Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Дослідження та порівняння атак на криптосистему RSA

Виконав: студент IV курсу, групи СБс-41
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

(підпис) Ремінник М.М. (прізвище та ініціали)

Керівник (підпис) Загородна Н.В. (прізвище та ініціали)

Нормоконтроль (підпис) Лобур Т.Б. (прізвище та ініціали)

Завідувач кафедри (підпис) Загородна Н.В. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль - 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Ремінnику Миколі Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та порівняння атак на криптосистему RSA

Керівник роботи Загородна Наталія Володимирівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 04 » 04 2023 року № 4/7-349

2. Термін подання студентом завершеної роботи 20.06.2023 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Розробка програмного засобу.
3. Результати проведених досліджень.
4. Безпека життєдіяльності, основи охорони праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета, задачі дослідження. 4. RSA.
5. Частотний аналіз. 6. Квадратичне решето, атаки Хастада та Вінера.
7 Розроблений програмний додаток. 8. Скрін-шоти фрагментів програмного коду.
9 – 12. Результати проведених програмних експериментів.
13. Основні результати дослідження

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	04.04 – 06.04	*Виконано*
2.	Підбір джерел про атаки на криптосистему RSA	07.04 – 11.04	*Виконано*
3.	Опрацювання джерел про атаки на криптосистему RSA	12.04 – 16.04	*Виконано*
4.	Проведення дослідження та порівняння атак на		
	криптосистему RSA	17.04 – 23.04	*Виконано*
5	Розроблення програмного коду	24.04 – 29.04	
6.	Оформлення розділу «Аналіз предметної області»	30.04 – 07.05	*Виконано*
7.	Оформлення розділу «Розробка програмного засобу»	08.05 – 15.05	*Виконано*
8.	Оформлення розділу «Результати проведених		
	досліджень»	16.05 – 21.05	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи охорони праці»	14.05 – 21.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	22.05 – 05.06	*Виконано*
11.	Нормоконтроль	06.06 – 12.06	*Виконано*
12.	Перевірка на плагіат	10.06 – 14.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	15.06 – 17.06	*Виконано*
14.	Захист кваліфікаційної роботи	21.06	

Студент _____ Ремінник М.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження та порівняння атак на криптосистему RSA // Ремінник Микола Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра кібербезпеки, група СБс–41 // Тернопіль, 2023 // С. – 52, рис. – 26, табл. – 6 , слайдів – 13, бібліогр. – 23.

Ключові слова: ФАКТОРИЗАЦІЯ, ЧАСТОТНИЙ АНАЛІЗ, КВАДРАТИЧНЕ РЕШЕТО, АТАКА, ВРАЗЛИВІСТЬ, RSA

Кваліфікаційна робота присвячена перевірці вимог до криптосистеми RSA, для свідчення доцільності використання шифрування блоками застосовано частотний аналіз, а для вимог до генерації напівпростих чисел - квадратичне решето.

Проаналізовано алгоритм RSA, сфери його використання. Докладно розглянуто особливості частотного криптоаналізу, факторизації (квадратичного решета). Досліджено атаки Хастада та Вінера. Для програмної реалізації дослідження створено додаток на основі мови C# із використанням зовнішніх бібліотек. Наведено опис основних його класів та файлів, які формуються під час його роботи. Для реалізації атак обрана мова Python.

Були отримані експериментальні результати і для згенерованого тексту, і для тексту, створеного реальною особою.

Досліджено, що при порушенні сучасних вимог та постійному адаптуванні під них, з'являється ризик зменшення надійності способу шифрування.

ANNOTATION

Research and comparison of attacks on the RSA cryptosystem // Reminnyk Mykola // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security // Ternopil, 2023 // P. - 52, Fig. - 26, Table - 6, Slides - 13, References - 23.

Keywords: FACTORIZATION, FREQUENCY ANALYSIS, QUADRATIC GRID, ATTACK, VULNERABILITY, RSA

Thesis deals with checking the requirements for the RSA cryptosystem, frequency analysis is used to demonstrate the expediency of using block encryption, and the quadratic sieve is used for the requirements for the generation of semiprime numbers.

The RSA algorithm and the areas of its use are analyzed. Features of frequency cryptanalysis, factorization (quadratic lattice) are considered in detail. Hustad and Wiener attacks are investigated. For the software implementation of the research, an application based on the C# language was created using external libraries. A description of its main classes and files, which are formed during its operation, is given. The Python language is chosen for the implementation of attacks.

Experimental results were obtained for both generated text and text created by a real person.

It has been studied that in violation of modern requirements and constant adaptation to them, there is a risk of reducing the reliability of the encryption method.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

RSA (Rivest, Shamir та Adleman) – криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

WPF – Windows Presentation Foundation.

ВПЧ – взаємно просте число.

НСД – найбільший спільний дільник.

СЛАР – система лінійних алгебраїчних рівнянь.

ПЗ – програмне забезпечення.

ЗМІСТ

ВСТУП

Методи криптографії для захисту даних є важливими, а розвиток математичного підґрунтя і технологій в загальному приносить як нові способи для зашифровування даних, так і нові способи зламування цих шифрів. Тому при розробці будь-якої системи шифрування слід детально вивчати будь-яку можливість злому.

У 1977 році завдяки трьом ученим: Рональду Рівесту, Аді Шаміру та Леонарду Адлеману з'явився метод RSA [1] — найвідоміший представник криптографічних алгоритмів, котрий є основоположником асиметричного шифрування та застосовується у різних криптографічних додатках. Для будь-якого алгоритму шифрування є закономірним питання його надійності, доцільності використання та формування вимог до нього.

Очевидно, що з плином часу вимоги до криптографічних алгоритмів не стають м'якшими, оскільки обчислювальні потужності зростають, а це означає, що з'являється можливість вирішувати ті завдання, на які раніше не вистачало ресурсів. На одній із таких задач і побудовано надійність алгоритму RSA, а саме, на факторизації чисел [2]. Тобто на розкладанні складеного напівпростого числа на його множники. В основі алгоритму лежить генерація простих чисел p і q і обчислення їх добутку n. Число n буде називатися напівпростим числом, так як воно може бути репрезентовано як добуток двох простих чисел.

В теперішній час є низка вимог для алгоритму:

– прості числа p і q повинні складатися більш, ніж з 512 біт;

– p та q не повинні бути близькими;

– неприпустиме використання посимвольного шифрування, шифруються блоки.

Серед багатьох алгоритмів факторизації натуральних чисел, таких як перебір дільників, метод Ферма, алгоритм Полларда, алгоритм Діксона і т.д. особливе місце займає метод квадратичного решета, так як він довгий час вважався найкращим субекспоненціальним методом факторизації [2].

Метою роботи є перевірка актуальних вимог до алгоритму RSA, де для

підтвердження доцільності використання блокового шифрування буде застосовано частотний криптоаналіз, а для вимог до генерації напівпростих чисел буде застосовано алгоритм квадратичного решета.

В процесі виконання роботи потрібно вирішити наступні завдання:

– вивчення матеріалів, що належать до сфери застосування RSA;

– реалізація алгоритму RSA з можливістю налаштування відповідних параметрів для тестування атак;

– дослідження матеріалів пов'язаних із частотним криптоаналізом;

– реалізація та застосування частотного аналізу;

– вивчення матеріалів, пов'язаних із методами факторизації;

– реалізація та застосування квадратичного решета;

– ознайомлення з матеріалами, які пов'язані з атаками Хастада та Вінера;

– реалізація та застосування цих атак.

Актуальність роботи зумовлена тим, що існують такі системи, де RSA активно застосовується, а відповідно, при недотриманні сучасних вимог та своєчасному адаптуванні під них, виникає ризик зниження надійності даного способу шифрування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні визначення та поняття

Всі наведена нижче інформація відповідно до [3, 4].

Прості числа — є такими натуральними числами, у котрих відсутні інші дільники, окрім одиниці і самого себе. Для прикладу, таким числом є 17, так як воно має лише два дільники 1 та 17.

Напівпрості числа - утворені як добуток власне простих чисел. Для прикладу, результатом добутку 7 і 17 буде 119, яке є числом напівпростим.

ВПЧ — в котрих відсутні спільні дільники, окрім одиниці. Наприклад, 8 та 15 будуть ВПЧ, оскільки 8 має наступні дільники {1, 2, 4, 8}, а 15 – {1, 5, 15}, тобто у перетин цих множин дільників потрапляє тільки число 1.

Функція Ейлера – є рівною числу ВПЧ менших n з самим n. Для напівпростих чисел обчислюється за допомогою формули $\varphi(n) = (p - 1)(q - 1)$, де p і q прості числа.

Розширений алгоритм Евкліда – окрім власне НСД відшукує також величини x і y такі, що $a * x + b * y = НСД(a, b)$.

Тест простоти - деякий алгоритм, що виконує функцію визначення того, чи є вхідне число простим чи ні. У RSA використовується імовірнісний тест Міллера-Рабіна.

Свідок простоти — таке число a, для якого можна виконати такі умови: $a^d \equiv 1 \pmod n$; $a^{2^r * d} \equiv -1 \pmod n$.

Факторизація — декомпозиція об'єкта до інших об'єктів. В роботі розглядатиметься декомпозиція напівпростих чисел у добутку простих. Так, напівпросте число 77 розкладається як добуток, властиво, простих чисел 7 та 11.

Квадратичний лишок — число a є таким за деяким модулем m, якщо можливе виконання такої умови $x^2 \equiv a \pmod m$. Наприклад, $4^2 \equiv 2 \pmod 7$, 2 є квадратичним лишком за модулем 7, оскільки існує x, що $x^2 \equiv 2 \pmod 7$.

Символ Лежандра є функцією, котра застосовується для визначення

квадратичного лишку, або квадратного нелишку. Приймає значення 0,1 чи -1, якщо число a ділиться на p, якщо a є квадратичним лишком по модулю p і a не ділиться на p, якщо a є квадратичним нелишком по модулю p відповідно.

Факторна база — такий набір простих чисел, для яких число, що факторизується, є квадратичним лишком.

Моноалфавітний шифр – такий метод шифрування, при якому для будь-якого знаку вихідного тексту є один і лише один знак у зашифрованому тексті.

Поліалфавітний шифр – сукупність застосування моноалфавітного шифру до повідомлення. З тією відмінністю, що для кожного *i*-го знаку вихідного тексту використовується алгоритм, таким способом, ховається частота появи знаків вихідного тексту.

1.2 RSA

Ще в 1976 році в роботі [1] Вітфілда Діффі та Мартіна Геллмана було продемонстровано ідею криптографії з відкритими ключами. Її суть у формуванні деякої функції f, за допомогою якої, знаючи значення аргументу x, можна ефективно обчислити $y = f(x)$. Але не повинно бути можливості за аналогічний проміжок часу вирішити обернену задачу. Такі функції одержали назву односторонніх, тобто це ті функції, для яких наявний поліноміальний алгоритм обрахунку, але відсутній алгоритм для зворотного завдання. При цьому абсолютно неважливо, рішення оберненої задачі потрібне для санкціонованого чи несанкціонованого рішення.

Функція множення двох простих чисел – одностороння функція, тому що відсутній поліноміальний алгоритм для зворотного обчислення.

Система RSA базується на таких принципах:

– обчислення $a = b^d \ (mod\ n)$ за умови, що відомі $b\ i\ d$ є простим завданням;

– обчислення b за умови, що відомі $d\ i\ a$ є складним завданням для $a = b^d \ (mod\ n)$;

– якщо відомо, що p і q прості числа, то обчислити $n = p * q$ легко, а

знайти декомпозицію n на прості множники складно.

Алгоритм:

1. Генерується два простих числа p та q.
2. Обчислюється напівпросте число $n = p * q$.
3. Обчислюється функція Ейлера $\varphi(n) = (p - 1) \cdot (q - 1)$.
4. Генерується число e, котрому справедливо $1 < e < \varphi(n)$ і e ВПЧ із числом $\varphi(n)$.
5. Обчислюється d, для якого істинно $d * e = 1 \pmod{\varphi(n)}$ за допомогою розширеного алгоритму Евкліда.
6. Повідомлення m шифрується як $c = m^e \pmod{n}$.
7. Повідомлення c дешифрується як $m = c^d \pmod{n}$.

Приклад. Припустимо, нам потрібно зашифрувати число 19 і перевірити, дешифрувавши його назад.

— p = 13, q = 23;

— n = 13 * 23 = 299;

— φ (299) = (13 - 1) (23 - 1) = 264;

— $e = 7$;

— d = 151;

— $m = 19, c = 19^7 \pmod{299} = 176;$

— $m = 176^{151} \pmod{299} = 19$.

Таким чином, ми маємо відкритий ключ (n, e) = (299, 7) та закритий ключ (n, d) = (299, 151) придатні для шифрування дешифрування повідомлення

1.2.1 Тест Міллера-Рабіна

Оскільки генерація простих чисел пов'язана з самим визначенням, чи є число простим, чи ні, вже на першому етапі алгоритму RSA з'являється необхідність у такому алгоритмі, який досить швидко і точно дасть таку відповідь [5]. Всі тести простоти поділяються на дві великі групи: детерміновані та недетерміновані (імовірнісні).

Тест Міллера-Рабіна відноситься до недетермінованих тестів простоти, але, незважаючи на це, його активно використовують у подібних алгоритмах

шифрування. Пов'язано це з його високою точністю та низьким часом роботи. Крім вхідного параметра n (найдосліджуванішого числа), також є параметр k - кількість раундів. k характеризує точність роботи і, зазвичай, обчислюється, якщо задача, в якій використовується алгоритм, не має на увазі вибору конкретної точності, як $k = log_2 n$. Точність: ймовірність помилки 4^{-k}.

Імовірність помилки (таблиця 1.1) будується на основі теореми Рабіна, де стверджується, що для $\forall n\ ,\ d \leq \frac{\varphi(n)}{4}$, де n - непарна складова числа, а d - кількість свідків простоти числа n.

Таблиця 1.1 - Ймовірність помилки при різній кількості ітерацій для тесту Міллера-Рабіна

k	1	2	3	4	5
Імовірність помилки	0,25	0,0625	0,015625	0,00390625	0,0009765625

Обчислювальна складність: $O(log^3 n)$.

Алгоритм:

1. Розкласти $n - 1$ на $2^s * t$, де t непарне число.
2. Цикл №1: виконується до k разів .
   2.1. Вибрати випадкове a на відрізку від 2 до $n - 2$.
   2.2. $x = a^t \pmod n$.
   2.3. Якщо $x = 1$ або $x = n - 1$, то перейти на наступну ітерацію.
   2.4. Цикл №2: виконується до s – 1 разів.
      2.4.1. $x = x^2 \pmod n$.
      2.4.2. Якщо $x = 1$, то n – складене число.
      2.4.3. Якщо $x = n – 1$, перейти на наступну ітерацію циклу №1.
   2.5. n - складене число.
3. n - ймовірно просте число.

Варто також відзначити, що досить важливо вибирати a випадковим, а не просто перебирати всі числа в порядку зростання, оскільки існують такі складові

числа, для яких свідки простоти, що йдуть підряд, дають погані результати.

Одне з таких чисел записується так: $n = p$ [313 (p - 1) + 1] * [353 (p - 1) + 1], де p = 29 674 495 668 685 510 550 154 174 642 905 332 730 771 991 37 99 753 171 770 199 594 238 596 4248 121 188 033 664 754 218 345 562 493 168 782 883. Дане число n є складеним , але перші 307 свідків не спростовують ймовірність того, що число є простим.

1.2.2 Вразливості

Для будь-якої системи безпеки найслабшою ланкою завжди був людський чинник. У цьому разі йдеться про помилки реалізації. Найбільш груба з них, яка зводить нанівець всі переваги RSA – це посимвольне шифрування повідомлення, оскільки перетворює його на шифр простої заміни, інакше кажучи, на моноалфавітний шифр [6-8].

Така помилка дозволяє застосувати прості методи злому шифру:

— мінімальна розмірність секретної експоненти. Атака Вінера заснована на малій розмірності d. Завдяки чому рекомендується брати таке число d, яке буде менше, ніж n не більше, ніж у чотири рази;

— невеликі значення відкритої експоненти. Ця вразливість зустрічається досить часто, тому що вибирають малу відкриту експоненту усвідомлено для прискорення процесу шифрування.

Інший набір вразливостей пов'язаний безпосередньо з числом n, а точніше його складових: простих чисел p і q:

— мінімальна довжина p або q. Неважливо, наскільки більшим буде одне з простих чисел і число n, якщо інше просте число можна отримати перебором;

— мала різниця між p і q. Обмеження пов'язане з тим, що існує такий метод як метод факторизації Ферма. Він дозволяє досить швидко розкласти число n на p і q при виконанні цієї умови.

Постає закономірне питання, яка розмірність вважається малою в контексті генерації p і q. Вважатимемо, що якщо один з алгоритмів факторизації здатний розкласти число n за прийнятний час при доступних на сьогоднішній

день обчислювальних потужностях, то таке число n вважається непридатним для використання, а числа p і q малими для завдання шифрування.

Існує множина, звана RSA -числами, що складається з 54 напівпростих чисел, що використовуються у конкурсі RSA Factoring Challenge.

Найбільше факторизоване число на сьогоднішній день має 768 біт, у десятковому поданні воно складається з 232 знаків, що було розкладено у грудні 2009 року. Але це не означає, що всі числа факторизують по порядку, таке останнє факторизоване число було розкладено в серпні 2018 року, а складалося воно з 230 десяткових знаків, що всього на два знаки менше за рекордсмена 2009 року. Зазвичай, для шифрування використовують числа, котрі мають 1024 біт, що досі актуально, так як числа такої розмірності ще не були факторизованими. Стандарт шифрування зміниться на користь 2048 біт, як тільки 1024 біт стануть непридатними для цих цілей.

1.3 Частотний аналіз

Частотний криптоаналіз – базується на частоті появи симовлів зашифрованого тексту [6-8].

Наприклад, в українській мові найчастіше трапляється буква «о», що дозволяє припустити для зашифрованого повідомлення, у якому є певний символ, який зустрічається найчастіше інших, що він і є буквою «о» [9].

Цей підхід до зламування різних алгоритмів шифрування має деякі особливості для вхідних даних:

- використане шифрування має бути посимвольним;
- кожному вихідному символу *A* відповідає лише один символ *B*;
- вихідне повідомлення має смислове навантаження.

Даний метод заснований на аналізі мови, а точніше, що випливає з назви, на складанні статистики літер, як для самої мови, так і досліджуваного повідомлення, яка представлена в табл. 1.2 – 1.5.

Таблиця 1.2 - Частоти букв у українській мові №1

Літера	Пробіл	о	а	н	и	і	т	в	р
Відносна частота, %	17,0	9,4	7,2	6,5	6,1	5,7	5,5	5,2	4,7

Таблиця 1.3 - Частоти букв у українській мові №2

Літера	с	у	л	д	к	м	п	я	ь
Відносна частота, %	4,1	4,0	3,6	6,5	3,5	3,1	2,9	2,9	2,9

Таблиця 1.4 - Частоти букв у українській мові №3

Літера	з	ч	б	е	г	х	ш
Відносна частота, %	2,3	1,8	1,7	1,7	1,6	1,2	1,2

Таблиця 1.5 - Частоти букв у українській мові №4

Літера	ж	є	й	ї	ц	ю	ф	щ	ґ
Відносна частота, %	0,9	0,8	0,8	0,6	0,6	0,4	0,1	0,1	близько 0

Як можна помітити, розподіл букв не є рівномірним. Якби розподіл українських букв у тексті був рівномірним, то частота для будь-якої букви дорівнювала б 33/100=0,33 і це без урахування пробілу. Такий розподіл пов'язаний з тим, що кожна мова має свою специфіку словотворення, для української мови до цього поняття входять префікси, суфікси, закінчення і т.д.

Також завдяки такій специфіці будуються m-грами, котрі відображають, як часто m символів зустрічаються разом. Очевидно, якщо виключити випадковий набір букв у тексті і залишити лише ту частину, яка має смислове навантаження, то неможливо зустріти «ь» і «ь», котрі стоять поруч. А поєднання літер "с" і "т", "н" і "о", "е" і "н" зустрічаються досить часто.

Важливим моментом є поєднання голосних та приголосних букв у узагальненому вигляді. Тобто статистика, в якій враховується не поєднання

конкретного символу з іншим, а поєднання типу символу з іншими типом. Під типом мається на увазі приналежність до однієї з двох множин: голосних або приголосних букв. Варто відзначити, що на досить довгих текстах межі таких умовних ймовірностей розмиваються, хоча на перший погляд може здатися, що пари голосна-приголосна і приголосна-голосна повинні зустрічатися частіше, ніж голосна-голосна та приголосна-приголосна [9].

Розберемо раніше обумовлені вимоги.

Вимога до посимвольного шифрування пов'язана з тим, що немає можливості складати статистику для блокового шифрування. Припустимо, є певний блок символів «Тест» і блок «стіл». Завдяки деякому шифруванню ми отримали блоки «Лама» і «стілець». Незважаючи на те, що в кінцевому повідомленні ми можемо обчислити частоту будь-якого блоку, нам нема з чим його зіставляти.

Вимога до єдиного можливого зіставлення символу обумовлено тим, що ми не маємо можливості припускати в конкретному випадку, про який символ йшлося, незважаючи на те, що ми матимемо частоти всіх символів у зашифрованому повідомленні. Наприклад, під час деякого шифрування повідомлення «Тест» було перетворено на повідомлення «йгне». Далі припустимо, що з статистичного аналізу ми з'ясували, що символи зашифрованого повідомлення мають аналогічну частоту, як і символи «н, е, з, г». Незважаючи на те, що ми визначили символи, які зустрічалися один раз, символ «т» так і залишився дешифрованим неправильно.

За дотримання всіх вимог, але не маючи в повідомленні смислового навантаження теж виникають проблеми, оскільки можна скласти статистику символів для зашифрованого повідомлення і навіть можна співвіднести її із заданою статистикою алфавіту, але ця статистика буде абсолютно невідповідною. Навіть якщо нам пощастить і статистика алфавіту підходитиме, ми не зможемо дізнатися, чи справді був зашифрований саме цей набір символів.

1.4 Квадратичне решето

Є другим за швидкістю методом факторизації чисел. В основі методу лежить метод Діксона, який у свою чергу є узагальненням методу Ферма. Розберемо всі алгоритми з їх узагальнення [6, 7].

Метод Ферма має одну мету: знайти такі x та y для заданого n, для яких буде виконуватися умова $x^2 - y^2 = n$. За допомогою отриманих x та y можна буде виконати розкладання $n = (x + y) * (x - y)$.

Алгоритм:

1. Обчислюємо цілу частину квадратного кореня n: $s = [\sqrt{n}]$.
2. Цикл: починається з k =1.
   - 2.1. Обчислюємо: $y = (s+k)^2 - n$.
   - 2.2. Обчислюємо: $[\sqrt{n}]$.
   - 2.3. Якщо $[\sqrt{n}]$. не дає цілочисельного рішення, то збільшуємо на 1 і переходимо до початку циклу.
3. Для числа $n = a * b$, $a = s + k - [\sqrt{n}]$, $b = s + k + [\sqrt{n}]$
4. Приклад n = 3103.
5. s = $[\sqrt{n}]$= 56 .

Для зручності, наступні обчислення циклі, запишемо в табл. 1.6.

Таблиця 1.6 - Обчислення методу Ферма

k	$x = s + k$	y	$[\sqrt{n}]$
0	56	33	5,74
1	57	146	12,08
2	58	261	16,15
3	59	378	19,44
4	60	497	22,29
5	61	618	24,85

Продовження таблиці 1.6

k	$x = s + k$	y	$[\sqrt{n}]$
6	62	741	27,22
7	63	866	29,42
8	64	993	31,51
9	65	1122	33,49
10	66	1253	35,39
11	67	1386	37,22
12	68	1521	39

Таким чином, $a = s + k - \sqrt{y} = 29$, $b = s + k + \sqrt{y} = 107$.

Метод Діксона будується на методі Ферма, з тією відмінністю, що тепер, по-перше, шукаються пари чисел типу $x^2 \equiv y^2 \pmod{n}$ і $x \neq \pm y \pmod{n}$, по-друге, вводиться таке поняття, як факторна база та гладкі числа.

Алгоритм:

1. Обчислити $L(n) = exp(\sqrt{ln\, n * ln\, ln\, n})$.
2. Скласти факторну базу B, яка складається з таких простих чисел, які $\leq L(n)^{1/2}$.
3. Задати порожню множину FB.
4. Цикл: до тих пір, поки кількість елементів множини FB не буде рівною $h + 1$ де h - кількість елементів множини B.

4.1. Вибрати число b, яке належить відрізку $[\sqrt{n}; n]$.

4.2. Обчислити $a = b^2 \pmod{n}$.

4.3. Якщо a є гладким числом, то додаємо його в множину FB, заповнюючи тим самим вектор.

5. Методом Гауса знайти лінійну залежність серед векторів.
6. Якщо $x \equiv \pm \pmod{n}$, повторити процедуру генерації FB.
7. Множники a та b рівні відповідно $НСД(x + y, n)$ $НСД(x - y, n)$.

Завдяки методу Померанца і утворюється квадратичне решето. Базується

воно на іншій побудові факторної бази, а точніше на просіюванні.

Алгоритм:

— побудуємо факторну базу, де перший елемент дорівнюватиме $p = 2$, далі кожне число є простим і для якого символ Лежандра дорівнює 1, тобто квадратичні лишки;

— побудувати t пар чисел a та b, для яких $a = x + m$ для $x = 0, \pm 1. \pm 2 \ldots, b = (x + m)^2 - n$ (перевіряючи на гладкість), де $t > k + 1$, k - розмір факторної бази.

Далі алгоритм повторює дії методу Діксона:

— методом Гауса знайти лінійну залежність серед векторів;

— якщо $x = \pm \pmod n$, повторити процедуру генерації FB;

— множники a та b рівні відповідно $НСД(x + y, n)$ та $НСД(x - y, n)$.

1.5 Атака Хастада

Як і будь-яка інша атака, має свої особливості. При її реалізації важливим моментом є те, що відкрита експонента має бути не меншою за кількість одержувачів зашифрованого повідомлення. Отже, особі, котра здійснює атаку, треба мати n пар відкритих ключів з однаковою відкритою експонентою та n шифротекстів, де вихідне повідомлення було однаковим [7, 8].

Алгоритм:

– для $C_i = M^m \bmod m$, де m - відкрита експонента і кількість шифротекстів, застосовуємо китайську теорему про лишки і отримаємо C';

– обчислюємо корінь ступеня, що дорівнює значенню експоненти, в тривіальному випадку - це 3, із C' тим самим отримуємо вихідне повідомлення;

– ця атака мало застосовна практично, так як значення відкритої експоненти вибираються, як правило, не настільки малими. Першою важливою складністю стає перехоплення одного повідомлення для великої кількості одержувачів

1.6 Атака Вінера

Цю атаку найкраще описувати, починаючи з самої теореми. Припустимо, що є число $n = p * q$ при цьому $q < p < 2q$. Також виконується умова, що $d < {}^{1}/_{3} * n^{1/4}$. Припустимо, що відома пара (n,e), для якої виконується умова $e*d = 1 \ mod \ \varphi(n)$. Тоді існує ефективний спосіб обчислення числа d.

Сам алгоритм атаки ґрунтується на безперервних дробах [7, 8].

Алгоритм:

1. Розкладання e/N в безперервний дріб $[a_1, a_2, a_3, ...]$.
2. Для отриманого безперервного дробу знаходиться множина відповідних дробів k_n/d_n (кінцевий ланцюговий дріб, значення якого є деяке раціональне число).
3. Перевірити відповідний дріб:
   - 3.1. Знаходження можливого значення $\varphi(N)$.
   - 3.2. Знайти рішення наступного рівняння: $x^2 - ((N - f_n) + 1)x + N = 0$. В результаті рішення вийде пара коренів p_n і q_n.
4. Якщо $N = p_n * q_n$, то закрита експонента d_n успішно знайдена, інакше повернення до пункту 3 з вибором наступного дробу.

2 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

2.1 Засоби реалізації

Для створення програмного засобу вибрана мова C# [10] із застосуванням системи для побудови клієнтських програм WPF [11]. Додаток у своєму складі має наступні класи:

– RSA;
– MR;
– QuadraticSieve;
– FrequencyAnalysis.

Опис цих класів буде відображено в подальшому тексті.

Частина атак реалізована мовою програмування Python з міркувань простоти реалізації, наприклад атака Хастада.

З бібліотек слід відзначити System.Numerics, котра дозволяє працювати з великими числами, так як більшість досліджуваних чисел не буде поміщатися в стандартні типи даних, такі як int, та System.Diagnostics, яка використовується для оцінки часу алгоритмів.

Додаток складається з п'яти основних текстових полів з назвами (рис. 2.1):

– Message Box;
– Frequency Box;
– Encrypted Box;
– Quadratic Sieve Box;
– Size prim.

Також містить чотири кнопки (див. рис. 2.1):

– Encrypt;
– Decrypt;
– Frequency;
– QS.

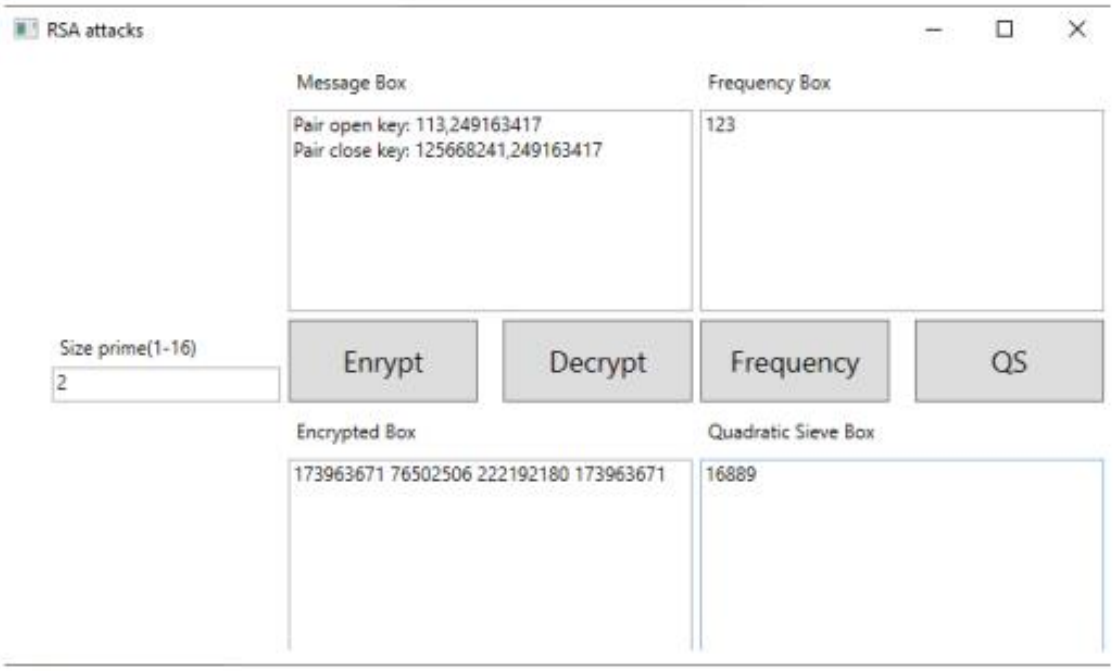

Рисунок 2.1 – Зовнішній вигляд програмного додатку

Варто описати основні дії, котрі здатна виконувати розробка.

Шифрування. У полі MessageBox вводиться вихідне повідомлення. Після цього водиться довжина в байтах для простого числа у полі SizePrime. Далі після натискання кнопки Encrypt, зашифроване повідомлення відображається у полі Encrypted Box, а в полі MessageBox залишається інформація про використані ключі шифрування. У полі Quadratic Sieve автоматично вводиться n, котре використовується при шифруванні.

Дешифрування: За натисканням кнопки Decrypt застосовується алгоритм дешифрування повідомлення за заданими ключами, повідомлення відображається в Message Box.

Застосування квадратичного решета: Алгоритм запускається після натискання кнопки QS, результат замінює поточні дані у полі Quadratic Sieve (рис. 2.2).

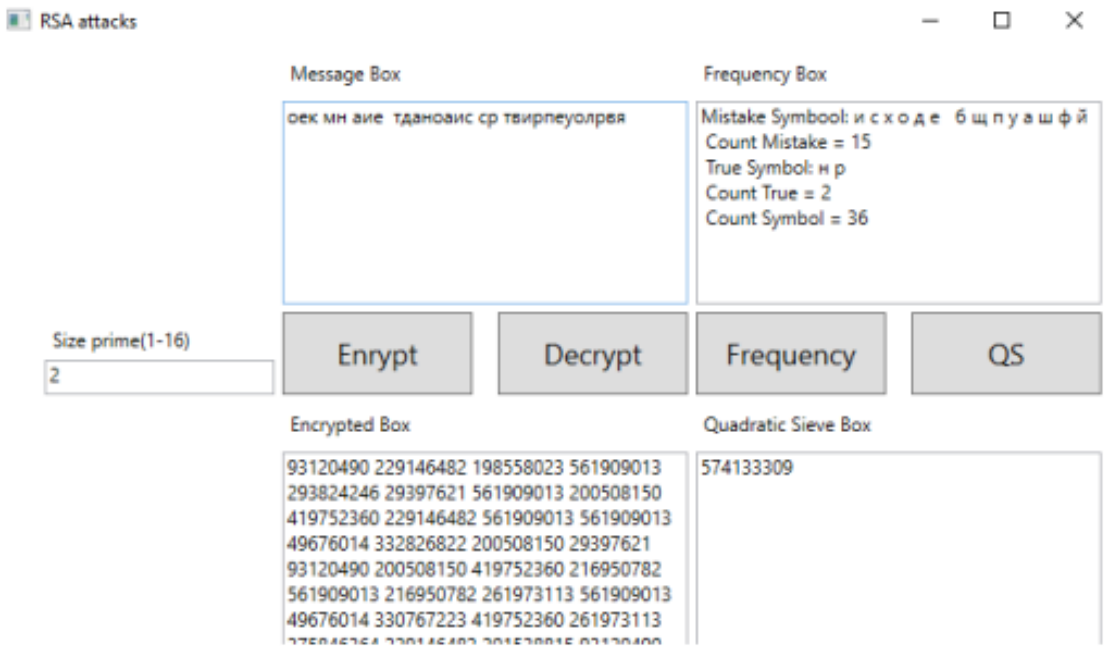

Рисунок 2.2 – Приклад роботи кнопки QS

Застосування частотного аналізу. Натисканням кнопки Frequency застосовується частотний криптоаналіз, інформація про його застосування заповнює поле Frequency, а передбачуваний варіант заноситься до поля Message.

2.2 Класи RSA та MR

Клас RSA призначений для генерації ключів, шифрування та дешифрування тексту на їх основі. Містить поля e, d, n типу BigInteger. Також має 3 конструктори, що дозволяють створити як новий екземпляр класу з параметрами, що генеруються, так і задавати їх самому.

Опис функцій класу RSA:

— ToString - є перевизначеною функцією, повертає пари відкритого та закритого ключа для заданого екземпляра у форматі "Pair open key: " + e + ',' + n + '\n' + "Pair close key: " + d + ',' + n + '\n';

— StrToBigInteger — як параметр приймає рядок, повертає переведений рядок до типу BigInteger;

— BigIntegerToStr — як параметр приймає BigInteger число, повертає переведений рядок;

— CharToBigInteger - як параметр приймає символ, повертає еквівалентне йому BigInteger число;

— encryptSymbol - як параметри приймає символ і екземпляр класу RSA, виконує шифрування заданого символу, повертає BigInteger число;

— decryptSymbol — як параметр приймає BigInteger число та екземпляр класу RSA, виконує дешифрування та повертає символ;

— encryptLetterByLetter — посимвольне шифрування, де параметрами є екземпляр класу RSA і рядок вихідного повідомлення;

— decryptLetterByLetter — посимвольне дешифрування, де як параметри виступає екземпляр класу RSA і рядок зашифрованого повідомлення;

— Encrypt – класичне шифрування повідомлення;

— Decrypt – дешифрування повідомлення;

— genKey - генерація ключів, має 1 вхідний параметр - довжина простого числа в байтах і 3 вихідних: e, d, n;

— generatePrime - генерація простого числа для заданого числа байт;

— generateE - генерація відкритої експоненти;

— RAE - реалізація розширеного алгоритму Евкліда, що має 2 вхідні параметри типу BigInteger, повертає число d для генерації ключів.

Фрагмент коду класу наведено на рис. 2.3.

```
Class RSA:
        BigInteger e, d, n;
        public BigInteger GetE() { return e; }
        public BigInteger GetN() { return n; }
        public BigInteger GetD() { return d; }
        /// <summary>
        /// <returns></returns>
        public override string ToString()
        {
            return "Pair open key: " + e + ',' + n + '\n' + "Pair close key:
" + d + ',' + n + '\n';
        }
```

Рисунок 2.3 – Фрагмент коду класу RSA

Опис класу MR - клас є статичним та містить 2 функції:

— власне реалізація тесту Міллера-Рабіна

— перевірку тривіальних дільників. Кількість очевидних дільників регулюється полем SIZESIMPLEDIVIDERS.

2.3 Клас FrequencyAnalysis

Призначений для частотного аналізу. Для нього є деякі статичні поля. Поле pathCSV відповідає за шлях створення файлу FrequencyBigInt.csv, поле FrequenctRusAlphabet є словником, де кожному ключу типу char відповідає його відносна частота типу double. Такі поля типу bool як AlphabetIsInitialized та AlphabetIsChanged, відповідають за стан словника: чи він ініціалізований вже і чи був він змінений.

Опис функцій класу FrequencyAnalysis:

— PreparationOfText — функція на вхід отримує повідомлення, після чого видає на його основі результат, що відповідає наступним вимогам: у повідомленні беруть участь лише літери нижнього регістру, всі символи, що не належать до українського алфавіту і пробіл;

— AnalysDef — функція порівняння двох повідомлень оригіналу та можливого варіанта дешифрування. Результат містить таку інформацію: кількість символів, відповідних оригіналу та їх список, кількість символів, що відрізняються від оригіналу та їх список, загальна кількість символів;

— InitializeRusAlphabet - ініціалізація словника FrequenctRusAlphabet заданими символами та їх відносними частотами;

— CountSpace — підрахунок кількості пропусків у рядку;

— GetBigIntArray — перетворює рядок, який містить пробіли і числа, масив типу BigInteger;

— ResultFrequencyToCSV — збереження таблиці відносних частот для кожного числа BigInteger у файлі формату csv;

— GetSymbolWithMinDef — повертає найближчий символ зі схожою заданою частотою, у своїй виключає його вихідного словника;

— GetPossibleVariant — функція на вхід отримує частотний словник символів зашифрованого повідомлення і повідомлення, в результаті на основі застосування функції GetSymbolWithMinDef повертає можливий варіант декодування;

— GetCounterSymbolDictionary — функція отримання словника статистики, де кожному числу BigInteger відповідає число int , що позначає кількість цього числа в зашифрованому повідомленні;

— GetFrequency - отримання словника частот числа BigInteger, де й застосовується функція GetCounterSymbolDictionary;

— RunAnalys - основна статична функція частотного криптоаналізу, що складається з чотирьох етапів: перетворення запису в масив BigInteger, отримання частотної статистики для цього масиву, запису отриманих даних у файл, отримання найближчого відповідного варіанту.

На рис. 2.4 показано фрагмент коду класу.

```
Class FrequencyAnalysis:
        static string pathCSV = Directory.GetCurrentDirectory() +
@"\FrequencyBigInt.csv";
        static Dictionary<char, double> FrequencyRusAlphabet = new
Dictionary<char, double>();
        static Dictionary<char, double> FrequencyEngAlphabet = new
Dictionary<char, double>();
        static bool AlphabetIsInitialized = false;
        static bool AlphabetIsChanged = false;
        public static string PreparationOfText(string Original)
        {
            string result="";
```

Рисунок 2.4 – Фрагмент програмного коду класу FrequencyAnalysis

2.4 Клас QuadraticSieve

Даний клас необхідний для факторизації чисел типу BigInteger методом квадратичного решета і містить статичні функції, які в сукупності є цим методом

з деякими особливостями, що використовуються для аналізу роботи.

Функції класу QuadraticSieve:

— isSqrt - функція перевіряє, чи є другий параметр root коренем числа n;

— SqrtBigInteger - знаходження кореня для числа n, який відноситься до типу BigInteger, результат має аналогічний тип;

— Gauss - функція розв'язання СЛАР методом Гаусса;

— LegendreSymbol — це обчислення символу Лежандра, який використовується для визначення квадратичного оишку або нелишку;

— EratosfenSieve - функція решета Ератосфена, яка до заданого числа залишає лише прості числа та формує з них набір;

— SemiprimeSortedSet - генерація набору напівпростих чисел, що використовується для аналізу роботи алгоритму;

— SemiprimeToCSV - записує отримані напівпрості числа у файл Semiprimes.csv, який знаходиться в директорії самої програми;

— Analys - функція, що виконує аналіз роботи алгоритму набору напівпростих чисел;

— Run - основна функція алгоритму, що містить усі етапи, які не винесені в окремі функції.

Фрагмент програмного коду класу наведено на рис. 2.5.

```
Class QuadraticSieve:
        static public void Analys(SortedSet<BigInteger> Semiprimes)
        {
            Dictionary<BigInteger, string> TimeForFactor = new
Dictionary<BigInteger, string>();
            FileStream fstream = new FileStream(PathAnalysCsv,
FileMode.Create);
            StreamWriter sw = new StreamWriter(fstream);
            foreach (var item in Semiprimes)
            {
                Stopwatch stopwatch = new Stopwatch();
                stopwatch.Start();
                BigInteger result = Run(item);
```

Рисунок 2.5 – Фрагмент коду класу QuadraticSieve

2.5 Створювані файли під час роботи

Файл Semiprimes містить напівпрості числа, для зручності можна відкрити як засобами Microsoft Excel (рис. 2.6), так і за допомогою звичайного блокнота (рис. 2.7). Шлях до цього файлу задається в полі класу QuadraticSieve, стандартним чином поле має шлях до поточної директорії.

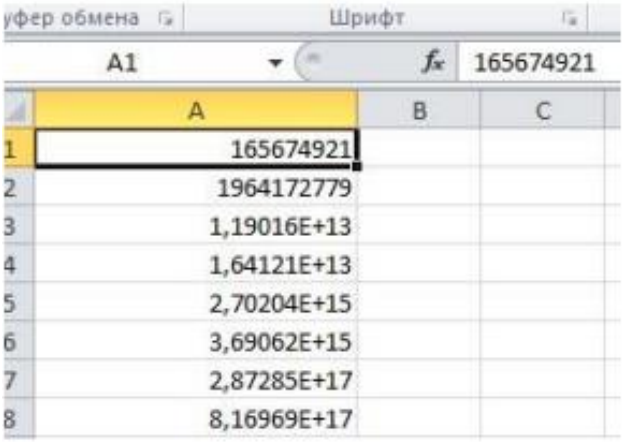

Рисунок 2.6 – Файл Semiprimes.csv відкритий у Excel

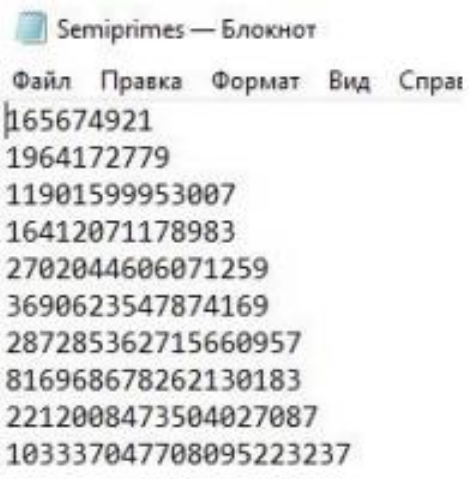

Рисунок 2.7 – Файл Semiprimes.csv відкритий у блокноті

Файл FrequencyBigInt.csv містить частотний аналіз BigInteger чисел в останньому досліджуваному повідомленні. Файл аналогічно можна відкривати як засобами Microsoft Excel, так і блокнотом, або за допомогою Notepad++. Шлях

до файлу задається полем класу FrequencyAnalysis і значенням за промовчанням є шлях до поточної директорії. Сам потік для створення файлу створюється у функції ResultFrequencyToCSV (рис. 2.8).

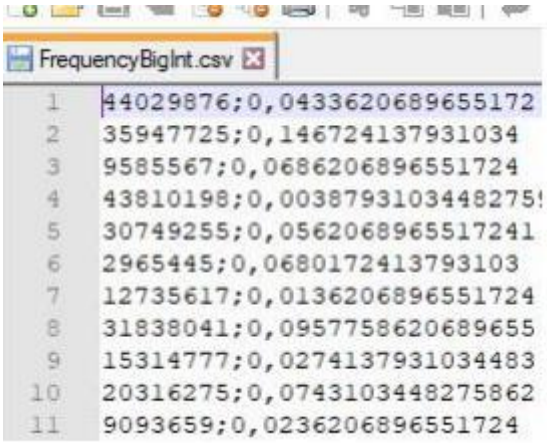

Рисунок 2.8 – Файл Semiprimes.csv відкритий у блокноті

Файл AnalysCSV.csv аналогічно до попередніх знаходиться у поточній директорії та відкривається за допомогою notepad (рис. 2.9), Microsoft Excel. Кожен рядок містить напівпросте число, його множник та загальний час, витрачений на факторизацію числа. Шлях до файлу задається полем PathAnalysCsv класу QuadraticSieve.

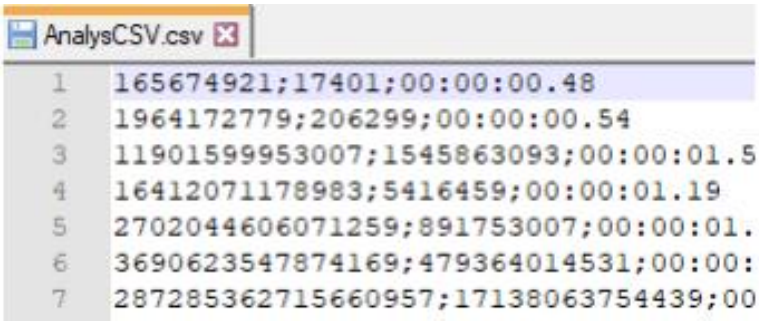

Рисунок 2.9 – Файл AnalysCSV.csv

2.6 Клас Hastad

Клас, реалізований на Python. Містить кілька коротких функцій, які разом реалізують цю атаку.

— chinese_remainder - обчислення за китайською теоремою про лишки;

— mul_inv - мультиплікативне зворотне;

— find_invpow - обчислення коренів різних ступенів.

2.7 Клас Wiener

Атаку аналогічно реалізовано на Python. Функції можна розділити на кілька блоків.

Перший блок пов'язаний з арифметикою та містить наступні функції:

— egcd – розширений алгоритм Евкліда;

— gcd - алгоритм Евкліда;

— totient – функція Ейлера;

— is_perfect_square - ідеальний квадрат (ціле число, квадратний корінь котрого також є цілим числом);

— issqrt – квадратний корінь;

— hack_RSA - основна функція атаки, яка викликає інші функції, тобто. це вихідна точка алгоритму.

Наступні функції потрібні до роботи з безперервними дробами: rational_to_contfrac, convergents_from_contfrac, contfrac_to_rational.

3 РЕЗУЛЬТАТИ ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

3.1 Частотний аналіз

Візьмемо кілька великих текстів і збільшуватимемо розмірність тексту для шифрування та наступного частотного аналізу. Збільшуватимемо від 200 до 10 000 символів з кроком у 200 символів.

Як перший текст візьмемо випадково згенерований текст за допомогою одного із сервісів для автоматичного заповнення контенту. Результати наведено на рис. 3.1.

Рисунок 3.1 – Кількість правильних символів для згенерованого випадково тексту

Незважаючи на те, що вихідний текст складався зі слів, які дійсно існують і занесені до тлумачного словника української мови, в ньому не було враховано важливу вимогу – змістовність (розуміння). Тому зі збільшенням розмірності тексту результати майже не змінювалися.

Далі на вхід подамо текст, написаний дійсним користувачем, де буде дотримуватися умова змістовності пропозицій, тобто ймовірно, зі збільшенням розмірності тексту частоти символів повинні прагнути до заданих в алфавіті, а

отже, і кількість помилок при дешифруванні знижується. Результати наведено на рис. 3.2.

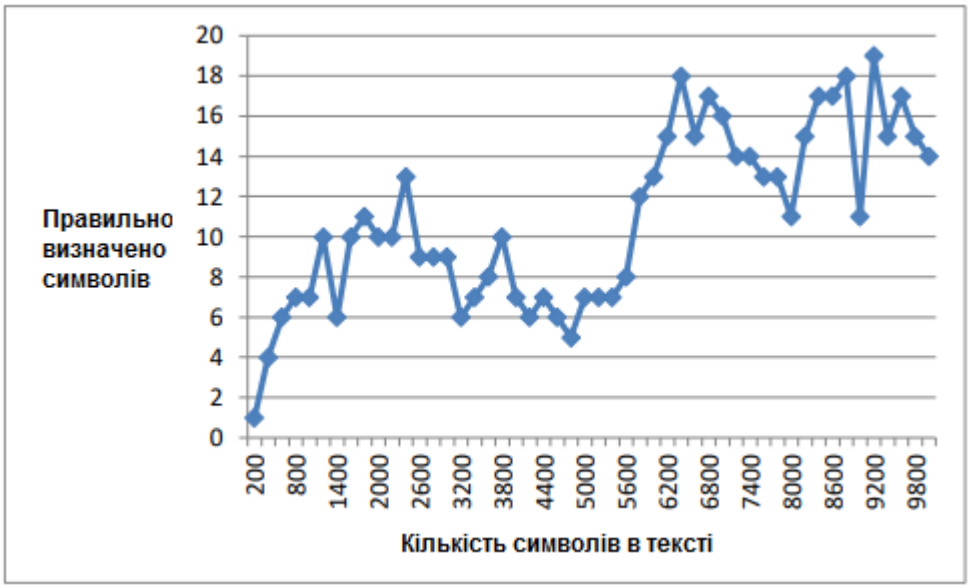

Рисунок 3.2 – Кількість правильних символів для тексту, створеного реальним користувачем

Як і було припущено, кількість чітко визначених символів досить швидко зростає зі збільшенням розмірності тексту. Варто також зазначити, що з подальшому аналізі тексту можна відновити все вихідне повідомлення, минаючи помилки. Починаючи з мінімальних розмірностей пропуск був одночасно визначено чітко, що досить важливим моментом, оскільки це дозволяє текст розбити на слова. Після цього можна шляхом підбору для слів, що складаються з 1 -3 букв визначити потрібні букви використовуючи сполучники і прийменники.

Розглянемо докладніше пік за 9200 символів. Було визначено правильно 19 символів, а помилки склали наступний набір «п е і н а ц з м б й х є щ». Насправді такий набір не є поганим показником, так як частина з цих літер міститься в сполучниках і прийменниках, а отже, їх досить легко відновити.

3.2 Факторизація методом квадратичного решета

Для початку перевіримо, за яких параметрів ми будемо досягати кращих

результатів. По-перше, ми можемо змінити початковий розмір факторної бази, позначимо його за l. При цьому стандартне значення l знаходиться за заданою формулою $\exp(\sqrt{\ln n * \ln \ln n})$. По-друге, ми можемо регулювати кількість пар чисел, де $t \geq k + 1$, де k - розмір факторної бази.

Як тест будуть використані наступні напівпрості числа:

1. 165674921
2. 1964172779
3. 11901599953007
4. 2702044606071259
5. 287285362715660957
6. 2212008473504027087
7. 103337047708095223237
8. 3602099130590696255339
9. 17020732667346627004129
10. 1284564637232988967387229
11. 122779349257007148076936 31
12. 17464306393977903129606487
13. 29355918493053952853404 4293
14. 11083969892933627756301 56033
15. 44775079919452148331616494007
16. 21875004005106592855625795843 87
17. 33796108178554770690550543875817

Для початку спробуємо збільшувати t на 1 (рис. 3.3), на 2 (рис. 3.4), на 5 (рис. 3.5), що збільшуватиме матрицю на останніх етапах, але також підвищуватиме ймовірність того, що рішення буде знайдено відразу і не виникне потреби повторювати генерацію чисел.

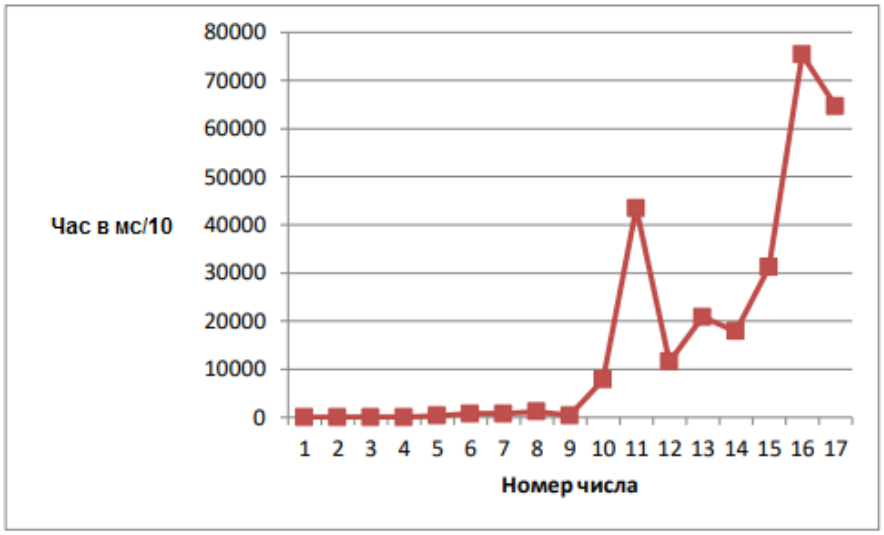

Рисунок 3.3 – Час при t = 1

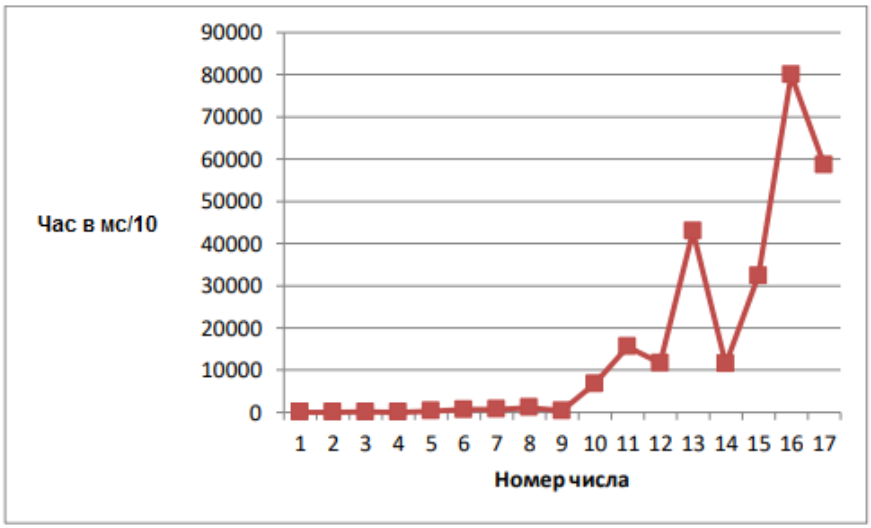

Рисунок 3.4 – Час при t = 2

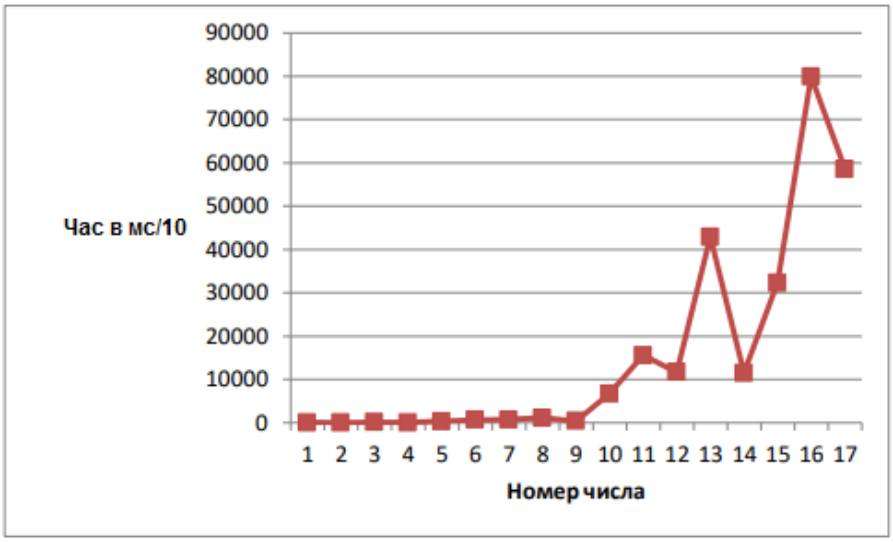

Рисунок 3.5 – Час при t = 5

Незважаючи на схожість отриманих графіків, варто зазначити, що для 11-го напівпростого числа показники стали кращими більш, ніж у 2 рази, це пов'язано з тим, що до збільшення t, повторювалася процедура генерації чисел.

Далі спробуємо збільшити початкову факторну базу чисел. Збільшуватимемо у 2 (рис. 3.6), у 3 (рис. 3.7) та у 4 (рис. 3.8) рази відповідно.

Нагадаємо, що в поточному контексті стандартний розмір позначатиметься як l.

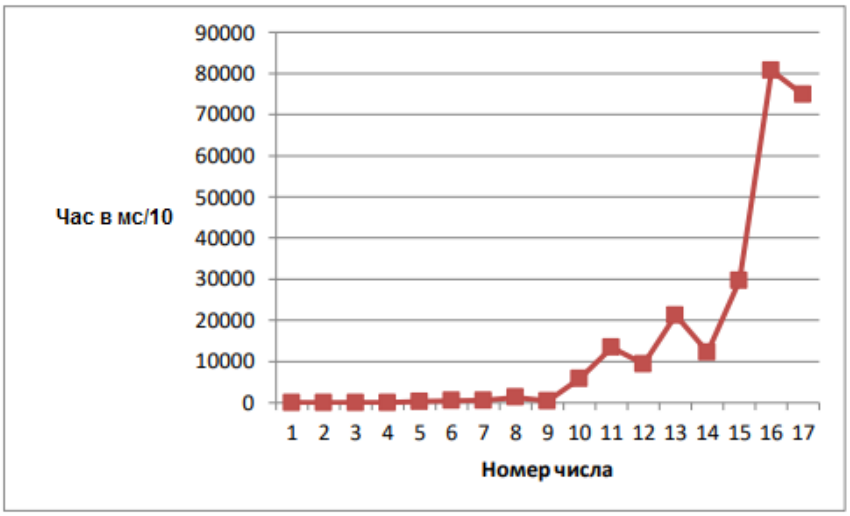

Рисунок 3.6 – Час при 2l

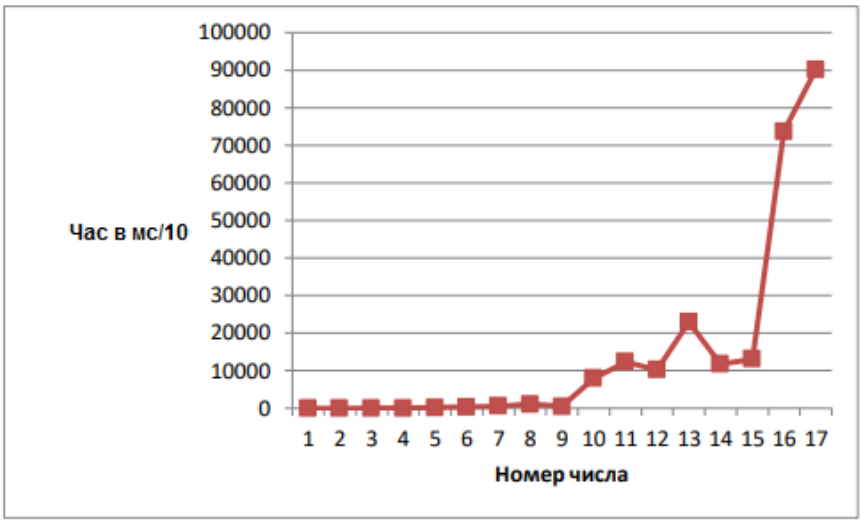

Рисунок 3.7 – Час при 3l

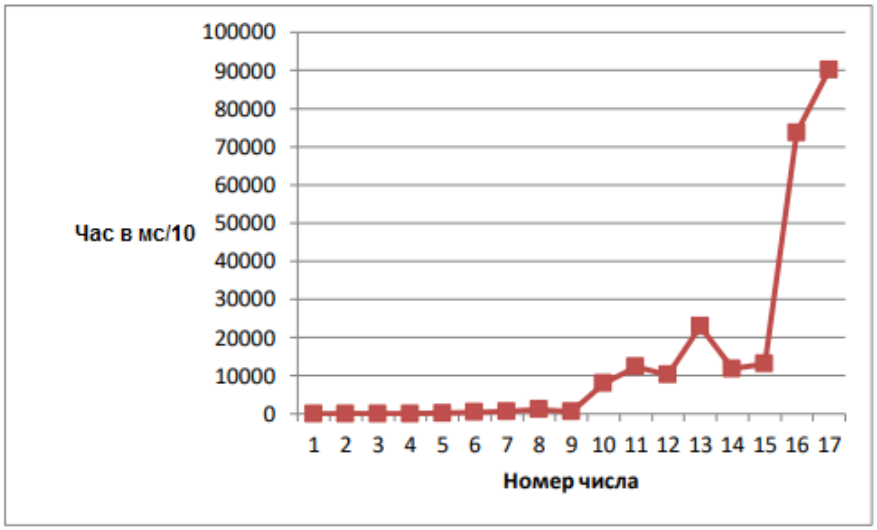

Рисунок 3.8 – Час при 4l

Також стало помітно згладжування деяких піків на графіку при збільшенні l. Для наочності можна порівняти загальний час роботи, затрачений на всі числа: 256.281 сек для l2, 250.523 сек для l3, 245.093 сек для l4. Таким чином, збільшення факторної бази допомогло досягти кращих результатів. Найбільше число, розглянуте нами, мало в бітовому поданні розмірність 105 біт. Спроби розкласти числа більшої розмірності було зупинено, оскільки процес займав багато часу.

3.3 Атака Хастада

При аналізі результатів ми не розглядатимемо підсумки малих чисел *e*. Під малим числом ми розумітимемо будь-яке менше, ніж 100. Зазначимо, що, наприклад, для числа $e = 3$, необхідно 3 одержувача і дешифрування повідомлення з використанням атаки Хастада відбувається менше секунди. Достатньо хороший показник, і саме з цієї причини на практиці не повинно бути таких малих чисел.

Для аналізу візьмемо кілька чисел: 113, 199, 317. Очевидно, що ці числа досить малі, але цього достатньо для відображення зниження швидкості роботи даної атаки. І різні розміри ключів в бітах: 64, 128 і 256.

Наочно показано, що відбувається при зміні розмірності ключа для різних

чисел на рис. 3.9, 3.10, 3.11 відповідно.

Рисунок 3.9 – Зростання часу в мс для *e* = 113

Рисунок 3.10 – Зростання часу в мс для *e* = 199

Рисунок 3.11 – Зростання часу в мс для *e* = 317

Графіки досить близькі до лінійних. Можна сказати, що для малого *e* буде помилкою покращення лише інших параметрів шифрування. Тепер же подивимося, які результати будуть отримані для кожної розмірності ключів зі збільшенням *e* на рис. 3.12, 3.13, 3.14, 3.15.

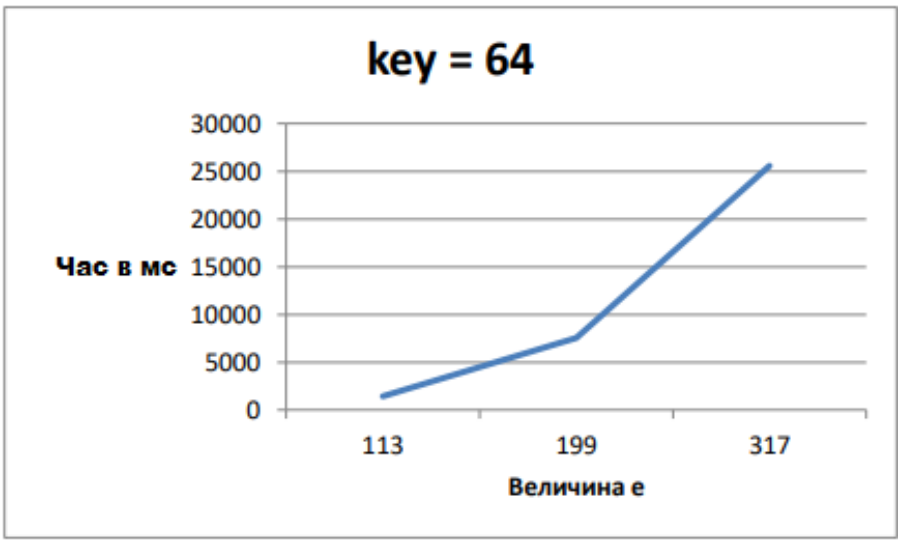

Рисунок 3.12 – Зростання часу в мс для розмірності ключа = 64

Рисунок 3.13 – Зростання часу в мс для розмірності ключа = 128

Рисунок 3.14 – Зростання часу в мс для розмірності ключа = 256

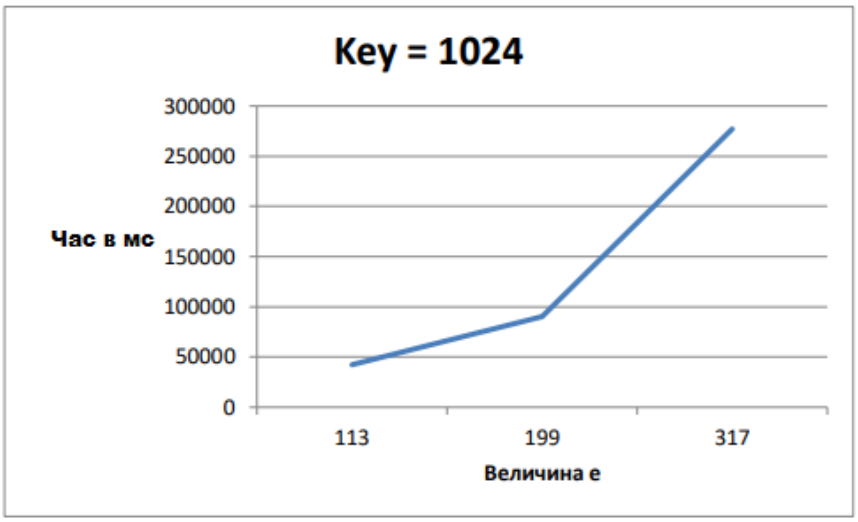

Рисунок 3.15 – Зростання часу в мс для розмірності ключа = 1024

З графіків видно, що збільшення *e* позитивно впливає на шифрування тексту. Більше того, це означає, що й атакуючому потрібна велика кількість перехоплених повідомлень. А за таких умов завдання атаки може стати практично неможливим. На загальному графіку показано результати всіх порівняльних аналізів (рис. 3.16).

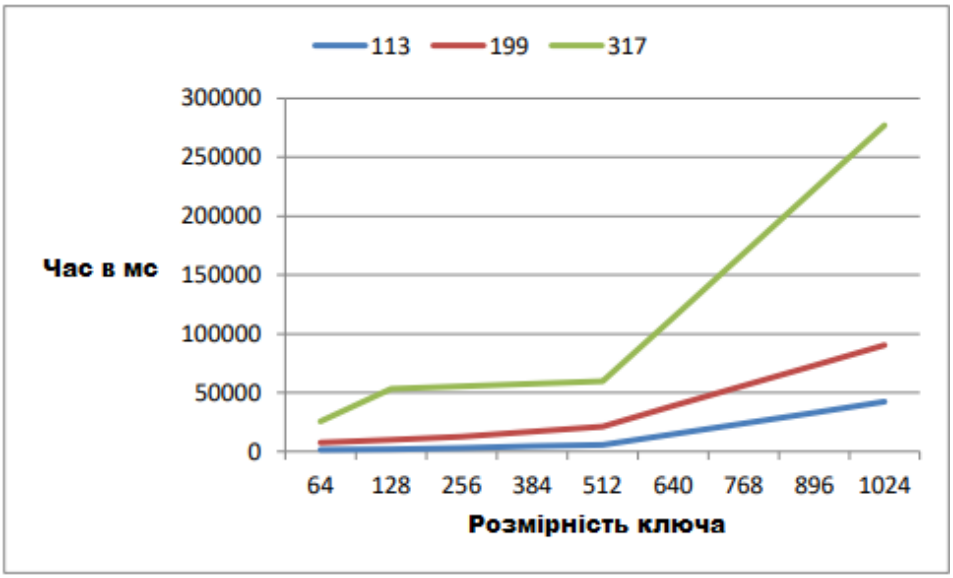

Рисунок 3.16 – Загальний графік

3.4 Атака Вінера

Візьмемо різні розмірності ключів. Для зручності почнемо із чисел 64 і 128, та будемо послідовно збільшувати на 128 біт. Загальні результати показані на рис. 31.7.

Рисунок 3.17 – Загальний графік часу атаки Вінера

Час на графіку вказано в мілісекундах вертикальної осі, а по горизонтальній вказана розмірність. Можемо відзначити, що після розмірності 1024 біта витрачений час різко зростає. Розглянемо докладніше відрізок до 896 біт рис. 3.18.

Рисунок 3.18 – Від 64 біт до 896 біт

Трохи більше 68 мілісекунд знадобилося для розмірності 512 біт — досить

хороший показник часу для атакуючого.

Далі розглянемо випадок із розмірністю до 4096 біт рис. 3.19.

Рисунок 3.19 – Від 1024 біт до 4096 біт в мс

Як виявилося, задання часу в мілісекундах для великих розмірностей - помилка. Тому повторимо графік із одиницею виміру - секунда (рис. 3.20).

Рисунок 3.20 – Від 1024 біт до 4096 біт в с

Для розмірності в 4096 біт результатом часу атаки Вінера буде понад 18 хвилин. З одного боку, цей показник хороший, але ця атака дійсна тільки для $d < \frac{1}{3} n^{\frac{1}{4}}$, що в порівнянні з малою відкритою експонентою зустрічається ще рідше.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при опіках

Опік — це ушкодження тканин, яке спричинене дією високої температури, електроструму, хімічних речовин, рентгенівських і сонячних променів. Опіки можуть бути первинними (миттєвими) та вторинними. Вторинні опіки є результатом займання одягу та охоплення полум'ям тіла.

Перша допомога при опіках, які спричинені полум′ям передбачає гасіння палаючого одягу водою, з використанням вогнегасника або потерпілого накривають плащем, пальто, ковдрою тощо. Якщо необхідно звільнити постраждалого від одягу, після гасіння вогню в деяких випадках для звільнення ураженої ділянки тіла одяг розрізають. При обмежених опіках, спричинених окропом, обпечену поверхню тіла охолоджують холодною водою з-під крану впродовж 10 хв. Після цього накладають асептичну суху пов′язку, застосовуючи чисту бавовняну тканину, бинт, індивідуальний пакет. У випадку поширених опіків потерпілого загортають у чисте простирадло, вкриваючи ковдрою зверху, та дають пити чай [20].

Під час надання першої лікарської допомоги бригадою швидкої допомоги, крім накладання асептичної пов′язки, вводять серцеві та знеболювальні засоби, здійснюють протишокові заходи під час транспортування в медичний заклад.

Під час надання долікарської допомоги застосовують: інфільтрацію уражених тканин 0,25 % розчином новокаїну, місцеву гіпотермію, новокаїнову блокаду, накладають фурацилінові пов'язки, шкіру змащують стерильною олією.

При термічних опіках накладають стерильну пов'язку, забезпечують тепло, обмотують людину простирадлом, дають солодку каву, теплий чай, обтирають горілкою, одеколоном, спиртом чи змащують ними бинти і пов'язки. При наявності шоку дають пити двадцять крапель валер'янки. При хімічних опіках 10-20 хвилин виконують промивання обпеченого місця струменем проточної води. У випадку отримання опіку лугом накладають марлеву пов'язку, яка змочена розчином борної кислоти (1 грам на склянку води) [21].

Заборонено відривати пухирі, а також відривати від них одяг, каніфоль, сургуч, бо це може спричинити інфекцію та затягнути тривалість загоєння ран. Заборонено засипати рани порошками, змащувати мазями та маслом. При обширних і важких опіках (більше 15-30% всієї поверхні тіла) виникає загальне ураження організму, яке супроводжується важким шоком (опікова хвороба), викликає інтоксикацію організму, зміни складу крові, зміни в роботі центральної нервової системи (біль). Чим більша опікова поверхня, тим більше нервових закінчень уражено і тим сильніше проявляються явища травматичного шоку [22]. При опіках з’являється велика кількість виділень крізь опікову поверхню плазми крові, виникає отруєння організму від продуктів розпаду змертвілої тканини, які із зони ушкодження всмоктуються організмом. З'являються такі симптоми як блювання, загальна слабкість, головний біль. Потерпілому потрошки і часто дають пити воду з питною сіллю, (одна чайна ложка солі + пів чайної ложки соди на один літр води).

Для пов'язок застосовують індивідуальний пакет, стерильний бинт. Опечену поверхню можна накрити чистою тканиною з бавовни, пропрасованою гарячою праскою чи змоченою перманганатом калію, горілкою або етиловим спиртом, які зменшують біль. Постраждалого тепло вкривають, для зменшення шоку вводять наркотичні речовини (морфій, промедол), дають пити гарячий чай з вином, каву, трохи горілки [22].

У випадку виникнення опіку фосфором ушкоджену частину тіла опускають в воду і там пінцетом знімають частинки фосфору, шкіру обробляють п’яти процентним розчином мідного купоросу і закривають чистою сухою пов'язкою.

При ураженні хімічною зброєю отруйні речовини всмоктуються в кров зі значно більшою швидкістю, ніж при їх потраплянні на неушкоджену ділянку шкіри: ними можуть бути органи дихання і травлення, очі, заражені шкірні ділянки. Отруйні речовини можуть потрапляти на поверхню опіків і ран у вигляді газоподібних речовин, аерозолів і крапель.

Невідкладні заходи першої долікарської допомоги включають: інгаляцію кисню, дегазацію отруйних речових, введення антидоту, введення

протисудомних і серцево-судинних засобів, нейтралізація отруйних речовин на шкірі [20].

4.2 Вимоги пожежної безпеки при гасінні електроустановок

Пожежа – це неконтрольований процес горіння, який спричиняє нищення матеріальних цінностей та загибель людей. Серед причин виникнення пожеж можна виділити природні явища (посуха, блискавка), порушення правил пожежної безпеки, недбалу поведінку людей з вогнем. Відомо, що лише 7-8% пожеж спричинені блискавками і близько 90% виникає внаслідок діяльності людини [21].

Пожежна безпека організацій та підприємств, у яких використовуються електроустановки, забезпечується шляхом здійснення організаційно-технічних та інших заходів з попередження виникнення пожеж, зменшення можливих матеріальних збитків, забезпечення безпеки людей, зниження негативних екологічних наслідків, створення умов для успішного гасіння пожеж та швидкого виклику пожежних підрозділів, а також евакуації з території виникнення та ймовірного розповсюдження пожежі людей, матеріальних цінностей і документів. Система для контролю пожежної безпеки передбачає використання блоків живлення для подачі напруги необхідного рівня. Таке електрообладнання при неналежному нагляді, наприклад, при короткому замиканні, може стати епіцентром спалаху.

При виникненні пожежі можна виділити два методи, які застосовуються для гасіння електроустановок:

– відведених від напруги мережі;

– які знаходяться під напругою.

Електроустановки повинні бути під’єднані до заземлення з гнучкого мідного голого провідника з поперечним перерізом не менше 25 мм^2 , який підключається до заземлених конструкцій. Місця підключення до заземлених конструкцій, які визначаються фахівцями енергетичних об’єктів разом з

представниками гарнізону пожежної охорони, вносяться до графічної частини плану пожежогасіння та позначаються знаком заземлення [20].

Для забезпечення безпеки пожежників та персоналу, який бере участь у гасінні пожежі електроустановок під напругою, застосовуються ізолюючі індивідуальні електрозахисні засоби (діелектричні килими, калоші, боти, рукавиці). Кількість індивідуальних ізолюючих захисних засобів та заземлень і місця їх зберігання затверджуються керівниками енергетичних об'єктів враховуючи подачу вогнегасних засобів на електроустановки, які перебувають під напругою. Випробування електрозахисного обладнання виконується енергетичним об'єктом в установленому порядку [22].

У разі виникнення пожежі на електроустановці особа, яка першою виявила факт загорання, повинна негайно повідомити відповідальних за пожежну безпеку осіб та керівника для уникнення подальшого загорання. Гасіння електрообладнання під напругою із застосуванням ручних стволів повинне виконуватися за умови [23]:

– застосування ефективних прийомів і способів подачі в зону горіння вогнегасних речовин;

– дотримання електробезпечних відстаней від електрообладнання, яке знаходиться під напругою, до пожежників, які використовують ручні пожежні стволи;

– забезпечення надійного заземлення пожежних автомобілів і стволів;

– використання індивідуальних ізолюючих електрозахисних засобів під час гасіння пожежі електроустановок без зняття напруги.

Як вогнегасні речовини під час гасіння електроустановок, які знаходяться під напругою, доцільно застосовувати: розпилені порошкові суміші й інертні гази, струмені води, комбіновану суміш, яка являє собою розпилену воду з порошком. Застосування усіх видів піни під час гасіння електроустановок під напругою за участю людей ручними засобами забороняється, через те що піна й розчин піноутворювача мають підвищену електропровідність в порівнянні з розпиленою водою. При гасінні пожежі на електроустановках під напругою

потрібно застосовувати прийоми та засоби подачі в зону горіння вогнегасних речовин, що забезпечують ефективне гасіння пожежі і безпечну роботу пожежників.

Компактні струмені води доцільно використовувати лише під час гасіння пожеж на електроустановках під напругою до 110 кВ, але лише в тих випадках, коли до осередку горіння не має можливості наблизитися для подачі розпиленої води. В цьому випадку пожежник повинен перебувати на безпечній відстані від найближчих струмоведучих частин електроустановок, до яких може торкнутися струмінь води [21].

Для гасіння пожеж електроустановок, що знаходяться під напругою, можна застосовувати воду з водопровідних мереж, а також із штучних і природних водойм. Забір води насосами пожежних автомобілів з водойм варто здійснювати зі спеціально обладнаних пірсів.

При гасінні пожежі на електроустановках, які перебувають під напругою до 220 кВ включно, тривалість перебування пожежників на бойових позиціях не лімітується. Заземлення ручних пожежних насосів і стволів пожежних автомобілів при гасінні пожеж на електроустановках, які знаходяться під напругою, повинно виконуватись за допомогою гнучких мідних провідників з поперечним перерізом не менше 12 мм2 , оснащених спеціальними струбцинами для під’єднання до заземлених конструкцій: шурфів, обсадних труб артезіанських свердловин, металевих опор повітряних ліній електропередач, гідрантів водогінних мереж [20].

Місця під’єднання до заземлених конструкцій повинні затверджуватися спеціалістами енергетичного об'єкта, позначатися відповідними знаками заземлення і вноситись у графічну частину плану пожежогасіння. Ручні пожежні насоси й стволи пожежних автомобілів необхідно заземлювати окремо. В процесі подачі води від внутрішнього водопроводу заземлюються лише стволи.

Індивідуальні електрозахисні ізолюючі засоби (діелектричні боти, рукавиці) потрібно застосовувати для електробезпечності пожежників та персоналу, який безпосередньо бере участь у гасінні пожежі на електроустановках, які знаходяться під напругою.

Автомобілі пожежних частин, що охороняють енергетичні об'єкти, повинні бути укомплектовані індивідуальними ізолюючими захисними засобами відповідно до чисельності бойової обслуги, яка безпосередньо бере участь у процесі гасіння пожежі [23]. Необхідна кількість індивідуальних ізолюючих захисних засобів на енергооб'єктах, у тому числі для пожежних підрозділів, що залучаються до гасіння пожеж з інших частин, затверджується під час розробки планів пожежогасіння. При пожежі на електроустановках, які знаходяться під напругою, обслуговуючий персонал зобов'язаний у першу чергу повідомити про пожежу начальника зміни (диспетчера, чергового) й пожежну охорону, а потім вжити усіх необхідних заходів відповідно до плану пожежогасіння.

За необхідності гасіння пожежі повітряно-механічною піною, з об'ємним заповненням приміщення (тунелю) піною виконується попереднє закріплення піногенераторів, їх заземлення, а також заземлення насосів пожежних машин. Водій пожежної машини повинен працювати в діелектричному взутті і рукавицях.

Після прибуття до місця виклику першого пожежного підрозділу старший начальник (заступник начальника частини, начальник варти і т. ін.) повинен швидко зв'язатися з начальником зміни або посадовою особою, яка відповідає за виконання робіт, з метою уточнення ситуації на пожежі, одержання інструктажу та письмового допуску на виконання гасіння пожежі на електроустановках, що перебувають під напругою [23]. Після узгодження маршрутів руху до осередку горіння та розміщення бойових позицій, із яких пожежники виконуватимуть подачу вогнегасних речовин, керівник групи повинен провести інструктаж всього особового складу, який бере участь у гасінні пожежі, й віддати розпорядження на бойове розгортання.

З метою набуття вольових якостей, удосконалення тактичних навичок, підвищення фахової майстерності і забезпечення психологічної підготовки з врахуванням дотримання правил безпеки праці особовий склад пожежних підрозділів гарнізону зобов’язаний безпосередньо на енергетичному об'єкті не рідше одного разу на рік проходити спеціальний інструктаж.

ВИСНОВКИ

У ході вивчення та реалізації частотного аналізу були отримані результати як для згенерованого тексту, так і для тексту реальної людини.

Застосування частотного аналізу для першого випадку показало незадовільні результати, оскільки розподіл символів для такого тексту далекий від того, що занесено до частотної таблиці алфавіту.

У випадку із справжнім текстом результати інші. Незважаючи на те, що графік не є строго зростаючим при кроці 200 символів, починаючи з 5800 символів, кількість правильно визначених не опускається нижче 11. Таким чином, можна сказати, що зі збільшенням розмірності тексту при застосуванні частотного аналізу ми будемо отримувати кращі результати.

У результаті вивчення та реалізації квадратичного решета було отримано такі параметри на формування факторної бази, у яких досягається максимальна швидкість на аналізованому відрізку.

При використанні атаки Хастада було з’ясовано, що неприпустимим є використання малих значень відкритої експоненти. А для великих значень процес атаки взагалі може бути неможливим.

При розгляді атаки Вінера були отримані досить хороші результати для сторони, що атакує, що означає дотримання умови $d < \frac{1}{3} n^{\frac{1}{4}}$ має завжди виконуватись для безпечного використання шифрування RSA.

Що стосується питання про актуальність поточних вимог шифрування, то за умови правильної реалізації на сьогоднішній день не варто турбуватися про безпеку даних. При факторизації числа розмірністю понад 100 біт процес стає ресурсомістким, а поточні вимоги ґрунтуються на числах, котрі складаються з 1024 біт.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Directions in Cryptography. Diffie W., Hellman M. E. 1976 — [Електронний ресурс] - Режим доступа: https://ee.stanford.edu/~hellman/publications/24.pdf (дата звертання: 12.04.2023)

2. Factoring numbers using singular integers, Proceedings 23rd Annual ACM Symposium on Theory of Computing (STOC) (1991)/ L.M. Adleman, 1991. — p. 64–71.

3. Елементи теорії чисел : навч. посіб. / О. І. Оглобліна, Т. С. Сушко, Ю. В. Шрамко. – Суми : Сумський державний університет, 2015. – 186 с.

4. Завало С.Т. та ін. Алгебра і теорія чисел: Практикум. Частина 2. К.: Вища школа, 1986. 264 с.

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

6. Криптологія у прикладах, тестах і задачах: навч. посібник / Т.В. Бабенко, Г.М. Гулак, С.О. Сушко, Л.Я. Фомичова. Д.: Національний гірничий університет, 2013. 318 с.

7. Горбенко І. Д. Прикладна криптологія. Теорія. Практика. Застосування [Текст] / І. Д. Горбенко, Ю. І. Горбенко. Х. : Форт, 2012. 870 с.

8. Основи криптографічного захисту інформації / Г.М.Гулак, В.А. Мухачов, В.О. Хорошко, Ю.Є. Яремчук. В.: ВНТУ, 2011. 198 с.

9. Сушко С.О., Фомичова Л.Я., Барсуков Є.С. Частоти повторюваності букв і біграм у відкритих текстах українською мовою. Захист інформації. Київ, 2010. Т. 12, № 3. С. 94-102 DOI: https://doi.org/10.18372/2410-7840.12.1968

10. Посібник по C#. [Електронний ресурс] - Режим доступа https://programm.top/uk/c-sharp/tutorial/introduction/ (дата звернення 22.04.2023).

11. The complete WPF tutorial. [Електронний ресурс] - Режим доступа :https://wpf-tutorial.com/ (дата звернення 22.04.2023).

12. Anderson R., Bond M., Clulow J., Skorobogatov, S. Cryptographic processors – a survey. — 17 pp. : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.1740&rep=rep1&type=pdf.

13. YongBin Zhou, DengGuo Feng. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. 2005. 34 pp.

14. Song Y.Yan.Cryptanalytic Attacks on RSA. Springer. 2008. 255 pp.

15. Kocher, P.C. (1996). Timing Attacks on Implementations of DiffieHellman, RSA, DSS, and Other Systems. In: Koblitz, N. (eds) Advances in Cryptology CRYPTO ’96. CRYPTO 1996. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, Heidelberg. — 110-113 pp.

16. B. Kaliski, “Timing Attacks on Cryptosystems”, RSA Laboratories Bulletin, Number 2, January 1996 :

17. Koc Cetin Kaya. Cryptographic Engineering. Springer. 2009. - 517 pp.

18. Hoffstein, J. An Introduction to Mathematical Cryptography [Text] / J. Hoffstein, J. Pipher, J.H. Silverman. Springer, 2008. 523 p.

19. Baigneres, T. A Classical Introduction to cryptography Exercise Book [Text] / T. Baigneres, P. Junod, Y. Lu, J. Monneart, S. Vaudenay. Springer, 2006. 254 p.

20. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. Львів: Манголія 2006, 2007. 499 с.

21. Желібо Є. П. Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. Навчальний посібник. К.: Каравела, 2004. 328 с.

22. Зеркалов Д.В. Безпека життєдіяльності. Навчальний посібник. К.: Основа. 2011. 526 с..

23. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навчальний посібник. 2011. 215 с.