МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки

Кафедра Інформатики

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка додатку для здійснення криптоаналізу асиметричних алгоритмів шифрування

Виконав студент 2 курсу групи МІС-19
спеціальності 122 Комп'ютерні науки
Сахаров Володимир Олександрович

Керівник к.т.н., доцент
Гнатовська Ганна Арнольдівна

Рецензент к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Одеса 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки
Кафедра Інформаційних технологій
Рівень вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
_____
" 26 " жовтня 2020 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сахарова Володимира Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка додатку для здійснення криптоаналізу асиметричних алгоритмів шифрування

керівник роботи Гнатовська Ганна Арнольдівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від " 16" жовтня 2020р. №194 «С»

2. Строк подання студентом роботи 7 грудня 2020р.

3. Вихідні дані до роботи Ресурси мережі Інтернет, асиметрична криптосистема RSA, біоінспірірований мурашиний алгоритм, мова програмування C#, середовище Microsoft Visual Studio, програмний пакет .NetFramework.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз алгоритмів асиметричних криптосистем та алгоритмів криптоаналізу. Дослідження методів криптоаналізу асиметричних та симетричних криптосистем. Інтелектуальні методи криптоаналізу. Розробка додатку для виконання криптоаналізу мурашиним алгоритмом. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 26 жовтня 2020р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Аналіз алгоритмів асиметричних криптосистем та алгоритмів криптоаналізу	26.10.20 – 30.10.20	98%	відмін.
2	Основні види атак, принципи криптоаналізу	30.10.20 – 03.11.20	96%	відмін.
3	Дослідження методів криптоаналізу асиметричних та симетричних криптосистем	04.11.20 – 07.11.20	98%	відмін.
4	Застосування мурашиного алгоритму в криптоаналізі	08.11.20 – 14.11.20	96%	відмін.
5	Розробка додатку для виконання криптоаналізу мурашиним алгоритмом	15.11.20 – 25.11.20	96%	відмін.
6	Рубіжна атестація	19.11.20	96%	відмін.
7	Тестування різних конфігурацій та параметрів алгоритму	25.11.20 – 30.11.20	96%	відмін.
8	Оформлення пояснювальної записки та презентації	01.12.20 – 06.12.20	98%	відмін.
9	Подання роботи на кафедру	07.12.20		
10	Перевірка на плагіат	08.12.20		
11	Рецензування	16.12.20		
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)		97%	відмін.

Студент _____ Сахаров В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гнатовська Г.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема магістерської роботи «Розробка додатку для здійснення криптоаналізу асиметричних алгоритмів шифрування».

Актуальність магістерської роботи полягає в застосуванні біоінспірірованіх алгоритмів для розв'язання завдань криптоаналізу асиметричних алгоритмів шифрування заснованих на методі факторизації чисел та розробки програмного додатку, що реалізує криптоаналіз асиметричних криптосистем за допомогою мурашиного алгоритму.

Об'єкт дослідження – процеси виконання криптоаналізу асиметричних криптосистем, заснованих на методі факторизації складених чисел.

Предмет дослідження – біоінспірірований мурашиний алгоритм для розв'язання завдань криптоаналізу асиметричних алгоритмів шифрування заснованих на методі факторизації чисел.

Мета роботи – розробка програмного додатку для виконання криптоаналізу асиметричних алгоритмів шифрування на основі факторизації складених чисел за допомогою мурашиного алгоритму.

В роботі було проведено аналіз алгоритмів асиметричних криптосистем, здійснено дослідження методів криптоаналізу асиметричних та симетричних криптосистем, а також інтелектуальних методів кріптоаналізу; здійснена розробка програмного додатку для розв'язання завдання криптоаналізу асиметричних алгоритмів шифрування на основі факторизації складених чисел, засобами мурашиного алгоритму. Практична цінність роботи полягає в тому, що розроблений додаток може бути використано для тестування різних конфігурацій мурашиного алгоритму з різними наборами початкових параметрів при здійсненні криптоаналізу асиметричних алгоритмів шифрування.

Ключові слова: КРІПТОАНАЛІЗ, АСИМЕТРИЧНИЙ АЛГОРИТМ ШИФРУВАННЯ, МУРАШИНИЙ АЛГОРИТМ.

Магістерська робота містить 74 сторінки, 2 таблиці, 13 рисунків, 11 посилань та 5 листів додатків.

ANNOTATION

The theme of master's work is "Development of an application for cryptanalysis of asymmetric encryption algorithms".

The relevance of the master's thesis is the use of bioinspired algorithms to solve problems of cryptanalysis of asymmetric encryption algorithms based on the method of factoring numbers and the development of a software application that implements cryptanalysis of asymmetric cryptosystems using an ant algorithm.

The object of research is the processes of cryptanalysis of asymmetric cryptosystems based on the method of factorization of composite numbers.

The subject of research is a bioinspired ant algorithm for solving problems of cryptanalysis of asymmetric encryption algorithms based on the method of factorization of numbers.

The purpose of the work is to develop a software application for cryptanalysis of asymmetric encryption algorithms based on factorization of compound numbers using an ant algorithm.

The analysis of asymmetric cryptosystems and cryptanalysis algorithms was carried out in the work, research of cryptanalysis methods of asymmetric and symmetric cryptosystems, as well as intelligent cryptanalysis methods was carried out; the development and implementation of a software application for solving the problem of cryptanalysis of asymmetric encryption algorithms based on the factorization of compound numbers, by means of an ant algorithm. The practical value of the work is that the developed application can be used to test different configurations of the ant algorithm with different sets of initial parameters when performing crypto analysis of asymmetric encryption algorithms.

Key words: CRYPTOANALYSIS, ASYMMETRIC ENCRYPTION ALGORITHM, ANT ALGORITHM.

The master's thesis contains 74 pages, 2 tables, 13 figures, 11 references.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ АЛГОРИТМІВ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ ТА АЛГОРИТМІВ КРИПТОАНАЛІЗУ 11
1.1 Асиметрична криптосистема RSA 12
1.2 Алгоритм шифрування Ель Гамаля 16
1.3 Порівняльний аналіз асиметричних криптосистем 18
1.4 Основні види атак, принципи криптоаналізу 18
1.5 Класифікація атак на симетричні криптоалгоритми 22
1.6 Класифікація атак на асиметричні криптоалгоритми 24
2 ДОСЛІДЖЕННЯ МЕТОДІВ КРИПТОАНАЛІЗУ АСИМЕТРИЧНИХ ТА СИМЕТРИЧНИХ КРИПТОСИСТЕМ 29
2.1 Криптоаналіз систем шифрування, заснованих на складності завдання дискретного логарифмування 29
2.2 Криптоаналіз систем шифрування, заснованих на складності задачі факторизації 30
2.3 ρ-алгоритм факторизації 32
2.4 Диференціальний криптоаналіз 37
2.5 Лінійний криптоаналіз 40
2.6 Постановка задачі 42
3 ІНТЕЛЕКТУАЛЬНІ МЕТОДИ КРИПТОАНАЛІЗУ 43
3.1 Застосування нейронної мережі в завданнях криптоаналізу 43
3.2 Використання генетичних алгоритмів при дійсненні криптоаналізу 44
3.3 Застосування мурашиного алгоритму в криптоаналізі 48
3.4 Модифікації класичного алгоритму 56
3.5 Виконання криптоаналізу мурашиним алгоритмом 58
4 РОЗРОБКА ДОДАТКУ ДЛЯ ВИКОНАННЯ КРИПТОАНАЛІЗУ МУРАШИНИМ АЛГОРИТМОМ 62
4.1 Програмне середовище розробки 62

4.2 Проектування діаграми класів 63
4.3 Основні методи, використовувані додатком..... 65
4.4 Алгоритм роботи з додатком 69
4.5 Тестування різних конфігурацій та параметрів алгоритму 72
ВИСНОВКИ..... 74
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 75
ДОДАТОК А Лістинг коду методу Ant.Iterator()..... 77
ДОДАТОК Б Матриця суміжності в текстовому поданні……....81

ВСТУП

Одним з головних аспектів аудиту безпеки інформаційних систем є оцінка надійності криптографічних алгоритмів, що вони використовують. Так, за даними звіту CSI / FBI Computer Crime and Security Survey 2019 року, зафіксовано збитки фізичних і юридичних осіб на суму понад 3,5 мільярда доларів. Найбільш затребувані з фінансовою точки зору жалоби стосувались компрометації ділової електронної пошти, шахрайства з довіренням, а також підробки або імітації облікових записів осіб або постачальників, відомих жертв для збору приватної або фінансової інформації. Змінюються підходи та застосовуються нові тактики та методи для здійснення вже існуючих видів шахрайства [1][1)].

З огляду на постійне зростання обчислювальних потужностей і методів захисту даних останнім часом розробляється відносно новий розділ криптоаналізу – natural (bio inspired) calculation, біоінспіровані обчислення, що розкривають математичні методи прийняття рішень на основі природних механізмів. Вони включають такі різноманітні методи, як моделювання відпалу, методи еволюційного моделювання, генетичні алгоритми, методи еволюційної адаптації, алгоритми ройового інтелекту, мурашині алгоритми. У моделях і алгоритмах таких природних механізмів основною частиною є побудова вихідної моделі і правил згідно з якими модель може змінюватися (еволюціонувати). Схем природних обчислень вже існує безліч, в тому числі генетичний алгоритм, ройові алгоритми, різноманітні стратегії, що розвиваються самостійно, еволюційне програмування.

На сьогодні відомо застосування рішень на основі біоінспірованних методів для оптимізації широкого спектру завдань, в тому числі задач криптоаналізу. Криптоаналіз і криптографія мають протилежні напрямки і є одними з підмножин криптології. Звичайний, лінійний чи диференційний крип-

---

[1)] [1] Internet Crime Report Released – FBI/ URL: https://www.fbi.gov /news/stories/2019-internet-crime-report-released-021120 (дата звернення: 16.05.2020)

тоаналіз або складний в реалізації, або вузько орієнтований, або не підходить з інших міркувань (час злому, витрати на злом і т. п). З іншого боку через складність співвідношень, які використовуються при аналізі доводиться вдаватися до допомоги різних діаграм і таблиць, які частіше всього не охоплюють весь спектр значень, і не можуть бути перераховані. Тому для вирішення такого завдання зручно використовувати природні алгоритми. З огляду на емерджентність (виникнення в системі властивостей, які не спостерігаються в її елементів) алгоритмів, багато з біоінспірованних алгоритмів можуть давати більш точні і більш швидкі результати ніж їх нативні аналоги. Багато еволюційних алгоритмів з безлічі природних реалізують так званий «сліпий» пошук, призводить до генерації рішень з порушеннями, збільшення числа ітерацій і, як наслідок, вимагають додаткового контролю з боку дослідника. З іншого боку, використання таких методів, як ройові алгоритми, дозволяє реалізувати контроль над системою засобами самої системи. Тому цікаві евристичні системи, такі як колективний інтелект, в яких здійснюється поетапна побудова рішення задачі (додавання приватного рішення до вже побудованого загального рішення). До методів даного типу відносяться такі системи, як мурашині і бджолині алгоритми, метод рою частинок, алгоритми гравітаційного пошуку і багато інших. Ці механізми забезпечують ефективну адаптацію флори і фауни до навколишнього середовища протягом декількох мільйонів років. З криптоаналізу можна відзначити алгоритм RSA, складність якого визначається складністю факторизації великих чисел. Факторизація – одна з головних проблем криптоаналізу, і застосування ройових алгоритмів для вирішення даного завдання може бути цілком виправдано. У разі симетричних криптосистем комбінування методів мурашиного і генетичного алгоритмів дає хороші результати [2][1)].

Актуальність магістерської роботи полягає в необхідності розробки програмного забезпечення, яке розв'язку завдання криптоаналізу асиметрич-

---

[1)] [2] С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития – PDF Free Download / URL: http://docplayer.ru/25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya.html (дата звернення: 05.06.2020)

них алгоритмів шифрування на основі факторизації складених чисел засобами застосування біоінспірованного мурашиного алгоритму.

Об'єкт дослідження – процеси виконання криптоаналізу асиметричних криптосистем, заснованих на методі факторизації складених чисел, за допомогою мурашиного алгоритму з застосуванням розробленого програмного додатку.

Предмет дослідження – біоінспірірований мурашиний алгоритм для розв'язання завдань криптоаналізу асиметричних алгоритмів шифрування заснованих на методі факторизації чисел.

Метою магістерської роботи є розробка програмного додатку для виконання криптоаналізу асиметричних алгоритмів шифрування на основі факторизації складених чисел за допомогою мурашиного алгоритму.

Для досягнення поставленої мети в роботі сформульовані такі завдання: проведення аналізу алгоритмів асиметричних криптосистем та алгоритмів криптоаналізу; виконати дослідження методів криптоаналізу асиметричних та симетричних криптосистем; дослідити і обрати метод інтелектуального крипто аналізу; здійснити розробку програмного додатку для виконання криптоаналізу мурашиним алгоритмом; провести тестування різними наборами вхідних даних та виконати аналіз отриманих результатів.

1 АНАЛІЗ АЛГОРИТМІВ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ ТА АЛГОРИТМІВ КРИПТОАНАЛІЗУ

Асиметричні алгоритми використовують для розв'язання двох основних задач: шифрування даних і цифрового підпису.

Принциповою відмінністю асиметричної криптосистеми від криптосистеми симетричного шифрування є те, що для шифрування інформації і її наступного розшифрування використовують різні ключі:

– відкритий ключ K використовують для шифрування інформації, і він визначається із секретного ключа k;
– секретний ключ k використовують для дешифрування інформації, яка зашифрована за допомогою парного йому відкритого ключа K.

Секретний і відкритий ключі генеруються попарно. Секретний (приватний) ключ повинен залишатися у його власника; його необхідно надійно захистити від несанкціонованого доступу (по аналогії з ключем шифрування в симетричних алгоритмах). Копія відкритого ключа (публічного) повинна знаходитись у кожного абонента криптографічної мережі, з яким відбувається обмін інформацією власника секретного ключа. Відзначимо характерні особливості асиметричних криптосистем: відкритий ключ K і шифротекст S можливо відправляти по незахищеним каналам зв'язку, тобто зловмиснику вони можуть бути відомі [3][1)]:

– алгоритми шифрування і дешифрування є відкритими. У. Діффі і М. Хеллман сформулювали вимоги, виконання яких забезпечує високу криптостійкість асиметричної системи;
– визначення пари ключів (K, k) одержувачем на базі початкової умови повинно бути простим. Відправник, знаючи відкритий ключ одержувача K і повідомлення C, може легко визначити криптограму S;

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

– одержувач, використовуючи свій секретний ключ k і криптограму S, може легко відновити початкове повідомлення C;
– зловмисник, знаючи відкритий ключ одержувача K, у разі спроби визначити секретний ключ одержувача k наштовхується на нездоланну обчислювальну проблему;
– зловмисник, знаючи відкритий ключ одержувача K і криптограму S, у разі спроби визначити початкове повідомлення C наштовхується на нездоланну обчислювальну проблему.

Однак далеко не всі стійкі алгоритми використовують на практиці. Декотрі з них потребують дуже великі ключі. Наприклад, розмір відкритого ключа криптосистеми HFE (Hidden Fields Equations) може достигати десятків мегабайт, що ускладнює розповсюдження таких ключів. При використанні декотрих алгоритмів розмір шифротексту значно перевищує розмір відповідного йому відкритого тексту. Концепція асиметричних криптографічних систем з відкритим ключем базується на використанні односпрямованих функцій. Односпрямованою називається функція $f(X)$, яка має наступні властивості:

– є алгоритм визначення значень функції $Y = f(X)$;
– немає алгоритму інвертування функції f, тобто не має розв'язання рівняння $f(X) = Y$ відносно X.

У якості прикладів односпрямованих функцій можливо вказати цілочисельне множення і цілочисельну модульну експоненту.

1.1 Асиметрична криптосистема RSA

Першим і самим розповсюдженим криптоалгоритмом асиметричного шифрування є алгоритм RSA, який отримав назву по першим буквам прізвищ його розробників: Рональда Райвеста (R. Rivest), Ади Шаміра (A. Shamir) і Леонардо Едельмана (L. Adleman). Алгоритм RSA, який було винайдено у 1977–1978 роках, став першим алгоритмом з відкритим ключем, який може

працювати як у режимі шифрування даних, так і у режимі електронного цифрового підпису. У 1993 році метод RSA було прийнято в якості стандарту. Сьогодні RSA є складовою частиною багатьох стандартів, у тому числі International Standards Organization (ISO) 9796, Society for Worldwide Interbank Financial Telecommunications (SWIFT), ANSI X9.31, французького стандарту ETEBAC 5, австралійського AS2805.6.5.3 та ін

Алгоритм RSA засновано на тому факті, що розкладання числа на прості множники доволі складна операція. Згідно теоремі Ейлера, окремий випадок якої стверджує, що якщо число n представлено у вигляді добутку двох простих чисел p і g, то для любого x має місце рівність:

$$\left(x^{(p-1)(q-1)}\right) mod\, n = 1 .$$

Процес передачі зашифрованої інформації у будь-якій асиметричній криптосистемі виконується наступним чином [3][1)].

Підготовчий етап:

– одержувач генерує пару ключів: секретний ключ k і відкритий ключ K;
– відкритий ключ K розсилається відправнику повідомлення і іншим абонентам комп'ютерної мережі (чи робиться доступним, наприклад, на поділюваному ресурсі).

Використання – обмін інформацією між абонентами:

– відправник шифрує відкрите повідомлення C за допомогою відкритого ключа одержувача K і відсилає шифротекст S по відкритому каналу зв'язку;
– одержувач дешифрує повідомлення S за допомогою свого секретного ключа k. Ніхто інший (у тому числі відправник повідомлення) не може розшифрувати дане повідомлення, оскільки не має секретного

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

ключа одержувача. Таким чином, захист інформації в асиметричній криптосистемі засновано на секретності ключа k одержувача повідомлення.

Генерація сполучених ключів RSA.

Для алгоритму RSA етап створення ключів складається з наступних операцій:

– визначають два випадкових великих простих числа p і q. Для забезпечення максимальної безпеки p і q беруть рівної довжини і зберігають їх у таємниці;

– визначають модуль RSA: $n = p \cdot q$;

– визначають функцію Ейлера m=φ(n)=(p-1)(q-1). Така функція вказує на кількість позитивних чисел на інтервалі від 1 до n, які є взаємно простими з модулем n;

– вибирають число d, 0<d<m, таке, що числа p, q, d є взаємно простими;

– визначають параметр e – мультипликативно зворотне числу d:

$$e \equiv d^{-1} \bmod \varphi(n),$$

тобто

$$ed \equiv 1 \bmod m.$$

Числа *e* і *d* називають секретним і відкритим показниками відповідно. Відкритим ключем є пара *{d,n}*, а секретним ключем – пара *{e,n}*.

Насправді, по домовленості, будь-який із згенерованих ключів можливо оголосити секретним або відкритим. Однак, після оголошення секретного ключа необхідно забезпечити його конфіденційність, тобто зберігати у таємниці секретний показник і параметри *p, q*.

Вважається, що знаючи відкритий ключ *{d,n}*, складно визначити секретний параметр *e*, однак, якщо удасться розкласти n на множники *p, q,* то можливо дізнатися про значення e. Таким чином, загальна безпека системи RSA заснована на труднощі задачі розкладання. Математично така задача поки не розв'язна.

Шифрування – криптоперетворення $S = f(C, K)$

Відправник розділяє відкрите повідомлення на блоки, які дорівнюють $k = [log_2(n)]$ біт, де дужки означають узяття цілої частини від дробового числа. Кожен такий блок може бути інтерпретовано як число з діапазону $(0;2^{k-1})$.

Отримані блоки C_i шифрують окремо один від одного, використовуючи формулу: $S_i = C_1^d (mod\, n)$.

Блоки S_i і є зашифроване повідомлення. Їх можливо передавати по відкритому каналу зв'язку. Це означає, що якщо зловмисник знає числа d і n, то по перехопленому значенню S_i прочитати початкове повідомлення він ніяк не може, окрім як повним перебором C_i.

Дешифрування – крипто перетворення $C = f(S, k)$

Процедуру дешифрування може провести тільки власник секретного ключа k. Він розшифровує отримані блоки, використовуючи формулу: $S_i = C_1^e (mod\, n)$

Слід зауважити, що відкрите повідомлення C, шифротекст S, відкритий ключ K і секретний ключ k належать до множини цілих чисел

$$Z_n = \{0,1,2,...,n-1\},$$

де n – модуль RSA.

Криптостійкість. Криптоалгоритм RSA всебічно досліджено і визнано стійким за умовою достатньої довжини ключів. Сьогодні довжина ключа 512 біт вважається недостатньою для забезпечення стійкості, а довжина 1024 біт – прийнятний варіант. Декотрі автори стверджують, що з ростом потужності процесорів криптоалгоритм RSA втратить стійкість щодо атаки повним перебором. Однак не слід забувати, що збільшення потужності процесорів дозволить застосовувати більш довгі ключі, що підвисить стійкість RSA.

Переваги. Асиметричні криптоалгоритми дозволяють здолати недоліки, які властиві системам симетричного шифрування [2][1)]:

[1)] [2] С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития - PDF Free Download / URL: http://docplayer.ru/25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya.html (дата звернення: 15.07.2020)

– не потрібна секретна доставка ключів; асиметричне шифрування, на відмінність від симетричного, має певну перевагу: воно дозволяє динамічно передавати відкриті ключі, тоді як для симетричного шифрування до початку сеансу захищеного зв'язку необхідно обмінятися секретними ключами;

– зникає квадратична залежність числа ключів від числа користувачів; нескладно побачити, що в асиметричній криптосистемі RSA кількість ключів, які використовуються під час обміну повідомленнями, зв'язано з кількістю абонентів лінійною залежністю (у системі з N користувачами необхідно створити 2N ключів), а не квадратичною, як в симетричних системах.

Недоліки. У асиметричних криптосистем і, як і у криптосистеми RSA, є і недоліки:

– на сьогодні не має математичного доказу необоротності функцій, які використовуються в асиметричних алгоритмах;

– порівняно з симетричним шифруванням асиметричне значно повільніше, оскільки під час криптоперетворень використовуються дуже ресурсоємні операції (наприклад, в RSA це піднесення одного великого числа у ступінь, яка є другим великим числом). Тому реалізувати апаратний шифратор з асиметричним алгоритмом значно складніше, ніж апаратно симетричний алгоритм;

– необхідно захищати відкриті ключі від підміни.

1.2 Алгоритм шифрування Ель Гамаля

Даний алгоритм шифрування було розроблено американцем арабського походження Тахером Ель Гамалем у 1984 році. Порівняно з RSA, запропонований алгоритм припускає узагальнення на випадок довільної кінцевої групи, наприклад, групи точок еліптичної кривої. Однак у такому випадку відкритий текст повинен бути елементом групи. Тому алгоритм шифрування пови-

нен передбачати стадію перетворення довільного тексту в елемент групи (під час шифрування) і зворотно (під час дешифрування). В алгоритмі шифрування Ель Гамаля відкритим ключем шифрування К є сукупність [3][1)]:

– простого числа p;

– утворюючої g групи Fp, порядок якої має великий простий дільник;

– експоненти $y = gx(mod\ p)$.

У якості секретного ключа розшифрування k беруть ціле число x, $0<x<p-1$.

Шифрування – криптоперетворення $S = f(C, K)$

Для шифрування відкритого повідомлення C, яке є елементом групи F_p, відправник виконує наступні дії:

– вибирає випадкове число v, $0<v<p-1$;

– визначає $r = yv(mod\ p)$.

Шифротекстом є пара (S_1, S_2), де

$$S_1 = g^v \pmod p,$$
$$S_2 = rC \pmod p.$$

Дешифрування – криптоперетворення $C = f(S, k)$

Одержувач повідомлення виконує алгоритм розшифрування, який передбачає наступні дії:

– відтворює r за допомогою свого секретного ключа x, тобто підраховує: $r = (S_1)^x \pmod p$.

– відтворює відкритий текст C за формулою: $C = r^{-1} S_2\ (mod\ p)$.

Криптостійкість алгоритму висока, оскільки його засновано на складності дискретного логарифмування. Розв’язання такої задачі виявляється значно більш складною проблемою, ніж розв’язання задачі, заснованої на

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

складності розкладання складного числа на прості множники, на яких базується криптосистема RSA і подібні до неї системи.

1.3 Порівняльний аналіз асиметричних криптосистем

В алгоритмі Ель Гамаля шифротекст у два рази довше, ніж початковий відкритий текст. Окрім того, в такому алгоритмі під час шифрування визначаються дві експоненти, а в RSA – тільки одна, тобто шифрування алгоритмом RSA виконується у два рази швидше. У алгоритмі Ель Гамаля, на відмінність від RSA, однакові відкриті повідомлення перетворюються у різні шифротексти, що ускладнює дослідження їх статистичних властивостей. Безпека алгоритму RSA заснована на складності розкладання складного числа, а безпека протоколу Ель Гамаля – на складності дискретного логарифмування у кінцевому полі. У разі застосування в алгоритмі RSA малих показників шифрування, тобто параметру d, можливо розкриття шифротексту без ключа за умовою, що відправник передає різним абонентам-одержувачам близькі відкриті повідомлення. Крім того, використання загального секретного ключа двома користувачами комп'ютерної мережі приводить до взаємного розкриття їх секретних ключів [3][1)].

Для уникнення розкриття шифротексту без ключа в протоколі Ель Гамаля необхідно тільки, щоб випадкові числа v були непередбачуваними і з великою вірогідністю неповторюваними.

1.4 Основні види атак, принципи криптоаналізу

Багато років тому, коли сторонами, що обмінювалися конфіденційною інформацією, були переважно дипломати та військові, можна було бути впевненими у надійності учасників обміну, довіряти один одному. Основною

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

проблемою сторін тоді було забезпечення неможливості втручання у конфіденційний зв'язок третіх сторін. Практично не було випадків, коли адресати поводили себе недобросовісно: відмовлялися від отриманих повідомлень або стверджували про якісь зміни у їх тексті. Основне завдання тогочасних криптографів полягало лише у забезпеченні конфіденційності інформації. Принципово інакше складається картина на сьогодні, коли інформацією обмінюються суб'єкти комерційної діяльності, які можуть не довіряти один одному. І якщо раніше важливо було забезпечити лише конфіденційність інформації, то зараз не менш важливо забезпечити її цілісність та юридичну чинність, а також захиститися від нав'язування хибних повідомлень. Отже, метою суб'єкта в такому разі є перешкодження здійсненню намірів законних учасників інформаційного обміну. В загальному випадку зловмисник може не лише перехоплювати зашифровані повідомлення, але й модифікувати їх, а також направляти фальсифіковані повідомлення легальним сторонам, що обмінюються інформацією, ніби від імені їх легальних партнерів. Таким чином, існує чотири можливих типи загроз з боку зловмисників [4][1)]:

– порушення конфіденційності інформації – дешифрування, повне чи часткове, переданого повідомлення або отримання додаткової інформації про його зміст;
– порушення цілісності інформації – внесення змін у повідомлення, що змінюють його зміст;
– забезпечення неможливості відмови від отриманого чи відправленого повідомлення;
– порушення істинності повідомлень – формування хибних повідомлень, які легальні учасники інформаційного обміну можуть класифікувати як істинні.

[1)] [4] Gordon L.A., Loeb M.P., Lucyshyn W., Richardson R. CSI/FBI Computer Crime and Security Survey 2015 – Computer Security Institute Publications, 2015. URL: https://www.researchgate.net/publication/323112056_Empirical_Evidence_on_the_Determinants_of_Cybersecurity_Investments_in_Private_Sector_Firms (дата звернення: 18.07.2020)

Дешифрування перехопленого повідомлення можливе у випадку обчислення криптографічного ключа або так званого «безключового читання», тобто за допомогою знаходження еквівалентного алгоритму, що не вимагає знання ключа. Процес, при якому реалізується спроба отримати відкритий текст, ключ або й те, й інше, називається криптоаналізом. Однією з можливих атак на алгоритм шифрування є атака грубою силою (лобова атака), тобто просте перебирання усіх можливих ключів. Якщо кількість ключів досить велика, то підібрати потрібний дуже й дуже складно. При довжині ключа n бітів кількість можливих ключів дорівнює 2^n. Таким чином, чим довший ключ, тим стійкішим вважається алгоритм для лобової атаки. Існують різні типи атак, які ґрунтуються на тому, що зловмиснику відома певна кількість пар відкритого тексту-шифротексту. При аналізі зашифрованого тексту зловмисник часто застосовує статистичні методи аналізу тексту. Криптоаналіз – це наука відновлення (дешифрування) відкритого тексту без доступу до ключа. Фундаментальне припущення криптоаналізу, вперше сформульоване Кірхгоффом, полягає в тому, що таємність повідомлення повністю залежить від ключа, тобто весь механізм шифрування, крім значення ключа, відомий противнику. Криптоаналіз ставить своїм завданням в різних умовах отримати додаткові відомості про ключі шифрування, щоб значно зменшити спектр можливих ключів. Результати криптоаналізу можуть варіюватися за ступенем практичного застосування. Так, криптограф Ларс Кнудсен пропонує таку класифікацію успішних результатів криптоанализа блокових шифрів в залежності від обсягу і якості секретної інформації, яку вдалося отримати: розробляє функціональний еквівалент досліджуваного алгоритму, що дозволяє зашифровувати і розшифровувати інформацію без знання ключа [4][1)]:

– повний злом – криптоаналітик отримує секретний ключ;

[1)] [4] Gordon L.A., Loeb M.P., Lucyshyn W., Richardson R. CSI/FBI Computer Crime and Security Survey 2015 – Computer Security Institute Publications, 2015. URL: https://www.researchgate.net/publication/323112056_Empirical_Evidence_on_the_Determinants_of_Cybersecurity_Investments_in_Private_Sector_Firms (дата звернення: 18.07.2020)

– глобальна дедукція – криптоаналітик розробляє функціональний еквівалент досліджуваного алгоритму, що дозволяє зашифровувати і розшифровувати інформацію без знання ключа;
– часткова дедукція – криптоаналітику вдається розшифрувати або зашифрувати деякі повідомлення;
– інформаційна дедукція – криптоаналітик отримує деяку інформацію про відкритий текст або ключ.

Однак, злом шифру зовсім не обов'язково має на увазі виявлення способу, що застосовується на практиці для відновлення відкритого тексту по перехопленому зашифрованому повідомленням. Шифр вважається зламаним, якщо в системі виявлено слабке місце, яке може бути використане для більш ефективного злому, ніж метод повного перебору ключів («brute-force approach»). Припустимо, для дешифрування тексту методом повного перебору потрібно взяти 2^{128} можливих ключів; тоді винахід способу, вимагає для дешифрування 2^{110} операцій з підбору ключа, буде вважатися зломом. Під зломом розуміється лише підтвердження наявності уразливості криптоалгоритму, що свідчить про те, що властивості надійності шифру не відповідають заявленим характеристикам. Як правило, криптоаналіз починається зі спроб злому спрощеної модифікації алгоритму, після чого результати поширюються на повноцінну версію [3][1)].

Спроба криптоаналізу називається атакою. Нехай для шифрування і розшифровки використовуються ключі k та k′ відповідно (в асиметричної криптографії k = k′); визначимо функцію шифрування E^k, розшифрування $D^{k'}$. Тоді виконуються наступне співвідношення $E_k(x) = y, D_{k'}(y) = x$.

Здатність криптосистеми протистояти атакам криптоаналітика називається стійкістю. Стійкість криптоалгоритмів визначається складністю розв'язання добре відомої математичної задачі, яка є загальновизнано складною. Як

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

приклад можна привести систему RSA (Ривеста-Шаміра-Адельмана), засновану на складності розкладання цілого числа на множники.

У теорії криптоаналізу відомі такі універсальні методи аналізу: метод повного перебору, атака по ключам, частотний аналіз.

1.5 Класифікація атак на симетричні криптоалгоритми

Розглянемо існуючу класифікацію атак, які використовують на симетричні крипто алгоритми.

Атака на основі лише шифротексту (ciphertext-only-attack). У цьому випадку зловмиснику відомі лише шифротексти, зашифровані на одному ключі. Це найслабший тип криптоаналітичної атаки, успіх якої зовсім неочевидний. Він залежить від багатьох чинників та визначається кваліфікацією аналітика (за умови ручного криптоаналізу) або повністю визначається потужністю комп'ютерів криптоаналітичної системи. Завжди процес криптоаналізу починається зі збирання інформації про відкритий текст [2][1)]:

– якою мовою написаний оригінал;
– які лінгвістичні особливості цієї мови;
– які слова або фрази може містити оригінал та у якій послідовності;
– якою приблизно може бути довжина оригінального тексту;
– які методи шифрування могли застосувати для зашифрування цього тексту;
– яка службова інформація може міститися у тексті (дата, контрольна сума, адреси тощо);
– якою апаратурою було зашифровано текст.

Ці або подібні дані збираються агентурними або аналітичними засобами і значно полегшують роботу криптоаналітиків.

[1)] [2] С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития - PDF Free Download / URL: http://docplayer.ru/25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya.html (дата звернення: 05.06.2020)

Атака на основі невибраного (відомого) відкритого тексту (known-plaintext attack). Супротивник знає або може встановити деякі частини відкритого тексту у криптограмі. Завдання полягає в тому, щоб дешифрувати увесь текст. Це можна виконати шляхом покрокового обчислення ключа. Атака на основі обраного відкритого тексту (chosen-plaintext attack). Зловмисник може обрати будь-який відкритий текст і отримати для нього зашифрований. Завдання полягає у визначенні ключа шифрування. Деякі алгоритми шифрування досить вразливі для атак цього типу.

Тому у випадку використання таких систем, серйозної уваги вимагає внутрішня безпека використання системи, щоб зловмисник ні в якому разі не зміг отримати доступ до відкритих текстів. Така атака буде називатися простою в разі, коли усі відкриті тексти зловмисник отримує до перехоплення першої криптограми, та адаптивною, коли зловмисник обирає черговий відкритий текст, маючи шифровки усіх попередніх.

Атака на основі обраного шифротексту (chosen-ciphertext attack). Зловмисник може обирати потрібну кількість криптограм та отримати для них відкриті тексти. Тут також, аналогічно до попереднього типу атак, існують різновиди простої та адаптивної атак. Атака на основі обраного тексту (chosen-text attack). Це найнебезпечніший тип атак, оскільки зловмисник може передавати спеціально підготовлені відкриті тексти для зашифрування, а потім отримувати відповідні шифровки. Завдання полягає у розкритті ключа. Ця атака також може бути простою та адаптивною. Якщо зловмисник здійснює атаку на основі лише шифротексту, у нього є дуже обмежений набір можливостей. До них можна віднести метод грубої сили або частотний криптоаналіз. У разі атаки на основі відомого відкритого тексту розкриття шифру буде примітивним після того, коли у текстах зустрінеться більшість літер абетки. Якщо зловмисник має можливість атакувати на основі обраного текс-

ту, то криптосистему буде розкрито вже після шифрування відкритого тексту типу: «АБВГД…ЬЮЯ» [2][1)].

1.6 Класифікація атак на асиметричні криптоалгоритми

Специфічний тип атаки, "посередництво" (Man-in-middle attack). Ця атака спрямована на злам криптографічних комунікацій та протоколів обміну ключами. Атака здійснюється приблизно таким чином. Припустимо, що зловмисник, назвемо його З, має змогу перехоплювати усі повідомлення сторін А та В. Припустимо також, що сторони хочуть обмінятися криптографічними ключами, тож А відправляє свій ключ шифрування КАВ стороні В. Зловмисник З перехоплює цей ключ, зберігає його, та відправляє стороні В вже свій криптографічний ключ, КЗВ, ніби від сторони А. Сторона В, отримавши цей ключ, у відповідь відправляє стороні А свій криптографічний ключ КВА. Зловмисник З також підміняє цей ключ своїм, КЗА і надсилає його стороні А. Таким чином, сторони А та В "вірять", що мають криптографічні ключі один одного, хоча насправді вони мають лише два різні ключі зловмисника З. Той, у свою чергу, має обидва ключі сторін та може читати усю їхню зашифровану інформацію (рис. 1.1).

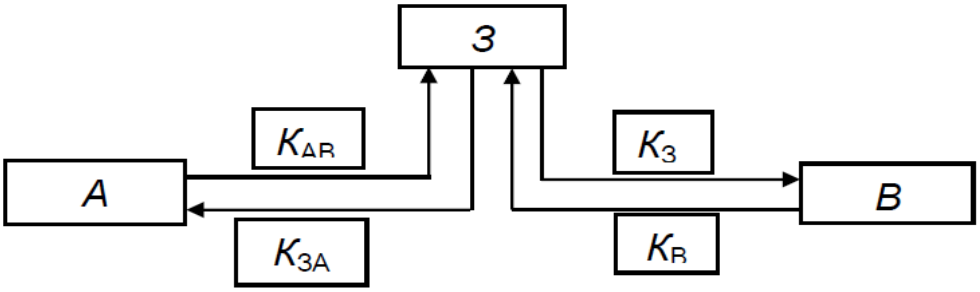

Рисунок 1.1 – Схема атаки Man-in-Middle при обміні ключами

[1)] [2] С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития - PDF Free Download / URL: http://docplayer.ru/25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya.html (дата звернення: 05.06.2020)

Спроба обміну інформацією між сторонами призводить до такого. А шифрує повідомлення ключем КАВ та надсилає його. З перехоплює цього листа, розшифровує його перехопленим ключем сторони А(КАВ), перешифровує ключем, який він надіслав В (КЗВ) та відправляє. Сторона В, отримавши зашифрованого листа, розшифровує його і, оскільки сеанс розшифрування пройшов успішно, «вірить», що обмін інформацією триває нормально. При спробі сторони В відповісти на лист від А, зловмисник виконує аналогічну процедуру. Таким чином, зловмисник читає усю інформацію, що курсує між сторонами А та В, оскільки володіє усіма чотирма криптографічними ключами, і продовжуватися це може доти, поки сторони А та В фізично не зустрінуться та зрозуміють, що оперують незнайомими криптографічними ключами, або З не використає проти них перехоплену інформацію. Посередництво – одна з найнебезпечніших атак проти систем зв'язку. Захист від нього потребував створення потужної системи – інфраструктури відкритих ключів та застосування геш-функцій. Успішний захист від посередництва полягає у обчисленні геш-образу повідомлень та підписуванні цього образу електронним цифровим підписом. У такому разі отримувач має змогу перевірити цілісність інформації порівнянням геш-функцій та власника цифрового підпису за допомогою сертифіката центра РКІ [3][1)].

Розглянемо деякі інші специфічні типи атак на криптосистему RSA.

Атака на близькі значення p і q. Однією з найвідоміших атак на криптосистему RSA є така, що використовує близькі значення p та q. Припустимо, що $p > q$, хоча це ніяк не обмежує подальші міркування. Тоді можна записати: $x = (p + q)/2$, $y = (p - q)/2$, а для n справедливе рівняння:

$$n = x^2 - y^2 \tag{1.1}$$

[1)] [3] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.

Для знаходження множників p та q досить підібрати числа x та y, що задовольняють формулі 1.1. Знайдемо $x_0 = \sqrt{n}$ та оберемо найближче до нього більше ціле x_1. Підставляючи його у формулу 1.1, знаходимо відповідне значення y. Повторюємо цю операцію доти, поки не отримаємо цілі значення x та y, що задовольняють формулі 1.1. У результаті знаходимо p = x + y, q = x − y. Розглянемо приклад. Нехай $n = p \cdot q = 851$.

Скористаємося описаним правилом для знаходження p та q. Оскільки $\sqrt{n} = 17.29$, то припустивши, що x = 30, знаходимо $y^2 = 30^2 - 851 = 49$. Отже, $y = 7$, з першої спроби знаходимо розв'язок (*): $x = 30, y = 7$. Отже, $p = 30 + 7 = 37$, а $q = 30 - 7 = 23$. Отримавши значення p та q, зловмисник може легко обчислити приватний ключ, зламавши таким чином криптосистему.

Атака з вибраним шифротекстом. Нехай перехоплено повідомлення $C = E_e(M)$, зашифроване на публічному ключі алгоритму RSA.

Обираємо число r < n. Тоді можна обчислити наступне:

$$x = r^e \bmod n;$$
$$y = xC \bmod n;$$
$$t = r^{-1} \bmod n.$$

Якщо $x = r^e \bmod n$, тоді $r = x^d \bmod n$.

Тепер просимо власника ключа підписати у приватним ключем d. Будемо мати $u=y^d \bmod n$.

Обчислимо $tu \bmod n = r^{-1}y^d \bmod n = r^{-1}x^d C^d \bmod n = C^d \bmod n = M$.

Таким чином, отримано розшифроване повідомлення M.

Захист від такої атаки може здійснюватися двома шляхами. Перший полягає в тому, що пари ключів для шифрування та електронного підпису повинні бути різними. Другий – підписувати завжди потрібно геш-образ повідомлення, а не власне повідомлення.

Атака на електронний підпис. Якщо зловмиснику необхідно отримати електронний підпис повідомлення M, він може діяти таким чином. Створю-

ються повідомлення M_1 та M_2 такі, що $M = M_1M_2 \ mod \ n,$ та підписуються окремо повідомлення M_1 та M_2.

Якщо є ЕЦП повідомлень M_1 та M_2, тоді $M_3^d = (M_1^d \ mod \ n) \ (M_2^d \ mod \ n)$. Отже, повідомлення M підписано.

Атака на спільний модуль. При реалізації RSA іноді буває вигідним роздати усім користувачам спільний модуль *n* та різні ключі *e та d.* Однак така схема працює доти, доки два користувача не зашифрують на різних ключах однакове повідомлення. Якщо e_1 *та* e_2 – взаємно прості числа, тоді можлива така атака.

Нехай *m* – повідомлення, e_1 *та* e_2 – два публічних ключа, *n* – спільний модуль. Процес зашифрування можна зобразити рівняннями:

$$C_1 = m^{e1} mod \ n \qquad C_2 = m^{e2} \ mod \ n$$

Отже, криптоаналітик знає n, e_1, e_2, C_1, C_2. За допомогою алгоритму Евкліда він знаходить такі r та s, які задовольняють рівняння:

$$se_1 + re_2 = 1 \ (r < 0; \ s > 0) \text{ та обчислює } C_1^{-1} \ mod \ n.$$

Тоді $(C_1^{-1})^{-r} \ C_2^{s} mod \ n = m$

Зважаючи на таку атаку, задавати спільний модуль для групи користувачів дуже небезпечно.

Атака дешифрування ітераціями. Суть методу полягає в тому, що перехоплену криптограму повторно шифрують на публічному ключі і при деякій кількості ітерацій отримують вихідне повідомлення. Кількість ітерацій, звичайно, залежить від довжини модуля та ключа. За невеликої довжини модуля цього можна досягти вже при кількох ітераціях. Розглянемо приклад. Нехай p $= 3, q = 11; n = 33; e = 7, d = 3;$ повідомлення $m = "02"$. Обчислення криптограми дає: $C = 27 \ mod \ 33 = 29.$ Спробуємо перешифрувати криптограму на публічному ключі (7,33):

– перша ітерація – $C1 = 297 \bmod 33 = 17;$

– друга ітерація – $C2 = 177 \bmod 33 = 8;$

– третя ітерація – $C3 = 87 \bmod 33 = 2 = m.$

Отже, як бачимо, вже третя ітерація дозволила отримати відкритий текст. На практиці це повинно було б виглядати таким чином. Перехоплена шифровка перешифровується на публічному ключі доти, поки на виході не отримується зв’язний текст. Зрозуміло, що це обов’язково станеться, однак час, потрібний для цього, не повинен бути меншим за час для повного перебору ключів чи розкладання модуля на множники.

2 ДОСЛІДЖЕННЯ МЕТОДІВ КРИПТОАНАЛІЗУ АСИМЕТРИЧНИХ ТА СИМЕТРИЧНИХ КРИПТОСИСТЕМ

Аналіз літературних джерел свідчить про те, що в переважній більшості алгоритми, що використовуються при рішенні завдань асиметричної криптографії засновані на вирішенні задачі факторизації та дискретного логарифмування, що мають застосування в різних алгебраїчних структурах.

Сучасний розвиток наявних алгоритмів свідчить про те, що на сьогоднішній день не знайдено поліноміальних алгоритм вирішення завдань асиметричної криптографії, а приналежність цих завдань до класу NP-повних задач не доведена.

Найчастіше, при вирішенні завдань криптоаналізу асиметричних криптосистем, використовують відомі універсальні методи криптоаналізу, але є і інші методи вирішення, що направлені на врахування особливостей систем з відкритим ключем. Всі вони спрямовані на вирішення математичної задачі, яка і складає основу асиметричного шифру.

2.1 Криптоаналіз систем шифрування, заснованих на складності завдання дискретного логарифмування

Розглянемо задачу дискретного логарифмування, яка вважається більш складною, ніж задача факторизації.

Якщо буде знайдений поліноміальний алгоритм її рішення, стане можливим і розкладання на множники (протилежне не доведено). Останні досягнення теорії обчислювальної складності показали, що загальна проблема логарифмування в кінцевих полях не може вважатися досить міцним фундаментом. Найбільш ефективні на сьогоднішній день алгоритми дискретного логарифмування мають вже не експоненційну, а субекспоненційну часову

складність. Це алгоритми «index-calculus», що використовують факторну базу [5][1)].

Алгоритм, який був запропонований Адлеманом – це один з перших таких алгоритмів, що з'являється та передбачає часову складність $L_p\left[\frac{1}{2};C\right]$ при обчисленні дискретного логарифму в простому полі:

$$Z_p\left(L_N[\gamma;c]\right) = e^{(c+o(1))(\log N)^{\gamma}(\log\log N)^{1-\gamma}}, \tag{2.1}$$

де $0 < \gamma < 1, c = const, c > 0)$.

Основа застосування методів «index-calculus» дозволяє стверджувати, що якщо обрати $\prod_{i=1}^{m} x_i = \prod_{j=1}^{n} y_j$ для деяких елементів кінцевого поля Z_p, то можливо отримати наступний вираз:

$$\sum_{i=1}^{m} \log_a x_i \equiv \sum_{j=1}^{n} \log_a y_i \bmod (p-1), \tag{2.2}$$

Отримавши досить багато таких відносин (причому хоча б одне з них повинно включати елемент g, для якого $\log_a g$ відомий), можна вирішити систему лінійних рівнянь відносно невідомих $\log_a x_i$ та $\log_a y_j$ в кільці відрахувань Z_{p-1} за умови, що кількість невідомих в рівняннях не надто велике.

2.2 Криптоаналіз систем шифрування, заснованих на складності задачі факторизації

У всіх асиметричних криптосистемах (в т.ч. RSA) використовують два ключа. В RSA відкритий ключ – це пара ($K_{відкр}$, N) і приватний ключ ($K_{секр}$,

---

[1)] [5] Чернышев Ю.О., Сергеев А.С., Венцов Н.Н., Рязанов А.Н. Исследование возможности применения генетических алгоритмов для реализации криптоанализа блочных криптосистем. Вестник ДГТУ. 2015;15(3):65-72. URL: https://doi.org/10.12737/12599 (дата звернення: 05.07.2020)

N), де *N – модуль. N = PQ, де P і Q* – два дуже великих простих числа. При цьому можливо кілька варіантів криптоаналізу [6][1)]:

– на основі відомого відкритого тексту і шифр текста підібрати секретний ключ Kсекр, щоб виконувалося рівність Mi = Ci Kcekp(modN), де Mi – відкритий текст, Ci – шифр текст;
– дослідним шляхом визначити функцію Ейлера *φ(n) = (p-1) (q-1),* розклавши модуль N на множники P і Q та секретний ключ Kcekp зі співвідношення Kсекр Kвідкр = 1 (mod φ(n));
– підібрати такі числа P і Q, щоб виконувалося співвідношення N = PQ.

Завдання пошуку ефективних засобів розкладання цілих чисел на множники потребує застосування методу, що виконує розкладання числа n на деякі множники, які передбачають перебір простих дільників, при умові що n не буде перевищуватись. Також використовують так званий метод Ферма, які вирішує завдання перебору, заснований на представленні числа n у вигляді різниці квадратів: $n = a^2 - b^2 = (a+b)(a-b)$. Ідея Ферма передбачає здійснення обчислення $НОД(n, a-b)$, що дозволяє в свою чергу знайти нетривіальний дільник n. Якщо число, яке розкладається, має два множника, при умові що ці число не дуже різняться за величиною, то застосування такого способу дозволяє їх швидше визначити, ніж при застосуванні простого перебору подільників. Лежандр звернув увагу на те, що при такому підході досить отримати порівняння $a^2 \equiv b^2 \, mod(n)$. За складністю як простий переборний метод, так і метод Ферма оцінюються величиною $O(\sqrt{n})$, однак останній метод може виявитися ефективніше, якщо подільники числа n близькі один до одного. Для досить великих чисел n час роботи таких алгоритмів стає неприйнятним. Перш ніж приступати до факторизації таких чисел, слід переконатися в тому, що вони дійсно складові. Для цього найкраще використовува-

---

[1)] [6] Мао В. Современная криптография. Теория и практика. М.: Вильямс, 2005. 763 с.

ти один з імовірнісних тестів на простоту, наприклад, алгоритм Міллера-Рабіна [5][1)].

Відомі деякі способи, що дозволяють здійснити злом шифру RSA. Також визначена ефективна атака, яка дозволяє: знайти private – секретний ключ, який зіставляється потрібному public – відкритому ключу. Застосування такої атаки зловмисниками забезпечує доступ до всіх повідомлень, які були зашифровані з використанням відкритого ключа та здійснювати підробку підпису. Така атака стає можливою, якщо зловмисником знайдені головні співмножники, які є факторами загального модуля *n: p і q*. Знаючи загальні показники *p, q і e,* зловмисник має можливість здійснити обчислення приватного показника *d.* Але в цієї задачі є деяка складність – це задача пошуку головних співмножників n, або так званий факторинг.

Завдання факторингу, яке є трудомістким завданням та і не має ефективних способів розв'язування, забезпечує достатньо високий рівень безпеки шифру RSA. Стрімкий розвиток обчислювальних потужностей та обчислювального устаткування не надає можливості вирішення такої задачі, якщо розробники будуть використовувати ключі достатньо великої довжини, що і зумовлює стійкість криптосистеми RSA.

2.3 ρ-алгоритм факторизації

Якщо розглядати різноманітні завдання з якими стикається дискретна теорія ймовірностей, то можливо визначити відоме класичне завдання, яке надає уяву про механізми вирішення завдань ймовірності. Припустимо, що у деякому класі є *a* людей, необхідно виявити яка ймовірність події, що щонайменше двоє з учнів класу мають співпадаючі дні народження. Для отримання рішення цього завдання не можливо стає узгодження з інтуїцією. Мо-

---

[1)] [5] Чернышев Ю.О., Сергеев А.С., Венцов Н.Н., Рязанов А.Н. Исследование возможности применения генетических алгоритмов для реализации криптоанализа блочных криптосистем. Вестник Донского государственного технического университета . 2015;15(3):65-72. URL: https://doi.org/10.12737/12599

жливі ймовірності настання такої події збільшується зі зростанням чисельності учнів класу. Таблиця 2.1 дає ймовірності для числа учнів класу, що змінюються від 5 до 50 із кроком 5.

Таблиця 2.1– Імовірність збігу днів народження

Число людей	Імовірність
5	0,0271
10	0,1169
15	0,2529
20	0,4114
25	0,5687
30	0,7033
35	0,8144
40	0,8912
45	0,9410
50	0,9774

Розглянемо покроковий алгоритм методу. Припустимо, що p – один з дільників N, і нехай удалося знайти два різних числа a і b, таких що $a \equiv b \bmod p$ *(нехай* $a>b$*)*. Тоді $a-b$ ділиться на p. Але і значення N ділиться на p, тоді, виконавши обчислення *НОД* $(a-b, N)$, одержимо p і другий множник, який дорівнює $q=N/p$. Винятком буде випадок за умови коми різниця a-b ділиться на N. Слід зауважити, що N складається із двох множників, а саме $N=p*q$. Зазначимо, що значення p – невідомо, тому слід продовжувати підбирати пари випадкових (псевдовипадкових) чисел a і b, і потім обчислювати $d=$*НОД* $(abs(a-b), N)$. Якщо знайдено таке d, яке ділиться на N, то це невдалий результат, тому що d ділиться на всі співмножники N. Але, якщо значення d дорівнює одиниці, то знов невдалий результат: d не ділиться ні на один співмножник N. А в випадку, коли $d>1$ і d не ділиться на N, то знайдено таке значення d при котрому d і є одним із співмножників N [7][1)].

[1)] [7] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.

Для отримання псевдовипадкових чисел за модулем p, використовують лінійний конгруентний датчик, який описується виразом:

$$r_i = f(r_{i-1}) = u * r_{i-1} + v \bmod M \ (M>N), \tag{2.3}$$

де M – максимальне представлене в машині ціле число, $r0$ – випадкова константа, u і v – регулюючі константи.

Але можливо розглянути цей метод з іншого боку. Припустимо, що функція f є однозначною, що передбачає, що фіксоване r_{i-1} породжує єдине r_i, але слід зауважити, що фіксоване r_i може надавати різні r_{i-1}. Область, де працює псевдовипадковий датчик, визначаємо як орієнтований граф з p вершинами з номерами вершин $0, 1, \ldots, p-1$, а функція f задає стрілки таким чином, що з кожної з вершин виходить одна стрілка, а входити може декілька. Тоді граф має вигляд, наведений на рисунку 2.1.

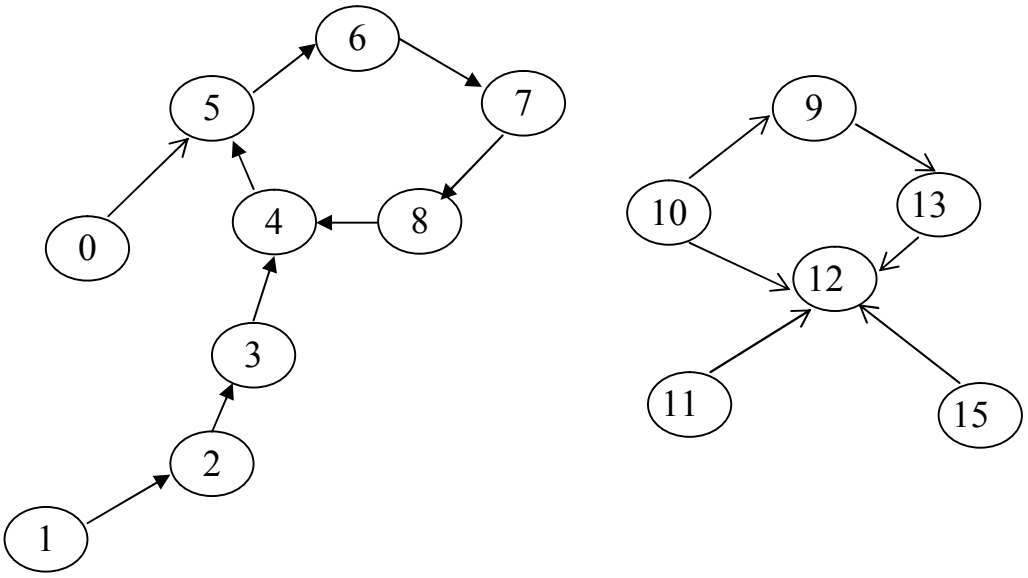

Рисунок 2.1 – Приклад схеми псевдовипадкового датчика при $N=16$

На рис. 2.1 видно, що декілька не зв'язаних між собою контурів, можуть мати «причеплені» до них ручки. Породження псевдовипадкових чисел здійснюється за наступним механізмом. Перш за все, випадково обирається початкова вершина (наприклад, вершина з номером 1), а в подальшому крок

за кроком відбувається перехід до ступної вершини по направленню стрілки. При умові, якщо початкова вершина належить контуру, то рух буде відбувається по контуру і нагадувати крутіння. При умові, розташування початкової вершини на одній з «ручок», рух буде спрямовано по ручці до контуру, і лише потім починається крутіння по цьому контуру. Тоді шлях позначений жирними стрілками на рис. 2.1 нагадує грецьку букву ρ, що й зумовило назву алгоритму. Ще одним завданням є визначення умов, при яких датчик почне зациклюватись. Вирішення цього завдання можливо здійснити якщо запам'ятовувати всі номери вершин, які вже пройдено. Потім зпівставляти номер наступної вершини з номером вершини у списку вже відвіданих вершин. Але такий алгоритм не задовольняє, тому що p може буди дуже великим числом, а значення кількості вершин на «ρ-шляху» може бути дуже великим, але не більш ніж p. Для великих значень p треба застосувати деякий алгоритм, що вимагає постійне визначене значення необхідної пам'яті і лінійний алгоритм, що виконує порівняння номерів за числом операцій.

Запропонований Флойдом алгоритм мав саме таки ознаки. Розглянемо алгоритм Флойда на прикладі, обравши за основу граф, який наведено на рисунку 2.1 Уздовж ρ-шляху вершини проходять в наступному порядку [7][1)]:

$$r_1=1,\ r_2=2,\ r_3=3,\ r_4=4,\ r_5=5,\ \ldots,\ r_9=4,\ \ldots,\ r_{14}=4,\ \ldots \qquad (2.4)$$

Для зберігання номерів вершин цьому алгоритму потрібно дві комірки пам'яті. Одно з вершин потрібна для для зберігання номера еталонної вершини E , з яким рівняються всі номери прохідних вершин. А для зберігання номера прохідної, поточної вершини C потрібна друга комірка пам'яті. Алгоритм вимагає організувати роботу тактами, де задається довжина такту L. Довжина такту дорівнює числу поточних вершин, яка порівнюється з еталон-

---

1) [7] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.

ною. Значення L збільшується на 1 від такта к такту, а новим еталонним номером призначається останній поточний з попереднього такту. Наприклад:

1-й такт 3-й такт 5-й такт

1 2 3 4 5 6 7 8 4 5 6 7 8 4 5 6 …

2-й такт 4-й такт

Очевидним є той факт, що повторення спостерігається на п'ятому такті (еталонне число $r11=6$ дорівнює поточному числу $r16=6$). Але при виконанні цього простого, на перший погляд, алгоритму, повторення знаходиться не відразу, хоча видно, що перше повторення можливо знайти вже при порівнянні $r_4=4$ *та* $r_9=4$. При виконанні завдання факторизації N на основі алгоритма Флойда, рівність двох елементів можливо визначити рівністю:

$$НОД\ (abs(r_k-r_l),\ N)=d\ (d>1,\ d \neq k) \qquad (2.5)$$

Процес ініціалізації передбачає введення N та завдання початкових даних для псевдовипадкового датчика: M – максимальне ціле, управляючи параметри u і v. Для ініціалізації датчику вводиться постійна величина $r0$. Наприкінці здійснюється видача значення d.

Наведемо приклад,

Припустимо, що N=209, M=256, u=131, v=1, r0=13.

1-й такт *3-й такт*

168 249 108 69 80 ...

2-й такт

Згідно з кожним тактом виконуються наступні розрахунки:

1-й такт: *НОД (249-168,209)=1.*

2-й такт: *НОД (249-108,209)=1;*

НОД (249-69,209)=1;

НОД (80-69,209)=11.

Кінець розрахунків, $p=11,\ q=209/11=19$.

Для виконання оцінки проведених розрахунків та ефективності використання саме цього алгоритму, потрібно розрахувати оцінку ймовірності *Pr,* яка свідчить, що при заданих *p* і *a (a<<p)* знайдуться хоча б два псевдовипадкових числа, порівнянних за модулем *p.* Але доцільно буде оцінити додаткову ймовірність *1-Pr* того, що всі числа будуть непорівнянні.

Перше число можна вибрати p способами, друге число можна вибрати p-1 способами, третє число можна вибрати p-2 способами і т.д. a-те число можна вибрати p-a+1 способами, так що всього є *p*(p-1)(p-2)*...*(p-a+1)* сприятливих комбінацій на *p* можливих. Отже отримаємо вираз:

$$1 - Pr = \frac{p \cdot (p-1) \cdot \ldots \cdot (p-a)}{p^a} = \left(1 - \frac{1}{p}\right) \cdot \left(1 - \frac{2}{p}\right) \cdot \ldots \cdot \left(1 - \frac{a-1}{p}\right) \qquad (2.6)$$

Відомо, що для |x|<1 має місце наближена рівність $x \sim ln\ (1-x)$. Здійснюючи логарифмування виразу (2.6), отримаємо наступний вираз:

$$ln(1-Pr)=ln(1-1/p) + ln(1-2/p) + \ldots + ln(1-(a-1)/p)\ (-1/p+2/p+\ldots+(a-1)/p =$$
$$= -1/p(1+2+\ldots+(a-1)) = -a(a-1)/(2p).$$

При цьому $a = \sqrt{p}$, оцінюваний логарифм дорівнює *–1/p*, при умові що $a = 2 * \sqrt{p}$ дорівнює приблизно –2; з цього випливає, що *(1-Pr)* у першому випадку буде дорівнювати $e^{-1/2} \sim 0.6$, а в другому випадку $-\ e^{-2} \sim 0.14$. Тоді отримаємо результат: Pr приблизно дорівнює *0.4* і *0.86*, відповідно.

2.4 Диференціальний криптоаналіз

Засновниками поняття диференціального криптоаналізу було введено двома фахівцями Елі Біхамом та Аді Шаміром у 1990 році, які і поклали початок розвитку цьому напрямку. Саме методи та алгоритми диференціально-

го криптоаналізу і застосовують для здійснення атак на симетричні криптосистеми, що передбачає засобами диференціального криптоаналізу отримання підключу раунду. Атаки на симетричні криптосистеми використовують властивості алгоритму, а саме властивості так званих S-блоків. Застосування конкретного способу для здійснення диференціального криптоаналізу залежить від обраного у системі алгоритму шифрування. При умові, коли в основі алгоритму шифрування лежить сітка Фейстеля, то можливо допустити, що деякий блок *m* має дві частини: *m0* і *m1*, тоді завдання диференціального криптоаналізу зводиться до визначення деяких відмінностей, що мають відбуватися в кожній з зазначених частин при здійсненні шифрування.

Наприклад, для алгоритму DES «відмінності» визначаються за допомогою операції XOR, для інших алгоритмів можливий інший спосіб. Обирається пара незашифрованих текстів з фіксованою відмінністю. Потім аналізуються відмінності, що вийшли після шифрування одним раундом алгоритму, і визначаються ймовірності різних ключів. Якщо для багатьох пар вхідних значень, що мають ту саму відмінність X, при використанні того самого підключа (Y) однаковими виявляються і відмінності відповідних вихідних значень, то можна припустити, що в X використано підключ Y з певною ймовірністю. Якщо ця ймовірність близька до одиниці, то підключ раунду знайдений з даною ймовірністю. Імовірність знаходження загального ключа визначається добутком ймовірностей підключів кожного раунду. Диференціальний криптоаналіз на основі відмов пристрою був оголошений у вересні 1996 року компанією Bellcore, що дозволяє ефективно розкривати секретний ключ, який зберігається в пам’яті портативного шифрувального обладнання. Вперше метод був успішно застосований для розкриття ключа криптосистеми RSA. Подальші дослідження нового методу, що одержав назву диференціального криптоаналізу на основі відмов обладнання (Differential Fault Analysis, DFA), продемонстрували його ефективність при атаці на DES й інші блокові шифри. Так, для розкриття ключа DES знадобилося проаналізувати двісті шифротекстів, отриманих шифруванням відкритих текстів, що зберігаються в

пам'яті обладнання. Було доведено, що навіть застосування потрійного DES не впливає на складність атаки [8][1)].

В основу крипоаналітичної атаки, запропонованої фахівцями компанії Bellcore, покладено наслідки від деяких видів випромінювання (наприклад, радіоактивного), які призводять до відмов електронного обладнання. А криптоаналітик немає доступу до обладнання, засобами якого проведене шифрування, тому самє відкритий текст і секретний ключ, які знаходяться в пам'яті обладнання, стають недоступними елементами. Піддаючи обладнання випромінюванню, робота обладнання зупиняється, що призводить до інверсії бітів в одному з регістрів на деякому проміжному етапі криптографічного перетворення. Така модельована відмова призводить до пошкодженню шифрованого тексту, тоді завданням крипто аналітика є розкриття секретного ключу, застосовуючи аналіз пошкоджених і непошкоджених шифротекстів.

Розглянемо цей метод при здійсненні криптоаналізу DES, де наявні деякі два різні шифротексти, що отримані при виконанні шифрування деякого відкритого тексту на фіксованому ключі. Але кріптоаналітику відомо шифротекст, який був отриманий при інверсії одиночного біту в процесі шифрування. Спочатку треба визначити номер циклу перетворення, на якому і сталась інверсія. Якщо припустити, що інверсія відбулася на останньому, шістнадцятому циклі DES-перетворення в правій частині блоку до заключної перестановки. Можливо припустити, якщо відмінність лівих частин блоків шифротексту визначається виходами одного або двох S-блоків. А інвертований біт розташований на вході. Якщо застосувати описаний вище метод криптоаналізу, то отримаємо шість бітів ключа для кожного з S-блоків. Якщо проаналізувати близько 200 шифротекстів, то отримаємо підключ останнього циклу перетворення. Надалі можливо здійснити пошук секретного ключа шляхом вичерпної перевірки кандидатів при заданому підключі. Але можливо і використання знання про підключ, отриманий на останньому циклі для аналізу

[1)] [8] Бауэр Ф. Расшифрованные секреты. Методы и принципы криптологии. М.: Мир, 2007. 550 с.

попередніх циклів. Останній варіант дозволяє успішно атакувати DES у режимі EDE-шифрування на трьох різних ключах. Описаний метод працює й у тих випадках, коли інверсія виникає всередині F-функції або процедури генерації підключів [8][1)].

Розглянутий метод диференціального криптоаналізу на основі відмов обладнання має широку сферу застосування, а саме: для атаки на блокові шифри, такі як IDEA, RC5, Feal.

Розглянутийв цій главі метод криптоаналізу забезпечує надійний механізм розкриття ключів шифрування навіть в тих випадках, коли сам алгоритм криптографічного перетворення невідомий. Прикладом може служити криптоалгоритму Fortezza, який засекречений Агентством національної безпеки США, хоча апаратна реалізація криптоалгоритму у вигляді мікросхеми Clipper входить до складу багатьох комерційних обладнань шифрування, наприклад, Fortezza PCMCIA.

2.5 Лінійний криптоаналіз

Наступним проаналізуємо метод лінійного криптоаналізу, який вперше був застосований для атаки блокового шифру FEAL, а через деякий час і для шифру DES. Назва методу походить від механізму лінійних наближень, який і застосовано у цьому методі. Якщо виконати операцію XOR над деякими бітами відкритого тексту, потім над деякими бітами шифротексту, а потім виконати XOR результатів попереднього сумування, можна одержати біт, який буде сумою кількох бітів ключа. Це і є лінійним наближенням, яке може бути вірним з деякою ймовірністю p. Якщо $p \neq 1/2$, то цей факт можна використовувати для розкриття бітів ключа. Розглянемо цей метод [2][2)].

[1)] [8] Бауэр Ф. Расшифрованные секреты. Методы и принципы криптологии. М.: Мир, 2007. 550 с.

[2)] [2] С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития - PDF Free Download / URL: http://docplayer.ru/25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya.html (дата звернення: 05.06.2020)

Розглянемо детальніше застосування методу лінійного криптоаналізу у загальному вигляді:

$$\sum_{i=1}^{a} P_i \oplus \sum_{j=1}^{b} C_j = \sum_{l=1}^{c} K_l$$

де $i_1, i_2,...,i_a, j_1, j_2 ..., j_b$ *та* $l_1, l_2,....,l_c$ – позиції деяких бітів відкритого тексту P_i, шифротексту C_j ключа K_l.

Успішність проведення лінійного криптоаналізу має значну залежність від структури S- блоків. При детальному розгляді шифру DES виявилось, що S-блоки уразливі і не оптимізовані проти цього способу розкриття, а стійкість до лінійного криптоаналізу не була закладена ще на етапі проектування DES, що робить його уразливим. Розробники не передбачили можливості захисту та стійкості проти лінійного криптоаналізу, хоча передбачили стійкість проти диференціального криптоаналізую, який є більш небезпечним.

При використанні методу лінійного наближення необхідно 247 відомих відкритих блоків, що надасть добрі результати застосування цього методу, а результатом буде лише отримання одного біта ключа, що дуже мало. Але, якщо використовувати для аналізу шифротекст, а не відкритий текст, а також розшифрування замість шифрування, то в результаті лінійного криптоаналізу можна отримати 2 біти ключа, що також є недостатньо. Але, якщо використати застосування механізму криптоаналізу паралельно 2^{12} разів та ґрунтуючись на ймовірностях передбачити правильний варіант, то таке рішення цього завдання надасть вже 12 біт ключа, що вже краще ніж розглянуті вище варіанти. В подальшому, змінивши процес шифрування на розшифрування можливо отримати ще 13 біт ключа. А на завершення, виконавши повний перебір отримати ще 30 біт.

Застосування такої схеми розкриття повного 16-раундового DES передбачає застосування близько 243 відомих відкритих текстів. А програмна реалізація цього методу на дванадцяти робочих станціях HP9735, розкрила

ключ DES за 50 днів. Саме метод лінійний криптоаналізу дає подальший розвиток для з'яви нових алгоритмів та ідей.

2.6 Постановка задачі

Метою магістерської роботи є дослідження, аналіз та виконання криптоаналізу асиметричних алгоритмів на основі факторизації складених чисел за допомогою мурашиного алгоритму та розробка додатку для здійснення криптоаналізу асиметричних алгоритмів шифрування.

Для досягнення цієї мети були визначені задачі, які необхідно реалізувати:

– здійснити дослідження та механізми реалізації мурашиного алгоритму для криптоаналізу асиметричних криптосистем;
– здійснити реалізацію програмного додатку для виконання криптоаналізу мурашиним алгоритмом;
– виконати тестування різних конфігурацій алгоритму та тестування алгоритму з різними наборами початкових параметрів.
– визначити переваги та недоліки застосування мурашиного алгоритму для виконання криптоаналізу.

Практична частина роботи, а саме розробка програмного додатку передбачає, що розроблений додаток може бути використано для проведення тестування різних конфігурацій мурашиного алгоритму з різними наборами початкових параметрів при здійсненні криптоаналізу асиметричних алгоритмів шифрування.

3 ІНТЕЛЕКТУАЛЬНІ МЕТОДИ КРИПТОАНАЛІЗУ

Сучасними методами, які все частіше використовуються при здійсненні криптоаналізу є такі алгоритми, як криптоаналіз за допомогою нейронних мереж, які можуть бути засновані на використанні біоінспірірованіх алгоритмів та різних відомих еволюційних методів.

3.1 Застосування нейронної мережі в завданнях криптоаналізу

Будь-яку криптосистему можливо розглядати як «чорний ящик», тобто як програму , внутрішня структура якої не відома для користувача, але, виконуючи подання сигналів різних команд або визначеному набору даних на вхід, можливо отримати деяку визначену реакцію на виході. Завдання криптоаналізу можливо визначити як процес ідентифікації обраної системи, а саме визначення її структури, що основана на сигналах, які можуть надходити на вхід цієї системи та які можуть бути одержані на виході. Для вирішення такого завдання можливими інструментами можуть бути обрані нейронні мережі.

Штучна нейронна мережа – це математична модель, а також пристрої паралельних обчислень, що представляють собою систему з з'єднаних і взаємодіючих між собою простих процесорів. Ці процесори зазвичай є прості, в порівнянні з процесорами, які використаються у ПК. Процесори такої мережі мають справу тільки з сигналами, які отримує та періодично посилає іншим процесорам. Будучи з'єднаними в досить велику мережу з керованим взаємодією, такі локально прості процесори разом здатні виконувати досить складні завдання. Поняття виникло при вивченні процесів, що протікають в мозку при мисленні, і при спробі змоделювати ці процеси. Отримані моделі називаються штучними нейронними мережами [9][1)].

[1)] [9] Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. К.:«Корнійчук», 2008. 446 с.

Ідентифікація системи полягає у визначенні правил функціонування системи – «чорного ящика» по вхідним і вихідним даним і апроксимації невідомої функції моделі нейронної мережі. Першим етапом є вибір нейронної моделі, яка характеризується певною архітектурою і алгоритмом навчання. Вибір здійснюється методом проб і помилок. Безліч відомих пар відкритих текстів і криптограми розділяється на дві підмножини, одне з яких використовується для навчання мережі, а інше – для перевірки відповідності отриманої моделі заданому критерію точності [9][1)].

З залучанням механізмів нейронної мережі виконувався криптоаналіз класичного поліалфавітного шифру Віжінера і поточних шифрів. Якщо встановити дякий поріг помилки на рівні 10^{-5} була отримана модель криптосистеми, яка надавала користувачам результат, що забезпечував 100%-ву точність. Якщо здійснювати порівняльний аналіз різних сучасних методів, то можливо зазначити переваги такого підходу, які полягають в незалежності результату від природної мови та її статистичних характеристик, обумовлений використанням адаптивного навчального процесу для отримання системою.

3.2 Використання генетичних алгоритмів при здійсненні криптоаналізу

Один з перших шифрів на основі завдання про укладання ранця був запропонований Мерхнули і Хеллманом в 1978. Це була одна з перших спроб створення системи шифрування з відкритим ключем. Незважаючи на те, що проблема укладання ранця ставиться до класу N_p-повних, було показано, що більшість версій алгоритму є нестійкими. В 1983 р. Брикел запропонував спосіб злому криптосистеми на основі ранця низької щільності. Рік тому

[1)] [9] Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. К.:«Корнійчук», 2008. 446 с.

Шамир розробив поліноміальний алгоритм для атаки на вихідну «рюкзачную» криптосистему. Після цього була запропонована безліч інших систем на основі алгоритму укладання ранця: кілька послідовних рюкзаків, рюкзаки Грєм-Шамира (Garham-Shamir) і ін. Для всіх цих систем були розроблені методи розкриття. Пропонується ще один метод криптоаналізу шифрів на основі алгоритму укладання ранця; відмінною рисою такого підходу є його універсальність, тобто можливість застосування до будь-якої версії «рюкзачної» криптосистеми, а також простота роботи. Метод базується на використанні генетичних алгоритмів. Генетичні алгоритми були розроблені Джоном Холландом і є модифікацією так званого «еволюційного програмування». Ідея Холланда полягала в тому, щоб створити алгоритм пошуку на основі механізмів природного відбору, відомого з біології, або алгоритм «спрямованого» випадкового пошуку. На етапі ініціалізації процедури створюється популяція можливих рішень. На основі цієї популяції виводиться нове покоління рішень, яке, в свою чергу, служить «вихідним матеріалом» для чергового покоління, і т.д. Цикл генетичного алгоритму в загальному вигляді включає стадії відбору, схрещування і мутації. Кращі представники покоління відбираються для відтворення популяції; таким чином, за припущенням, кожне нове покоління повинно містити кращі рішення, ніж попереднє. У багатьох випадках це відповідає дійсності. Популяція складається з набору бінарних рядків. Кожен бінарний рядок представляє рішення проблеми і називається хромосомою [9][1)].

Відзначимо, що на початку генетичного алгоритму передбачено механізм відбіру. У процесі здійснення відбору визначаються рядки, які будуть використовуватися при створенні нового покоління. «Батьки» вибираються довільно, там чином, що «кращі особини» популяції мають більшу вірогідність бути обраними, тому такий алгоритм має тенденцію просуватися в най перспективному із можливих напрямів здійснення пошуку.

[1)] [9] Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. К.:«Корнійчук», 2008. 446 с.

На наступному етапі відбувається схрещування, яке забезпечується наступним механізмом: пари відібраних рядків довжини *r,* кожна з яких обирається довільним чином число $S \in \{1,...,r\}$. «Батьки» здійснюють обмін бітами від $S+1$ до r; що дозволяє з'являться хромосомам нащадків.

А останній етап полягає у проявленні так званої мутації. При здійсненні ініціалізації алгоритму з'являється маленька ймовірність мутації, що є фіксованою, що передбачає надання їй новоутворених хромосом. Всі вище перелічені етапи повторюються до тих пір, поки не будуть досягнені умови виходу із циклу для його генетичного алгоритму. Прикладом умови виходу може бути значне перевищення чисельної кількості популяцій.

Для здійснення постановки задачі ознайомимося з задачею «про укладення ранця», яка є типовою при розгляді цього алгоритму. Припустимо, що за умовою задачі надано деяка кількість предметів, що мають різну вагу. Для вирішення задачі необхідно визначити: чи можливо розташувати деяку кількість предметів у ранець таким чином, щоб вага цього ранця дорівнювала деякому визначеному значенню. Формалізуємо задачу: надано набір значень M_1, M_2, ..., M_n , а сумарне їх значення дорівнює S, необхідно здійснити обчислення значення b_i так, щоб виконувалась наступна вимога:

$$S = b_1M_1 + b_2M_2 + ... + b_nM_n \tag{3.1}$$

де b_i може дорівнювати тільки одному з двох можливих значень: нуль або одиниця. Одиниця визначає, що предмет поклали у рюкзак, а нуль – не поклали. Тому вміст рюкзака можливо представити, як хромосоми, а біти цих хромосом відповідають значенням b. Реалізація функції вибору «кращих хромосом» може здійснювати оцінювання близькості ваги заданого рюкзака до заданого числа. Область значення функції знаходиться в діапазоні [0; 1], де одиниця дає точний збіг із вагою, яку шукаємо. Але, якщо вага одного з рюкзаків перевищує задане деяке значення S на число x, а вага другого рюкзака, навпаки, менше необхідного на те ж число x, то можливо зробити висновок,

що «найкращим» буде другий рюкзак. Визначений алгоритм заповнення рюкзака можливо описати формалізованою моделлю.

Перш за все необхідно обчислити максимальну розбіжність, яка може бути між будь-якою хромосомою та визначеним значенням *S.,* тому $\Delta_{max} = \max(S, \hat{S} - S)$, де $\hat{S}$ –сума всіх предметів, які можливо застосувати при укладенні рюкзака.

На наступному етапі, відповідного до поточної хромосоми, необхідно обчислити вагу рюкзака, яку визначимо як S'. При умові, що $S' \leq S$, якість хромосоми можливо оцінити, використовуючи вираз:

$$\alpha = 1 - \sqrt{\frac{|S' - S|}{S}}$$

При умові, що $S' > S$, якість хромосоми можливо обчислити за виразом:

$$\alpha = 1 - \sqrt{\frac{|S' - S|}{\Delta_{max}}}$$

Визначимо послідовно дії алгоритму, що може бути використаний у загальному виді для здійснення криптоаналізу [7][1)]:

- на початку буде створена випадкова популяція двійкових хромосом;
- надалі для кожної хромосоми виконується функція оцінки: вичислюється значення α;
- відбувається природний добір, якій визначається отриманими коефіцієнтами;
- механізм схрещування відбувається тільки до обраних особин;
- очевидно, що мутації піддаються нащадки;
- отримана нова популяція піддається аналізу, результатом якого є отримання «кращих хромосом».

[1)] [7] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.

Алгоритм буде виконуватись до тих пір, поки кількість поколінь зможе перевищити деяке визначене значення, а «кращі хромосоми», які зявляються в результаті виконання алгоритму, можуть бути обрані для проведення операції злому шифру.

Застосування генетичного алгоритму для простого шифру підстановки, навів ефективність використання, про що свідчить показник досягнення оптимальної крапки, тобто секретного ключа: засобами алгоритму досліджено не більш двох відсотків усього ключового простору. Тому застосування генетичних алгоритмів успішно використовується при здійсненні криптоаналізу шифрів для перестановки і підстановки.

3.3 Застосування мурашиного алгоритму в криптоаналізі

Розглянуті вище структури генетичних алгоритмів дозволяють зробити висновок, що застосування таких структур надають невизначені пошукові механізми, що мають достатньо велику кількість недоліків. Це зумовило появу евристичних методів, основні механізми яких визначені живою природою або фізичними процесами, що обумовлено алгоритмом поетапного відбудовування рішення, що дозволяє до побудованого рішення поступово додавати новий компонент.

До евристичних методів, що диктує сама природа, відносять і мурашині алгоритми. З використанням методу мурашиних колоній реалізується криптоаналіз шифрів перестановки. Відмінною особливістю застосування алгоритмів «мурашиних колоній» є необхідність подання завдання у вигляді графових моделей, на якій мурахи можуть будувати рішення. Даний алгоритм використовується також для криптоаналізу асиметричних алгоритмів [10][1)].

[1)] [10] Сергеев А.С., Третьяков О.П., Васильев А.Е., Чернышёв Ю.О. Биоинспирированные методы криптоанализа асимметричных алгоритмов шифрования на основе факторизации составных чисел. Вестник ДГТУ 2011;11(9):1544-1554. URL: https://www.vestnik-donstu.ru/jour /article/ view/888 (дата звернення: 10.07.2020)

Розглянемо механізм, що покладено в основу мурашиного алгоритму. Деяка мураха прямує шляхом від мурашника (М) у випадковому напрямку. Коли мураха знаходить їжу (Ї), то мураха прямує назад, до мурашника, але при цьому залишає за собою на своєму шляху слід з феромону. Схематично цей процес наведено на рисунку 3.1.

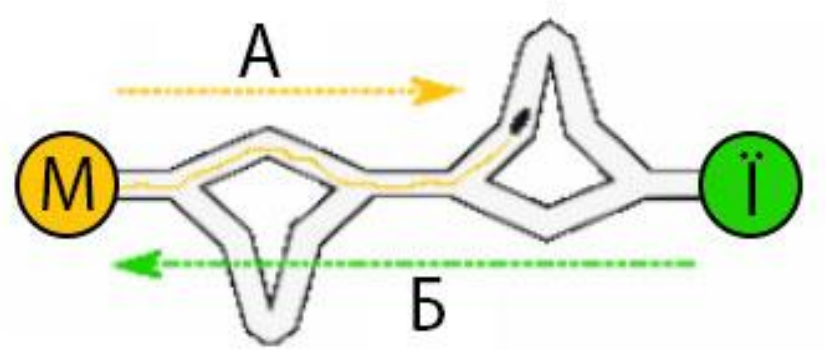

Рисунок 3.1 – Процес пошуку найкоротшого шляху від колонії до джерела живлення і назад

Феромони, які залишила мураха з найденою їжею, приваблюють інших мурах, що знаходяться поруч, і ці мурахи скоріш за все оберуть саме цей маршрут. Отримавши їжу, повертаючись до мурашника, ці мурахи теж залишають на своєму шляху свої феромони, що приведе через деякий час к укріпленню доріжки, що наведено на рисунку 3.2.

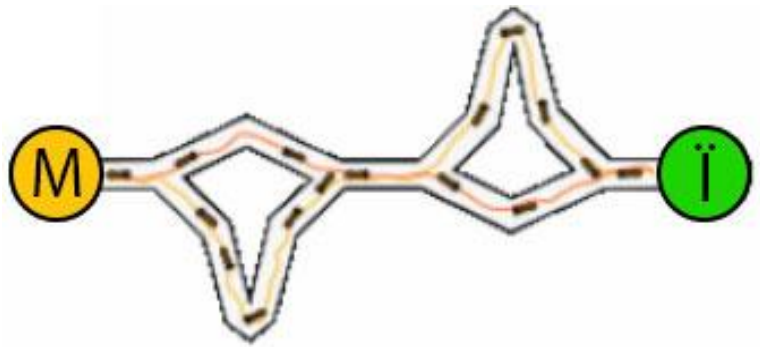

Рисунок 3.2 – Процес руху мурах двома шляхами (коротким та довгим) від колонії до джерела живлення

Припустимо, що існує деякі два маршрути: один з них довгий, а другий коротший, то мурах, які користуються коротким маршрутом, пройде більше ніж мурах, які пройшли довгим маршрутом. Тому коротший маршрут стане більш популярним і привабливим для мурах. Як наслідок, з часом, довгі доріжки будуть зникати за рахунок випаровування з феромонів, що наведено на рисунку 3.3.

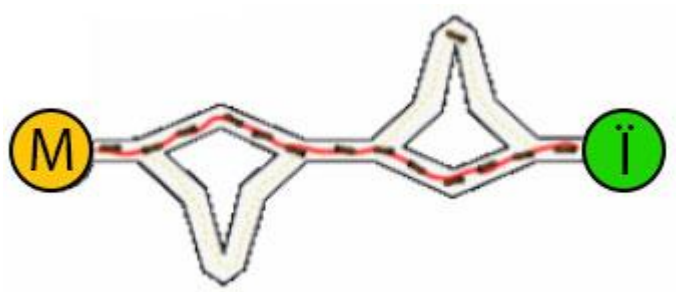

Рисунок 3.3 – Процес випарювання феромону на довгому шляху і укріплення коротшого шляху від колонії до джерела живлення

Основна ідея мурашиного алгоритму очевидна: виконання оптимізації, що виконується шляхом непрямого зв'язку між автономними агентами. Мурашиний алгоритм наводить модель багатоагентної системи з агентами «мурахи».

Ці визначені агенти досить прості: як і у справжній природі, ці мурахи для виконання своїх завдань потребують невелику кількість пам’яті, а кожен крок роботи мурахи зумовлено виконанням незначних обчислень [10] [1)] .

В пам’яті кожного агента-мурахи зберігається інформація що до вузлів, які їм були пройдені. Такий список називають пам'яттю агента-мурахи або tabu list. Агент-мураха тримає в пам’яті вузли, які нею були пройдені та для виконання наступного кроку не розглядає їх як можливі для здійснення пере-

[1)] [10] Сергеев А.С., Третьяков О.П., Васильев А.Е., Чернышёв Ю.О. Биоинспирированные методы криптоанализа асимметричных алгоритмов шифрования на основе факторизации составных чисел. Вестник ДГТУ 2011; 11(9):1544-1554. URL: https://www. vestnik-donstu. ru /jour /article/ view/888 (дата звернення: 10.07.2020)

ходу. Здійснюючи крок за кроком цей список заборон у кожної мурахи поповнюється новим вузлом, а перед новою ітерацією алгоритму – тобто перед тим, як агент-мураха знову здійснює пошук їжі, і проходиться шлях від мурашника до їжі і назад, шлях очищується. Але мураха керується не лише списком заборон при виборі вузла для переходу, але має значення і деяка привабливість ребер, які мураха збирається пройти. Мураху приваблює відстань між вузлами (ваги ребра для переходу), а також товщина шляху феромонів, що залишили на ребрах мурахи, які пройшли раніше цим шляхом. В цьому алгоритмі постійними величинами будуть ваги кожного з ребер, а сліди феромонів будуть при кожній ітерації алгоритму різними: одні випаровуються, а другі, навпаки посилюються значною кількістю проходів мурах. Наведемо узагальнену блок-схему мурашиного алгоритму (рис. 3.4)

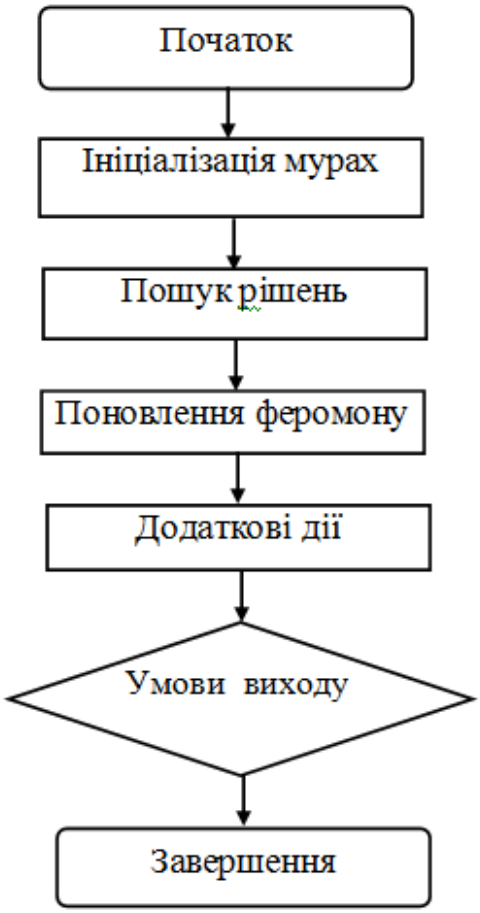

Рисунок 3.4 – Схема базового мурашиного алгоритму

Формалізуємо описаний вище алгоритм та визначимо покроковий опис загальної схеми.

Визначимо, що навколишнє середовище для агентів-мурах є повнозв’язний неорієнтований граф, а кожне з ребер цього графа визначається як відстань між двома вершинами, що ним з’єднується, яка і буде визначать вагу ребер. Той факт, що мураха може рухатись в будь-якому напрямку, визначає цей граф, як двох скерований (рис. 3.5).

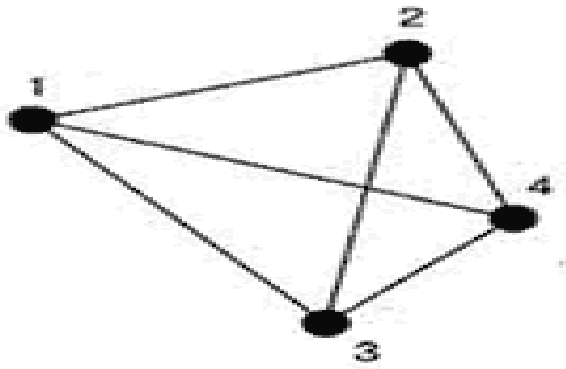

Рисунок 3.5 – Граф є двохскерованим

Зазначимо, що ймовірність, що деяке з ребер буде обране маршрутом агента-мурахи, пропорційно до кількості феромонів на цьому ребрі, а кількість відкладеного феромону пропорційне довжіні маршруту. Чим коротший маршрут, тим більше феромону буде відкладено на його ребрах. Тому, очевидно, що найкоротший маршрут буде обирати більша кількість агентів-мурах. Використання такої моделі передбачає збіжність: більшість мурашок обирає локально-оптимальний шлях і як наслідок маршрут. Це передбачає додатній зворотний зв'язок, що призводить до передчасної збіжності. Для уникнення цього додамо від’ємний зворотний зв'язок, що передбачає врахування випаровування феромону.

Але, якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, а, збільшення часу випаровуван-

ня феромону призводить до отримання стійкого локального оптимального рішення.

Використовуючи мурашиний алгоритм необхідно визначити програмного агента, яким і є мураха, що зумовлено наявністю колонії таких агентів, які можуть бути використаними для рішення будь-якого завдання. Цей програмний агент має набір правил, що забезпечує мураху наданням повноважень обирати шлях у графі.

Агент-мураха пам’ятає список вузлів, які вже відвідала і може пройти через кожен вузол лише один раз. Шлях між двома вузлами графа, за яким мураха відвідала кожен вузол, називається шляхом Гамільтона.

В пам’яті мурахи вузли розташовуються в тому порядку, в якому їх відвідала мураха, що дозволяє в подальшому використовувати цей список для визначення довжини шляхів між вузлами. Крім того, кожна мураха при переміщенні обов’язково залишає на ребрах графа феромони, але тільки на шляху до мурашника, тобто при завершення подорожі, маючи здобуту їжу.

Для кожного завдання, яке буде вирішене засобами мурашиного алгоритму, позиціонування мурахів буде залежати від накладених обмежень умовами самого завдання. Мурахи можуть бути розкидані по вузлах графа, або всі находитись в одній начальній точці. При формулюванні завдання також визначається кількість феромону, а саме початковий рівень. Якщо ввести деяке невелике додатне число, то при першому виконанні проходу алгоритму, ймовірності переходу в наступну вершину будуть нульовими.

Визначимо рівняння, яке буде описувати переміщення агента-мурахи. До тих пір поки агент-мураха не відвідає всі вузли зазначеного в завданні графа, щоб визначити наступне ребро графа використовується наступне ймовірнісне рівняння (3.2):

$$P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}. \tag{3.2}$$

де $\tau(r,u)$ – інтенсивність ферамону на ребрі між вузлами r і u;

$\eta(r, u)$ – функція, яка представляє вимір зворотної відстані для грані,

α – вага ферамону, а β – коефіцієнт евристики. Параметри α і β визначають відносну значимість двох параметрів, а також їх вплив на рівняння.

В зазначеному списку обмежень визначено, що мурахі дозволено рухатись тільки по ребрам до вузлів, які ще не були відвідані, то ймовірність доцільно обчислювати тільки для ребер, які ведуть до не відвіданих мурахою вузлів. Тоді, змінна k – ребра, які ведуть до не відвіданих мурахою вузлів.

Оскільки алгоритм передбачає використання списку обмежень, то цикли будуть заборонені, а шлях мурахи відобразиться тільки тоді, коли мураха відвідає всі вузли наведеного графа. Для підрахування довжини шляху, яким подорожувала мураха, треба дочекатися завершення подорожі. Довжина шляху визначається, як сума довжин всіх ребер, що пройшла мураха. Визначимо рівняння, що буде визначати кількість феромону, який залишила мураха на кожному ребрі шляху k (3.3). Змінна Q є константою.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}. \tag{3.3}$$

Результат, отриманий при обчисленні цього рівняння визначає і довжину шляху: найкоротшій шлях буде мати більшу величину концентрації феромонів, а найдовший шлях буде мати меншу величину концентрації феромонів. Цей результат використовується в рівнянні (3.4), для збільшення значення, що визначає кількість феромону на кожному з ребер, які пройшла мураха:

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t) \times \rho). \tag{3.4}$$

Наведене рівняння (3.4) застосовується для визначення значення всього шляху, а значення, яке визначає кількість феромону для кожного з ребер буде

пропорційне до довжини шляху. Тому рівні феромону оновлюються лише після закінчення мурахою подорожі, інакше довжину шляху неможливо буде визначити. Константа *p* має значення між нулем та одиницею.

Розглянемо алгоритм випаровування феромону. На початку алгоритму (при першій ітерації) у кожного з ребер графа, якими визначається майбутній шлях є можливість бути обраним. Але при подальшому виконанні алгоритму, щоб здійснювати поступове видалення ребер, застосовується застосовується механізм випаровування феромону з тих шляхів, які є найдовшими. Використовуючи константу *p* з рівняння (3.4), отримуємо рівняння (3.5):

$$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho). \tag{3.5}$$

Випаровування феромону обумовлене застосуванням зворотнього коефіцієнту оновлення шляху. Алгоритм має зупинку якщо виконується умова: шлях агента-мурашки завершено. При цьому, виконується оновлення ребер з урахуванням довжини шляху, а також відбувається на всіх ребрах графа випаровування феромону, і лише тоді алгоритм повторюється. На початку алгоритму список обмежень оновлюється, а довжини шляхів дорівнюють нулю. А агенти-мурахи знову обирають шляхи переміщення по ребрам графу, використовуючи результати обчислень рівняння (3.1). Цей алгоритм може буде адаптовано як для постійної кількості шляхів або до моменту, коли при наступній ітерації не відбуваються ні яки зміни. Рішенням є кращій шлях.

Визначимо характерні особливості для здійснення програмної реалізації мурашиного алгоритму. Необхідними є наступні початкові данні: визначити закони виділення і випаровування феромону, кількість агентів-мурах та початкові місця їх розташування. Обираються початкові данні враховуючи особливості кожного з вирішуемих завдань. Мурашиний алгоритм реалізує пошук наближених рішень; має поліноміальну складність; є одним з видів імовірнісних алгоритмів (закони виділення випаровування – імовірнісні закони).

3.4 Модифікації класичного алгоритму

Простота класичного мурашиного алгоритму («мурашиної системи») залишала можливості для доробок –і саме алгоритмічні вдосконалення стали предметом подальших досліджень фахівців у галузі комбінаторної оптимізації. В основному, ці вдосконалення пов'язано з великим використанням історії пошуку та більш ретельним дослідженням областей навколо вже знайдених вдалих рішень [5][1)].

Elitist Ant System. Введення в алгоритм так званих «елітних мурах». Досвід показує, що проходячи ребра, що входять в короткі шляхи, мурахи з більшою ймовірністю будуть знаходити коротші шляхи. Ефективною стратегією є штучне збільшення рівня феромонів на найвдаліших маршрутах. Для цього на кожній ітерації алгоритму кожна з елітних мурах проходить шлях, який є найкоротшим із знайдених на даний момент.

Експерименти показують, що, до певного рівня, збільшення числа елітних мурах є досить ефективним, дозволяючи значно скоротити число ітерацій алгоритму. Однак, якщо число елітних мурах занадто велике, то алгоритм досить швидко знаходить субоптимальне рішення і застряє в ньому. Як і інші змінні параметри, оптимальне число елітних мурах слід визначати дослідним шляхом.

Ant-Q. Мурашиний алгоритм, який отримав свою назву за аналогією з методом машинного навчання Q-learning. В основі алгоритму лежить ідея про те, що мурашину систему можна інтерпретувати як систему навчання з підкріпленням. Ant-Q підсилює цю аналогію, запозичуючи багато ідей з Q-навчання.

Алгоритм зберігає Q-таблицю, що співставляє кожному з ребер величину, яка визначає «корисність» переходу по цьому ребру. Q-таблиця змінюється в процесі роботи алгоритму – відбувається навчання системи. Значення

[1)] [5] Чернышев Ю.О., Сергеев А.С., Венцов Н.Н., Рязанов А.Н. Исследование возможности применения генетических алгоритмов для реализации криптоанализа блочных криптосистем. Вестник ДГТУ 2015;15(3):65-72. URL: https://doi.org/10.12737/12599

корисності переходу по ребру обчислюється виходячи із значень корисності переходу за наступними ребрам в результаті попереднього визначення можливих наступних станів. Після кожної ітерації корисності оновлюються виходячи з довжин шляхів, до складу яких було включено відповідні ребра.

Ant Colony System. Для підвищення ефективності в порівнянні з класичним алгоритмом введено три основних зміни.

По-перше, рівень феромонів на ребрах оновлюється не лише в кінці чергової ітерації, але і при кожному переході мурах з вузла у вузол. По-друге, наприкінці ітерації рівень феромонів підвищується тільки на найкоротшому із знайдених шляхів. По-третє, алгоритм використовує змінене правило переходу: або, з певною часткою ймовірності, мураха безумовно вибирає краще ребро у відповідності до довжини і рівня феромонів, або робить вибір так само, як і в класичному алгоритмі.

Max-min Ant System. Мурашиний алгоритм, в якому підвищення концентрації феромонів відбувається тільки на кращих шляхах з пройдених мурахами. Така велика увага до локальних оптимумів компенсується введенням обмежень на максимальну і мінімальну концентрацію феромонів на ребрах, які вкрай ефективно захищають алгоритм від передчасної збіжності до субоптимальних рішень.

На етапі ініціалізації, концентрація феромонів на всіх ребрах встановлюється рівною максимальній. Після кожної ітерації алгоритму тільки одна мураха залишає за собою слід – або найбільш успішна на даній ітерації, або, аналогічно до алгоритму з елітизмом, елітна. Цим досягається, з одного боку, більш ретельне дослідження області пошуку, з іншого – його прискорення.

ASrank. Модифікація класичного мурашиного алгоритму, в якому в кінці кожної ітерації мурахи ранжуються у відповідно до довжин пройдених ними шляхів. Кількість феромонів, що залишається мурахою на ребрах, таким чином, призначається пропорційно до її позиції. Для ретельного дослідження околу вже знайдених вдалих рішень, алгоритм використовує елітних мурах.

3.5 Виконання криптоаналізу мурашиним алгоритмом

Основне слабке місце RSA-подібних алгоритмів – задача факторизації. Знаючи це, будемо використовувати в якості початкових параметрів відкритий ключ *(e, N)*, де *N* – просте число [11][1)] .

Зі своєї природи, мурашиний алгоритм використовує уявлення задачі як набір компонент і переходів або як набір неорієнтованих зважених графів, на яких можна будувати рішення. У цьому випадку задача криптоаналізу зводиться до задачі знаходження найменшого шляху в графі [10][2)]. Необхідно визначити його подільники на відрізку $\lfloor n_i, n_j \rfloor$ із заданим ступенем точності, тобто знайти такі числа (x, y), для яких $\left(\left(\frac{N}{x_i}\right)-\left(\frac{N}{x_j}\right)\right)\to min$. Нехай числа $x_k \in \lfloor n_i, n_j \rfloor$ містяться в оперативній пам'яті одного процесора (або «острова» процесорів) і є вершинами повного графа $G=(X,U)$, де $|X|=n_j-n_i+1$ – безліч вершин, $|U|=\frac{(n_j-n_i+1)\cdot(n_j-n_i)}{2}$ – безліч ребер, при цьому вага вершини x_j дорівнює $F(x_j)=\left(N/x_j\right)-\left(N/x_j\right)$, тобто дорівнює дробової частини приватного, отриманого від ділення *N* на *x*. Формулювання задачі знаходження найкоротшого шляху буде виглядати наступним чином: знайти маршрут *T* в графі *G*, що містить задане число *m* вершин, що задовольняє умові

$$\sum_{i=1}^{m} F(x_i) \to min$$

Вага кожного маршруту T_k можливо визначити як (L_k, F_k), де

[1)] [11] Остапов С. Е., Валь Л.О. Основи криптографії: навчальний посібник. Чернівці: Книги–XXI, 2008. 188 с.

[2)] [10] Сергеев А.С., Третьяков О.П., Васильев А.Е., Чернышёв Ю.О. Биоинспирированные методы криптоанализа асимметричных алгоритмов шифрования на основе факторизации составных чисел. ВДГТУ. 2011;11(9):1544-1554. URL: https://www.vestnik-donstu.ru/jour /article/ view/888 (дата звернення: 10.07.2020)

$$L_k = \sum_{x \in T_k} F(x_i) \qquad F_k = \min_{x_i \in T_k} F(x_i)$$

Очевидно, що завданням затосування мурашиного алгоритму в цьому-випадку є визначення маршруту, що має мінімальну вагу L_k мА має *m* вершин і вершину графа x_k, для якої $F(x_k) = (N/x_k) - (N/x_k) \to min$, тобто яка є найбільш точним дільником числа *N*. Виходячи з умов мурашиного алгоритмом та системи обмежень, цикл, який визначає життя мурашиної колонії буде виконуватись до тих пір поки не будуть пройдені всі вершини графа *G,* тоді вершина графу $x_k \in \lfloor n_i, n_j \rfloor$ буде визначена за число тимчасових циклів С алгоритму:

$$C = \frac{n_j - n_i + 1}{m \cdot M}$$

де M – розмір популяції мурахів, *m* – число вершин у маршруті мураха.

Виходячи з початкових даних, ваги вершин графу $F(x_j)$ є невідомими величинами, а можуть бути визначені тільки при здійсненні ітерацій алгоритму, то мурахи k здійснить перехід з вершини x_i до x_j з ймовірністю, яка визначається співвідношенням:

$$P_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{l \in J_k} \tau il(t)}$$

де J_{ik} – безліч вершин, доступних мурасі *k*, що знаходиться в вершині *i*, $\tau_{ij}(t)$ – рівень феромону в момент часу *t* на ребрі x_i, x_j, *t* – параметр циклу часу життя колонії [10][1)].

[1)] [10] Сергеев А.С., Третьяков О.П., Васильев А.Е., Чернышёв Ю.О. Биоинспирированные методы криптоанализа асимметричных алгоритмов шифрования на основе факторизации составных чисел. ВДГТУ. 2011;11(9):1544-1554. URL: https://www.vestnik-donstu.ru/jour /article/ view/888 (дата звернення: 10.07.2020)

Довжина шляху, яка може бути визначена після проходження *k*-ю мурахою маршруту з *n* вершин дорівнює сумі всіх ваг вершин, які були пройдені мурахою на своєму шляху. Тоді можливо визначити значення кількісті відкладеного феромону на кожному з ребер x_i, x_j графу на визначеному шляху:

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L_k(t)}$$

le Q – параметр порядку довжини оптимального шляху (що визначає зменшення $\Delta\tau_{ij}^{k}$ зі збільшенням довжини маршруту $L_k(t)$).

Для здійснення оновлення значення кількості феромона на шляху застосовується наступне рівняння:

$$\tau_{ij}(t) = (1-\rho)\cdot\left(\tau_{ij}(t) + \sum_{k\in F}\Delta\tau_{ij,k}(t)\right)$$

де F –кількість мурахів, які відвідували на своєму шляху ребро x_i, x_j, ρ – інтенсивність випаровування.

Треба зауважити, що на першому етапі необхідно виконати позиціювання мурах з мурашиної колонії у вершинах графа, а деяка обмежена кількість агентів-мурах розміщується в вершинах графа випадковим чином, але без дозволу повторень.

Сформулюємо механізм виконання мурашиного алгоритму для вирішення задачі знаходження дільників числа N та виконаємо покроковий опис:

1) на початку виконаємо визначення початкових значень наступних параметрів: $Q, M, \rho, t_{макс}$;
2) далі задамо значення мінімальних ваг ребер, що зумовлена значеннями початкової концентрації феромону;
3) необхідно задати довжину найкоротшого маршруту – L^* , та визначити з якою точністю потрібно одержати результати – F^*;
4) необхідно ввести значення циклу по часу життя колонії: $t = 1$;

5) здійснити розміщення колонії M мурах не допускаючи збігів, в випадкові вершини графа;
6) надати значення циклу по мурахам: $k = 1$;
7) використуючи розподіл ймовірності по ребрах, кожна з мурах k створює маршрут $T_k(t)$, який має значення довжини $L_k(t)$;
8) Перевіряємо, якщо $k \le M$, то повертаємося до пункту 7, при $k = k + 1$;
9) перевірка всіх маршрутів T_k на краще рішення в порівнянні з L^* та F^*;
10) у разі, якщо отриманий маршрут $T_k(t)$ довжиною $L_k(t)$, який краще L^*, який краще x_j, для якої $F(x_j) < F^*$, оновити L^* та F^*;
11) цикл по ребрах графу $i = 1$;
12) $j = 1$;
13) оновити не мінімальні сліди феромону на ребрі x_{ij} ;
14) $j = j + 1$, якщо $j \le n_j - n_i + 1$, перейти до пункту 13;
15) $i = i + 1$, якщо $i \le n_j - n_i$, перейти до пункту 12;
16) $t = t + 1$, якщо умови зупинки не виконані, перехід к пункту 5;
17) вивести маршрут найкоротшої довжини L_k, а також вершину x, для якої $F(x)$ мінімально.

Таким чином, даний алгоритм дозволяє визначати маршрут, який містить задане число вершин x_i, для яких $\sum F(x_i) = min$ (тобто є найточнішими дільниками числа N на відрізку $\lfloor n_i, n_j \rfloor$), а також знайти вершину x_j, для якої $F(x_j) = min$, тобто яка є дільником числа N із заданим ступенем точності.

4 РОЗРОБКА ДОДАТКУ ДЛЯ ВИКОНАННЯ КРИПТОАНАЛІЗУ МУРАШИНИМ АЛГОРИТМОМ

4.1 Програмне середовище розробки

Якщо відомо відкритий ключ та розкладення простого числа на множники, завдання криптоаналізу зводиться до розширенного алгоритму Евкліда з відомим відкритим ключем та знайденими множниками. Тому даний програмний додаток був розроблений для рішення задачі факторизації.

Для реалізації даного програмного продукту була вибрана мова програмування C# і виконавче середовище Microsoft Visual Studio.

Для реалізації операцій з числами довжиною більше стандартних обмежень була використана структура BigInteger з простору імен System.Numerics програмного пакету .NetFramework 4.0 в Microsoft Visual Studio. Дана розробка дозволяє використовувати які завгодно величезні числа, обмежені тільки об'ємом пам'яті на жорсткому диску.

NET довгий час розвивалась головним чином як платформа для Windows під назвою .NET Framework. В 2019 вийшла остання версія цієї платформи – .NET Framework 4.8. Вона більше не розвивається. З 2014 Microsoft став розвивати альтернативну платформу – .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Потім Microsoft послідовно випустив ряд версій цієї платформи: .NET Core 1, .NET Core 2, .NET Core 3.

Для реалізації інтерфейсу була використана технологія WinForms з простору імен System.Windows.Forms програмного пакету .NetFramework 4.0 в Microsoft Visual Studio. Дана технологія дозволяє створювати віконні додатки для виконання в операційній системі Windows.

4.2 Проектування діаграми класів

Для реалізації генерації випадкових чисел був використаний RNGCryptoServiceProvider з простору імен System.Security .Cryptography програмного пакета .NetFramework 4.0 в Microsoft Visual Studio. Дана технологія дозволяє використовувати байтові масиви збільшеної розмірності, в поєднанні з бібліотекою BigConverter дозволяє створювати числові послідовності необхідної довжини [11][1)].

Клас Ant реалізує сутність мурашиного алгоритму із заданими параметрами.

Клас Ant_Functions містить статичні методи і реалізує допоміжні методи і функціонал мурашиного алгоритму.

Клас RandomGenerator реалізує генерацію випадкових чисел на основі сутності RNGCryptoServiceProvider.

Клас asimmetricalACOForm реалізує інтерфейс основної форми, що відображає результати роботи алгоритму, в тому числі поітераціонно.

Клас test реалізує інтерфейс форми із заготовленими заздалегідь тестами. Клас є спадкоємцем asimmetricalACOForm.

Клас startForm реалізує інтерфейс початкової форми, що надає користувачам вибір подальших дій.

Клас getParameters реалізує інтерфейс форми вибору початкових параметрів алгоритму.

Програмний додаток реалізовано одинадцятою створеними класами. Реалізовані у додатку для здійснення криптоаналізу асиметричних алгоритмів шифрування класи, наведено на рисунку 4.1.

[1)] [11] Алекс Макки. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. М. Вильямс. 2010. 416 с.

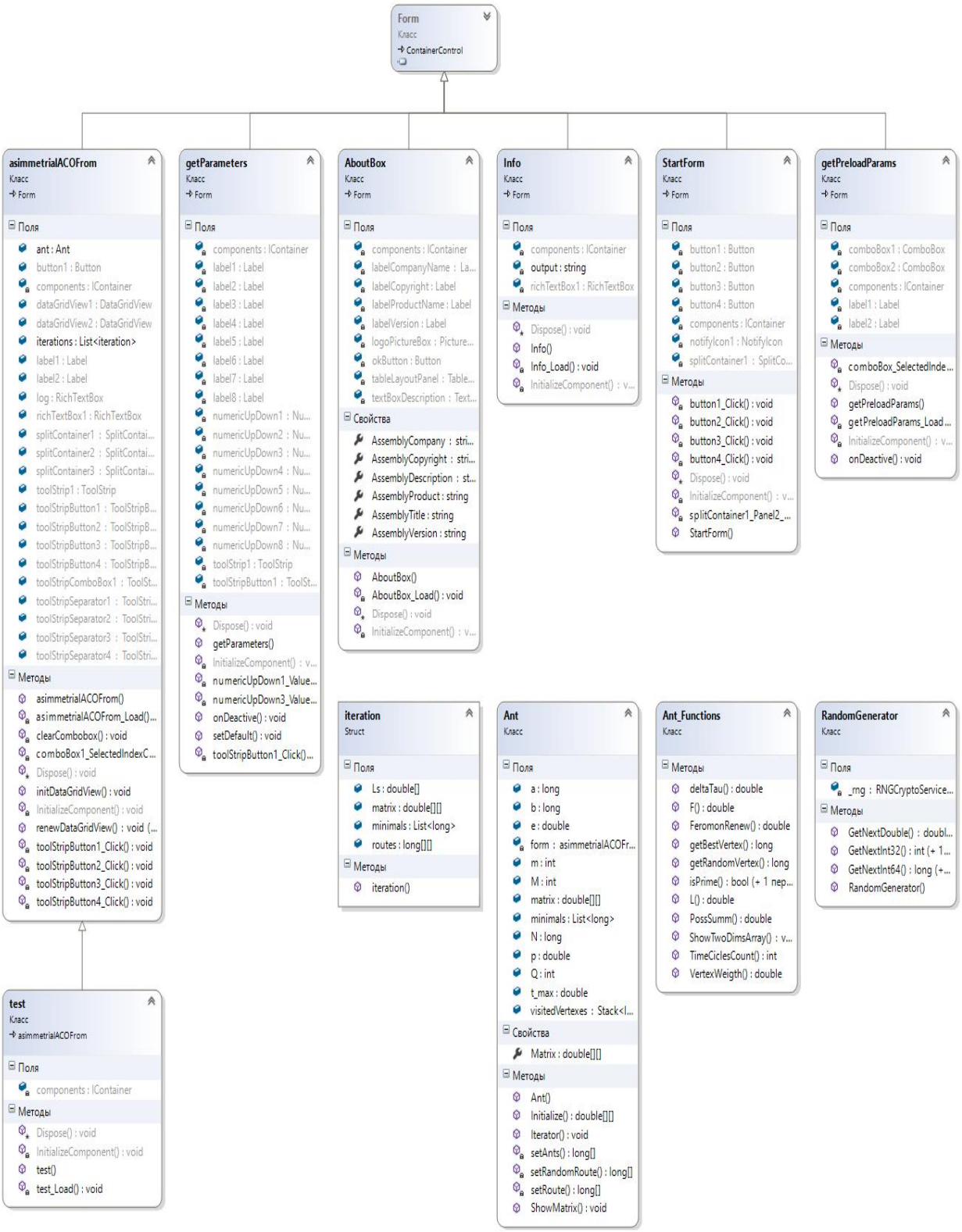

Рисунок 4.1 – Діаграма класів додатку

4.3 Основні методи, використовувані додатком

Конструктор Ant() класу Ant, що ініціалізує об'єкт цього класу по заданих параметрах, представлений наступним кодом:

```
    public Ant(int Q, int M, double p, double t_max, int m,
long N, long a, long b, asimmetrialACOFrom form)
            {
                this.Q = Q;
                this.M = M;
                this.p = p;// интенсивность испарения
                this.t_max = t_max;
                this.m = m;
                this.N = N;
                this.a = a;
                this.b = b;
                Initialize();
                e = 0.14;
                this.form = form;
            }
```

Граф викликів даного методу представлений на рисунку 4.2

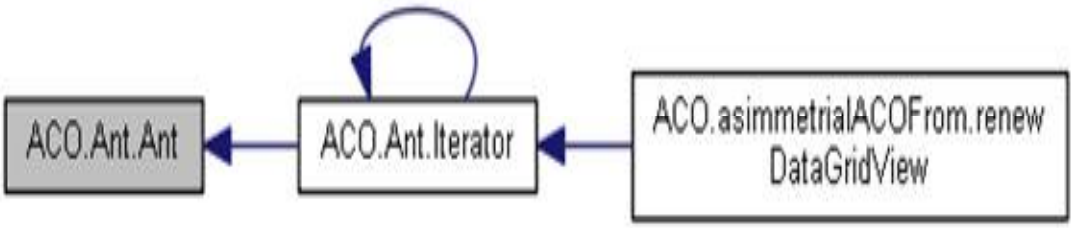

Рисунок 4.2 – Граф викликів методу Ant()

Метод ініціалізації Ant.Initialize() задає матрицю суміжності графа, за якої відбувається побудова маршрутів, в межах, заданих початковими параметрами і з урахуванням мінімального сліду феромону, заданого в конфігурації алгоритму.

```
public double[][] Initialize()
        {
            this.matrix = new double[b - a][];
            for (long i = 0; i < b - a; i++)
            {
                matrix[i] = new double[b - a];
                for (long j = 0; j < b - a; j++)
```

```
            {
                if (j == i)
                {
                    this.matrix[i][j] = 0;
                }
                else
                {
                    this.matrix[i][j] = t_max;
                }      }            }
        return matrix;   }
```

Граф викликів даного методу представлений на рисунку 4.3

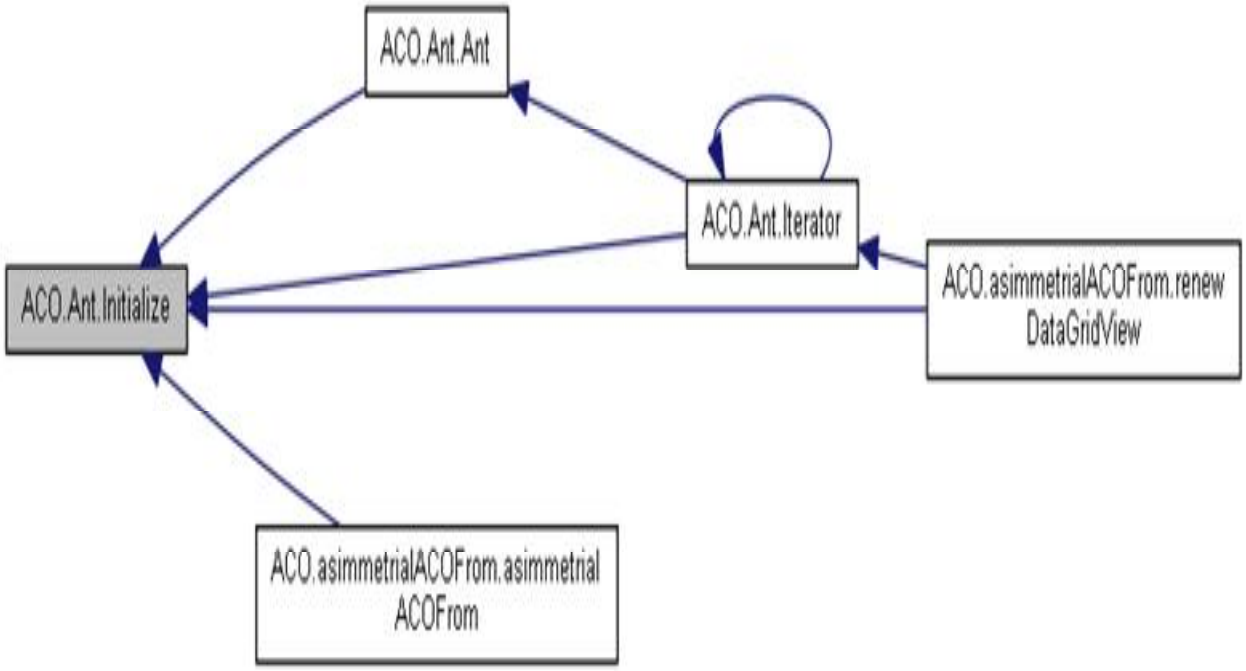

Рисунок 4.3 − Граф викликів методу Ant.Initialize()

Метод Ant.Iterator () реалізує ітераційний прохід алгоритму до тої пори, доки не буде досягнуто максимальну кількість ітерацій або не прийнята мінімальна вершина, вага якої буде дорівнювати нулю.

Після цього будуть обстежені всі мінімуми і, в разі, якщо це буде можливо, метод буде викликаний рекурсивно для пошуку подмножітелів, щоб були знайдені всі множники даного N. Лістинг коду методу Ant.Iterator() представлений в Додатоку А.

Граф викликів методу Ant.Iterator() представлений на рисунку 4.4.

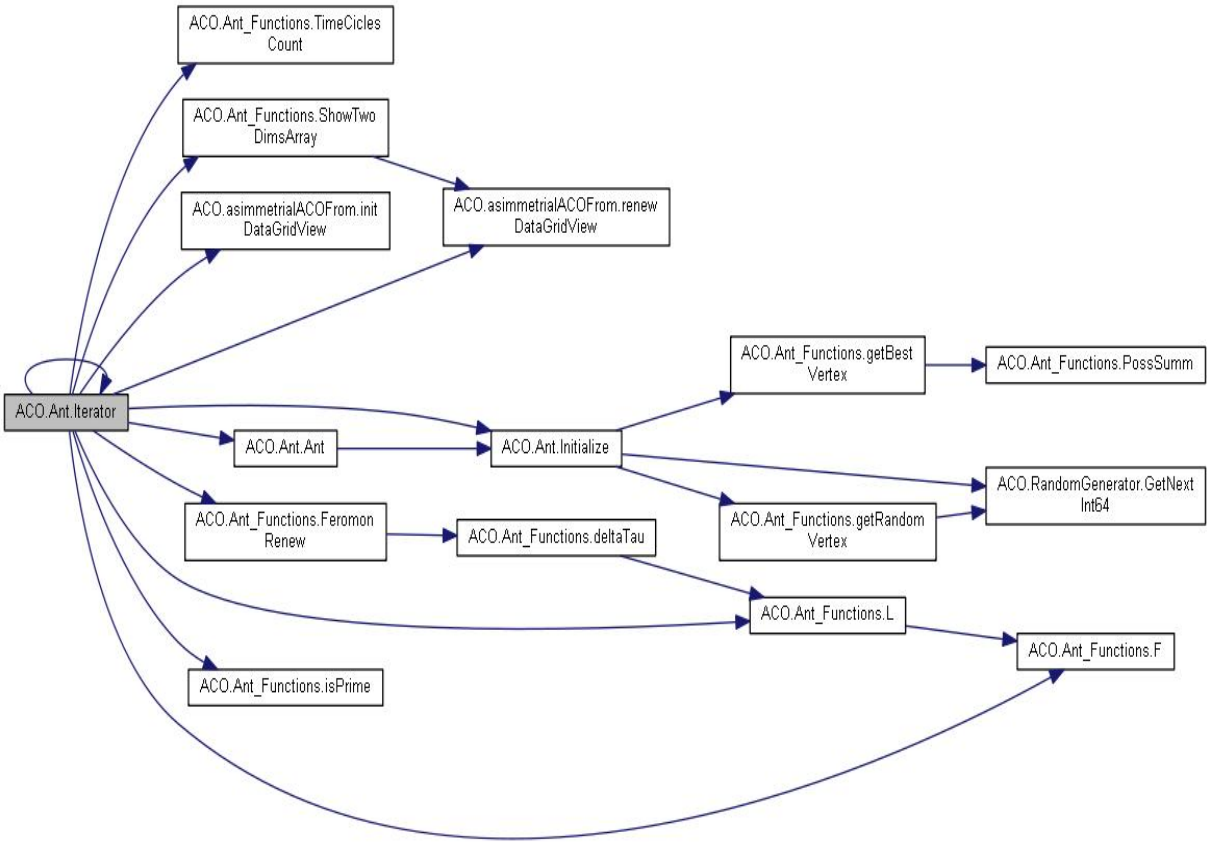

Рисунок 4.4 − Граф викликів методу Ant.Initialize()

Перевірка на простоту методом Міллера-Рабина реалізована в методі Ant_Functions.isPrime(). Він задіє допоміжні методи: modPow і mod. ModPow реалізує швидке піднесення до ступеня за модулем числа, mod реалізує швидке піднесення до ступеня чисел типу long. Код методу:

```
int modPow(int a, int x, int m)
        {
            if (x == 0) return 1;
            if (x == 1) return a % m;
            if (x % 2 == 0)
         return (modPow(a, x / 2, m) * modPow(a, x / 2, m)) % m;
            else
                return (modPow(a, x - 1, m) * a % m) % m;
        }
long mod(int x, int y)
        {
            long tmp = x;
            for (y; y > 1; y--)
            {  tmp *= x;    }
            return tmp;
        }
```

```
Public static bool isPrime(int N)
        {
            int n = 0;
            do
            {    cin >> n;
            } while (n % 2 == 0);
            int b = 0;
            int q = (n - 1) / 2;
            int k = 1;
            while (q % 2 == 0)
            {
                q = (q) / 2.0;
                k++;
            };
            int i = 0;
            int r = modPow(b, q, n);
            if (r == 1)
            {  return false;  }
            do
            {     i++;
                r = modPow(r, 2, n);
                if (r == n - 1 || r == 1)
                    return true;
            } while (i < k);
     return false;
}
```

Граф викликів методу Ant_Functions.isPrime() представлений на рисунку 4.5.

Рисунок 4.5 − Граф викликів методу Ant.Prime()

Метод пошуку найкращої вершини Ant_Functions.getBestVertex() використовується приватним методом пошуку найкращого маршруту. Метод визначає найкращу вершину для переходу на основі розподілу ймовірностей в графі.

```
public  static  long  getBestVertex(long  i,  double[][]  matrix,
long[] exclude, long shift)
        {
            long best = -1;
            List<long> bestVs = new List<long>();
```

```
            double max = 0;
  for (long j = matrix[i - shift].LongLength - 1; j >= 0; j--)
            {
                if (j != i && !exclude.Contains(j+shift))
                {
   var poss = matrix[i - shift][j] / PossSumm(i, matrix, new long[0], shift);
                    if (poss >= max)
                    {
                        max = poss;
                        best = j+shift;
                        bestVs.Add(best);
                    }
                }
            }
            Console.WriteLine("Possibility from {0} to {1} is Max, P = {2}", i, best, max);
            // искать из максимальных случайное
            Random r = new Random();
            int ind = r.Next(bestVs.Count);
            Console.WriteLine("Random Vertex from {1} max is {0}", ind+shift, bestVs.Count);
            return bestVs[ind];        }
```

Граф викликів методуAnt_Functions.getBestVertex() представлений на рисунку 4.6.

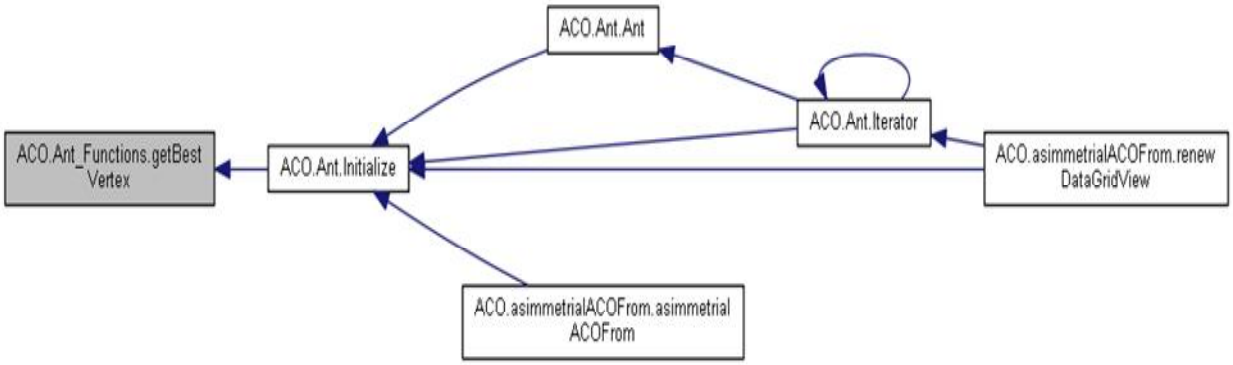

Рисунок 4.6 − Граф викликів методу Ant_Functions.getBestVertex()

4.4 Алгоритм роботи з додатком

При запуску програмного додатку користувачу надається можливість використовувати кілька варіантів дій: запустити алгоритм для роботи з власними даними, які користувач сам може задавати; запустити алгоритм з гото-

вим набором тестів, прочитати документацію щодо вихідного коду програми; та коротко ознайомитися з додатком (рис. 4.7).

Рисунок 4.7 − Стартова форма додатка

При виборі роботи з власним набором даних користувачу надається друга форма додатка, що виводить данні про алгоритм, де він може переглянути матрицю суміжності графа, створеного його набором параметрів, вивести матрицю в текстовому вигляді для завдання графа сторонніми сервісами, переглянути маршрути і матрицю графа для кожної окремої ітерації, задіяної алгоритмом і переглянути історію виконання алгоритму (рис. 4.8 –4.9).

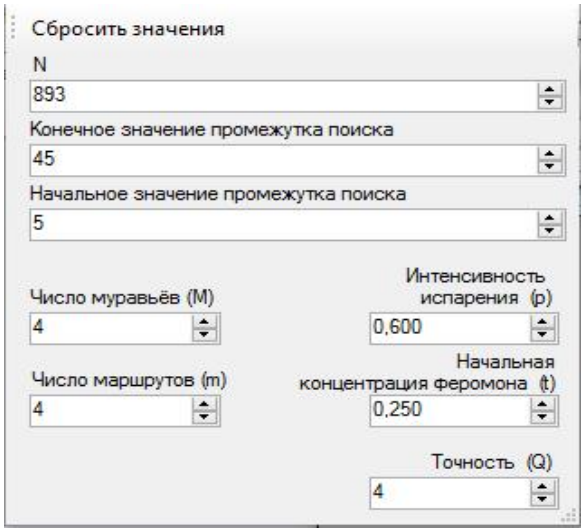

Рисунок 4.8 − Форма введення параметрів алгоритму

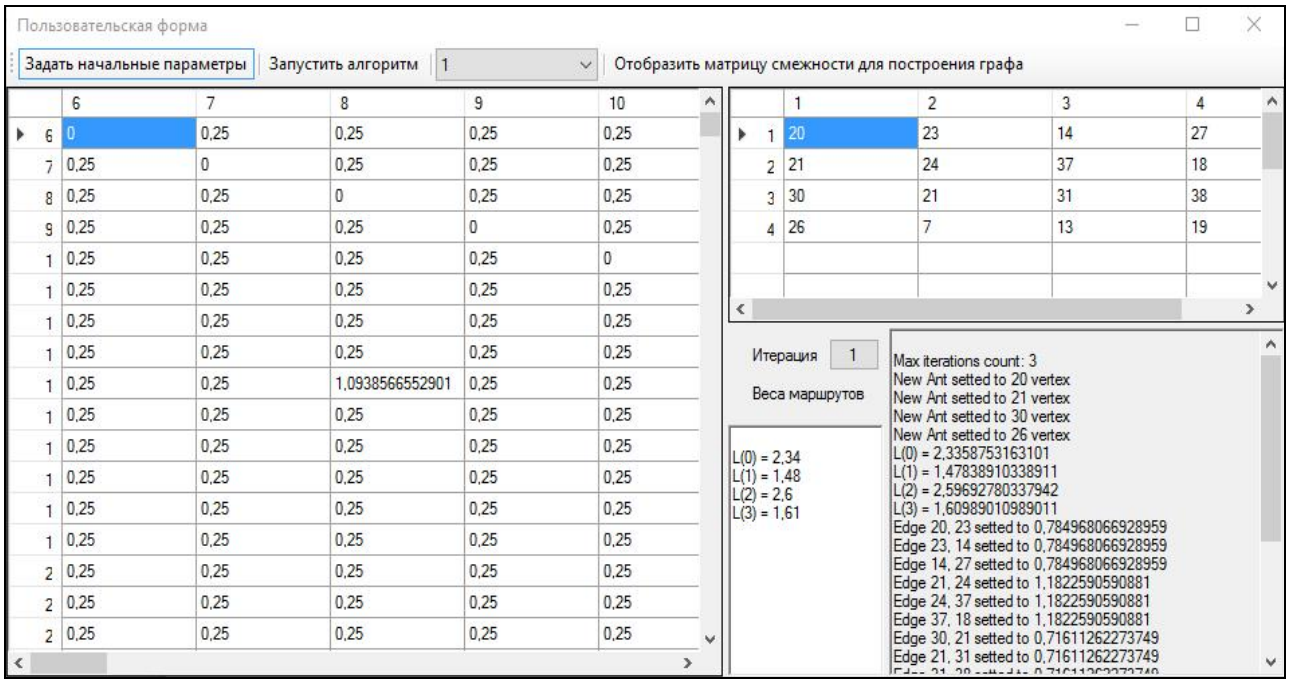

Рисунок 4.9 – Форма з результатом роботи алгоритму

При виборі роботи з готовими тестовими даними, користувачеві буде запропоновано вибрати дані з заготовленого набору тестів і переглянути роботу алгоритму з цими даними. Візуально форма тестів повторює інтерфейс форми алгоритму з одними даними, за винятком можливості задати дані самостійно (рис. 4.10).

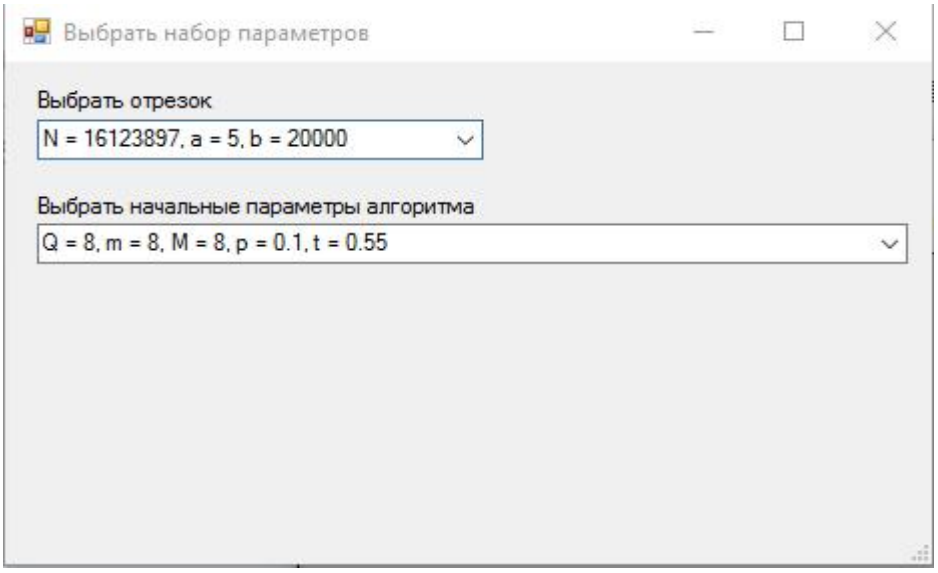

Рисунок 4.10 – Форма вибору набору тестових даних

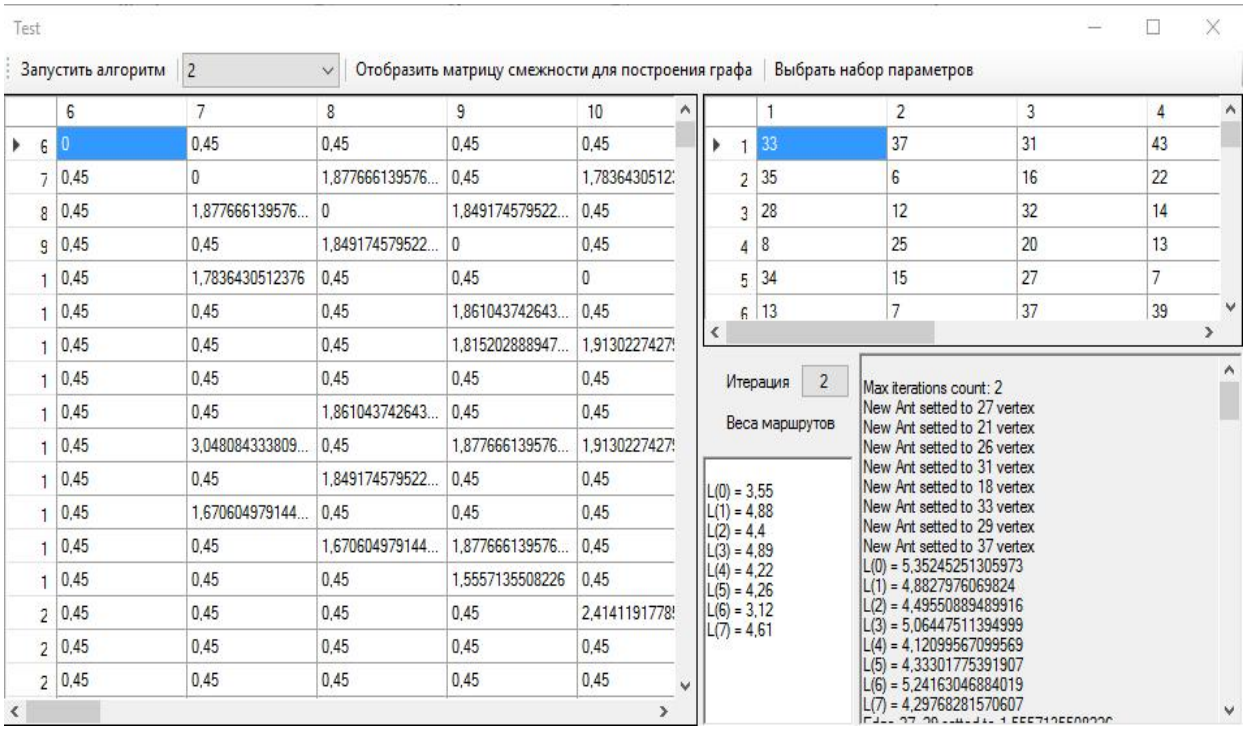

Рисунок 4.11 − Форма роботи алгоритму за обраним набором даних

При виборі пункту «Показати матрицю суміжності для побудови графа» користувач відкриє форму з текстовим полем, що містить текстове представлення матриці, яку можна використовувати для генерації графа сторонніми сервісами (Додаток Б).

4.5 Тестування різних конфігурацій та параметрів алгоритму

В ході тестування роботи алгоритму були використані деякі набори початкових параметрів, які відрізняються точністю, кількістю маршрутів, інтервалом пошуку і початковою кількістю вершин в маршрутах. Результати представлені в таблиці 4.1.

Таблиця 4.1 – Результати тестування алгоритму

N	Множники	Початкові параметри	Кількість ітерацій
893	19, 47	[5,45], Q=6, m=6, M=6	1
		[5,45], Q=4, m=4, M=4	3
15238657	7,7,353,881	[5,2000], Q=4, m=6, M=6	543
		[5,2000], Q=4, m=5, M=5	550
		[5,2000], Q=4, m=4, M=4	568
16123897	23,37,18947	[5,20000], Q=4, m=6, M=6	599
		[5,30000], Q=4, m=4, M=4	611
		[5,20000], Q=4, m=5, M=5	638
4154963851	3943,1053757	[3000,2000000], Q=4, m=12, M=8	2346

Як видно з таблиці 4.1, в залежності від початкової конфігурації алгоритму, спостерігається зміна кількості ітерацій, необхідних для пошуку мінімальних значень. Це залежить не тільки від величини обраного за для факторизації числа, а й від кількості агентів, використаних в алгоритмі.

ВИСНОВКИ

В процесі виконання магістерської роботи був розроблений програмний додаток, що виконує криптоаналіз асиметричних криптосистем, заснованих на методі факторизації складених чисел, за допомогою мурашиного алгоритму.

Тема магістерської роботи є актуальною та полягає в застосуванні біоінспірірованіх алгоритмів для розв'язання завдань криптоаналізу асиметричних алгоритмів шифрування заснованих на методі факторизації чисел та розробки програмного додатку, що реалізує криптоаналіз асиметричних криптосистем за допомогою мурашиного алгоритму.

В роботі було проведено аналіз алгоритмів асиметричних криптосистем та алгоритмів крипто аналізу, здійснено дослідження методів криптоаналізу асиметричних та симетричних криптосистем, а також інтелектуальних методів кріптоаналізу; здійснена розробка та реалізація програмного додатку для розв'язку завдання криптоаналізу асиметричних алгоритмів шифрування на основі факторизації складених чисел, засобами мурашиного алгоритму. Програмний додаток був розроблений на мові C# з використанням технологій .NET в середовищі виконання Microsoft Visual Studio Community.

Проведене тестування різних конфігурацій мурашиного алгоритму, а також тестування готового додатку з різними наборами початкових параметрів. В ході тестування була виявлена залежність кількості ітерацій від кількості маршрутів, заданого в початковій конфігурації алгоритму, а також від таких параметрів, як величина ітераційного випаровування «феромону», точність визначення маршруту і довжина маршруту.

Практична цінність роботи полягає в тому, що розроблений додаток може бути використано для проведення тестування різних конфігурацій мурашиного алгоритму з різними наборами початкових параметрів при здійсненні криптоаналізу асиметричних алгоритмів шифрування, та використаний в якості сервісу для інших сторонніх додатків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Internet Crime Report Released — FBI/ URL: https://www.fbi.gov /news/stories/2019-internet-crime-report-released-021120 (дата звернення: 16.05.2020).
2. С.М. Авдошин, А.А. Савельева. Криптоанализ: современное состояние и перспективы развития – PDF Free Download / URL: http://docplayer.ru/ 25985958-Kriptoanaliz-sovremennoe-sostoyanie-i-perspektivy-razvitiya. html (дата звернення: 05.06.2020).
3. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. М., Гелиос АРВ, 2002. 480 с.
4. Gordon L.A., Loeb M.P., Lucyshyn W., Richardson R. CSI/FBI Computer Crime and Security Survey 2015 – Computer Security Institute Publications, 2015. URL: https: //www.researchgate.net /publication/ 323112056_Empirical_Evidence_on_the_Determinants_of_Cybersecurity _Investments_in_Private_Sector_Firms (дата звернення: 18.07.2020).
5. Чернышев Ю.О., Сергеев А.С., Венцов Н.Н., Рязанов А.Н. Исследование возможности применения генетических алгоритмов для реализации криптоанализа блочных криптосистем. Вестник ДГТУ. 2015;15(3):65-72. URL: https://doi.org/ 10.12737/12599 (дата звернення: 05.07.2020).
6. Мао В. Современная криптография. Теория и практика. М.: Вильямс, 2005. 763с.
7. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2003. 806 с.
8. Бауэр Ф. Расшифрованные секреты. Методы и принципы криптологии. М.: Мир, 2007. 550 с.
9. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. К.:«Корнійчук», 2008. 446 с.

10. Сергеев А.С., Третьяков О.П., Васильев А.Е., Чернышёв Ю.О. Биоинспирированные методы криптоанализа асимметричных алгоритмов шифрования на основе факторизации составных чисел. ВДГТУ. 2011;11(9):1544-1554. URL: https://www.vestnik-donstu.ru/jour /article/ view/888 (дата звернення: 10.07.2020).

11. Остапов С. Е., Валь Л.О. Основи криптографії: навчальний посібник. Чернівці: Книги–XXI, 2008. 188 с.

ДОДАТОК А

Лістинг коду методу Ant.Iterator()

```
public void Iterator()
        {
            if (N == 0) return;
            minimals = new List<long>();
            int limit = Ant_Functions.TimeCiclesCount(a, b, m,
M);
            Console.WriteLine("Iterations count: {0}", limit);
            form.log.Text += "\nMax iterations count: " + limit;
            int t = 0;
            bool stopTrigger = false;
            if (t == 0)
            {
                this.Initialize();
            }
            do
            {
                long minimalVertex = -1;
                long[] vertexes = setAnts();
                long[][] routes = new long[this.M][];
                form.initDataGridView(form.dataGridView2,
this.M, this.m);
                if (t < 2)
                {
                    for (int i = 0; i < vertexes.Length; i++)
                    {
                        routes[i] = setRandomRoute(vertexes[i]);
                    }
                }
                else
                    for (int i = 0; i < vertexes.Length; i++)
                    {
                        routes[i] = setRoute(vertexes[i]);
                    }
                Ant_Functions.ShowTwoDimsArray(routes, form);

                double[] Ls = new double[routes.Length];
                for (int i = 0; i < routes.Length; i++)
                {
                    Ls[i] = Ant_Functions.L(routes[i], N);
                    Console.WriteLine("L{1} = {0}", Ls[i], i);
                    form.log.Text += "\nL(" + i + ") = " +
Ls[i];
                }

                Stack<int> visitedPairs = new Stack<int>();
                for (long i = 0; i < routes.LongLength; i++)
                {
                    for (long j = 1; j < routes[i].LongLength;
j++)
                    {
                        long fr = routes[i][j - 1], sec =
routes[i][j];
                        if
(!visitedPairs.Contains((fr.ToString() +
sec.ToString()).GetHashCode()) ||
```

```
!visitedPairs.Contains((sec.ToString() + fr.ToString()).GetHashCode()))
                        {
                            Console.WriteLine("Search for {0}, {1} counts", fr, sec);
                            //form.log.Text += "\nSearch for "+ fr + ", "+sec+" counts";
                            int count = 0;
                            for (long i1 = 0; i1 < routes.LongLength; i1++)
                            {
                                for (long j1 = 1; j1 < routes[i].LongLength; j1++)
                                {
                                    if (routes[i1][j1 - 1] == fr && routes[i1][j1] == sec || routes[i1][j1 - 1] == sec && routes[i1][j1] == fr)
                                    {
                                        count++;
                                    }
                                }
                            }
                            Console.WriteLine("Count = {0}, fr-a = {1}, sec-a = {2}", count, fr - a, sec - a);
                            double renew = Ant_Functions.FeromonRenew(count, Q, Ls[i], p, matrix[fr - a][sec - a]);
                            matrix[fr - a][sec - a] = renew;
                            matrix[sec - a][fr - a] = renew;
                            try
                            {
                                form.renewDataGridView(form.dataGridView1, (int)(fr - a), (int)(sec - a), renew.ToString());
                                form.renewDataGridView(form.dataGridView1, (int)(sec - a), (int)(fr - a), renew.ToString());
                            }
                            catch { }
                            Console.WriteLine("Edge {0} {1} setted to {2}", fr, sec, renew);
                            form.log.Text += "\nEdge " + fr + ", " + sec + " setted to " + renew;
                            if (Ant_Functions.F(N, fr) < e || Ant_Functions.F(N, sec) < e)
                            {
                                if (Ant_Functions.F(N, fr) < e && !minimals.Contains(fr))
                                {
                                    this.minimals.Add(fr);
                                }
                                if (Ant_Functions.F(N, sec) < e && !minimals.Contains(sec))
                                {
                                    this.minimals.Add(sec);
                                }
                            }
                            if (Ant_Functions.F(N, fr) == 0 || Ant_Functions.F(N, sec) == 0)
                            {
```

```
                                    if (Ant_Functions.F(N, fr) == 0)
                                    {
                                        minimalVertex = fr;
                                    }
                                    else
                                    {
                                        minimalVertex = sec;
                                    }
                                    stopTrigger = true;
                                }
                                Console.WriteLine("F({0})   =   {1},
F({2})    =    {3}",    fr,    Ant_Functions.F(N,    fr),    sec,
Ant_Functions.F(N, sec));
                                visitedPairs.Push((fr.ToString()    +
sec.ToString()).GetHashCode());
                                visitedPairs.Push((sec.ToString()   +
fr.ToString()).GetHashCode());
                            }

                        }
                    }
                    //ShowMatrix();
                    //Console.WriteLine("");
                    //this.ShowMatrix();

                    iteration   iteratee   =   new   iteration(matrix,
routes, minimals, Ls);
                    form.iterations.Add(iteratee);
                    //end of iteration
                    t++;
                    Console.WriteLine(t + " Iteration Done");
                    form.log.Text += "\nIteration "+t+" Done";
                    if (minimalVertex != -1)
                    {
                        if (Ant_Functions.isPrime(minimalVertex))
                        {
                            if ((N / minimalVertex) == (N * 1.0 /
minimalVertex)   &&   !Ant_Functions.isPrime((long)(N   *   1.0   /
minimalVertex))   &&   !minimals.Contains((long)(N   *   1.0   /
minimalVertex)))
                            {
                                Ant ant = new Ant(this.Q, this.M, p,
t_max, m, (N / minimalVertex), a, b, form);
                                Console.WriteLine("New    Iterations
with N = {0}", (N / minimalVertex));
                                MessageBox.Show("New Iterations with
N = " + (N / minimalVertex));
                                ant.Iterator();
                            }
                            else
                            {
                                Console.WriteLine("Minimal  vertex  =
{0}", (N / minimalVertex));
                                form.log.Text += "\nMinimal = " + (N
/ minimalVertex);
                            }
                        }
                        else
                        {
```

```
                        Ant ant = new Ant(this.Q, this.M, p,
t_max, m, minimalVertex, a, b, form);
                        Console.WriteLine("New Iterations with N
= {0}", minimalVertex);
                        MessageBox.Show("New Iterations with N =
" + minimalVertex);
                        ant.Iterator();
                    }
                    Console.WriteLine("Minimal vertex = {0}",
minimalVertex);
                    form.log.Text += "\nMinimal = " +
minimalVertex;
                }
            } while ((t < limit) && !stopTrigger);
//Ant_Functions.TimeCiclesCount(a, b, m, M)
            Console.WriteLine("Minimal vertexes: ");
            form.log.Text += "\nMinimal vertexes: ";
            for (int i = 0; i < minimals.Count; i++)
            {
                Console.WriteLine("{0}:     {1},     {2}",     i,
minimals[i], Ant_Functions.F(N, minimals[i]));
                form.log.Text += "\n" + i + ": " + minimals[i] +
" = " + Ant_Functions.F(N, minimals[i]);
            }
        }
```

Graph 1

0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.00	1.88	0.45	1.78	0.45	0.45	0.45	0.45	3.05	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	1.56	0.45	0.45	0.45	0.45	1.91	0.45	2.41	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	1.88	0.00	1.85	0.45	0.45	0.45	0.45	1.86	0.45	1.85	0.45	1.67	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	1.88	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	1.86	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45
0.45	0.45	1.85	0.00	0.45	1.86	1.82	0.45	0.45	1.88	0.45	0.45	1.88	1.56	0.45	0.45	0.45	0.45	0.45	0.45	1.67	1.78	1.85	0.45	1.56	0.45	0.45	0.45	0.45	1.82	0.45	0.45	0.45	0.45	1.86	0.45	0.45	0.45	0.45	1.78	0.45	0.45
0.45	1.78	0.45	0.45	0.00	0.45	1.91	0.45	0.45	1.91	0.45	0.45	0.45	0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.78	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	1.86	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.41	0.45	0.45	1.84	0.45	0.45	0.45	2.16	0.45	0.45	0.45	0.45	0.45	1.84	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.86	0.45	2.16	0.45	0.45	0.45
0.45	0.45	0.45	1.82	1.91	0.45	0.00	0.45	1.75	0.45	1.75	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.91	0.45	0.45	0.45	0.45	0.45	0.45	1.82	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.82	0.45	0.45	0.45	1.82	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	1.86	0.45	0.45	0.45	1.75	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	1.56	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	1.56	0.45	0.45	0.45	0.45	1.67	0.45	3.05	0.45
1.88	1.91	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.85	0.45	1.84	0.45	0.45	1.82	0.45	0.45	1.84	0.45	0.45	0.45	0.45	1.82	0.45	0.45	1.85	0.45	0.45	0.45
0.45	1.85	0.45	0.45	0.45	1.75	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.88	1.75	0.45	0.45	0.45	0.45	1.91	1.88	1.67	1.85	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	1.67	1.88	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	1.84	0.45	1.84	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	1.56	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	1.58	0.45	0.45	1.91	2.16	1.62	0.45	0.45	1.62	0.45	0.45	1.58	0.45	0.45	1.56	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	2.41	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	2.41	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.84	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.84	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	1.56	1.67	0.45	0.45	1.84	0.45	0.45	1.56	0.45	0.45	0.45	1.84	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.41	2.41	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.58	0.45	1.67	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.65	0.45	1.75	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	1.78	0.45	0.45	0.45	0.45	1.67	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	1.88	1.85	0.45	2.16	0.45	0.45	0.45	1.85	1.88	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	3.05	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	1.91	0.45	0.45	1.78	0.45	0.45	1.82	0.45	0.45	1.75	0.45	0.45	1.91	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.62	1.75	1.78	1.62	0.45	0.45
0.45	0.45	0.45	1.56	0.45	0.45	0.45	0.45	0.45	1.84	0.45	0.45	0.45	2.16	0.45	0.45	1.84	0.45	0.45	0.45	0.45	0.45	3.05	0.45	0.00	0.45	0.45	0.45	0.45	1.58	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	1.62	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	1.58	0.45	0.45	2.41	0.45	0.45	0.45	1.58	1.62	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	1.62	0.45	0.45	0.45	2.16	0.45	0.45	0.45	0.45	0.45	2.16
0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.82	0.45	1.82	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	1.67
0.45	0.45	0.45	0.45	0.45	1.84	1.91	0.45	0.45	0.45	1.91	0.45	0.45	1.62	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.62	0.45	0.00	0.45	0.45	0.45	2.16	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	1.67	1.82	0.45	0.45	0.45	0.45	0.45	0.45	1.88	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.58	1.58	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.84	1.67	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	1.85	0.45	0.45	1.58	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	1.85	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	1.86	0.45	2.41	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.41	2.16	0.45	2.16	0.45	0.45	1.85	0.00	0.45	1.86	0.45	0.45	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.65	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	3.21	0.45	1.75	1.78
0.45	0.45	0.45	1.86	0.45	0.45	0.45	0.45	1.56	0.45	0.45	0.45	0.45	1.56	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.62	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.86	0.45	0.00	0.45	0.45	0.45	1.62	0.45
0.45	0.45	0.45	0.45	0.45	0.45	1.82	0.45	0.45	1.82	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.75	0.45	0.45	1.75	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.00	0.45	0.45	0.45	0.45
0.45	0.45	0.45	0.45	0.45	1.86	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.78	0.45	1.58	0.45	0.45	0.45	0.45	0.45	0.45	0.45	3.21	0.45	0.45	0.00	1.86	0.45	0.45
0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.62	0.45	1.62	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.86	0.00	0.45	0.45
0.45	0.45	0.45	0.45	0.45	2.16	0.45	0.45	0.45	1.85	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	2.16	0.45	0.45	0.45	0.45	0.45	0.45	1.75	1.62	0.45	0.45	0.45	0.00	0.45
0.45	0.45	0.45	1.78	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	1.67	0.45	0.45	0.45	0.45	0.45	1.78	0.45	0.45	0.45	0.45	0.45	0.00

ДОДАТОК Б

Матриця суміжності в текстовому поданні