблю математику» за допомогою матриць 2×2

Функція шифратор для матриць 3×3:

```
function ReadList() {
    let elem = document.getElementById("1");
    let St = elem.value;
    let St_length = St.length;
    let Arr_St = new Array();
    let Shif = '';
    let KEY = getKEY();
    let T_matrix = new Array();
    let KEY_2 = Obern(KEY);
        let eee = 9 - (St.length % 9);
```

```
 if (eee != 9) {
 for(let ij = St_length; ij < (St_length + eee); ij++) {
     St = St + ' ';
     }
}
let Num_Str = '';
let kt_cikl = ((St.length + 1) / 9).toFixed(0);
for(let i = 0; i < kt_cikl; i++ ) {
     Arr_St[i] = new Array();
     for (let ii = 0; ii<3; ii++){
        Arr_St[i][ii] = new Array();
        for(let iii = 0; iii<3; iii++){
           Arr_St[i][ii][iii] = 0;
        }
     }
}
let y = 0, u = 0, t = 0;
for(let j = 0; j < St.length; j++) {
   for(let k = 0; k<Slovnik.length; k++) {
        if(St[j] == Slovnik[k]) {
           Arr_St[y][u][t] = k;
           t = t + 1;
           if (t == 3) {
             u = u + 1;
             if (t == 3 && u == 3) {
               y = y + 1;
               u = 0;
               t = 0;
             }
             t = 0;
```

```
                }
            }
        }
    }
    let kontrol = '';
    let Shifr = '';
    let B = new Array();
    let C = new Array();
    for(let e = 0; e < kt_cikl; e++ ) {
        B = Arr_St[e];
        C = MultiplyMatrix(B,KEY);
        kontrol = kontrol + Number(Det(B)) + ' ';
        for(i = 0; i<3; i ++){
          for(j = 0; j<3; j ++){
              Shifr = Shifr + C[i][j] + ' '
    }}}
    let elem_2 = document.getElementById("4");
    elem_2.innerHTML = Shifr;
    let elem_3 = document.getElementById("8");
    elem_3.innerHTML = kontrol;
}
```

Функція дешифратор для матриць 3×3:

```
function obr_read() {
    let elem = document.getElementById("5");
    let str = elem.value;
    console.log(str);
    let elem_2 = document.getElementById("6");
    let KEY_back = elem_2.value;
    let target = ' ';
    let c = 0;
```

```
let pos = -1;
while ((pos = str.indexOf(target, pos + 1)) != -1) {
    c = c + 1;
}
let kt_cikl = ((c+1) / 9).toFixed(0);
let Arr_St = new Array();
for(let i = 0; i < kt_cikl; i++ ) {
       Arr_St[i] = new Array();
       for (let ii = 0; ii<3; ii++){
          Arr_St[i][ii] = new Array();
          for(let iii = 0; iii<3; iii++){
             Arr_St[i][ii][iii] = 0;
          }
       }
  }
let Sss = '';
let y = 0, u = 0, t = 0;
for(let j = 0; j < str.length; j++) {
      if (str[j]!= ' ') {
       Sss = Sss + str[j];
      }
       if (str[j] == ' ' || j == (str.length - 1)) {
       Arr_St[y][u][t] = Number(Sss);
       Sss = '';
       t = t + 1;
       if (t == 3) {
           u = u + 1;
           if (t == 3 && u == 3) {
              y = y + 1;
              u = 0;
```

```
                }
                t = 0;
             Sss = '';
            }
          }
        }
let KEY_back_num = new Array();
KEY_back_num[0] = new Array();
KEY_back_num[1] = new Array();
KEY_back_num[2] = new Array();
let det_t = ' ',det_c, stt = '';
let r = 0, q = 0;
let posz = 0;
let zzz = KEY_back.indexOf(' ', posz);
let jj;
      for(jj = 0; jj < zzz; jj++) {
          det_t = det_t + KEY_back[jj];
      }
      det_c = Number(det_t);
      for(jj = (zzz+1); jj < KEY_back.length; jj++) {
         if (KEY_back[jj] != ' ') {
           stt = stt + KEY_back[jj];
         }
         if (KEY_back[jj] == ' ' || jj == (KEY_back.length - 1)) {
           KEY_back_num[r][q] = Number(stt);
           stt = '';
           q = q + 1;
           if(q == 3) {
             q  = 0;
             r = r + 1;
```

```
                }
            }
        }
    let B = new Array();
    let C = new Array();
    let kontrol = '';
    let E_m = new Array();
    let Strok = '';
        for(let er = 0; er < kt_cikl; er++ ) {
            B = Arr_St[er];
            kontrol = kontrol + String(Det(B) * det_c) + ' ';
            C = MultiplyMatrix(B,KEY_back_num);
            E_m = multMatrixNumber((1/det_c), C);
            for(let y_q = 0; y_q < 3; y_q++) {
              for(let y_r = 0; y_r < 3; y_r++) {
              Strok = Strok + Slovnik[E_m[y_q][y_r].toFixed(0)];
            }
            }
        }
    let elem_3 = document.getElementById("7");
    elem_3.innerHTML = Strok;
    let elem_4 = document.getElementById("9");
    elem_4.innerHTML = kontrol;
}
```

Результати роботи програми приведено нижче.

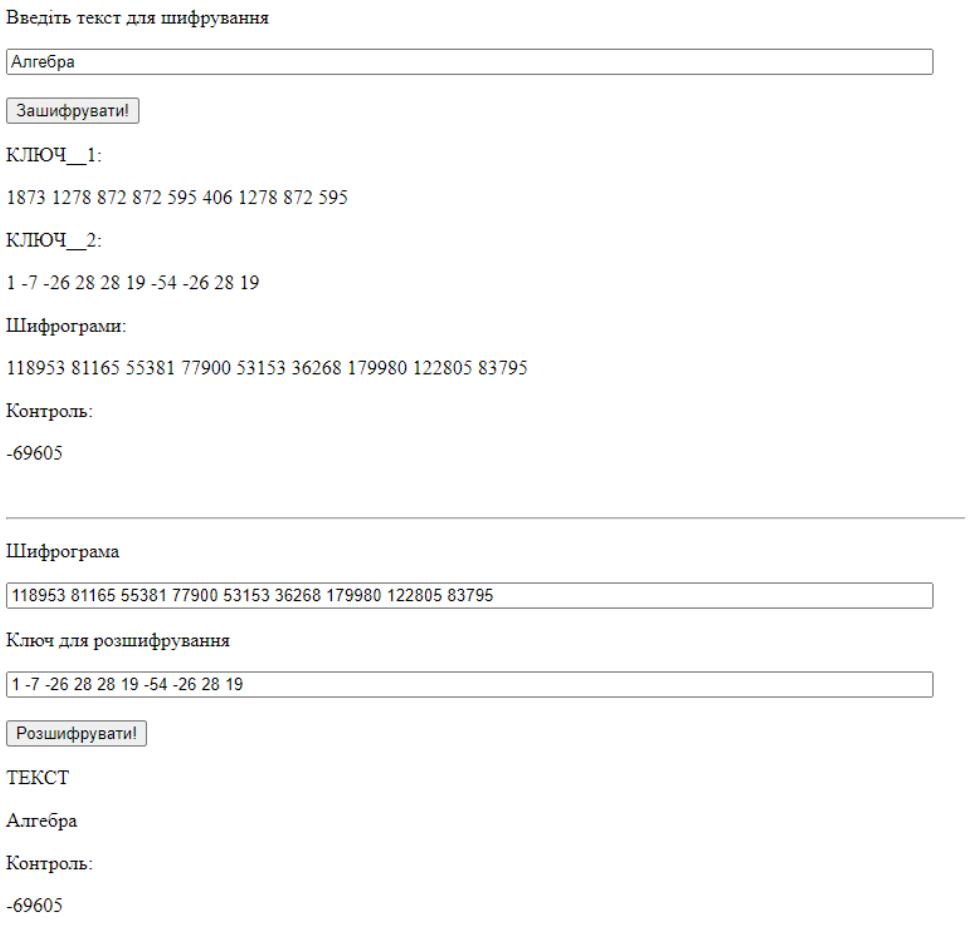

Рис 3.4 – Результати шифрування і дешифрування слова «Алгебра» за допомогою матриць 3×3

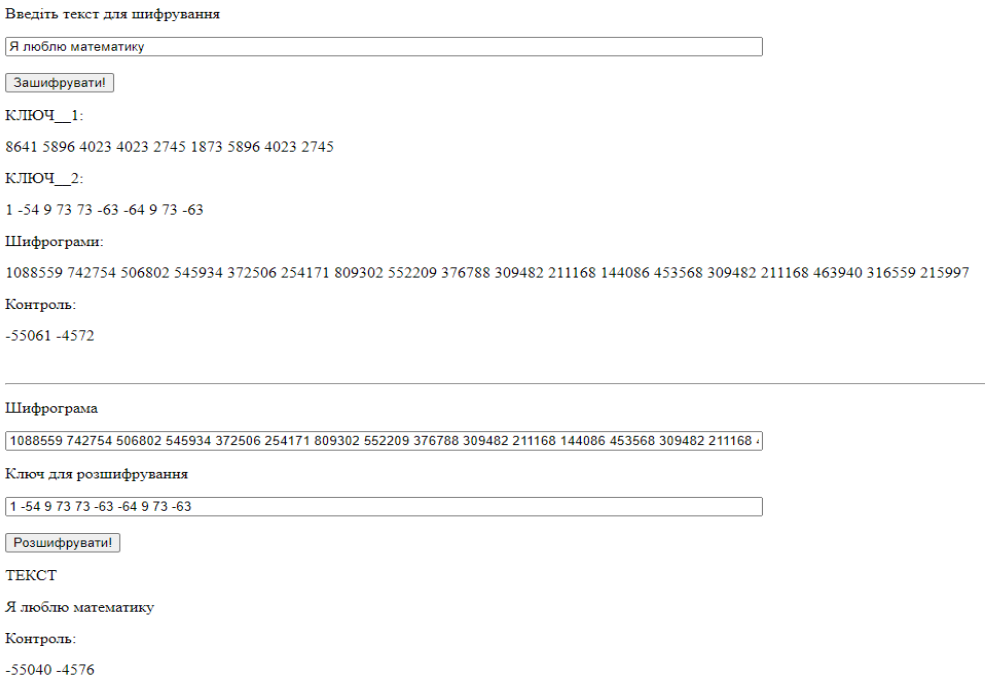

Рис 3.5 – Результати шифрування і дешифрування фрази «Я люблю математику» за допомогою матриць 3×3

При написанні коду програми були використані наступні джерела [24; 25; 26; 28].

Отже, як висновок випливають такі факти:

1. Програма шифратор модульна і містить наступні функції - блоки:
   - вибір довільного числа та знаходження чисел Фібоначчі;
   - множення матриць;
   - множення матриці на число;
   - знаходження визначника матриці;
   - знаходження оберненої матриці;
   - формування ключової матриці;
   - функція шифратор;
   - функція дешифратор.
2. Універсальної програми для всіх матриць не існує через неможливість написання універсального алгоритму формування Q матриць.

ВИСНОВКИ

В ході наукового дослідження поставлені цілі були досягнуті в повному обсязі. Були зібрані та систематизовані знання про використання чисел Фібоначчі в криптографії. На основі цих даних був розширений, на основі робіт Ю. Грицюка, матричний метод шифрування за допомогою чисел Фібоначчі. Тепер за допомогою цього методу можна шифрувати текстову інформацію, а також через введення словника стало можливо легко виправити помилку в отриманому повідомленні.

Також була написана програма, яка демонструє можливості даного криптографічного методу. Дана розробка може послужити основою для захищеного клієнта миттєвого обміну текстовими повідомленнями.

Коротко про сам метод шифрування. Матричний метод шифрування даних це асиметричний метод шифрування з двома ключам. Суть матричного методу шифрування полягає в тому, що існує повідомлення: $m_1...m_n$ яке потрібно зашифрувати. Для цього записуємо його у вигляді квадратної матриці M, і вибираємо ключ Q у такому ж вигляді квадратної матриці. Перемноживши матриці M та Q, отримаємо матрицю T, яка і є нашим зашифрованим повідомленням. Для розшифрування повідомлення T помножимо на матрицю Q^{-1}, яка є оберненою матрицею до Q.

Переваги:

- Висока крипостійкість. Це пов’язано зі складністю відшукання ключа, а отже з неможливістю частотного аналізу.
- Простий алгоритм шифрування / дешифрування – матриці досить просто перемножуються.
- Перевірка помилок при передачі повідомлень в реальному часі за рахунок контрольних детермінантів.

Недоліки:

- Важкість знаходження Q матриць ключів.

- Надмірність (збільшення) тексту, що передається, у відношенні до вихідного, через множення матриць.

В майбутньому більш професійні програмісти скориставшись можливостями нейронних мереж зможуть написати повністю автоматизоване програмне забезпечення, яке зможе виявляти помилки при прийомі даних. Крім того подальше вивчення матриць та чисел Фібоначчі зможе допомогти в складенні алгоритму формування Q матриць $n \times n, n = 3,4...$ без використання p – чисел Фібоначчі. Таким чином програму можна зробити універсальну в якій можна буде вибирати довжину ключа чим підвищувати криптостійкість або швидкодію програми. Бо ми зможемо написати один універсальний алгоритм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алфьоров А.П., Зубов А.Ю., Кузьмін А.С., Черьомушкін А. В. Основи криптографії. М .: Геліос АРВ, 2001. 479 с.
2. Бауер Ф. Розшифровані секрети. Методи і принципи криптології. М .: Світ, 2007. 550 с.
3. Воробьев Н.Н. Числа Фибоначчи. – М. : Изд-во "Наука", 1978. – 141 с.
4. Гантмахер Ф.Р. Теория матриц / Ф.Р. Гантмахер. – М. : Изд-во "Физматлит", 2010. – 560 с.
5. Голуб Дж. Матричные вычисления / Дж. Голуб, Ч. ван Лоун. – М. : Изд-во "Мир", 1999. – 548 с.
6. Грицюк П.Ю. Особливості реалізації матричної Афінної криптосистеми захисту інформації / П.Ю. Грицюк, Ю.І. Грицюк // Науковий вісник НЛТУ України : зб. наук.-техн. праць. – Львів : РВВ НЛТУ України. – 2015. – Вип. 25.5. – С. 346-356.
7. Грицюк Ю.І., Цубера С.В. Числа Фібоначчі в алгоритмах шифрування числової інформації – Збірник науково-технічних праць. – Львів : РВВ НЛТУ України. – 2009. Вип. 19.12. – 308-314 с.
8. Грицюк Ю.І. Методи і засоби генерування Q_p-матриць Фібоначчі - ключів для реалізації криптографічних перетворень / Ю. І. Грицюк, П. Ю. Грицюк // Науковий вісник НЛТУ України. - 2015. - Вип. 25.6. - С. 334-351.1.
9. Ємець В. Сучасна криптографія: Основні поняття / В. Ємець, А. Мельник, Р. Попович. – Львів : Вид-во БаК, 2003. – 144 с.
10. Красиленко В.Г. Матричні афінно-перестановочні алгоритми для шифрування та дешифрування зображень / В.Г. Красиленко, С.К. Грабовляк // Системи обробки інформації: зб. наук. праць. – Харків : Вид-во ХУПС ім. Івана Кожедуба. – 2012. – Вип. 3(101), т.2. – С. 53-61.
11. Мао В. Сучасна криптографія. Теорія та практика. М : Вільямс, 2005. 763с.

12. Сингх С. Книга шифрів. Таємна історія шифрів і їх розшифровки. М .: Аст, Астрель, 2006. 447 с.
13. Стахов А.П. Введение в алгоритмическую теорию измерения / А.П. Стахов. – М. : Изд-во "Советское Радио", 1977. – 246 с.
14. Стахов А.П. Гармония Мироздания и Золотое Сечение: древнейшая научная парадигма и ее роль в современной науке, математике и образовании / А.П. Стахов. – У 2-ох ч. – Ч. 1.
15. Стахов А.П. Гармония Мироздания и Золотое Сечение: древнейшая научная парадигма и ее роль в современной науке, математике и образовании / А.П. Стахов. – У 2-ох ч. – Ч. 2.
16. Хорошко В.О. Методи та засоби захисту інформації : навч. посібн. / В.О. Хорошко, А.О. Четков. – К. : Вид-во "Юніор", 2003. – 502 с.
17. Шнайер Б. Прикладна криптографія. Протоколи, алгоритми, вихідні тексти на мові Сі. М .: Тріумф, 2003. 806 с.
18. Hoggat, V.E. Fibonacci and Lucas Numbers / V.E. Hoggat. – Houghton-Mifflin, Palo Alto, California, 1969.
19. Stakhov A.P. Brousentsov's ternary principle, Bergman's number system and ternary mirrorsymmetrical arithmetic / A.P. Stakhov // The Computer Journal. – 2002. – Vol. 45, No. 2. – Pp. 222-236.
20. Stakhov A.P. Introduction into Fibonacci Coding and Cryptography / A.P. Stakhov, V. Massingua, A.A. Sluchenkova. – Харьков : Изд-во "Основа" Харьковского университета, 1999 г.
21. URL: http://algolist.manual.ru/defence/intro.php
22. URL: https://books.ifmo.ru/file/pdf/901.pdf
23. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript
24. URL: https://github.com/
25. URL: https://habr.com/ru/
26. URL: http://htmlbook.ru/
27. URL: https://ichip.ru/tekhnologii/chto-takoe-kriptografiya-prosto-o-slozhnom-224532

28. URL: https://learn.javascript.ru/
29. URL: http://lits.ua/article/blogs/algorithms-fibonacci-numbers
30. URL: http://mathhelpplanet.com/static.php?p=javascript-operatsii-nad-matritsami
31. URL: https://prog-cpp.ru/fibonacci/
32. URL: https://ru.onlinemschool.com/math/assistance/matrix/multiply/
33. URL: https://tproger.ru/tag/javascript/
34. URL: https://uk.wikipedia.org/wiki/Числа_Фібоначчі
35. URL: https://uk.wikipedia.org/wiki/Фібоначчі
36. URL: https://works.doklad.ru/view/b3BAOOFo2ls.html