НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.9

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Алгоритмічно-програмний метод асиметричного шифрування даних»

Виконав:
студент ІІ курсу, групи КП-01мп
Квітка Олександр Вячеславович _____

Керівник:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доцент кафедри СПіСКС, к.т.н., доцент,
Клятченко Ярослав Михайлович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2020 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Квітці Олександру Вячеславовичу

1. Тема роботи «Алгоритмічно-програмний метод асиметричного шифрування даних», науковий керівник дисертації Онай Микола Володимирович, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету від «05» листопада 2021 р. № 3682-с

2. Термін подання студентом роботи «16» грудня 2021 р.

3. Об'єкт дослідження: процес асиметричного шифрування

4. Предмет дослідження: модифіковані алгоритми реалізації методів асиметричного шифрування.

5. Перелік завдань, які необхідно розробити:
   - провести аналіз методів асиметричного шифрування;
   - розробити модифікований метод асиметричного шифрування;
   - розробити програмне забезпечення, що реалізує запропонований модифікований метод шифрування;
   - провести тестування роботи розробленого методу шифрування;
   - порівняти ефективність розробленого методу з іншими.

6. Перелік ілюстративного матеріалу:
   - загальна схема роботи методу шифрування;
   - графіки, що ілюструють час роботи запропонованого методу;
   - канва бізнес-моделі;
   - дерево проблем.

7. Перелік публікацій:

- Тези доповіді «Модифікований метод генерації ключів для алгоритму асиметричного шифрування RSA»

8. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС, к.т.н., доцент		

9. Дата видачі завдання «20» жовтня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	16.12.2020	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	02.02.2021	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	13.03.2021	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	08.05.2021	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	14.08.2021	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2021	19.09.2021	
7.	Підготовка матеріалів третього розділу магістерської дисертації	07.11.2021	
8.	Підготовка матеріалів четвертого розділу магістерської дисертації; оформлення текстової і графічної частини магістерської дисертації	07.12.2021	

Студент _____ Олександр КВІТКА

Науковий керівник _____ Микола ОНАЙ

РЕФЕРАТ

Актуальність теми. Розвиток інформаційно-телекомунікаційних технологій значно ускладнює завдання збереження надійності захисту даних. Для цього необхідно підвищувати ефективність криптографічних алгоритмів, які використовуються для вирішення даного завдання. Серед криптографічних методів одним із найпоширеніших можна назвати метод асиметричного шифрування RSA, який постає в якості стандарту для великої кількості шифрувальних додатків та сервісів. Головною проблемою забезпечення надійності шифрування методу RSA на сьогоднішній день постає необхідність використовувати великі ключі шифрування, що залишаться недосяжними для обчислювальної потужності сучасних процесорів. Це негативно впливає на обчислювальну складність роботи алгоритму.

Об'єктом дослідження є процес асиметричного шифрування даних.

Предметом дослідження є методи асиметричного шифрування даних.

Мета роботи: вдосконалити процес асиметричного шифрування для роботи з великими ключами шифрування довжиною понад 10 тисяч біт, що забезпечить криптостійкість асиметричного шифрування на найближчі 20 років шляхом зменшення його обчислювальної складності та збільшення резистентності до криптографічних атак.

Методи дослідження. В роботі використовуються методи абстрактної алгебри, методи математичного програмування, метод інтерполяції.

Наукова новизна роботи полягає в наступному: запропоновано метод асиметричного шифрування, що відрізняється від існуючих застосуванням складеного модуля, що складається з простих чисел, кількість яких наближено дорівнює $\frac{1}{25}$ від довжини модуля, у поєднанні із застосуванням наслідку з Китайської теореми про лишки, що дозволяє

зменшити обчислювальну складність експоненціально пропорційно до збільшення довжини модуля шифрування.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований метод асиметричного шифрування має зменшену обчислювальну складність порівняно з існуючими методами для довжини модуля понад 10 тисяч біт.

Апробація роботи. Основні положення і результати роботи доповідалися та обговорювалися на XIV науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'ятьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У першому розділі проаналізовано існуючі методи асиметричного шифрування. виявлено основні переваги та недоліки кожного з методів. Для подальшого аналізу обрано метод RSA. Проаналізовано математичний апарат методу асиметричного шифрування RSA, існуючі методи модифікацій методу RSA. Для подальшого застосування обрано метод, що використовує наслідок з Китайської теореми про лишки та метод, що використовує складений модуль.

У другому розділі описано запропонований метод асиметричного шифрування та проаналізовано його особливості із використанням складеного модуля та способу розшифрування з використанням наслідку з Китайської теореми про лишки, а також введення рандомізації вибору кількості простих множників для генерації модуля n. Проаналізовано переваги даного методу, серед яких зменшення обчислювально складності роботи та зменшення використання пам'яті. Проаналізовано особливості найбільш популярних методів атаки на класичний метод асиметричного шифрування RSA у разі їх застосування до запропонованого методу

асиметричного шифрування та вплив внесених модифікацій на резистентність до зазначених методів атаки.

У третьому розділі проаналізовано переваги та недоліки мов програмування Python та R. Для реалізації запропонованого методу асиметричного шифрування та програмного забезпечення для аналізу обчислювальної складності методу шифрування вирішено використати мову Python. Проаналізовано використані бібліотеки Matplotlib та NumPy. Проаналізовано структуру розробленого програмного забезпечення, його модулі та взаємозв'язки між різними частинами застосунку.

У четвертому розділі проведено експериментальне дослідження ефективності розробленого алгоритмічно-програмного методу асиметричного шифрування, порівняно час його роботи із часом роботи класичного методу асиметричного шифрування RSA. В результаті виведено формулу отримання оптимальної кількості великих простих чисел у складеному модулі, використання якої дозволяє отримати найменший час роботи виконання алгоритму.

У п'ятому розділі проаналізовано сферу криптографічних систем, визначені основні користувачі системи та їх основні проблеми. За виконаним аналізом побудовано бізнес-модель кінцевого продукту та розраховані фінансові показники системи.

У висновках проаналізовано отримані результати дослідження.

Робота виконана на 80 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 29 найменувань. У роботі наведено 16 рисунків та 4 таблиці.

Ключові слова: методи асиметричного шифрування, RSA, обчислювальна складність, криптографічна система.

ABSTRACT

Actuality of theme. The development of information and telecommunication technologies significantly complicates the reliability of data protection. To do this, you need to increase the efficiency of cryptographic algorithms required for this task. Among the cryptographic methods, one of the most common is the asymmetric encryption method RSA, which is becoming the standard for a large number of encryption applications and services. The main problem in ensuring the reliability of the RSA encryption method today is the need to use large encryption keys, which remains unattainable for the computing power of modern processors. This has a negative impact on the computational complexity of the algorithm.

Object of research is the process of asymmetric encryption.

Subject of research is modified methods of asymmetric encryption.

Goal of the work is to improve the process of asymmetric encryption to work with large encryption keys over 10 thousand bits, which will ensure cryptographic stability of asymmetric encryption for the next 20 years by reducing its computational complexity and increasing resistance to cryptographic attacks.

Methods of research: the methods of abstract algebra, the method of mathematical programming, interpolation method.

Scientific novelty of the work is as follows: proposed a asymmetric encryption method, which differs from the existing ones by using a composite module consisting of prime numbers, the number of which is approximately equal to $\frac{1}{25}$ of the module length, combined with the consequence of the Chinese Remainder Theorem exponentially proportional to the increase in the length of the encryption module.

Practical value of the results obtained in the work is that the proposed asymmetric encryption method has reduced computational complexity compared to existing methods for module lengths over 10 thousand bits.

Approbation. The main provisions and results of the work were reported and discussed at the XIV scientific conference of undergraduates and graduate students «Applied Mathematics and Computing» AMC-2021.

Structure and scope of work. The master's dissertation consists of an introduction, five chapters and conclusions.

The introduction presents a general description of the work, assesses the current state of the problem, substantiates the relevance of research.

The first section analyzes the existing methods of asymmetric encryption. identified the main advantages and disadvantages of each method. The RSA method was chosen for further analysis. The mathematical apparatus of the asymmetric RSA encryption method, the existing methods of RSA method modifications are analyzed. For further application, a method using the consequence of the Chinese Remainder theorem and a method using the composite module are chosen.

The second section describes the proposed method of asymmetric encryption and analyzes its features using a composite module and the method of decryption using the consequence of the Chinese Remainder theorem, as well as the introduction of randomization of the choice of the number of prime factors to generate module n. The advantages of this method are analyzed, including the reduction of computational complexity and the reduction of memory usage. The peculiarities of the most popular methods of attack on the classical method of asymmetric encryption RSA in the case of their application to the proposed method of asymmetric encryption and the impact of modifications on the resistance to these methods of attack are analyzed.

The third section analyzes the advantages and disadvantages of Python and R programming languages. To implement the proposed method of asymmetric encryption and software for analyzing the computational complexity of the encryption method, it was decided to use Python. Used Matplotlib and NumPy libraries are analyzed. The structure of the developed software, its

modules and interrelations between different parts of the application are analyzed.

In the fourth section, an experimental study of the effectiveness of the developed algorithmic-software method of asymmetric encryption was compared, compared with the time of its operation with the operating time of the classical method of asymmetric encryption RSA. As a result, the formula for obtaining the optimal number of large prime numbers in a compound module is derived, the use of which allows to obtain the shortest execution time of the algorithm.

The fifth section analyzes the field of cryptographic systems, identifies the basics of system users and their main problems. According to the analysis, the business model of the final product was built and the financial indicators of the system were calculated.

The conclusions contains brief summary of study results.

The work is performed on 80 sheets, contains 3 appendices and links to a list of used literature sources from 29 titles. The paper presents 16 figures and 4 tables.

Keywords: asymmetric encryption methods, RSA, computational complexity, cryptographic system.

ЗМІСТ

СПИСОК ТЕРМІНІВ ТА СКОРОЧЕНЬ

Задача про укладання ранця – задача комбінаторної оптимізації, що полягає у визначенні для заданої множини предметів, кожен з яких має цінність та вагу, яку кількість кожного з даних предметів необхідно взяти таким чином, щоб сумарна вага ранця не перевищувала задану, а сумарна цінність була найбільшою.

DES (Data Encryption Standard) – алгоритм із симетричним ключем для шифрування цифрових даних, що було прийнято Національним Бюро Стандартів США. Незважаючи на те, що його довжина ключа в 56 біт занадто коротка та робить його небезпечним для використання у програмному забезпеченні, даний алгоритм мав великий вплив на розвиток криптографії.

Ethernet – один з найвідоміших протоколів кабельних комп'ютерних мереж, що працює на канальному та фізичному рівні мережевої моделі OSI.

SSL – криптографічний протокол, що створено для забезпечення захищеного з'єднання між сервером та клієнтом.

SSH – мережевий протокол прикладного рівня, що дозволяє дистанційно керувати операційною системою.

PGP – комп'ютерна програма, а також бібліотека функцій для виконання операцій шифрування та цифрового підпису для інформації, представленої у цифровому вигляді, а також для прозорого шифрування даних на запам'ятовуючих пристроях.

Еліптична крива – математичний об'єкт у вигляді певної множини точок, що може бути визначений над деяким довільним полем.

SCA (Side channel attacks) – клас атак заснований на побічних каналах отримання інформації про криптосистеми, використовуючи інформацію про енергоспоживання, збої у роботі та інші процеси, що відбувається під час роботи алгоритму.

Просте число – ціле додатне число, що ділиться лише на самого себе та на одиницю.

Факторизація – розкладання числа на добуток простих чисел.

НСД – найбільший спільний дільник для двох чи більше невід'ємних чисел, тобто найбільше натуральне число, на яке дані числа діляться без остачі.

LSB (Least Significant Bit) – найменший значущий біт у двійковому числі, найдальший праворуч.

MSB (Most Significant Bit) – найбільший значущий біт у двійковому числі, найдальший ліворуч.

PKCS (Public Key Cryptography Standards) – специфікація, що була розроблена компанією RSA Security спільно з розробниками систем безпеки, залучених з усього світу з метою прискорення розробки криптографії з відкритим ключем.

CPython – найбільш популярна та еталонна реалізація мови програмування Python, являє собою написаний мовою C інтерпретатор байт-коду.

PSFL (Python Software Foundation License) – BSD-подібна пермісивна ліцензія на вільне програмне забезпечення, що призначена для розповсюдження мови програмування Python та продукти, що нею написані.

MATLAB (Matrix Laboratory) – пакет прикладних програм для проведення обчислення технічних задач.

ВСТУП

Останнім часом у світовому суспільстві відзначається глобальна цифрова трансформація, що призводить до зростання обсягу приватної інформації, що транслюється через мережу Інтернет. Це призводить до підвищення ризику порушення конфіденційності даних, в наслідок чого особиста інформація стане доступна зловмисникам. Тому задача забезпечення захисту інформації є однією із найбільш пріоритетних на сьогоднішній день.

Розвиток інформаційних, телекомунікаційних та інформаційно-телекомунікаційних технологій значно ускладнює завдання збереження надійності захисту даних. Для цього необхідно підвищувати ефективність криптографічних алгоритмів, які використовуються для вирішення даного завдання.

Серед криптографічних методів одним із найпоширеніших можна назвати метод асиметричного шифрування RSA, який постає в якості стандарту для великої кількості шифрувальних додатків та сервісів. Не зважаючи на винайдення нових методів асиметричного шифрування, метод RSA залишається досить широко використовуваним у різноманітних криптографічних засобах захисту інформації сучасних інформаційно-телекомунікаційних систем.

Головною проблемою забезпечення надійності шифрування методу RSA на сьогоднішній день постає необхідність використовувати великі ключі шифрування, що залишаться недосяжними для обчислювальної потужності сучасних процесорів. Це негативно впливає на час роботи алгоритму, що реалізує метод RSA. У випадку збереження даної тенденції, у майбутньому час роботи методу RSA із ключами актуальної довжини зробить його непридатним до використання.

Більшість модифікацій криптографічного методу RSA направлені на подолання зазначеної тенденції уповільнення роботи шляхом спрощення його математичного апарату.

Для вдосконалення методу шифрування RSA необхідно виконати наступні завдання:

1. Дослідити існуючі підходи до покращення даного методу.
2. Експериментальним шляхом виокремити найбільш ефективний підхід та створити модифікацію на його основі.

У даному дослідженні пропонується алгоритмічно-програмний метод асиметричного шифрування на основі методу шифрування RSA із зменшеною обчислювальною складністю та підвищеною стійкістю до криптоаналітичних атак.

1. МЕТОДИ АСИМЕТРИЧНОГО ШИФРУВАННЯ ДАНИХ

1.1. Класичні методи асиметричного шифрування

Перші ідеї, що лягли в основу схеми асиметричного шифрування, були представлені 1976 року у роботі В. Діффі та М. Геллмана «Нові напрямки в сучасній криптографії» [1]. Вони запропонували у своїй роботі новий спосіб організації закритої комунікації без попереднього обміну ключами. Даний спосіб було названо шифруванням з відкритим ключем. В ньому використовувались різні ключі для шифрування та дешифрування, що поодинці не надають інформації один про одного. Завдяки даній властивості дозволяється відкривати ключ шифрування, оскільки це жодним чином не впливає на стійкість шифру. Закритим залишається тільки ключ дешифрування, завдяки цьому криптографічні системи з відкритим ключем називають асиметричними криптосистемами.

1.1.1. Метод Меркле-Геллмана

У 1978 році Р. Меркле та М. Геллман нову криптосистему, що використовувала перший метод асиметричного шифрування загального призначення. Дана криптосистема належить до ранцевих алгоритмів. Перший варіант методу Меркле-Геллмана був придатний лише до шифрування повідомлень, згодом вийшла модифікація А. Шаміра, що крім шифрування підтримувала функцію цифрового підпису.

Безпека ранцевих алгоритмів заснована на відомій математичній задачі про укладання ранця. В основі такого алгоритму полягає ідея шифрування повідомлення через розв’язок набору завдань стосовно укладання ранця. Оскільки розв’язати дану задачу для невеликої кількості елементів просто, для забезпечення практичної придатності ранець повинен містити понад 250 елементів, довжина кожного з яких повинна перебувати на проміжку від 200 до 400 біт, а довжина модуля алгоритму –

від 100 до 200 біт. Для отримання чисел вказаної довжини використовувались генератори випадкових чисел.

Даний підхід до створення криптосистем виявився ненадійним. Криптосистема Меркле-Геллмана була зламана Шаміром та Циппелом, які у своїх дослідженнях виявили її вразливі місця та відновили швидкозростаючу послідовність ранця. Згодом було винайдено ще декілька варіантів криптосистем на основі задачі про укладення ранця, проте жодна з них не виявилась стійкою до криптоаналітичних атак.

1.1.2. Метод RSA

У 1978 році, незадовго після створення першої асиметричної криптосистеми Меркле-Геллмана, Рон Рівест, Аді Шамір і Леонар Адлеман випустили статтю, у якій описали алгоритм із публічним ключем, що можна було використовувати для шифрування і водночас для створення цифрового підпису [2]. Даний алгоритм названо за першими літерами його авторів. Він зміг замінити використовуваний на той час менш захищений алгоритм DEC.

На сьогоднішній день криптосистема RSA використовується у великій кількості як комерційних, так і некомерційних продуктів у різних галузях. Вона використовується у операційних системах таких компаній як Microsoft, Apple, Novell і Sun. Також метод RSA застосовується в захищених телефонах, на смарт-картах та на мережних платах Ethernet, у протоколі захисту SSL, мережевому протоколі SSH та криптографічній програмі PGP.

1.1.3. Метод Діффі-Геллмана

У 1976 році В. Діффі і М. Геллман запропонували криптографічну систему з відкритим ключем, що заснована на складності обчислення дискретного логарифма [3]. Даний метод використовується для генерації

ключів, їх нормального розподілу, та в комутативних цілях, однак він не придатний до шифрування.

Також існує варіант даного методу, який було запропоновано Ш. МакКерлі та К. МакКерлі. Вони внесли використання модуля у вигляді складеного числа. Згодом М. Кобліц та Н. Кобліц розширили дану версію методу, ввівши в нього використання еліптичних кривих.

Метод Ель-Гамаля, що на відміну від методу Діффі-Геллмана придатний для шифрування та цифрового підпису, також засновано на складності обчислення дискретного логарифма. Це був перший алгоритм асиметричного шифрування, що був придатний як для повідомлень, так і для цифрових підписів, та не був обмежений патентами США.

Важливою особливістю даного методу є його непридатність до використання на лініях зв'язку, шо незахищені від модифікацій, оскільки метод Діффі-Геллмана вразливий до типу атаки «людина посередині».

Сутність алгоритму Діффі-Геллмана зводиться до наступного. Згідно даного алгоритму, учасники обміну повідомленнями А та В домовляються про значення великого простого числа p і його простого дискретного кореня – числа a. Абонент А обирає випадкове число k_a, а абонент В – випадкове число k_b таким способом, щоб виконувалися умови $1 < k_a < p - 1$ та $1 < k_b < p - 1$. Ці числа k_a та k_b зберігаються абонентами А та В у секреті. Абонент А формує відкритий ключ за допомогою використання формули

$$Y_a \equiv a^{k_a} \pmod{p}. \quad (1.1)$$

Аналогічно абонент В формує відкритий ключ, використовуючи формулу

$$Y_b \equiv a^{k_b} \pmod{p}. \quad (1.2)$$

Після обміну відкритими ключами Y_a та Y_b абоненти обчислюють значення секретного числа K:

$$K \equiv Y_a^{k_b} \pmod{p} \equiv a^{k_a k_b} \pmod{p}; \quad (1.3)$$

$$K \equiv Y_b^{k_a} (\text{mod } p) \equiv a^{k_b k_a} (\text{mod } p). \quad (1.4)$$

Отримане число K для потенційного зловмисника є секретним, тому що розв'язання рівнянь Y_a та Y_b для великих чисел є неможливим.

Алгоритм Діффі-Геллмана можна використовувати у випадку з трьома і більше учасниками.

1.1.4. Криптосистеми на еліптичних кривих

Криптосистеми, що засновані на використанні еліптичних кривих, належать до класу асиметричних криптосистем [4]. Їх безпека здебільшого ґрунтується на складності розв'язку задачі дискретного логарифмування у групі точок еліптичної кривої, що проходить над скінченним полем. Така система зумовлює високий рівень криптостійкості.

Окремі види криптосистем на еліптичних кривих основані на складності розкладання великих цілих чисел у випадку, коли еліптична крива задана над скінченним полем за складеним модулем, проте такий тип системи зустрічаються досить рідко.

Для використання у криптографії, як правило, обираються скінченні поля. Для точок, що знаходяться на еліптичній кривій, вводиться операція додавання, що відіграє ту саму роль, що й операція множення у методах RSA та Ель-Гамаля.

Найбільш значною перевагою криптосистем, що використовують еліптичні криві, є високий показник швидкості опрацювання інформації, що, однак, досягається лише при використанні алгоритмів швидкого обчислення.

Також підвищений рівень криптостійкості даних систем дозволяє використовувати ключі шифрування меншої довжини порівняно до інших типів асиметричних криптосистем. Це пояснюється тим, що для обчислення обернених функцій на еліптичних кривих можна використати лише алгоритми з експоненціальним зростанням обчислювальної

складності, а для інших асиметричних криптосистем придатні також субекспоненціальні методи.

Криптосистеми на еліптичних кривих є найбільш розповсюдженим способом для створення цифрового підпису, а їх використання було внесено у стандарти по всьому світу в кінці 90-х років.

Безпека даних систем спирається не лише на стійкість алгоритму з використанням еліптичних кривих, а ще й на стійкість використовуваної геш-функції, що використовується для підготовки документів до процедури шифрування шляхом зменшення їх об’єму.

1.2. Класичний метод асиметричного шифрування RSA

Для подальшого дослідження було обрано метод асиметричного шифрування RSA, оскільки даний метод є одним з найбільш стійких. Також він є найбільш розповсюдженим, тож створена на його основі модифікація буде сумісна з класичним методом RSA.

Метод RSA було створено саме вчасно, перед початком активного розповсюдження використання електронної пошти. RSA втілював наступні важливі ідеї:

1. Шифрування з публічним ключем. Даний принцип позбавляє необхідності у наявності окремого захищеного каналу, через який будуть пересилатись ключі користувачів перед початком обміну повідомленнями, оскільки у методі RSA закриті ключі не потребують пересилання.
2. Цифровий підпис. У користувача може виникнути необхідність перевірити особистість відправника. Для цього відправник ставить свій цифровий підпис за допомогою закритого ключа дешифрування, а для перевірки підпису використовується відкритий ключ. Завдяки цьому цифрові підписи неможливо підробити. Також користувач у майбутньому не зможе заперечити

факт підписання ним повідомлення. Дана властивість також використовується для грошових переказів.

Обмін повідомленнями відбувається за наступною схемою: нехай Аліса хоче надіслати Бобу повідомлення. Для цього Боб спочатку створює на своєму боці унікальні замок і ключ до нього (відкритий і закритий ключі шифрування відповідно). Далі Боб поширює свій замок у мережу, щоб будь-який користувач, що має намір відправити йому повідомлення, зміг його закрити. Оскільки ключ від замка знаходиться лише у Боба, ніхто закрите повідомлення прочитати не зможе. Аліса, отримавши замок Боба, закриває ним своє повідомлення та надсилає його Бобу, який, в свою чергу, відкриває повідомлення своїм ключем. За даною схемою влаштовані всі асиметричні криптосистеми. Схема обміну повідомленнями у асиметричній криптосистемі зображена на рис. 1.1.

Рис. 1.1. Схема обміну повідомленнями у асиметричній криптосистемі

Для створення цифрового підпису Боб шифрує своє повідомлення закритим ключем, який в даному випадку використовується у якості унікального ідентифікатора Боба. Далі Аліса, отримавши відкрите повідомлення і його підпис, розшифровує цей підпис за допомогою відкритого ключа. Якщо отриманий результат співпадає з повідомленням, підпис вважається вірним.

Для спрощення створення підпису документів вони спершу оброблюються геш-функцією.

1.2.1. Математична база методу RSA

Метод шифрування RSA починається з вибору відкритого ключа, що складається з пари цілих позитивних чисел (e, n). Для цього потрібно у довільний спосіб обрати два великі прості числа p і q, що зберігаються в секреті. Число n обчислюється як добуток даних чисел, тобто $n = pq$. Також обчислюється функція Ейлера:

$$\varphi(n) = (p - 1)(q - 1). \tag{1.5}$$

Далі у довільний спосіб обирається просте число e – ключ зашифровування, що задовольняє умови $e < \varphi(n)$; НСД $(e, \varphi(n)) = 1$. Ключем розшифровування постає пара чисел (d, n), число d є оберненим до числа e, тобто:

$$d \equiv e^{-1} (\text{mod } \varphi(n)). \tag{1.6}$$

Для зашифрування повідомлення воно повинне бути представлене у якості цілого числа величиною від 0 до $n - 1$. Якщо повідомлення представлене числом більшого розміру, воно розбивається на цифрові блоки, які шифруються окремо. Тобто якщо n має розмір 256 розрядів, то кожен із цифрових блоків повідомлення повинен бути довжиною не більше як 255 розрядів. Ту саму довжину будуть мати цифрові блоки після шифрування завдяки модульній конгруенції.

Щоб отримати шифротекст C, повідомлення M шифрується за допомогою операції піднесення до степеня e по модулю n:

$$C \equiv M^e \ (\text{mod } n). \tag{1.7}$$

Дешифрування відбувається, згідно з теоремою Ейлера, за допомогою операції піднесення до степеня d по модулю n:

$$M \equiv C^d \ (\text{mod } n). \tag{1.8}$$

Схема генерації ключів, шифрування та дешифрування представлена на рис. 1.2.

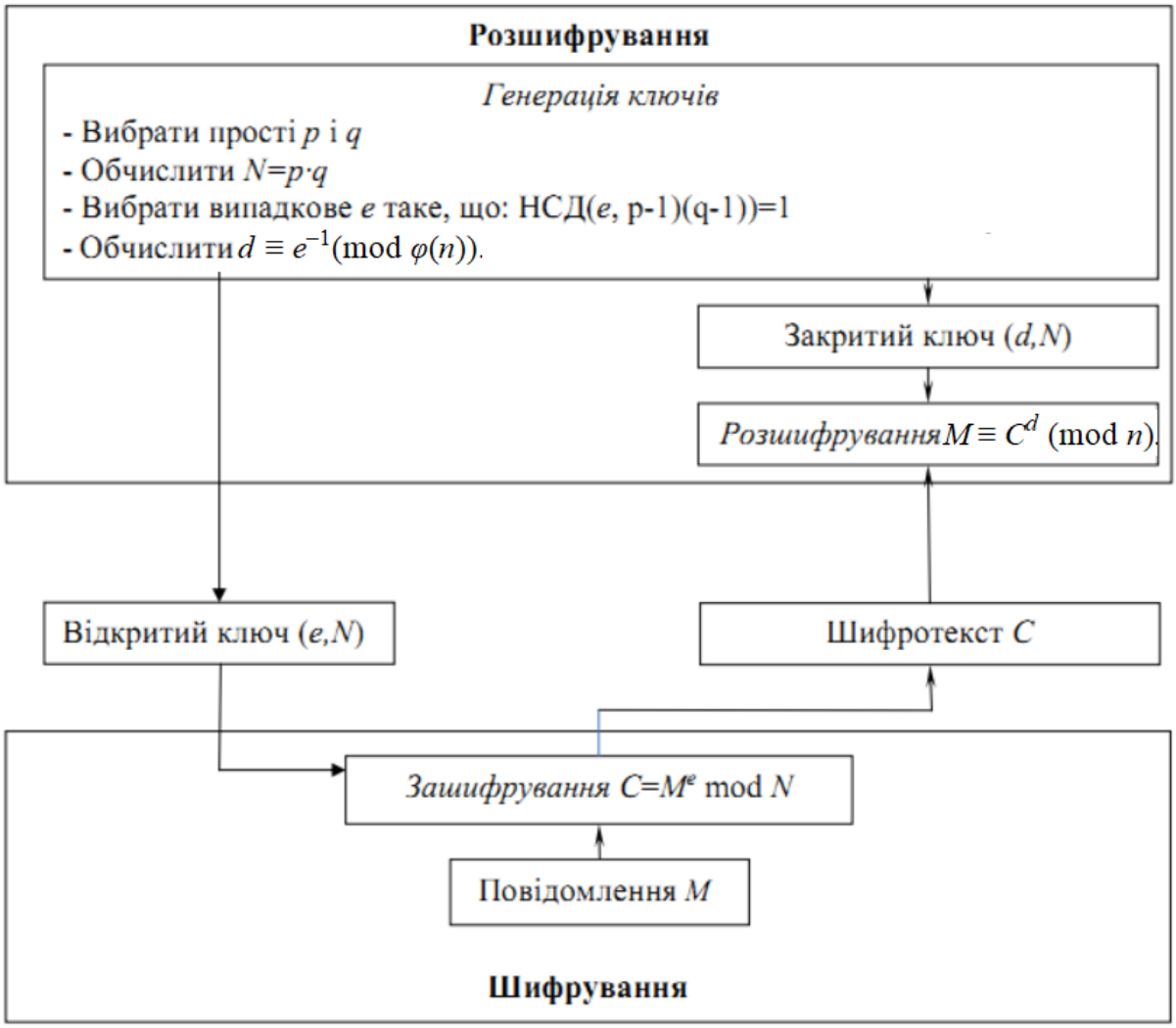


Рис. 1.2. Схема роботи методу RSA

1.2.2. Алгоритм виконання методу RSA

У методі RSA піднесення у степінь відбувається за допомогою повторення операцій піднесення у степінь квадрату та ділення. Виконується наступна послідовність дій:

1. Нехай бінарне представлення e виглядає як $e_k e_{k-1} \ldots e_1 e_0$.
2. Визначається змінна $C = 1$.
3. Для $i = k, k-1, \ldots, 0$ повторюються кроки:
   - 3.1. C присвоюється залишок від ділення C^2 на n.
   - 3.2. Якщо $e_1 = 1$, то C присвоюється значення, рівне залишку від ділення добутку $C \times M$ на n.

4. Отримується зашифроване повідомлення C.

Для зашифрування повідомлення, тобто для операції $M^e \pmod n$, потрібно не більше як $2\log_2(e)$ операцій ділення. Даний показник визначає необхідний час проведення відповідних операцій. Час шифрування одного блоку зростає не швидше ніж квадрат довжини числа n.

1.2.3. Оцінка безпеки методу RSA

Складність атаки на метод RSA базується на трудомісткості рішення проблеми факторизації числа n, тобто його розкладання на множники. На даний час досі не знайдено ефективного рішення обчислювальної задачі факторизації. Це забезпечує статус надійності алгоритму RSA, проте твердження про його залежність від задачі факторизації великих чисел є гіпотетичним, оскільки теоретично не виключена можливість існування алгоритму факторизації із поліноміальною складністю на звичайному комп'ютері. На даний момент найшвидший алгоритм факторизації довільних цілих додатних чисел має субекспоненціальну оцінку часу. Це означає, що теоретично кількість його кроків до завершення значно більша, ніж значення поліному $\ln n$. Така властивість забезпечує досягання прийнятного рівня безпеки у протоколі шифрування RSA при використанні числа n відносно невеликого розміру, що надзвичайно важливо для його практичного використання [5].

Числа, які використовуються у якості n називають RSA-числами [6]. Це множина великих напівпростих чисел, тобто чисел, представлених у вигляді добутку двох простих чисел. Актуальна довжина числа n дорівнює 2048 біт або більше.

Останній досягнутий рекорд факторизації було встановлено 12 грудня 2009 року. Тоді вдалося розкласти на прості множники число довжиною 768 біт, тобто 232 десяткових знаки, використовуючи алгоритм решета числового поля. Загальний витрачений на дану роботу час складає два з половиною роки. Обчислення виконувались розподілено на сотнях

комп’ютерів, що виконали понад 10^{20} операцій. Це еквівалентно двом тисячам років обчислень на процесорі AMD Opteron 2.2 ГГц.

Збільшення довжини ключа призводить до значного уповільнення роботи алгоритму, тому в залежності від потреб в безпеці і швидкодії та специфіки певної задачі, що висуваються до алгоритму.

1.3. Криптоаналіз методу асиметричного шифрування RSA

На сьогоднішній день проведено велику кількість досліджень на тему потенційно ефективних методів криптографічного аналізу методу RSA. Це питання регулярно включене в список обговорюваних тем на таких авторитетних щорічних криптографічних наукових конференціях, як, наприклад, EUROCRYPT, ASIACRYPT і CRYPTO.

1.3.1. Методи факторизації для криптоаналізу RSA

Одним із пріоритетних напрямків для вирішення задачі криптоаналізу методу RSA є застосування обчислювальних методів факторизації [7]. Серед загальної множини таких методів виділяють чотири основні:

- метод Ферма – використовується для знаходження ключів, які мають дуже близькі один до одного множники p і q;
- P-метод Полларда – застосовується перед використанням потужніших алгоритмів факторизації з метою відокремити невеликі прості дільники ключа;
- метод решета числового поля – найбільш ефективний метод факторизації чисел розміром понад 110 знаків у десятковій системі, за допомогою якого були факторизовано найбільші на даний момент числа;
- метод квадратичного решета – вважається наступним за швидкістю після методу решета числового поля, значно простіший за нього.

Методи решета числового поля та квадратичного решета базуються на знаходженні гладких чисел відносно порядку $\sqrt{n}$. Для підвищення швидкості обчислень вони відбуваються у числових полях, що ускладнює даний алгоритм.

1.3.2. Непрямі методи криптоаналізу RSA

Через високу складність математичного апарату криптографічних алгоритмів було винайдено інший підхід до атаки на них – використання так званих непрямих методів. Такі методи атаки, переважно, експлуатують різноманітні вразливості криптографічних алгоритмів, що виникають при некоректній реалізації, наприклад:

- неправомірне використання системи;
- неправильний вибір приватної або публічної експоненти;
- послідовності у шифрованому тексті.

Прикладом подібної атаки є атака Вінера [8], що використовує наближені дроби. Дана атака була запропонована М. Вінером для зламу методу RSA при малому значенні закритого ключа d. Він стверджував, що при виконанні даної умови, а саме:

$$d < \frac{1}{3} n^{\frac{1}{4}}, \tag{1.9}$$

закритий ключ може бути знайдено за лінійний час.

Атака виконується наступним чином:

1. Спочатку $\frac{e}{n}$ розкладається у ланцюговий дріб $[a_1, a_2 \ldots]$.
2. Для ланцюгового дробу $[a_1, a_2 \ldots]$ знаходиться множина всіх можливих дробів $\frac{k_n}{d_n}$.
3. Дослідити підходящий дріб $\frac{k_n}{d_n}$:
   3.1. Визначити можливе значення $\varphi(n)$, обчисливши $f_n = \frac{ed_n - 1}{k_n}$.

3.2. Вирішити рівняння $x^2 - ((n - f_n) + 1)x + n = 0$, отримавши пару коренів (p_n, q_n).

4. Якщо добуток отриманих коренів (p_n, q_n) дорівнює значенню n, то атака успішна, і далі можна легко отримати закритий ключ d. В іншому разі обирається інший дріб $\frac{k_{n+1}}{d_{n+1}}$, для якого виконується крок 3.

Час роботи алгоритму Вінера O(log(n)), тобто закритий ключ відновлюється за лінійний час, що залежить від довжини n.

Ще один вартий уваги ряд алгоритмів, для яких необхідно знання декількох найбільш або найменш значущих бітів закритого ключа. Припускається, що криптоаналітику відоме деяке d_0 для певного відомого повідомлення M таке що $d \equiv d_0 \bmod M$. У такому разі ми можемо записати закритий ключ як

$$d = d_1 M + d_0, \tag{1.10}$$

де єдиним невідомим є d_1. Далі покладається $M = n^{\beta - \delta}$, тоді невідома частина закритого ключа d_1 обмежується як $|d_1| < n^{\beta}$.

Аналогічний метод застосовується при наявності знань про найбільш або найменш значущі біти відкритого ключа.

Відомі приклади успішних атак на метод RSA переважно використовували саме непрямі методи атаки, проте в загальному випадку вони не є ефективнішими за задачу криптоаналізу, що заснована на математичній проблемі складності виконання факторизації великих чисел, що складають криптомодуль методу RSA.

1.3.3. Методи криптоаналізу RSA по стороннім каналам

Ще один клас атак заснований на побічних каналах отримання інформації про криптосистеми, використовуючи інформацію про енергоспоживання, збої у роботі та інші процеси, що відбувається під час роботи алгоритму. Вони дістали назву SCA (Side channel attacks) [9].

Сторонні або приховані канали являють містять у собі два або більше процеси, що пов’язані між собою через певний загальний ресурс, на який вони мають вплив. Зловмисники експлуатують такі канали для обминання захисту операційної системи. На відміну від інших криптоаналітичних методів, SCA аналізує не методи шифрування, а їх реалізацію. Таким чином, вони відстежують кореляцію між характеристиками, такими як споживання енергії та час роботи алгоритму, та внутрішнім станом процесів, пов’язаних із закритим ключем, всередині шифрувального пристрою. Такі атаки набагато простіші та ефективніші за традиційний криптоаналіз, заснований на математичному аналізі.

Одною з перших винайдених атак SCA є часова атака, або атака синхронізації [10]. Вона має місце, коли із стороннього каналу отримується інформація про кількість часу, затрачену на виконання певної операції. Нехай криптоаналітик робить часову атаку на функцію перевірки паролю, реалізовану через звичайне порівняння рядків. Він спостерігає, як система завантажує введений рядок у пам’ять та послідовно перевіряє кожен байт. Число порівнянь, виконуваних функцією, напряму пов’язане з кількістю співпадаючих байт у рядку.

Часові атаки на криптографічні алгоритми, такі як RSA, базуються на тому, що обчислювально затратні криптографічні функції виконуються за змінний час.

Ще одна атака SCA заснована на аналізі потужності. Стороннім каналом, який використовується для такої атаки, виступає струм, що тече через транзисторні переходи, в той час як даний транзистор увімкнено. Таким чином, криптоаналітик може спостерігати за змінами стану роботи шифрувального пристрою через загальну зміну кількості спожитого струму, оскільки транзистори у відкритому стані збільшують кількість струму, а транзистори у закритому стані – навпаки зменшують. Така інформація про зміни та закономірності у спожитій кількості

електроенергії дозволяє отримати детальне представлення низького рівня даних про процеси всередині шифрувального пристрою.

Для атаки на алгоритм RSA аналізують модульне піднесення у степінь. Один із найбільш розповсюджених алгоритмів піднесення у степінь – це піднесення у квадрат і множення, де степінь зчитується побітово зліва направо. Тоді, починаючи з регістра, де встановлено 1, операція піднесення у квадрат виконується до першого біта, що дорівнює 0, а далі виконується операція множення. Таким чином, у спожитій електроенергії можна проспостерігати серію операцій, що відокремлені одна від одної спадними піками. Низьке споживання електроенергії відповідає операціям піднесення у квадрат, а високе споживання – множенню. При проведенні аналогії між отриманим сигналом і залежністю вхідного біта та відповідної операції, можна визначити реакцію на експоненту. На рис. 1.3 зображено сигнал, отриманий в результаті виконання подібної атаки.

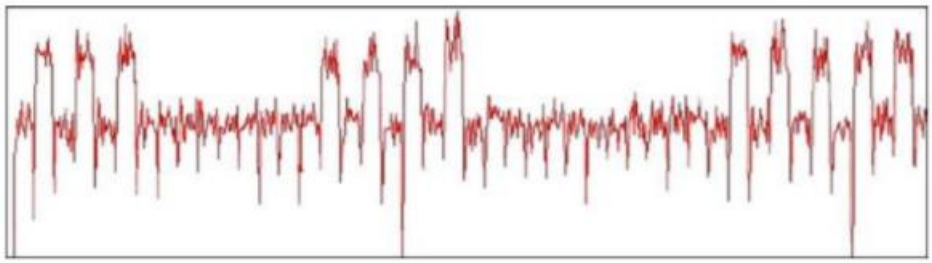


Рис. 1.3. Сигнал споживання електроенергії при піднесенні у степінь

Кількість помилок, що виникають в процесі роботи шифрувального модуля, також є досить інформативним стороннім каналом, що використовують криптоаналітики. При застосуванні такого виду атаки аналізується два види помилок:

- обчислювальні помилки, що виникають внаслідок неправильної реалізації криптографічного модуля, або ж викликані ззовні маніпулюванням напругою;
- помилки, що виникли внаслідок навмисної відправки пошкоджених вхідних даних у криптографічний модуль, що атакується.

Схожим типом атаки є електромагнітний аналіз. Для його використання криптоаналітику необхідно замість захоплення енергоспоживання шифрувального пристрою вимірювати його електромагнітні еманації. Як правило, такий аналіз дає більш детальну інформацію, ніж аналіз потужності.

1.4. Модифікації методу асиметричного шифрування RSA

Після того, як світ побачив односторонні функції та їх використання у методі RSA, почали з’являтись модифікації даного методу.

1.4.1. Модифікація RSA з використанням Китайської теореми про лишки

У 1982 році Дж.-Дж. Квісквотер та К. Коуврер запропонували нову схему розшифрування для методу RSA, яку було названо метод Квісквотера-Коуврера. У даному методі використовується наслідок з Китайської теореми про лишки [11].

При дешифруванні (або підписуванні) повідомлення, використовуючи алгоритм RSA, необхідно провести операцію піднесення у степінь по модулю n. Метод Квісквотера-Коуврера дозволяє скорочувати кількість таких операцій, виконуючи замість операції за формулою (1.11) дві аналогічні:

$$\begin{aligned} m_p &\equiv C^d \pmod{p}, \\ m_q &\equiv C^d \pmod{q}. \end{aligned} \tag{1.11}$$

Оскільки числа p і q мають у 2 рази менший порядок, ніж число n, набагато швидше зробити 2 піднесення у степінь по модулю $\frac{n}{2}$, ніж одне піднесення у степінь по модулю n. В кінці необхідно відновити повідомлення M, використовуючи m_p і m_q, що і робиться за допомогою наслідку з Китайської теореми про лишки.

Формулюється теорема наступним чином. Якщо натуральні числа $a_1, a_2,\ldots,a_n$ попарно взаємно прості, то для будь-яких цілих $r_1, r_2,\ldots,r_n$, таких, що $0 \le r_i \le a_i$ для всіх $i \in \{1, 2,\ldots,n\}$, знайдеться число N, яке при діленні на a_i дає залишок r_i при всіх $i \in \{1, 2,\ldots,n\}$. Крім того, якщо знайдуться два такі числа N_1 та N_2, що задовольняють попередню умову, то $N_1 \equiv N_2 \pmod{\prod_{i=1}^{n} a_i}$.

Дешифрування із використанням методу Квісквотера-Коуврера дає приріст у швидкості приблизно у 4 рази.

1.4.2. Модифікація RSA з використанням еліптичних кривих

Окремий клас асиметричних алгоритмів використовує еліптичні криві над певним довільним полем – в множині точок (x, y), що задовольняють рівняння:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1.12)$$

Для точок, що знаходяться на зазначеній еліптичній кривій, вводять операцію додавання, що виступає у якості аналога операції множення у криптоалгоритмі RSA.

Першу роботу у даному напрямі випустили у 1991 році К. Кояма, Ю.М. Моркер, Т. Окамото та С.А. Ванстоун запропонували модифікацію методу RSA на еліптичних кривих над кільцем Z_n, яку пізніше стали називати *KMOV* [12]. Вони створили три нових класи односторонніх функцій, кожен з яких призначений для різних задач. Перший клас не придатний для використання у криптосистемі з відкритим ключем, а лише для створення цифрового підпису. Другий клас односторонніх функцій

позбавлений даної проблеми та слугує альтернативою методу RSA. Третій клас подібний до криптосистеми Рабіна. Вони також використовують модуль $n = pq$, а стійкість запропонованих криптоалгоритмів також базується на трудомісткості рішення проблеми факторизації числа n. Серед переваг запропонованих криптоалгоритмів зазначалось підвищення стійкості до атак з невеликим ключем (атака Вінера), хоча дане твердження пізніше було спростовано експериментально. Серед недоліків варто зазначити уповільнення роботи алгоритму порівняно із показниками ефективності методу RSA.

Згодом з'явилось більше асиметричних криптоалгоритмів, заснованих на використанні еліптичних кривих. У 1993 році Н. Демитко запропонував покращення криптосистеми *KMOV* [13]. Повідомлення виступало у якості першої координати точки $M(x, y)$, що належить кривій

$$y^2 \equiv (x^3 + ax + b) \pmod{n}. \qquad (1.13)$$

Шифрування виконується шляхом e-кратного додавання точки M, а перша координата точки C визначає шифротекст. Закритий ключ d обирається в залежності від символів Лежандра $\left(\frac{z}{p}\right)$ і $\left(\frac{z}{q}\right)$, де

$$z \equiv (x_c^3 + ax_c + b) \pmod{n}. \qquad (1.14)$$

Для прискорення розшифрування також можливе використання Китайської теореми про лишки.

У 1995 році К. Кояма описав ще одну криптосистему, що була побудована на сингулярній еліптичній кривій, що на етапі дешифрування давала виграш у часі удвічі порівняно з методом RSA[14].

На початку 21 століття криптосистеми із використанням еліптичних кривих розвились настільки, що увійшли у світові стандарти безпеки. Велика кількість досліджень засвідчують перевагу криптосистем з використанням еліптичних кривих над іншими криптосистеми з відкритим ключем як з точки зору ефективності роботи алгоритму, так і з точки зору міри захищеності пропорційно до довжини ключа.

1.4.3. Модифікація RSA з використанням n-розширення

У 1997 році Т. Такагі створив модифікацію методу RSA, використовуючи *n*-розширення [15]. Модуль обчислюється як

$$n^k = (pq)^k, \tag{1.11}$$

в якому p і q це довільні великі прості числа, а число k становить кількість блоків, на які розбивається повідомлення M: $M_0, M_1, \ldots, M_{k-1}$, де всі числа $M_i < n$, $i \in [0, k-1]$.

Шифрування відбувається аналогічно до методу RSA, як і розшифрування першого блоку. Для всіх наступних блоків створюються лінійні рівняння, час для розв'язання яких співпадає з часом звичайного дешифрування у методі RSA. Це пояснюється твердженням про те, що якщо на множині відкритих текстів M задано рівномірний розподіл, то складність знаходження відкритого тексту за відомим шифротекстом аналогічна складності криптосистеми RSA.

У 1998 році Т. Такагі створив нову модифікацію [16], в якій в якості модуля обчислювалось число

$$n = p^k q. \tag{1.12}$$

Довільні великі прості числа p і q обираються таким чином, щоб застосувати алгоритми факторизації решета числового поля було неможливо.

Шифрування відбувається аналогічно до методу RSA, а розшифрування – частинами, тобто спочатку окремо обчислюється частина повідомлення за модулем p^k, використовуючи алгоритм Такагі для *n*-розширення, а друга частина обчислюється за модулем q. У кінці застосовується наслідок за Китайської теореми про лишки.

Для невеликих значень модуля n така криптосистема швидша за RSA приблизно у 1,5 рази.

У 2000 році С. Лім, С. Кім, І. Йие та Х. Лі [17] доповнили модифікацію Такагі використанням модуля

$$n = p^k q^l. \quad (1.13)$$

У разі якщо числа k та l близькі за значенням, складність розшифрування наближається до мінімального значення. Для забезпечення стійкості такої криптосистеми необхідно, щоб НСД(k, l) = 1.

1.4.4. Модифікація RSA з використанням складеного модуля

У 1998 році Т. Коллінз, Д. Хопкінс, С. Лангфорд та М. Сабін створили патент на використання складеного модуля в класичному методі RSA [18]. Такий модуль обчислюється як добуток більше ніж двох простих чисел, таким чином для створення модуля певної довжини можна генерувати прості числа меншої розрядності, що дає значний виграш у часі на етапі формування ключів без втрати стійкості криптоалгоритму. Автори досліджували використання модуля із трьох та чотирьох простих чисел.

1.4.5. Методи модифікацій RSA для протидії криптоаналізу по стороннім каналам

Для протидії криптоаналітичних атак по стороннім каналам було введено певну кількість методів, що здатні забезпечити стійкість криптографічних алгоритмів від такого типу атак. Виділяють декілька загальних підходів до протидії:

- введення часових зсувів, станів очікування, зміна порядку виконання операцій та встановлення фіктивних інструкцій для алгоритму;
- перепроєктування асемблерних інструкцій виконання математичних операцій та перезаписів пам’яті з метою приховати зміни у споживанні електроенергії;
- введення алгоритмічних змін, таких як маскування, приховуючи згенеровані ключі довільною маскою окремо для кожного циклу шифрування.

Серед усіх видів протидій техніки алгоритмічних змін є найбільш ефективними, потужними та універсальними. Окрім того, такий спосіб підвищення криптографічної стійкості є найпростішим та найдешевшим у реалізації. Виділяють дві групи методів протидії криптоаналітичних атак по стороннім каналам:

1. Програмні методи, що включають рандомізацію послідовностей виконання запрограмованих інструкцій алгоритму, розщеплення біт даних, та додавання фіктивних інструкцій до алгоритму.
2. Апаратні методи, що включають рандомізацію виконавчого часу та споживання електроенергії (або її компенсація), а також рандомізація виконання регістрів.

Програмні методи протидії криптоаналітичних атак по стороннім каналам значно ускладнюють криптографічні алгоритми, особливо з точки зору затрат часу, тому масштаб їх впровадження у криптосистему залежить від значення даних, що захищаються, у кожному конкретному продукті. При розробці переважно обирається комбінація програмних та апаратних методів для зменшення впливу їх недоліків на криптосистему.

Найбільш розповсюджений метод захисту від атак SCA – певним чином рандомізувати дані про час виконання, спожиту потужність та електромагнітні випромінювання. Особливо практичним для використання даний метод є у криптосистемах, що використовують еліптичні криві, де є можливість довільним чином рандомізувати проективні координати поля еліптичних кривих. Також використовується стійкий до атак SCA метод скалярного множення, що може приймати довільну кількість попередньо обчислених точок. Є три стандартні методи рандомізації:

1. Маскування базової точки довільною.
2. Закритий скаляр рандомізується множником порядку еліптичної кривої.
3. Базова точка рандомізується проективними координатами.

При проєктуванні криптографічних систем також уникають процедур з використанням проміжних ключів та секретів, що маскує багато інформації із сторонніх каналів. Критичний код не повинен містити умовні вирази та елементарні операції такі як І, АБО та сума по модулю два, а також операції відгалуження. Наявність умовно виконуваного коду, що залежить від вхідних даних, легко розкриває їх властивості, тому необхідно, щоб код виконувався незалежно від вхідних даних. В такому разі дані про затрачені на виконання алгоритму час і кількість електроенергії втратить інформативність.

Ще одним методом захисту від атак SCA є так звана «сліпота» – метод, що передає обчислення математичних функцій певному провайдеру, що не має доступу до вхідних та вихідних даних. Така концепція дозволяє користувачу не обчислювати математичні функції самостійно. Метод використовує гомоморфні властивості цифрового підпису RSA.

Додавання затримок виконується для протидії часовому аналізу з метою виконання всіх операцій за однаковий час [19]. Такий метод дуже складно реалізувати, оскільки при використанні таймеру все ще можливо відстежити відклик системи на запити. Крім того, час виконання всіх операції потрібно прирівняти до найповільнішого у алгоритмі. Тож такий метод вважається неефективним. Додавання випадкових затримок також підвищує кількість необхідних шифротекстів, які криптоаналітик зможе компенсувати збільшенням кількості заміряних та проаналізованих даних. Кількість таких зразків для успішного результату необхідно збільшувати приблизно на квадрат синхронізації шуму, що хоч і ускладнює криптоаналіз, проте не унеможливлює його.

Техніка балансування потужності не завжди можливо реалізувати, оскільки вона вимагає апаратного додавання фіктивних регістрів, що виконують довільні незначущі операції, утримуючи споживання електроенергії на певному константному рівні. Така техніка надійно

захищає від всіх типів аналізу споживання струму. Серед недоліків варто зазначити підвищення споживання електроенергії.

На елементарному рівні протидія атакам на основі аналізу потужності базується на використанні логічних елементів, що використовують константне споживання енергії, що не залежить від виконуваних операції і оброблюваних даних

1.5. Висновки

В даному розділі було розглянуто роботу методів асиметричного шифрування, таких як метод Меркле-Геллмана, метод RSA, метод Діффі-Геллмана та метод із використанням еліптичних кривих. Для подальшого аналізу було обрано метод RSA.

Далі було проаналізовано математичний апарат методу асиметричного шифрування RSA, проаналізовано його криптостійкість та ефективність використання. Також було проаналізовано вразливості та слабкі місця даного методу, способи криптоаналітичних атак, до яких він вразливий.

Далі було розглянуто існуючі методи модифікацій методу RSA, що були направлені на покращення часових показників роботи алгоритму або на посилення його стійкості до окремих видів криптоаналітичних атак.

Найбільш перспективними для подальшого застосування при створенні нових модифікацій асиметричних методів шифрування виглядають:

- метод, що використовує наслідок з Китайської теореми про лишки;
- метод, що використовує складений модуль.

Дані методи було обрано, оскільки саме вони дають найкращий приріст швидкості у роботі алгоритму, при цьому мінімізуючи його ускладнення.

2. РОЗРОБЛЕННЯ АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ АСИМЕТРИЧНОГО ШИФРУВАННЯ ДАНИХ

2.1. Запропонований метод асиметричного шифрування

В основі запропонованого методу, що має на меті покращити показники швидкості роботи порівняно з існуючими реалізаціями без втрати криптостійкості, полягає класичний метод асиметричного шифрування RSA у поєднанні із модифікаціями, що використовують складений модуль та наслідок з Китайської теореми про лишки. Загальна схема роботи запропонованого методу асиметричного шифрування представлена на рис. 2.1.

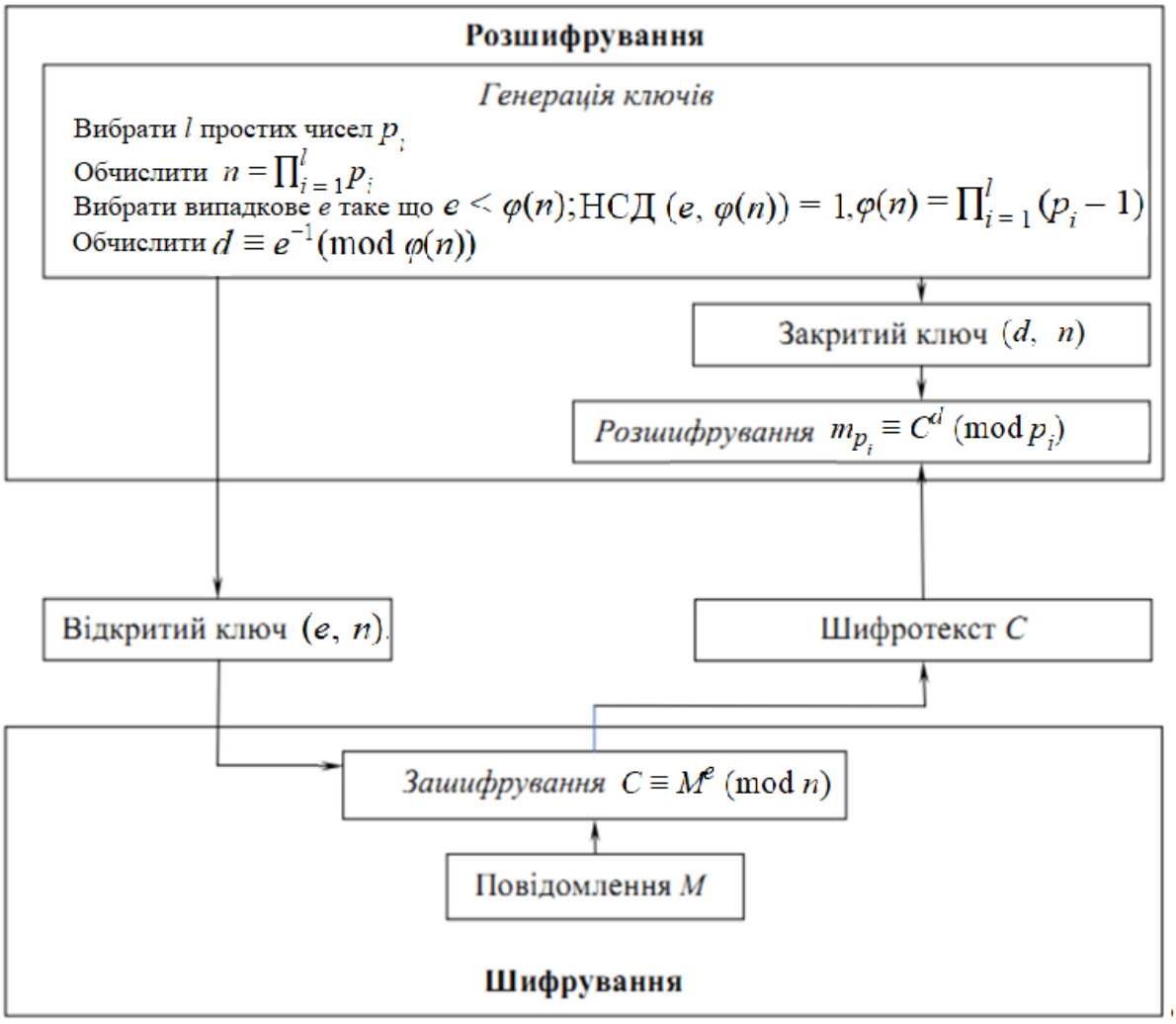


Рис. 2.1. Схема роботи запропонованого методу шифрування

2.1.1. Етап генерації ключів

Відкритий ключ складається з пари цілих позитивних чисел (e, n). Для отримання числа n довільним чином генерується l великих простих чисел p_i та обчислюється їх добуток:

$$n = \prod_{i=1}^{l} p_i \tag{2.1}$$

Також обчислюється функція Ейлера:

$$\varphi(n) = \prod_{i=1}^{l} (p_i - 1). \tag{2.2}$$

Кількість великих простих чисел l обмежена як $3 \le l \le \frac{n}{4}$, тобто розрядність кожного окремого множника не менше 4 знаків, оскільки в протилежному випадку алгоритм генерації простих чисел із використанням тесту Міллера-Рабіна не спрацює. Оптимальна кількість простих чисел буде визначена після експериментального дослідження поведінки запропонованого методу шифрування на предмет швидкості виконання та криптостійкості. Також для ускладнення атак пропонується використати метод рандомізації та використовувати змінну кількість простих чисел, які відрізняються довжиною на декілька розрядів, що буде довільно визначатись у певному діапазоні для кожного випадку шифрування. Таким чином, криптоаналітик не буде заздалегідь знати, скільки простих чисел йому необхідно обчислити для повного розкриття приватної експоненти.

Далі у довільний спосіб обирається просте число e – ключ зашифровування, або публічна експонента, яке задовольняє умови $e < \varphi(n)$; НСД (e, $\varphi(n)$) = 1. Ключем розшифровування постає пара чисел (d, n), число d, тобто приватна експонента, є оберненим до числа e, тобто:

$$d \equiv e^{-1} (\text{mod } \varphi(n)). \tag{2.3}$$

2.1.2. Етап шифрування та дешифрування

Щоб отримати шифротекст C, повідомлення M шифрується за допомогою операції піднесення до степеня e по модулю n:

$$C \equiv M^e \pmod{n}. \tag{2.4}$$

Дешифрування відбувається, згідно з теоремою Ейлера, за допомогою операції піднесення до степеня d по модулю p_i для всіх $i \in [1; l]$:

$$m_{p_i} \equiv C^d \pmod{p_i} = (\mathrm{C} \bmod p_i)^{d \bmod (p_i - 1)} \bmod p_i. \tag{2.5}$$

Для підвищення швидкості обчислень використовується теорема Ферма.

В кінці необхідно відновити повідомлення M, використовуючи m_{p_i} за допомогою Китайської теореми про лишки. Для цього потрібно вирішити систему рівнянь наступного вигляду:

$$\begin{cases} M \equiv m_{p_1} (\bmod p_1) \\ M \equiv m_{p_2} (\bmod p_2) \\ \dots \\ M \equiv m_{p_l} (\bmod p_l) \end{cases} \tag{2.6}$$

2.1.3. Аналіз переваг та недоліків запропонованого методу асиметричного шифрування

Перевагами даної криптографічної моделі є:

1. Обчислювальна складність. Використовуючи наслідок з Китайської теореми про лишки та виконуючи паралельні обчислення, кількість бітових операцій, необхідних для розшифрування зашифрованого тексту, не перевищує $\frac{3}{2l^3} (\log_2 n)^3$. Отже, час, необхідний для дешифрування, зменшується з кожним додатковим простим числом у модулі. Також збільшення кількості простих чисел спрощує їх генерацію, таким чином зменшуючи обчислювальну складність алгоритму на етапі генерації ключів шифрування.
2. Використання пам’яті. Використовуючи наслідок з Китайської теореми про лишки, обсяг використовуваної пам’яті, необхідний

для всіх обчислень дешифрування до останнього кроку рекомбінації, вимагає лише $\log_2 p_l$ біт, де p_l це найбільше просте число у модулі. Якщо всі прості числа об'ємом приблизно у $\frac{\log_2 n}{l}$ біт, тобто є збалансованими, то обсяг використовуваної пам'яті зменшується з кожним додатковим простим числом.

Серед недоліків варто зазначити те, що збільшення кількості простих множників у модулі полегшує його факторизацію. Тому для практичності використання необхідно визначити оптимальну кількість множників, досягши компромісу між обчислювальною складністю алгоритму та стійкістю до атак методами факторизації.

2.2. Аналіз криптостійкості запропонованого методу асиметричного шифрування

Далі буде розглянуто поведінку основних криптоаналітичних методів при проведенні атаки на запропонований метод асиметричного шифрування. Всі зазначені атаки виконуються для модулів із збалансованими простими числами.

2.2.1. Аналіз складності факторизації в залежності від кількості множників

Для порівняння складності факторизації звичайного псевдопростого RSA-числа, тобто такого, що є добутком двох простих чисел, із числом, що є добутком трьох і більше чисел, приймемо ймовірність факторизації як 1, тоді ймовірність знаходження одного із двох простих множників буде дорівнювати 0,5 [20].

У випадку складеного множника для $l > 2$ ймовірність знаходження одного числа буде дорівнювати $1 - \frac{1}{2^{l-1}}$. Тобто із збільшенням кількості множників ймовірність знайти один наближається до 1, а криптостійкість для модуля фіксованого розміру падає вдвічі порівняно з класичним

методом RSA. Тоді для збереження того ж рівня криптостійкості при збільшенні на одне просте число у модулі необхідно подвоювати довжину модуля. Дане твердження застосовне при проведенні атак грубою силою, на зразок розширеного алгоритму Евкліда.

2.2.2. Атака з використанням методу решета числового поля

Виходячи з роботи Х. Ленстри [21], евристичний очікуваний час виконання методу решета числового поля для обчислення нетривіального коефіцієнта p_i складеного числа n буде дорівнювати

$$L[n] = e^{1{,}923\,(\log n)^{1/3}(\log\log n)^{2/3}}. \qquad (2.7)$$

До того ж, час виконання $L[n]$ залежить лише від розміру розряду цілого числа n, яке потрібно розкласти. Таким чином, при використанні методу решета числового поля очікується, що час, необхідний для розкладання модуля RSA довжиною n біт, буде таким самим, як і для розкладання n-бітового складеного модуля для будь-якого $l > 2$.

2.2.3. Атака з використанням методу еліптичної кривої для факторизації

Метод еліптичної кривої для факторизації може обчислити нетривіальний коефіцієнт p_i складеного цілого числа n значно швидше, ніж метод решета числового поля, якщо коефіцієнт значно менший за $\sqrt{n}$. Числа, що піддаються даному методу, значно менші ніж для методу решета числового поля. Найбільше факторизоване число мало розмір 224 біта.

Виходячи з роботи Х. Ленстри [22], евристичний очікуваний час виконання методу еліптичної кривої для знаходження множника p_i складеного цілого числа n буде дорівнювати

$$\mathrm{E}[n, p_i] = (\log_2 n)^2\, e^{\sqrt{2}\,(\log p_i)^{1/2}(\log\log p_i)^{1/2}}. \qquad (2.8)$$

Таким чином, при використанні методу еліптичної кривої очікується, що час, необхідний для розкладання звичайного модуля RSA, буде більшим, ніж час, необхідний для розкладання складеного модуля запропонованого методу асиметричного шифрування для будь-якого $l > 2$ завдяки використанню простих множників меншого розміру для його отримання.

2.2.4. Атака з малою експонентою

Найвідомішою атакою з малою приватною експонентою є атака Вінера. Результат такої атаки буде виглядати наступним чином.

Дано дійсний відкритий ключ (n, e) з відповідним закритим ключем (n, d). Якщо виконується умова

$$d < \frac{n^{1/2l}}{\sqrt{2\,(2l-1)}}, \tag{2.9}$$

то приватна експонента може бути обчислена за поліноміальний час $\log n$.

Ще однією відомою атакою є атака на основі алгоритму редукції на решітках, що розробили Боне та Дерфі [23]. Вона розроблена на основі методу Коперсміта для пошуку малих розв'язків одновимірних модульних рівнянь.

Однак, на відміну від атаки Вінера, атака на основі алгоритму редукції на решітках є лише евристичною, а отримані результати стосуються граничного випадку великих модулів RSA та великого розміру решітки, хоча вона добре працює на практиці. Найсильніша атака Боне і Дарфі на основі алгоритму редукції на решітках використовує підгратки та вводить геометрично прогресивні матриці, щоб визначити межі об'єму використовуваних підграток. В результаті атаки виконується наступне твердження.

Для будь-якого $\varepsilon > 0$ та будь-якого $l \geq 2$ існує таке число n_0, що для довільного $n > n_0$ має місце наступне: нехай n складений модуль, відкритий

ключ (n, e), а закритий ключ – (n, d), де $e = n^{\alpha}$ та $d = n^{\delta}$. Тоді якщо виконуються умови

$$\alpha \approx 1,$$
$$\delta \leq 1 - \sqrt{1 - \frac{1}{l}} - \varepsilon, \tag{2.10}$$

то закритий ключ d може біти відновлено за поліноміальний час $\log n$.

2.2.5. Атака з частково відомим ключем

Існує декілька основних видів атак із частково відомим ключем на RSA, коли публічна або приватна експонента невелика, які були створені Ернстом, Йохемсом, Меєм і де Вегер [24]. Вони поділяються на так звані LSB (Least Significant Bit), коли відомо найменш значущий біт, та MSB (Most Significant Bit), коли відомо найбільш значущий біт.

Перший тип атаки проводиться наступним чином. Для будь-якого $\varepsilon > 0$ та будь-якого $l \geq 2$ існує таке число n_0, що для довільного $n > n_0$ має місце наступне: нехай n складений модуль, відкритий ключ (n, e), а закритий ключ – (n, d), де $e = n^{\alpha}$ та $d = n^{\delta}$. Дано публічний ключ, d_0 та M, де $d \equiv d_0 \bmod M$ і $M = n^{\beta - \delta}$. Тоді якщо

$$\delta \leq \frac{2}{3} + \frac{1}{3l} - \frac{2}{3l}\sqrt{(l-1)\,(3\alpha l + 3\beta l - 2l - 1)} - \varepsilon \tag{2.11}$$

то закритий ключ d може біти відновлено за поліноміальний час $\frac{1}{\varepsilon} \times \log n$.

Атака MSB проводиться аналогічним чином, для знаходження d використовується рівняння з апроксимацією $\hat{d}$, що задовольняє вираз $|d - \hat{d}| \leq n^{\delta}$.

Експерименти М. Хінека у 2008 році підтвердили емпірично, що збільшення кількості простих множників знижує ефективність атаки з використанням даних методів [25]. Тож використання даного виду криптоаналітичної атаки до запропонованого методу асиметричного шифрування можна вважати недоцільним.

2.2.6. Атака з використанням факторизації на решітчастій основі

Д. Боне, Г. Дерфі та Н. Гоугрейв-Грехем розробили метод факторизації для складених цілих чисел, який пізніше був доповнений М. Хінеком [26]. Метод атаки засновано на наступній теоремі.

Нехай n складений модуль RSA, що складається з l простих множників. Для будь-якого $s \in [2, l]$ дано $l - s$ простих чисел у факторизованому n, $(s - 1) / s$ найбільш або найменш значущих бітів одного з невідомих простих чисел, $(s - 2) / (s - 1)$ найбільш або найменш значущих бітів іншого невідомого простого числа…, $\frac{2}{3}$ найбільш або найменш значущих бітів іншого невідомого простого числа та $\frac{1}{2}$ найбільш або найменш значущих бітів одного з двох залишившихся невідомих простих чисел. Тоді складений модуль n може бути факторизовано за поліноміальний час $l \times \log n$. Варто зазначити, що дана теорема передбачає наявність верхньої границі для мінімальної частки бітів, необхідних для факторизації складеного модуля RSA. Фактична кількість необхідних бітів дорівнює добутку $\log_2 n$ та границі, зазначеної у теоремі. М. Хінек обчислив значення границі для перших значень l і всіх можливих відповідних значень s, що наведені у табл. 2.1.

Таблиця 2.1

Частина відомих бітів, необхідна для факторизації n

	Значення різниці $(l - s)$			
	0	1	2	3
$l = 2$	0,25			
$l = 3$	0,389	0,5		
$l = 4$	0,479	0,542	0,626	
$l = 5$	0,543	0,583	0,633	0,7

Значення границі наведене як частка сумарної кількості відомих бітів від загальної довжини модуля, тобто, розглядаючи класичний метод RSA, для успішної атаки необхідно знати кількість бітів, сумарна довжина яких дорівнює принаймні чверті довжини модуля.

Із даної таблиці можна зробити висновок, що збільшення кількості простих чисел у модулі збільшує необхідну для успішної атаки часку відомих бітів. Тож використання даного виду криптоаналітичної атаки до запропонованого методу асиметричного шифрування можна вважати недоцільним.

2.3. Висновки

У даному розділі було описано запропонований метод асиметричного шифрування та проаналізовано його особливості із використанням складеного модуля та способу розшифрування з використанням наслідку з Китайської теореми про лишки, а також введення рандомізації вибору кількості простих множників для генерації модуля n. Також було проаналізовано переваги даного методу, серед яких:

- зменшення обчислювально складності роботи;
- зменшення використання пам’яті.

Далі було проаналізовано особливості найбільш популярних методів атаки на класичний метод асиметричного шифрування RSA у разі їх застосування до запропонованого методу асиметричного шифрування та вплив внесених модифікацій на резистентність до зазначених методів атаки.

В результаті аналізу атак із використанням методів факторизації типу решета числового поля було продемонстровано, що для заданого розміру модуля n безпека однакова для будь-якої кількості множників, а складність факторизації не залежить від зміни їх кількості.

Атаки, що використовують математичні властивості схеми шифрування RSA, такі як метод Вінера, метод на решітчастій основі, а

також атаки з використанням відомих біт приватної або публічної експоненти можливі, навіть ефективні у певних випадках для модуля з трьох або чотирьох простих множників, проте для їх застосування необхідно виконання багатьох умов. Крім того, всі вони втрачають ефективність із збільшенням кількості простих множників у модулі. Однак, використання експонент невеликого розміру не рекомендується, оскільки створює вразливість для будь-якої асиметричної криптосистеми.

Шанс успішності атаки грубою силою із збільшенням кількості простих множників у модулі зростає у 2 рази після додаванням кожного нового множника, проте атаки такого типу в разі використання ключів актуальної довжини, тобто 1024 біта і більше, не ефективні.

3. АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ АСИМЕТРИЧНОГО ШИФРУВАННЯ

3.1. Аналіз використаної мови програмування та бібліотек

В рамках даної роботи буде реалізовано запропонований метод асиметричного шифрування, а також програмне забезпечення для експериментального дослідження його обчислювальної складності.

Для аналізу та візуалізації статичних даних, що будуть отримані під час експериментального дослідження, найбільш часто вживаними є мови програмування Python та R.

3.1.1. Мова програмування Python

Мова Python – це високорівнева інтерпретована мова програмування, що має строгу динамічну типізацію та підтримує більшість використовуваних парадигм розробки програмного забезпечення, таких як функціональне, об'єктно-орієнтоване, імперативне, процедурне та аспектно-орієнтоване програмування [27]. Дана мова направлена на швидку розробку програмного забезпечення та підвищення продуктивності розробника завдяки зручності читання коду та простий синтаксис. Також для даної мови програмування було створено велику кількість бібліотек для задач різного типу, зокрема для роботи із статичними даними та їх візуалізації.

Серед архітектурних рис окрім динамічної типізації варто зазначити автоматичне керування пам'яттю, механізм обробки виключень, підтримку високорівневих структур даних, повну інтроспекцію часу виконання, а також багатопоточні обчислення. Також підтримується розбиття на модулі, які можна об'єднувати у пакети. Крім того, присутній інтерактивний режим роботи.

Інтерпретатор для даної мови CPython підтримується більшістю сучасних платформ і забезпечує крос-платформність. Даний інтерпретатор можна розширювати типами даних та функціями, розробленими на будь-якій мові, що можна викликати із мови С.

Мова Python поширюється під вільною ліцензією PSFL, що дозволяє використовувати дану мову програмування в будь-яких додатках без обмежень. Завдяки цьому вона широко використовується великою кількістю великих компаній у різнопланових проєктах найрізноманітнішої направленості, у якості як основної мови програмування, так і для створення додаткових розширень розроблених програмних застосунків.

До переваг мови програмування Python можна віднести:

- простий зрозумілий синтаксис;
- велика кількість багатофункціональних бібліотек, що надають засоби роботи з крос-платформними додатками та застосунками, графікою та математичними функціями;
- портованість до більшості відомих платформ, таких як Mac OS, IOS, Microsoft Windows, UNIX та Android;
- наявність якісної документації до самої мови програмування та написаних нею бібліотек;
- легка інтеграція даної мови досягається завдяки потужним можливостям керування, до яких належать виклик функцій напряму через C, C++ або Java через Jython. Також Python обробляє довільні мови розмітки, оскільки працює за допомогою однакового байт-коду для всіх платформ.

Серед недоліків даної мови варто зазначити:

- низьку швидкодію виконання програм через інтерпретованість даної мови, порівняно з компільованими мовами;
- відсутність статичної типізації, що заважає реалізовувати на етапі компіляції механізми перевантаження функцій;

- ускладнене тестування через неможливість виявлення помилок до виконання програми.

3.1.2. Мова програмування R

Мова R – це мультипарадигменна мова програмування для статичної обробки даних та розробки графіки. Дана мова широко розповсюджена для створення статичного програмного забезпечення для аналізу даних, та стала стандартом для спільноти спеціалістів у галузі статистики. Вона також належить до інтерпретованих мов.

Мова R має широкий спектр можливостей для проведення статистичного аналізу різних типів, серед яких лінійна і нелінійна регресія, аналіз часових рядів, класичні статистичні тести та кластерний.

Мова R поширюється під ліцензією GNU General Public License безкоштовно у вигляді вільнодоступного коду, що дозволяє використовувати дану мову програмування в будь-яких додатках без обмежень.

До переваг мови програмування R належить:

- простий синтаксис;
- крос-платформність;
- здатність використовувати пакети, що написані не лише на мові програмування R;
- здатність мови до розширення завдяки підключеним пакетам;
- здатність до використання для матричних обрахунків з швидкодією на рівні математичних пакетів MATLAB.

Серед недоліків даної мови варто зазначити:

- обмеженість та незручність засобів опису даних для аналізу;
- низьку швидкодію виконання програм через інтерпретованість даної мови, порівняно з компільованими мовами.

Після проведеного аналізу даних мов програмування для розробки програмної реалізації запропонованого методу асиметричного шифрування

та програмного забезпечення для аналізу обчислювальної складності методу шифрування було вирішено використати мову Python.

3.1.3. Аналіз використаних бібліотек

Для реалізації розробки програмної реалізації запропонованого методу асиметричного шифрування та програмного забезпечення для аналізу обчислювальної складності методу шифрування було використано бібліотеки Matplotlib та NumPy.

Matplotlib – бібліотека для мови Python, що призначена для візуалізації двовимірних та тривимірних графічних зображень [28]. Згенеровані графіки можуть бути використані у наукових публікаціях, графічному інтерфейсі користувача та інтерактивній графіці. Дана бібліотека легко конфігурується під задані потреби та підтримує усі існуючі види графіків та діаграм.

В даній роботі бібліотека Matplotlib використовується для побудови та візуалізації графіків, що ілюструють певні метрики ефективності застосування запропонованого методу асиметричного шифрування.

NumPy – бібліотека для мови Python, призначена для виконання загальних високорівневих математичних та числових операцій у вигляді швидких функцій, створених для роботи із багатомірними масивами даних, для яких вони оптимізовані [29]. Дана бібліотека представляє собою альтернативу використання MATLAB завдяки високій швидкості роботи.

В даній роботі бібліотека NumPy використовується для швидкого обчислення складних операцій, присутніх у запропонованому методі асиметричного шифрування.

3.2. Архітектура програмного забезпечення

Програмне забезпечення, що реалізовує запропонований метод асиметричного шифрування, створено у вигляді додатку на мові Python. Для запуску програми необхідно виконати команду «python main.py».

Розроблена програма складається з модулів, структурну схему яких наведено на рис. 3.1.

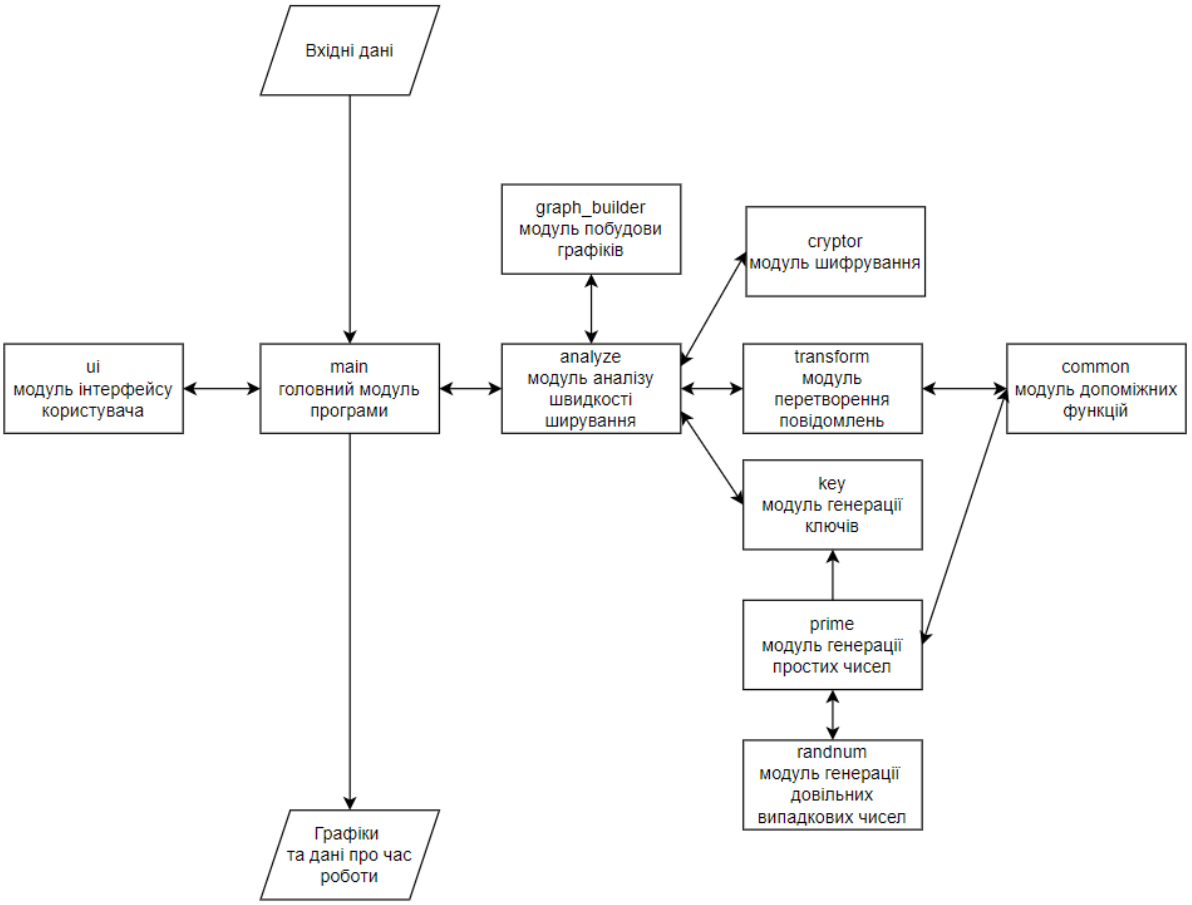


Рис. 3.1. Структурна схема програмного забезпечення

Далі описано кожен розроблений модуль, його функції та особливості.

1. Cryptor – модуль, в якому реалізовано функції шифрування та дешифрування. Функція дешифрування реалізована у двох варіантах – у класичному вигляді, тобто за формулою (1.7) та із використанням Китайської теореми про лишки, тобто за формулою (2.5).
2. Transform – модуль, що перетворює числа у звичайному форматі у біти та навпаки. Даний модуль містить дві функції, що виконують відповідні дії.

3. Common – модуль, що містить допоміжні математичні функції, що використовуються у методі шифрування, серед яких функції перевірки довжини числа у бітах та байтах, функцію перевірки знаходження НСД та функцію знаходження взаємно оберненого числа.
4. Key – модуль, що містить всі необідні функції для створення ключів шифрування. Функція, що повертає згенеровані ключі, реалізована у двох варіантах - для класичного методу RSA та для запропонованого методу асиметричного шифрування з використанням складеного модуля.
5. Prime – модуль, що генерує великі прості числа заданої довжини. Для цього використовується тест Міллера-Рабіна – тест числа на простоту, що виконується за поліноміальний час. Даний метод перевіряє ряд рівностей, які мають виконуватися для простих чисел. Результат тесту не гарантує простоту отриманого числа, а лише дає високу ймовірність цього. Проте даний тест показує найкращий час роботи порівняно з іншими детермінованими тестами на простоту числа, тож для генерації простих чисел у методах шифрування використовують переважно саме його.
6. Randnum – модуль, що генерує випадкові числа заданої довжини у бітах.
7. Analyze – модуль, що містить функції для вимірювання часу роботи методу шифрування.
8. Graph_builder – модуль, що містить функції побудови графіків на основі структури, у якій зберігається інформація про точки, на основі яких будується графік, а також назва графіка і назви вісей.
9. UI – модуль, що реалізує інтерфейс користувача, що представляє список доступних дій, серед яких отримання

графіків залежності часу виконання реалізації запропонованого методу шифрування для модуля заданої довжини, графіків часу виконання алгоритму із використанням наслідку з Китайської теореми про лишки та без, графіки часу виконання класичного методу RSA та запропонованого методу шифрування. Окремий сценарій створено для просто обчислення шифротексту заданого повідомлення.

3.3. Висновки

В даному розділі було проаналізовано переваги та недоліки таких мов програмування, як Python та R. Для створення програмної реалізації запропонованого методу асиметричного шифрування та програмного забезпечення для аналізу обчислювальної складності методу шифрування було вирішено використати мову Python.

Далі було проаналізовано можливості використаних бібліотек, а саме Matplotlib та NumPy для обчислення складних операцій та візуалізації отриманих в результаті проведеного аналізу даних.

Також було проаналізовано структуру розробленого програмного забезпечення, його модулі та взаємозв’язки між різними частинами застосунку.

4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ АСИМЕТРИЧНОГО ШИФРУВАННЯ

4.1. Дослідження швидкості роботи методу шифрування для визначення оптимальної кількості простих множників у модулі

Для визначення оптимальної кількості великих простих чисел l, що обмежена як $3 \le l \le \frac{n}{4}$, тобто розрядність кожного окремого множника не менше 4 знаків, необхідно провести часові заміри часу роботи реалізації запропонованого алгоритму асиметричного шифрування на різних довжинах модуля n. Для цього сто довільно згенерованих повідомлень будуть проходити повний цикл роботи алгоритму, тобто генерація ключів, шифрування та дешифрування, для кожного значення l у діапазоні можливих значень. Далі буде обчислено середнє значення часу роботи реалізованого алгоритму для кожного значення l у діапазоні можливих значень та побудовано графік залежності часу роботи алгоритму від значення l для різних розмірів модуля n. В якості розміру модуля n будуть використані актуальні значення, тобто 1024 біта і більше.

Оскільки верхня границя буде зростати із збільшенням довжини модуля n, а також враховуючи недоцільність використання настільки великого значення l з точки зору криптостійкості, її необхідно обмежити певним константним значенням, що, однак, дозволить проспостерігати поведінку алгоритму на великих значеннях даного числа. Виходячи з цього, верхню границю допустимих значень числа l для проведення експериментального аналізу роботи реалізації запропонованого методу шифрування буде обмежено значенням, для якого настає значне уповільнення роботи.

Результати проведеного дослідження наведено у вигляді графіків на рис. 4.1 – 4.5.

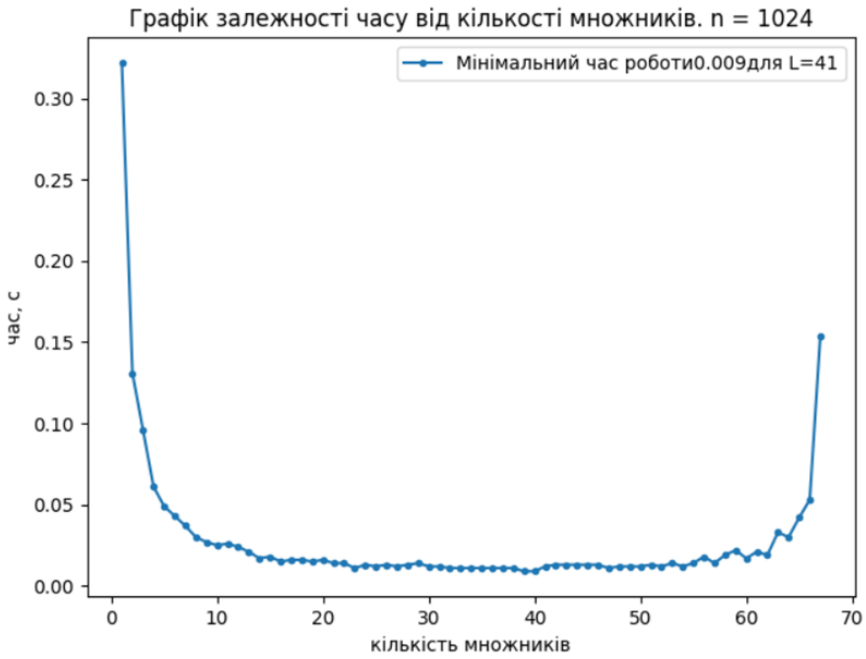


Рис. 4.1. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 1024

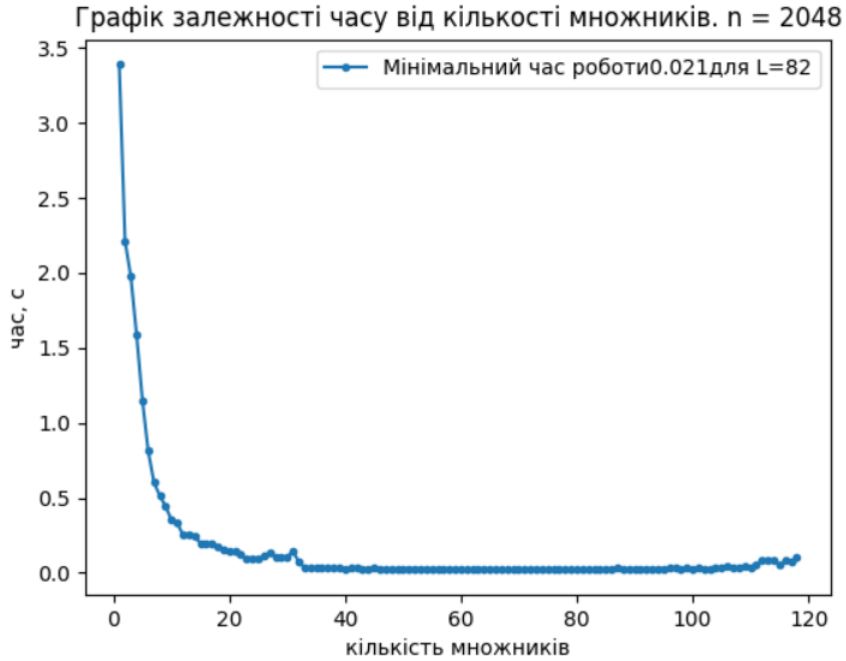


Рис. 4.2. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 2048

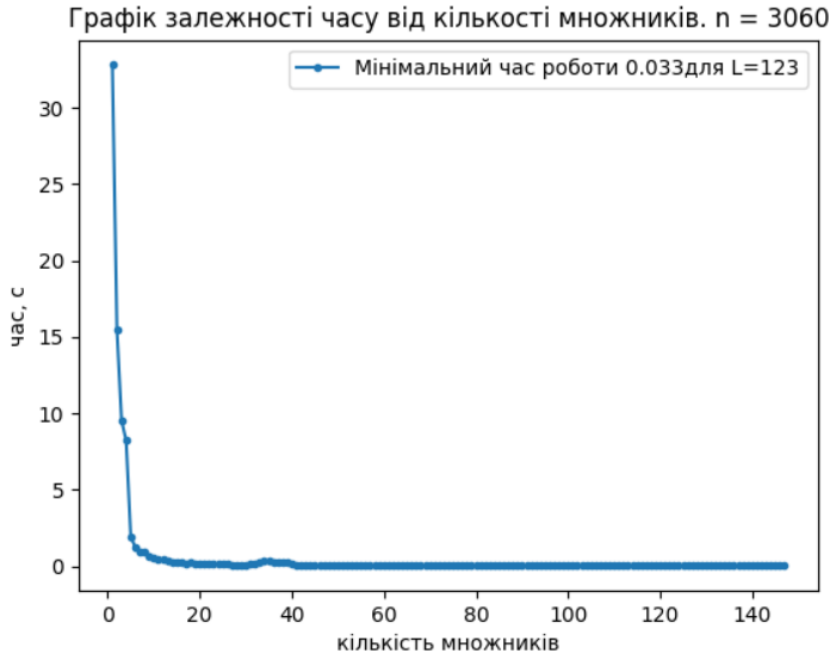


Рис. 4.3. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 3060

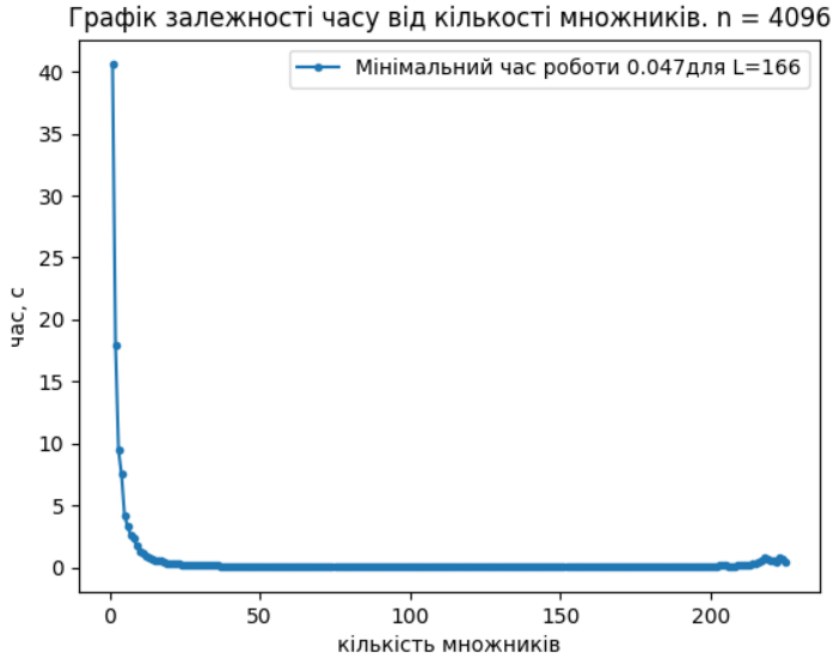


Рис. 4.4. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 4096

Аналізуючи графіки зміни швидкості роботи алгоритму в залежності від кількості простих множників у модулі, можна помітити тенденцію

розміщення найменшого значення часу виконання алгоритму на одному інтервалі, пропорційно довжини модуля. Використовуючи отримані значення, можна побудувати наступне відношення:

$$\frac{n}{l} = \frac{1024}{41} \approx \frac{2048}{82} \approx \frac{3060}{123} \approx \frac{4096}{166} \approx 25 \quad (4.1)$$

Отже, можна зробити висновок, що найменший час виконання алгоритму можна отримати, обираючи кількість простих чисел l як

$$l = \frac{n}{25}. \quad (4.2)$$

Для введення рандомізації з метою підвищення складності атаки пропонується обирати кількість простих чисел l як

$$l = \frac{n}{m}, m \in [20, 30]. \quad (4.3)$$

Для доказу ефективності запропонованого способу обирати кількість простих чисел l, буде порівняно графік часу роботи алгоритму, що буде використовувати запропонований спосіб та графік, побудований по мінімальним значенням часу роботи алгоритму, що були отримані раніше, з доданням проміжних значень. Відповідні графіки зображені на рис. 4.5.

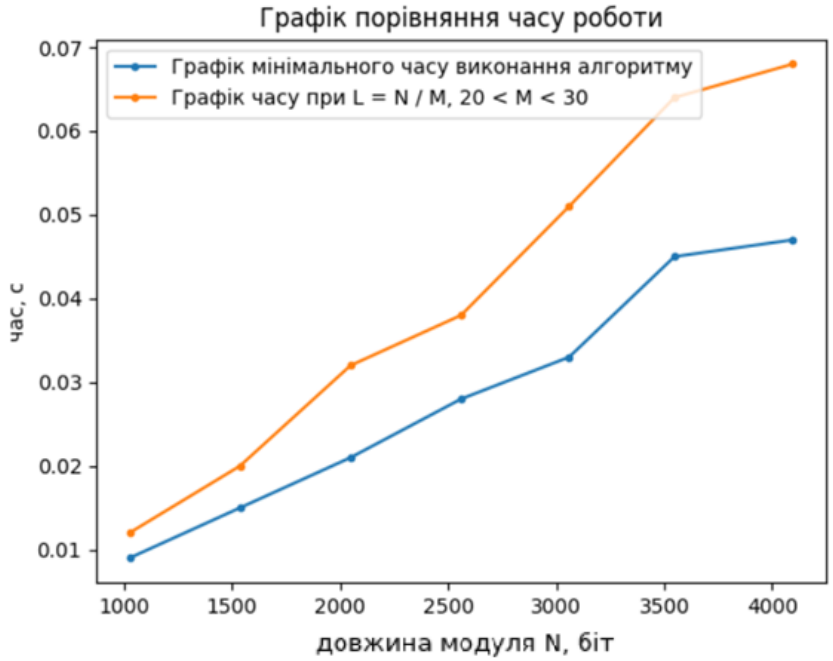


Рис. 4.5. Графіки часу виконання реалізації запропонованого методу шифрування та часу мінімального виконання

Отримані графіки ілюструють, що використання запропонованого способу обирати кількість простих чисел l за допомогою формули (4.3) дозволяє отримати час виконання алгоритму запропонованого методу шифрування, що є максимально наближеним до мінімального. Теоретично, досягти кращих показників можливо, звузивши діапазон числа m у формулі (4.3), проте це спростить виконання криптоаналітичних атак на алгоритм, тож дані дії не рекомендуються.

4.2. Доцільність використання наслідку з Китайської теореми про лишки на обчислювальну складність роботи запропонованого методу асиметричного шифрування

Теоретично, використання наслідку з Китайської теореми про лишки у будь-якій модифікації методу асиметричного шифрування RSA призводить до зменшення його обчислювальної складності, що відповідно призводить до зменшення часу виконання алгоритму. Використання даної теореми для класичної реалізації методу дає зменшення часу виконання у 4 рази. Проте необхідно дослідити, чи доцільно комбінувати використані підходи для зменшення часу роботи алгоритму, тобто збільшення кількості простих чисел у модулі та використання наслідку з Китайської теореми про лишки (КТЛ), оскільки можливість того, що використання даної теореми переускладнює обчислення для великої кількості простих множників, не виключена.

Для дослідження доцільності комбінування використаних підходів до прискорення роботи алгоритму буде порівняно графік часу роботи алгоритму, що буде використовувати запропонований спосіб у повному вигляді, тобто із використанням зазначеної теореми, та графік часу роботи алгоритму, що її не використовує. Відповідні графіки зображені на рис. 4.6.

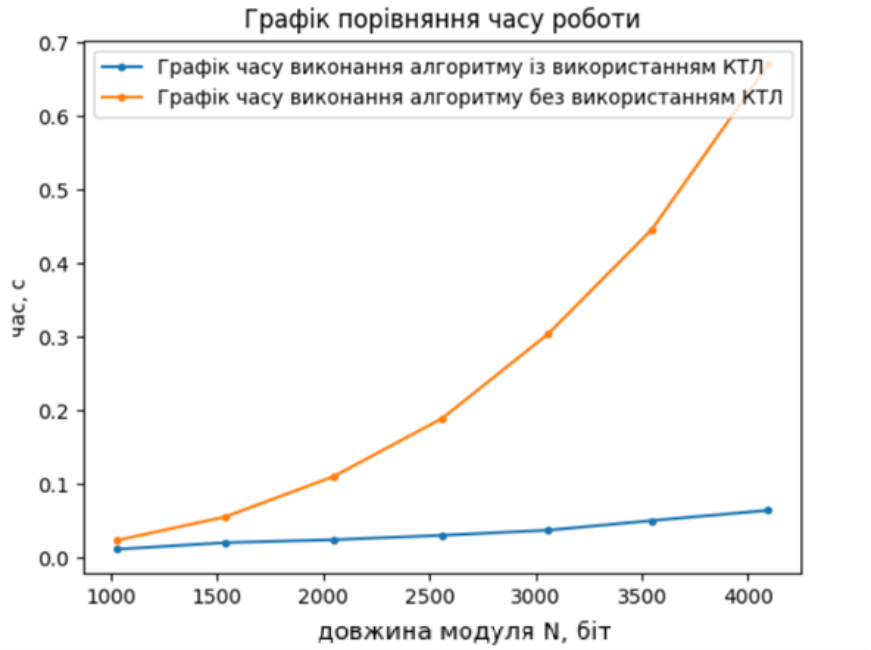


Рис. 4.6. Графіки часу виконання алгоритму із використанням КТЛ та без використання КТЛ

Наведені графіки часу виконання алгоритму доводять доцільність комбінування використаних підходів до прискорення роботи алгоритму, тобто збільшення кількості простих чисел у модулі та використання КТЛ. Аналіз даних графіків показує, що приріст швидкості роботи алгоритму збільшується відносно зростання розміру модуля.

4.3. Доцільність використання запропонованого методу асиметричного шифрування у порівнянні із класичною реалізацією методу шифрування RSA

Для дослідження доцільності використання запропонованого методу асиметричного шифрування у порівнянні із класичною реалізацією методу шифрування RSA буде побудовано та проаналізовано графіки часу виконання відповідних методів. Графіки наведено на рис. 4.7.

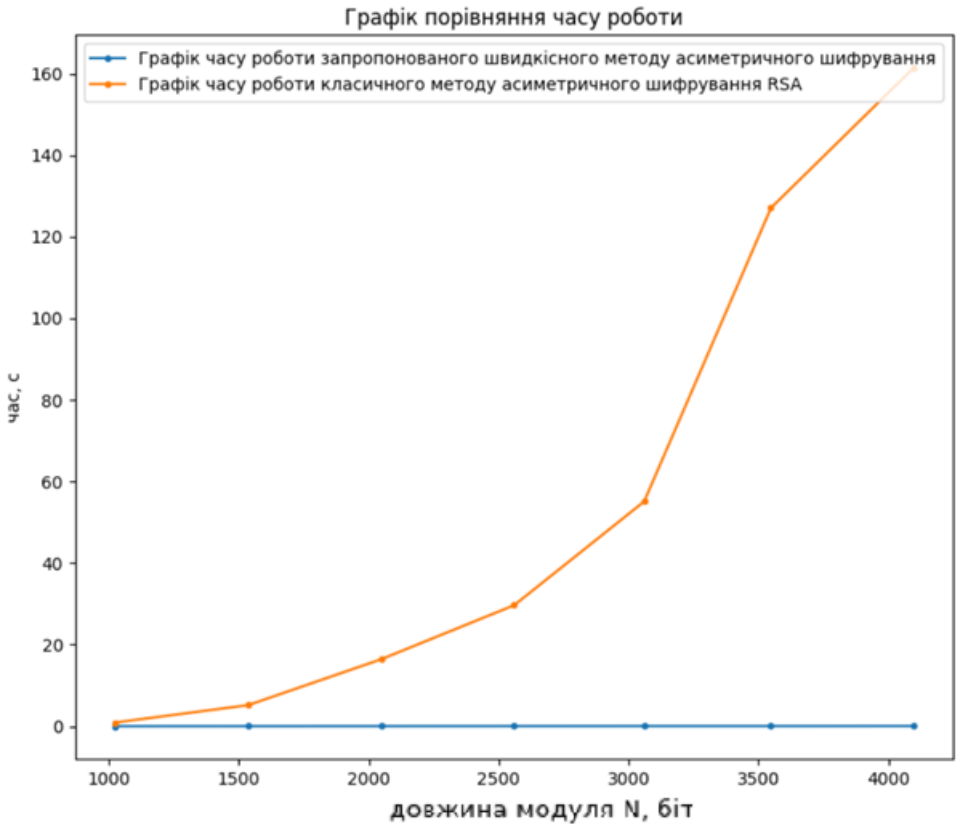


Рис. 4.7. Графіки часу виконання класичного методу RSA та запропонованого методу шифрування

Для виведення точного значення приросту швидкості роботи запропонованого методу асиметричного шифрування у порівнянні із класичним методом шифрування RSA відносно довжини модуля було проаналізовано графіки на рис. 4.7 та обчислено відношення часу роботи зазначених методів. Отримані значення були використані для побудови графіка, що ілюструє зміну значення відношення обчислювальної складності класичного алгоритму RSA до обчислювальної складності запропонованого методу асиметричного шифрування.

Застосувавши функцію інтерполяції до отриманих значень, було знайдено функцію зменшення обчислювальної складності роботи алгоритму:

$$f(x) = -1{,}275 \times 10^{-13}x^5 + 1{,}72 \times 10^{-9}x^4 - 8{,}508 \times 10^{-6}x^3 + 1{,}9609 \times \\ \times 10^{-2}x^2 - 20{,}606\,x + 8009 \qquad (4.4)$$

Відповідний графік наведено на рис. 4.8.

Рис. 4.8. Графік зміни значення відношення обчислювальної складності класичного алгоритму RSA до обчислювальної складності запропонованого методу асиметричного шифрування

На рис. 4.9. зображено побудований графік отриманої функції (4.4).

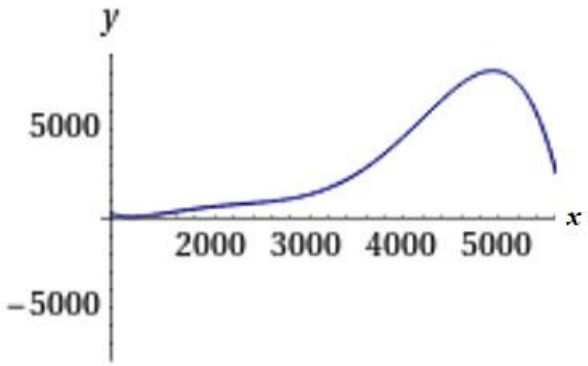


Рис. 4.9. Графік функції зменшення обчислювальної складності роботи алгоритму

Аналіз отриманої функції показує, що із збільшенням довжини модуля понад 5000 біт різниця у часі поступово починає зменшуватись.

Найкращий результат з точки зору зменшення обчислювальної складності запропонований метод асиметричного шифрування досягає на інтервалі [4000; 5000].

4.4. Дослідження обчислювальної складності запропонованого методу асиметричного шифрування із збільшенням довжини модуля

Для перевірки обчислювальної складності запропонованого методу асиметричного шифрування на великих значеннях модуля буде побудовано графік залежності часу роботи методу від довжини модуля *n* від 1024 біт до 10240 біт включно. Відповідний графік зображено на рис. 4.10.

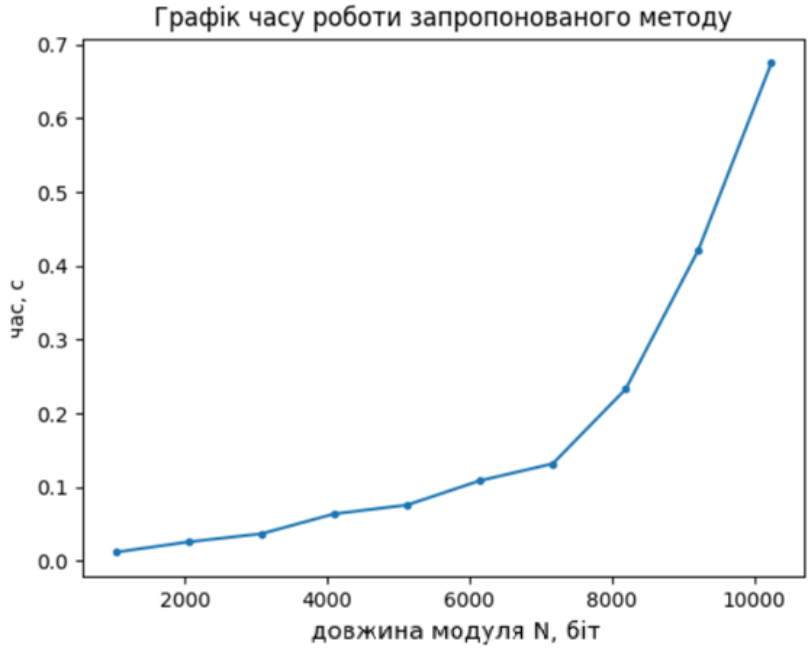


Рис. 4.10. Графік залежності часу роботи методу від довжини модуля *n* на інтервалі [1024; 10240] біт

Графік показує, що час роботи запропонованого методу асиметричного шифрування збільшується пропорційно збільшенню довжини модуля *n*, тобто зростає експоненціально.

4.5. Висновки

У даному розділі було проведено експериментальне дослідження ефективності розробленого алгоритмічно-програмного методу асиметричного шифрування, порівняно час його роботи із часом роботи класичного методу асиметричного шифрування RSA.

Всі заміри часу роботи проводились на процесорі Intel Core i7-8550U 1,8 ГГц.

В результаті проведеного аналізу було виведено формулу отримання оптимальної кількості великих простих чисел у складеному модулі, використання якої дозволяє отримати найменший час роботи виконання алгоритму, тобто найкращий результат з точки зору зменшення обчислювальної складності.

Також було проаналізовано доцільність використання наслідку з Китайської теореми про лишки у поєднанні із складеним модулем та доведено ефективність застосування поєднання даних способів зменшення обчислювальної складності алгоритму асиметричного шифрування.

Порівняння часу роботи класичного алгоритму RSA із часом роботи запропонованого методу асиметричного шифрування проілюструвало, що запропонований метод має меншу обчислювальну складність.

За допомогою методу інтерполяції було знайдено функцію зміни значення відношення обчислювальної складності даних методів шифрування, а саме поліном п’ятої степені. Аналіз графіка даної функції продемонстрував, що найкращий результат з точки зору зменшення обчислювальної складності запропонований метод асиметричного шифрування досягає на інтервалі [4000; 5000], а далі значення обчислювальної складності зазначених методів шифрування починають наближатись один до одного.

5. ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ КОМЕРЦІЙНОЇ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

5.1. Опис проблеми

Останнім часом у світовому суспільстві відзначається глобальна цифрова трансформація, що призводить до зростання обсягу приватної інформації, що транслюється через мережу Інтернет. Це призводить до підвищення ризику порушення конфіденційності даних, в наслідок чого особиста інформація стане доступна зловмисникам. Тому задача забезпечення захисту інформації є однією із найбільш пріоритетних на сьогоднішній день.

Розвиток інформаційних, телекомунікаційних та інформаційно-телекомунікаційних технологій значно ускладнює завдання збереження надійності захисту даних. Для цього необхідно підвищувати ефективність криптографічних алгоритмів, які використовуються для вирішення даного завдання.

Серед методів криптографії одним із найпоширеніших можна назвати метод асиметричного шифрування RSA, який постає в якості стандарту для великої кількості шифрувальних додатків та сервісів. Не зважаючи на винайдення нових методів асиметричного шифрування, метод RSA залишається досить широко використовуваним у різноманітних криптографічних засобах захисту інформації сучасних інформаційних, телекомунікаційних та інформаційно-телекомунікаційних систем.

Основною проблемою криптографії є потреба постійно випереджати у розвитку методи криптоаналізу для виконання своєї функції захисту інформації. У зв’язку з розвитком методів криптоаналізу, а також накопиченням обчислювальної потужності, збільшення ефективності роботи криптографічних методів є постійною актуальною задачею.

Стандартним методом боротьби з падінням криптостійкості асиметричних методів шифрування є збільшення довжини ключів

шифрування, що також породжує ряд проблем, пов’язаних з уповільненням роботи, а отже зменшенням можливості застосування у сферах, де важлива швидкість виконання.

RSA – перший широко використовуваний алгоритм асиметричної криптографії, який до цих пір популярний в індустрії. Математична основа даного методу приховує у собі достатню кількість вразливостей, які недостатньо досвідчений розробник може не врахувати при реалізації криптосистеми на основі методу RSA.

Головною проблемою забезпечення надійності шифрування методу RSA на сьогоднішній день постає необхідність використовувати великі ключі шифрування, що залишаться недосяжними для обчислювальної потужності сучасних процесорів. Це негативно впливає на час роботи алгоритму, що реалізує метод RSA. У випадку збереження даної тенденції, у майбутньому час роботи методу RSA із ключами актуальної довжини зробить його непридатним до використання.

Безпека RSA заснована на тому факті, що маючи велике число n, що є добутком двох простих чисел p і q, розкласти дане число n на прості множники без додаткової інформації про числа p і q за поліноміальний час неможливо. Розробники несуть відповідальність за вибір простих чисел, що становлять модуль RSA. Цей процес надзвичайно повільний у порівнянні з генерацією ключів для інших криптографічних протоколів, де досить просто вибрати кілька випадкових байтів. Тому, замість того, щоб генерувати дійсне випадкове просте число, розробники часто намагаються створювати числа певної форми. Дане рішення несе у собі ряд певних загроз.

Існує велика кількість умов, виконання яких необхідно при виборі простих чисел p і q для збереження необхідного рівня складності факторизації модуля n, що зможе гарантувати надійність шифрування. Наприклад, p і q повинні бути глобально унікальними. Якщо p або q коли-небудь повторно використовуються в інших модулях RSA, то обидва

множники можна легко обчислити за допомогою алгоритму GCD. Погані генератори випадкових чисел роблять цей сценарій досить імовірним, і дослідження показали, що приблизно 1% трафіку TLS в 2012 році було піддано такій атаці.

Оскільки використання закритого ключа великого розміру негативно впливає на час розшифрування та підпису, розробники іноді реалізують вибір невеликого числа у якості приватної експоненти d, особливо у випадках пристроїв з низьким споживанням енергії, такому як смарт-карти. Тим не менш, зловмисник може відновити закритий ключ, коли число d менше кореня 4 ступеня з n. Натомість розробникам варто обирати велике значення числа d, а для прискорення дешифрування використовувати спеціальні математичні прийоми, такі як використання наслідку з Китайської теореми про лишки. Однак складність даного підходу збільшує ймовірність виникнення незначних помилок реалізації, що може призвести до відновлення ключа.

Для запобігання даної вразливості спочатку, як правило, обирають відкриту експоненту e, що також має бути достатньо великим числом, чим часто нехтують.

Більш того, p і q повинні бути обрані незалежно один від одного. Якщо p і q спільно використовують приблизно половину своїх старших бітів, то модуль n може бути обчислено за допомогою методу Ферма. Насправді, навіть вибір алгоритму тестування простоти може мати наслідки для безпеки.

Найпоширенішою вразливістю методу RSA вважається ROCA в RSALib, яка торкнулася багатьох смарт-карток, модулів довірених платформ і навіть ключів Yubikey. Тут при генерації ключів використовуються тільки прості числа певної форми для прискорення обчислень. Задача по виявленню простих чисел, що згенеровані таким чином, вирішується з використанням хитрих прийомів теорії чисел тривіально. Як тільки слабка система була розпізнана, спеціальні

алгебраїчні властивості простих чисел дозволяють зловмисникові використовувати метод Копперсміта для розкладання *n*.

Варто враховувати, що ні в одному з цих випадків генерація простих чисел таким чином не є очевидним фактом, що приводить до повної відмови системи. Все тому, що незначні теоретико-числові властивості простих чисел істотно впливають на безпеку методу RSA. Очікування того, що звичайний розробник буде мати необхідну кваліфікацію та обізнаність у математичному аналізі, криптографії та теорії чисел для самостійної розробки засобів захисту програмного забезпечення, серйозно підриває його безпечність.

Отже, існуючі проблеми можна підсумувати наступним чином:

- збільшення часу роботи асиметричних криптосистем при збереженні необхідного рівня захисту;
- залучення недостатньо кваліфікованих розробників до забезпечення захисту програмних продуктів;
- розвиток криптоаналізу та розширення можливостей для атаки.

Все вище описане можна узагальнити у схемі «Дерева проблем», яка зображена на рис. 5.1

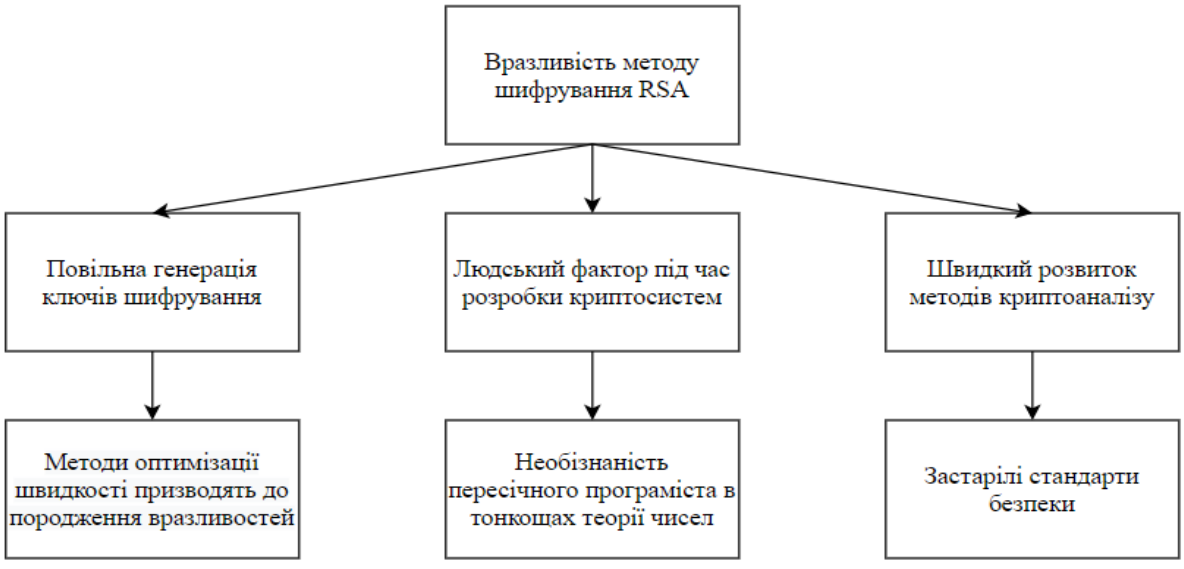


Рис. 5.1. Дерево проблем

5.2. Зацікавлені сторони

У вирішенні описаної вище проблеми існує декілька зацікавлених сторін.

Основною зацікавленою стороною є компанії, що працюють з персональними даними, а також створюють програмне забезпечення, що їх використовує, таке як додатки та сервіси у сфері медицини, пошти, фінансів, страхування та державних послуг. Всі вони гарантують своїм користувачам надійність та конфіденційність особистої інформації.

Особливо це стосується компаній, що не володіють власним відділом розробки програмного забезпечення для захисту інформації, а використовують сторонні модулі. Це компанії, що поставляють програмні продукти, такі як:

- соціальні мережі;
- фінансові веб-додатки;
- державні сервіси;
- поштові сервіси;
- сервіси страхування;
- медичні веб-додатки;
- інтернет-магазини.

Окремою зацікавленою стороною є користувачі програмних продуктів, яким необхідно мати посилений захист даних у разі зберігання на своїх пристроях особливо цінних даних.

Також в якості зацікавленої сторони можна виділити команду проєкту, тобто керівника та розробників, оскільки керівник має дуже високий інтерес у проєкті, тому що він отримує основну користь від його впровадження. Розробники зацікавлені у меншій мірі, проте вони значною мірою впливають на якість продукту.

У табл. 5.1 зведено всі групи зацікавлених сторін, їх інтереси, та вплив (міра зацікавленості у вирішенні наявних проблем).

Таблиця 5.1

Зацікавлені сторони проєкту

Зацікавлена сторона	*Інтерес зацікавленої особи*	*Вплив зацікавленої особи*	*Стратегії приваблення зацікавлених сторін*
Компанії, що працюють з даними користувачів	Наявність криптосистеми, що можна додати до своїх продуктів з мінімальними затратами на виробництво та гарантією ефективності її роботи	Високий	Проведення презентації для представників зацікавлених осіб. Участь у спеціалізованих виставках та форумах.
Користувачі ПЗ, що гарантує конфіденційність даних	Необхідність у підвищеному захисті персональної інформації.	Низький	
Керівник проєкту	Створення продукту. Отримання прибутку.	Високий	Створення та просування продукту. Досягнення основних цілей проєкту.
Розробники проєкту	Створення продукту. Збереження посади. Професійний розвиток.	Високий	Впровадження у проєкт використання актуальних технологій розробки програмного забезпечення та залучення якісних методик розробки ПЗ

5.3. Комерційне рішення. Основні характеристики

Відповідно до вищезазначених проблем, можна описати кінцевий продукт, що має їх вирішувати. Даний програмний продукт буде реалізовувати модуль шифрування, що використовує розроблений метод асиметричного шифрування на основі методу RSA. Даний метод забезпечує високий рівень захисту даних та має наступні переваги над конкурентними існуючими рішеннями:

- зменшена обчислювальна складність методу на етапі генерації ключів шифрування та на етапі розшифрування і, відповідно, створенні цифрового підпису, що призводить до зменшення часу виконання алгоритму;
- підвищена стійкість проти непрямих криптоаналітичних методів атаки, що використовують математичні вразливості методу шифрування.

Зменшена обчислювальна складність методу є важливою перевагою даного програмного продукту, оскільки розширює сфери його використання у порівнянні із класичним методом RSA. Дана особливість дозволяє швидко шифрувати великі обсяги даних, в наслідок чого запропонований програмний продукт для асиметричного шифрування можна застосовувати до мультимедійних даних.

Даний програмний продукт підтримує наступні функції:

- шифрування;
- дешифрування;
- створення та верифікація цифрових підписів;
- генерація ключів шифрування у відповідності до стандарту PKCS1.

Також при створенні даного модуля необхідно врахувати підтримку зручної інтеграції у довільні програмні продукти, в яких знадобиться його використання, а також надання зрозумілого інтерфейсу користувача та документацію.

Ще однією особливістю даного програмного продукту є сумісність із продуктами, що використовують класичну реалізацію методу асиметричного шифрування RSA. Дана особливість забезпечує можливість використання розробленого програмного продукту для шифрування даних не лише всередині однієї системи, а і для обміну зашифрованими даними із сторонніми системами, що використовують інші захисні програми, що реалізують класичний метод RSA і підтримують відповідні стандарти шифрування, зокрема PKCS1.

Таким чином, Боб, використовуючи запропонований програмний продукт, зможе безпечно спілкуватись з Алісою, яка використовує іншу криптосистему із класичним методом RSA, не втрачаючи переваг, що надає запропонований метод асиметричного шифрування. Дана можливість досягається завдяки тому, що внесені покращення змінюють окремі етапи алгоритму шифрування, не втручаючись у схему роботи алгоритму в цілому.

Враховуючи значну розповсюдженість використання методу шифрування RSA, така особливість значно розширює можливості впровадження на ринок даного програмного продукту.

5.4. Конкурентні переваги рішення

На сьогоднішній день ринок кібербезпеки наповнено великою кількістю систем шифрування даних. Методи асиметричного шифрування показали свою високу ефективність з точки зору надійності їх використання, проте вони значно програють методам симетричного шифрування у швидкості роботи, тому їх переважно використовують разом із симетричними методами.

Проте розвиток обчислювальної потужності комп'ютерів, а також напрацювання у методах криптоаналізу постійно підвищують можливості до зламу. Даний фактор створює одну із основних проблем криптографії, яка породжує необхідність постійно випереджати у розвитку методи

криптоаналізу для виконання своєї функції захисту інформації. Тому у зв’язку з розвитком методів криптоаналізу, а також накопиченням обчислювальної потужності, тому з метою підтримки рівня надійності шифрування даних, стандарти шифрування постійно змінюються і ускладнюються.

Для забезпечення збереження конфіденційності захищених даних та надійності використання криптосистеми використовується декілька підходів.

1. Стандартизація довжини ключів шифрування. Збільшення обчислювальної потужності, доступної криптоаналітикам, стандарти безпечної довжини ключів доводиться збільшувати. Даний фактор негативно впливає на час роботи алгоритму через підвищення обчислювальної складності. Прикладом такого стандарту є PKCS.
2. Внесення змін у криптосистеми і створення модифікацій методів шифрування з метою покращення їх характеристик.

Для створення запропонованого програмного продукту було використано другий підхід. Внесені модифікації забезпечують високий рівень захисту даних та надають переваги над конкурентними існуючими рішеннями, такі як зменшення обчислювальної складності методу на етапі генерації ключів шифрування та на етапі розшифрування і, відповідно, створенні цифрового підпису, що призводить до зменшення часу виконання алгоритму, а також підвищення стійкості проти непрямих криптоаналітичних методів атаки, що використовують математичні вразливості методу шифрування. Таким чином, запропонований метод асиметричного шифрування долає проблему збільшення часу роботи при нарощенні довжини ключів шифрування.

5.5. Клієнти. Сегменти ринку споживання

Як вже було зазначено вище, клієнтами даного програмного забезпечення є:

- компанії, що працюють з персональними даними клієнтів, а також створюють програмне забезпечення, що їх використовує;
- користувачі програмних продуктів, яким необхідно мати посилений захист даних у разі зберігання на своїх пристроях особливо цінних даних.

Відповідно, сегментацію ринку можна провести за типом ринку B2B та B2C.

На ринку B2B великі компанії мають можливості створювати власне програмне забезпечення для гарантування надійності обміну даними. Тому основна орієнтація на невеликі компанії, що використовують в розробці своїх програмних продуктів стороннє програмне забезпечення. Найбільш перспективними для співпраці постають розробники соціальних мереж, де дуже активно використовується асиметричне шифрування, та інші продукти, які передбачають обмін текстовим, аудіо та відео контентом.

Серед потенційних проблем потрібно зазначити:

- високу інтенсивність конкуренції в даному сегменті;
- небажання бізнесу фінансово вкладатись у безпеку свого продукту через нерозуміння відмінностей дешевих та дорогих рішень.

На ринку B2C орієнтація направлена на клієнтів, які прагнуть забезпечити власні системи додатковими засобами безпеки. Серед проблем даного сегменту варто зазначити:

- низький попит на програмне забезпечення у сфері безпеки;
- популярність використання неліцензійного програмного забезпечення.

5.6. Унікальна ціннісна пропозиція

У дереві проблем було виділено, які саме завдання даний програмний продукт має вирішити, а у зацікавлених сторонах було визначено, які саме очікування від продукту наявні у потенційних користувачів.

Ціннісну пропозицію, тобто спосіб вирішення поставлених завдань за допомогою запропонованого продукту, сформульовано наступним чином.

Невеликі компанії, які створюють програмне забезпечення, що використовує особисті конфіденційні дані користувачів, а також надає можливості обміну інформацією між користувачами, та користувачі, яким необхідно мати додатковий захист своїх особливо важливих даних по власним причинам, мають потребу у наявності засобів для забезпечення надійності зберігання та пересилання інформації, що будуть відповідати актуальним стандартам безпеки без надмірної обчислювальної складності, а також будуть сумісними із поточними протоколами безпечного пересилання даних та не накладатимуть інших обмежень на використання будь-якого характеру.

Запропонований криптографічний модуль, що використовує розроблений метод асиметричного шифрування, відповідає даним вимогам, вирішує зазначені проблеми та задовольняє очікування усіх груп потенційних користувачів.

Отже, основною унікальною ціннісною пропозицією є розроблений метод асиметричного шифрування, що має зменшену обчислювальну складність і, відповідно, зменшений час роботи алгоритму, а також підвищену стійкість проти непрямих криптоаналітичних методів атаки, що використовують математичні вразливості методу шифрування, порівняно із існуючими конкурентними рішеннями.

5.7. Доходи і витрати

Сумарний дохід від розробки даного програмного продукту складається з доходу від продажу ліцензії на використання створеного програмного забезпечення та його підтримки.

Передбачається продаж двох типів ліцензії:

- перший тип ліцензії для комерційного використання програмного забезпечення;
- другий тип ліцензії для приватного використання програмного забезпечення, що накладає обмеження у правах використання програмного забезпечення лише в особистій роботі, тобто із забороною експлуатації даного програмного продукту в комерційних цілях.

Крім того, обидві ліцензії забороняють вивчення, внесення змін, тиражування, розповсюдження та перепродаж вихідного коду програми або його частин.

Технічна підтримка передбачає зворотній зв’язок із розробником з подачею скарг на некоректну роботу програмного забезпечення, та доступ до оновлень. В окремих випадках підписані угоди із крупними компаніями передбачають особливі умови співпраці та підтримки програмного забезпечення, що описуються індивідуально.

Серед витрат на створення даного програмного продукту варто зазначити:

- утримання персоналу для розробки, тестування та подальшої підтримки програмного забезпечення;
- утримання приміщення для розміщення офісу;
- податкові витрати;
- послуги юриста, бухгалтера;
- розміщення програмного продукту на торгових цифрових платформах;
- розповсюдження продукту, реклама.

Детально доходи та витрати від проєкту описано у табл. 5.2.

Таблиця 5.2

Доходи та витрати проєкту

	1-й місяць, т. $	*2-й місяць, т. $*	*3-й місяць, т. $*	*4-й місяць, т. $*	*5-й місяць, т. $*	*6-й місяць, т. $*
Витрати	10	10	10	10	10	10
Заплановані доходи	0	0	0	0	0	0
Результат без оподаткування	–10	–10	–10	–10	–10	–10
	7-й місяць, т. $	*8-й місяць, т. $*	*9-й місяць, т. $*	*10-й місяць, т. $*	*11-й місяць, т. $*	*12-й місяць, т. $*
Витрати	20	15	15	15	15	15
Заплановані доходи	30	35	37	40	42	50
Результат без оподаткування	20	25	27	30	32	40

Загальна сума витрат, т. $	*Загальна сума доходів, т. $*	*Загальний результат без оподаткування, т. $*
155	234	79

5.8. Бізнес-модель

Результати створеного стартап-проєкту буде підсумовано як бізнес-модель за шаблоном Lean Canvas.

Споживачі: невеликі компанії, які створюють програмне забезпечення, що використовує особисті конфіденційні дані користувачів, а також надає можливості обміну інформацією між користувачами, та

користувачі, яким необхідно мати додатковий захист своїх особливо важливих даних по власним причинам.

Проблема: постійне ускладнення стандартів надійності шифрування призводить до збільшення обчислювальної складності методів асиметричного шифрування, що зменшує їх придатність до використання.

Рішення: програмний модуль, що реалізує створений метод асиметричного шифрування.

Унікальна ціннісна пропозиція: розроблений метод асиметричного шифрування, що має зменшену обчислювальну складність і, відповідно, зменшений час роботи алгоритму, а також підвищену стійкість проти непрямих криптоаналітичних методів атаки, що використовують математичні вразливості методу шифрування, порівняно із існуючими конкурентними рішеннями.

Потоки доходів: доходи від продажу ліцензій; доходи від підтримки програмного забезпечення.

Структура витрат:

- утримання персоналу для розробки, тестування та подальшої підтримки програмного забезпечення;
- утримання приміщення для розміщення офісу;
- податкові витрати;
- послуги юриста, бухгалтера;
- розміщення програмного продукту на торгових цифрових платформах;
- розповсюдження продукту, реклама.

Канали розповсюдження: через рекламну кампанію продукту.

Ключові метрики: кількість проданих ліцензій.

Прихована перевага: сумісність із продуктами, що використовують класичну реалізацію методу асиметричного шифрування RSA.

Бізнес-модель наведена у зведеному вигляді у табл. 5.3.

Таблиця 5.3

Канва бізнес-моделі

<table>
<tr>
<td>Проблема
повільна генерація ключів шифрування

недосконаліть стандартних реалізацій методів шифрування

Розвиток методів криптоаналізу</td>
<td>Рішення
програмний модуль, що реалізує модифікований метод шифрування RSA</td>
<td>Унікальна ціннісна пропозиція
модифікований метод асиметричного шифрування RSA, що при збереженні криптостійкості дає виграш у швидкості роботи</td>
<td>Прихована перевага
поєднуваність використання даного програмного продукту з будь-якими системами, які використовують інші шифрувальні програми</td>
</tr>
<tr>
<td>Споживачі
компанії, що працюють з даними клієнтів і користувачі, що хочуть посилити захист своїх систем</td>
<td>Ключові метрики
кількість проданих ліцензій</td>
<td>Потоки доходів
доходи від продажу ліцензій
доходи від підтримки програмного забезпечення</td>
<td>Канали розповсюдження
через рекламну кампанію продукту</td>
</tr>
<tr>
<td colspan="4">Структура витрат
утримання персоналу для розробки та підтримки програмного забезпечення
податкові витрати
послуги юриста, бухгалтера
розповсюдження продукту, реклама</td>
</tr>
</table>

Отже, підсумовуючи дані у табл. 5.3, можна зробити висновок, що запропонований проєкт, що реалізує розроблений у даній дисертації метод асиметричного шифрування, є перспективним для реалізації.

5.9. Висновки

У даному розділі було створено стартап-проєкт, що реалізує програмний продукт із використанням створеного методу асиметричного шифрування.

Було проведено аналіз поточної ситуації у сфері асиметричної криптографії та побудовано відповідне дерево проблем. Також було виділено основні зацікавлені сторони у вирішенні існуючих проблем, та їх ступінь впливу на вирішення даних проблем.

В результаті проведеного аналізу було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених сторін, у ньому було виділено унікальну ціннісну пропозицію.

Також було проаналізовано сегменти ринку криптографічних продуктів з метою визначення майбутніх клієнтів. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту.

У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його подальший дохід від проєкту.

ВИСНОВКИ

Отже, виходячи із результатів проведеного дослідження, що було виконане у даній магістерській дисертації, можна зробити наступні висновки.

1. Проаналізовано роботу методу асиметричного шифрування RSA, його криптостійкість та ефективність використання. Також проаналізовано вразливості та слабкі місця даного методу, способи криптоаналітичних атак, до яких він вразливий, а саме методи факторизації, непрямі методи атаки, та методи атаки по стороннім каналам. Далі проаналізовано існуючі методи модифікацій методу RSA, що направлені на покращення показників обчислювальної складності роботи алгоритму або на посилення його стійкості до окремих видів криптоаналітичних атак.
2. Запропоновано асиметричний метод шифрування, за основу якого було взято класичний метод асиметричного шифрування RSA. В якості модифікації було використано складений модуль та наслідок з Китайської теореми про лишки з метою зменшити обчислювальну складність методу та підвищити стійкість проти непрямих криптоаналітичних методів атаки. Проаналізовано застосування існуючих методів криптографічних атак, таких як атака методом решета числового поля, атака з методом еліптичної кривої, атака Вінера, атака із використанням методу на решітчастій основі, атака з використанням малої експоненти, атака з використанням частково відомих біт до запропонованого методу асиметричного шифрування та вплив внесених модифікацій на резистентність до них. Також змодельовано зміну ефективності проведення атаки із використанням даних методів в залежності від кількості простих множників у

складеному модулі. Аналіз атак показав, всі вони втрачають ефективність із збільшенням кількості простих множників у модулі або не чутливі до нього. Аналіз атаки грубою силою показав, що шанс її успішності із збільшенням кількості простих множників у модулі зростає у 2 рази після додаванням кожного нового множника, проте атаки такого типу в разі використання ключів актуальної довжини, тобто 1024 біта і більше, не ефективні.

3. Проведено аналіз підходящих інструментів та технологій для розроблення програмного забезпечення, що реалізує запропонований метод асиметричного шифрування, та дозволяє проводити аналіз роботи розробленого методу. Прийнято рішення розроблювати дане програмне забезпечення у вигляді програмного модуля на мові Python із використанням бібліотек Matplotlib та NumPy. Розроблено та проаналізовано відповідну архітектуру даного програмного забезпечення та основні реалізовані функції.
4. Проведено експериментальне дослідження ефективності створеного методу асиметричного шифрування при використанні складеного модуля різної довжини. В результаті проведеного дослідження виведено формулу отримання оптимальної кількості великих простих чисел у складеному модулі, використання якої дозволяє отримати найменший час роботи виконання алгоритму, тобто найкращий результат з точки зору зменшення обчислювальної складності.
5. Проведено порівняння часу роботи запропонованого методу асиметричного шифрування із часом роботи класичного алгоритму RSA. В результаті проведеного дослідження було доведено, що запропонований метод має меншу обчислювальну складність. За допомогою методу інтерполяції було знайдено

функцію зміни значення відношення обчислювальної складності даних методів шифрування, а саме поліном п’ятої степені. Аналіз графіка даної функції продемонстрував, що найкращий результат з точки зору зменшення обчислювальної складності запропонований метод асиметричного шифрування досягає на інтервалі [4000; 5000], а далі значення обчислювальної складності зазначених методів шифрування починають наближатись один до одного.

Подальшими напрямками роботи є експериментальне дослідження криптостійкості запропонованого методу асиметричного шифрування.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Diffie W., Hellman M.E. New Directions in Cryptography IEEE Trans. Inf. Theory / F. Kschischang — IEEE, 1976.
2. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM — New York City: ACM, 1978.
3. Онацкий А.В., Йона Л.Г. Асимметричные методы шифрования. – Модуль 2 Криптографические методы защиты информации в телекоммуникационных системах и сетях: учеб. пособие, 2011 – 148 с.
4. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходный код на С / Б. Шнайер. - М.: Вильямс, 2016 – 842 с.
5. Bakhtiari M., Maarof M.A. Serious security weakness in RSA cryptosystem, International Journal of Computer Science Issues, 2012 – сс. 175-178.
6. RSA-числа [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/RSA, дата доступу 15.12.2021.
7. Pomerance C., Lenstra H., Analysis and comparision of some integer factoring algorithms, Computational methods in number theory, №1, 1982 – сс. 89-139.
8. Wiener M.J., Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, 1990.
9. Joye M., Olivier F. Side-channel analysis, Encyclopedia of Cryptography and Security, 2005, сс. 571–576.
10. Gutmann P., Naccache D., Palmer C.C. Side Channel attacks on cryptographic software, copublished by the IEEE computer and reliability societies, 2009.
11. Großschadl J. The Chinese Remainder Theorem and its Application in a High-Speed RSA Crypto Chip.

12. Koyama K. New public-key schemes based on elliptic curves over the ring. Annual International Cryptology Conference. Berlin, Heidelberg, 1991 – cc. 252-266.
13. Demytko N. A new elliptic curve based analogue of RSA. Workshop on the Theory and Application of of Cryptographic Techniques. Berlin, Heidelberg, 1993 – cc. 40-49.
14. Koyama K. Fast RSA-type schemes based on singular cubic curves $y^2 + ax = x^3 (\mathrm{mod}\ n)$. International Conference on the Theory and Applications of Cryptographic Techniques, Berlin, Heidelberg, 1995 – cc. 329-340.
15. Takagi T. Fast RSA-type cryptosystems using n-adic expansion. Annual International Cryptology Conference, Berlin, Heidelberg, 1997 – cc. 372-384.
16. Takagi T. Fast RSA-type cryptosystem modulo $p^k q$. Annual International Cryptology Conference, Berlin, Heidelberg, 1998 – cc. 318-326.
17. Lim S. et al. A Generalized Takagi-Cryptosystem with a Modulus of the Form $p^k q^l$. International Conference on Cryptology in India, Berlin, Heidelberg, 2000 – cc. 283-294.
18. Collins T. et al. Public key cryptographic apparatus and method, пат. 5848159 США, 1998.
19. Schindler, Werner. A Timing Attack against RSA with the Chinese Remainder Theorem, Cryptographic Hardware and Embedded Systems CHES 2000, volume 1965 of Lecture Notes in Computer Science, Berlin, Heidelberg, 2000 – cc.109–124.
20. Stinson D.R. Cryptography : Theory and Practice. CRC Press LLC, 1995.
21. Lenstra H.W. Elliptic curves and number-theoretic algorithms, International Congress of Mathematicians, 1986 – cc. 99-120.
22. Lehman R.S., Factoring large integers, Math. Comp., 1976 – cc. 637-646.

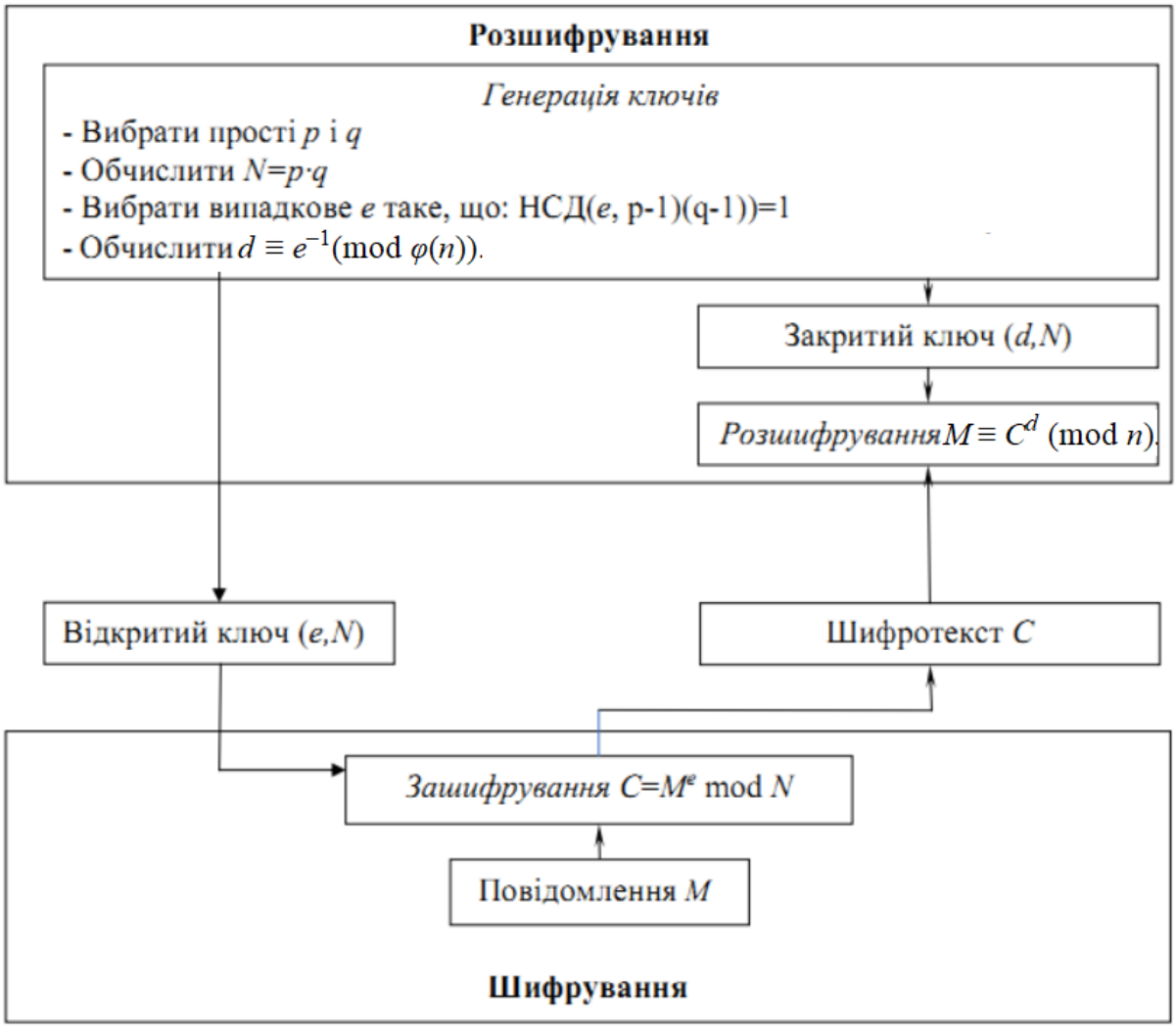
23. Boneh D., Durfee G. Cryptanalysis of RSA with private key d less than $x^{0,292}$, IEEE transactions on Information Theory, 2000 – cc. 1339-1349.
24. Ernst M., Jochemsz E., May A., de Weger B. Partial Key Exposure Attacks on RSA up to Full Size Exponents. Advances in Cryptology – Proceedings of EURO-CRYPT, 2005, cc. 371–387.
25. Hinek M.J. On the security of multi-prime RSA //Journal of Mathematical Cryptology, 2008 – cc. 117-147.
26. Hinek M.J., Low M.K., Teske E. On some attacks on multi-prime RSA, International Workshop on Selected Areas in Cryptography, Berlin, Heidelberg, 2003 – cc. 385-404.
27. Python [Електронний ресурс] — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python, дата доступу 15.12.2021.
28. Matplotlib [Електронний ресурс] — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/ Matplotlib, дата доступу 15.12.2021.
29. NumPy [Електронний ресурс] — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/NumPy, дата доступу 15.12.2021.

ДОДАТКИ

Додаток 1

Копії графічних матеріалів

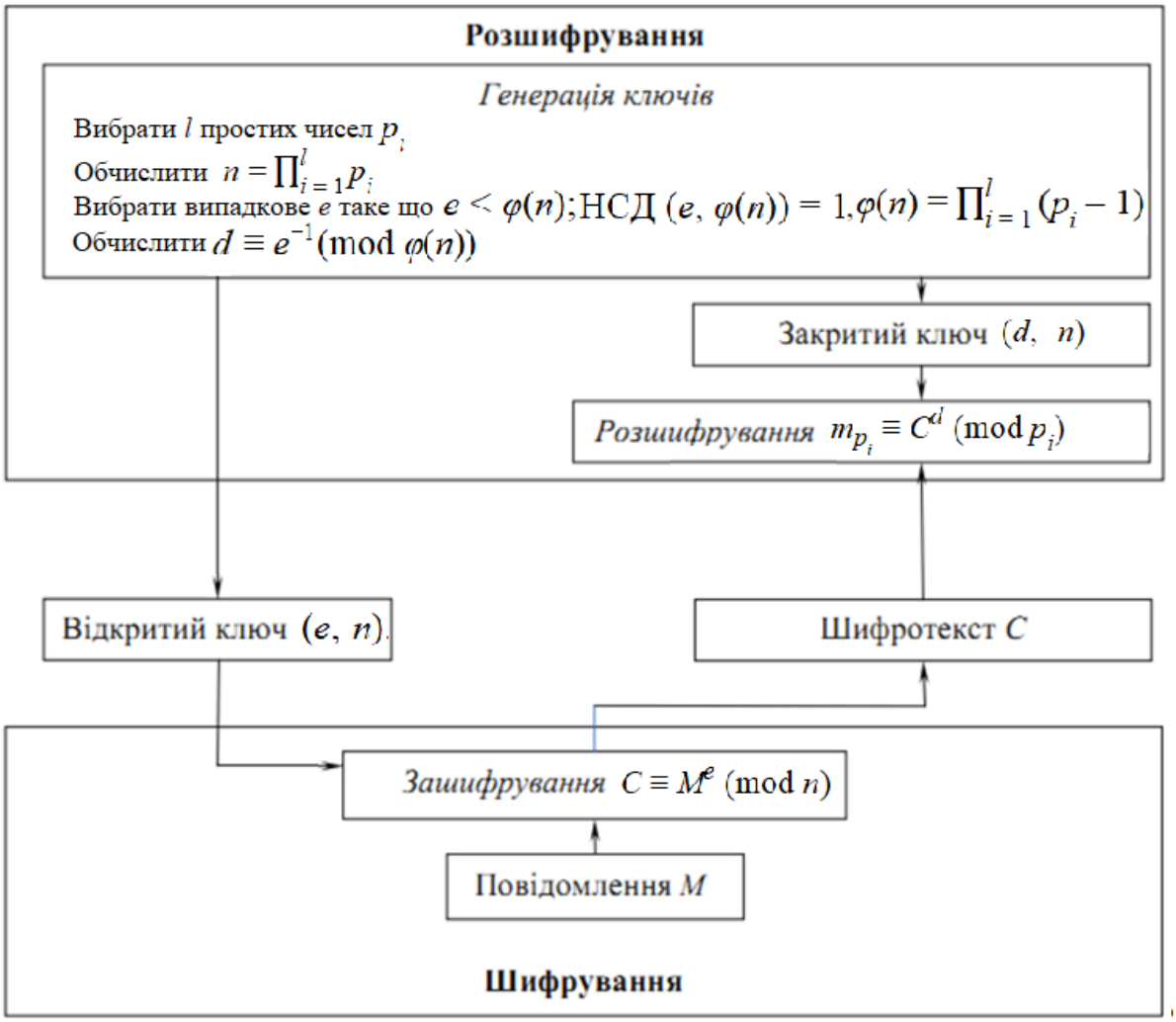
Схема роботи класичного методу RSA



Квітка Олександр Вячеславович

КП-01мп

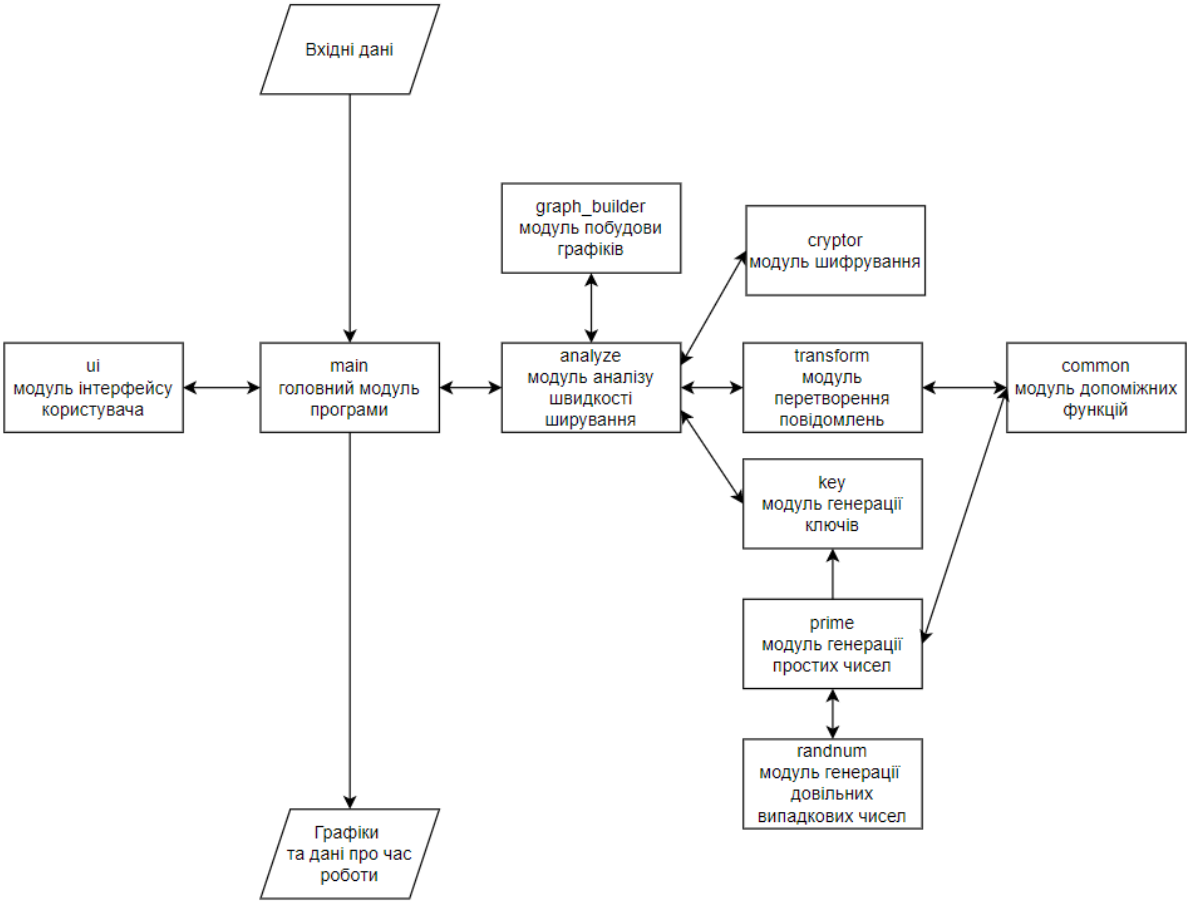
Схема роботи запропонованого методу асиметричного шифрування



Квітка Олександр

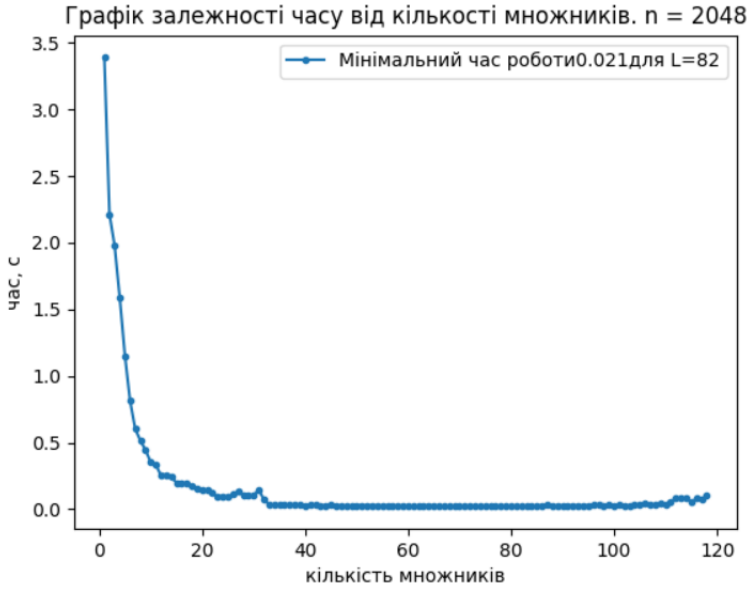
КП-01мп

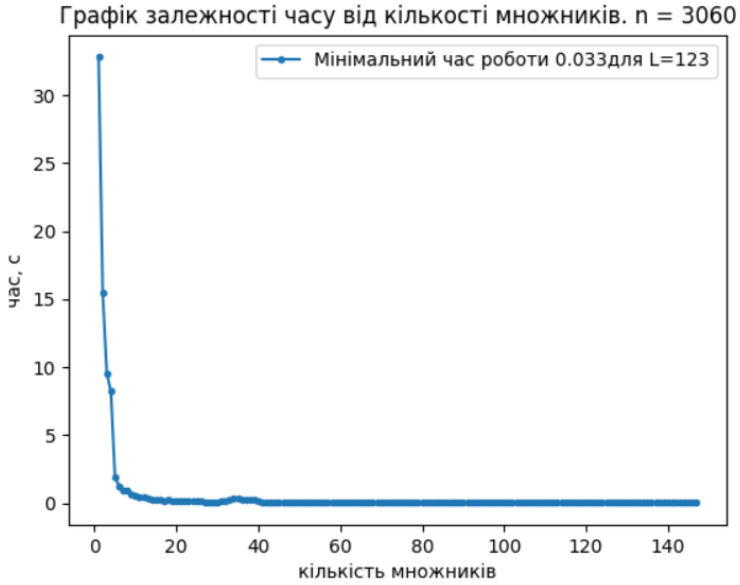
Структурна схема програмного забезпечення

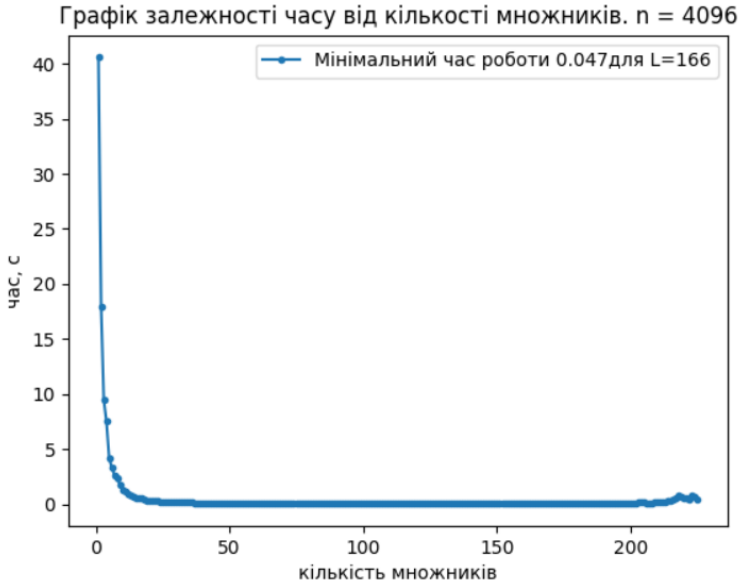


Квітка Олександр

КП-01мп

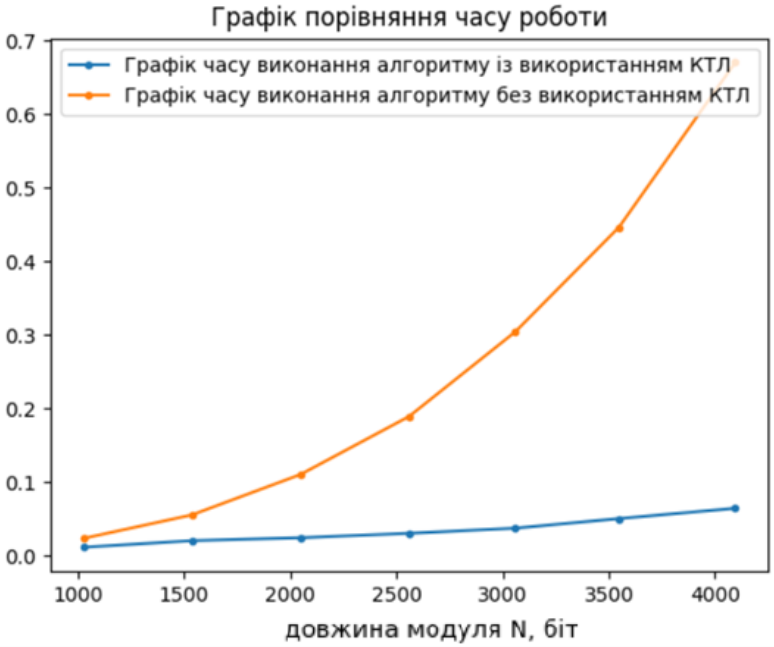




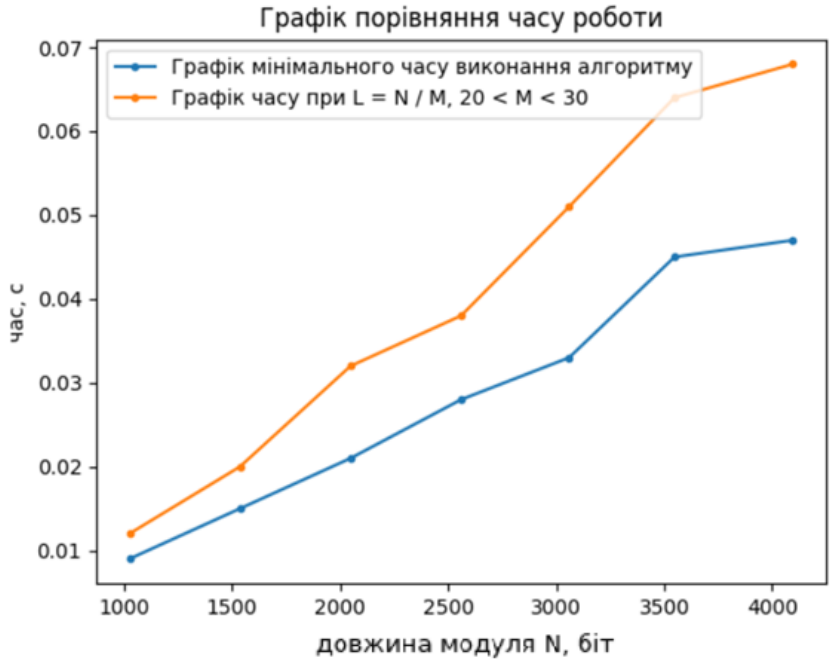


Квітка
Олександр
КП-01мп

Графіки часу виконання алгоритму із використанням КТЛ та без використання КТЛ



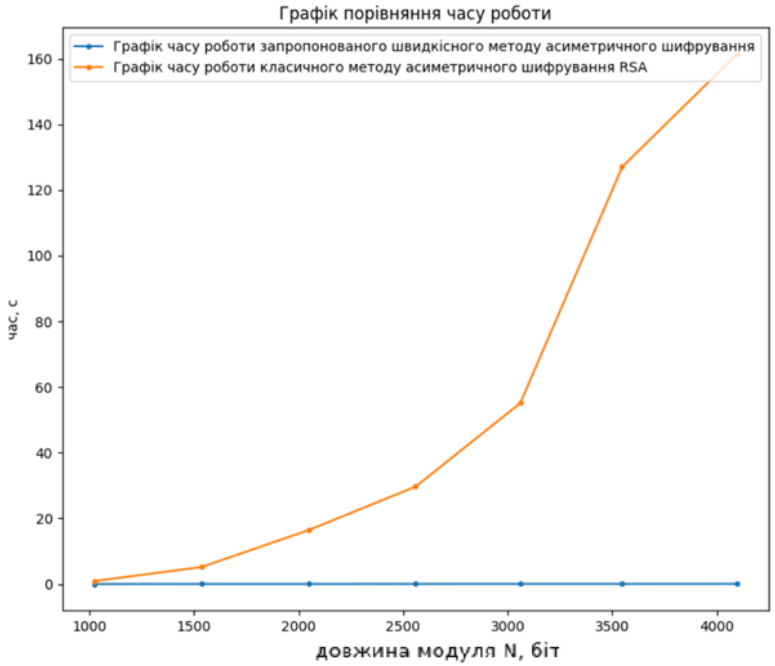
Графіки часу виконання реалізації запропонованого методу шифрування та часу мінімального виконання



Квітка Олександр

КП-01мп

Графіки часу виконання класичного методу RSA та запропонованого методу шифрування



Графік зміни значення відношення обчислювальної складності класичного алгоритму RSA до обчислювальної складності запропонованого методу асиметричного шифрування

Квітка Олександр

КП-01мп

Додаток 2

Текст програми

```
class NotRelativePrimeError(ValueError):
    def __init__(self, a: int, b: int, d: int, msg: str = '') -> None:
        super().__init__(msg or "%d and %d are not relatively prime,
divider=%i" % (a, b, d))
        self.a = a
        self.b = b
        self.d = d


def bit_size(num: int) -> int:
    """
    Number of bits needed to represent a integer excluding any prefix
    0 bits.

    Usage::

        >>> bit_size(1023)
        10
        >>> bit_size(1024)
        11
        >>> bit_size(1025)
        11

    :param num:
        Integer value. If num is 0, returns 0. Only the absolute value of
the
        number is considered. Therefore, signed integers will be abs(num)
        before the number's bit length is determined.
    :returns:
        Returns the number of bits in the integer.
    """

    try:
        return num.bit_length()
    except AttributeError as ex:
        raise TypeError('bit_size(num) only supports integers, not %r' %
type(num)) from ex


def byte_size(number: int) -> int:
    """
    Returns the number of bytes required to hold a specific long number.

    The number of bytes is rounded up.

    Usage::

        >>> byte_size(1 << 1023)
        128
        >>> byte_size((1 << 1024) - 1)
        128
        >>> byte_size(1 << 1024)
        129

    :param number:
        An unsigned integer
    :returns:
        The number of bytes required to hold a specific long number.
    """
    if number == 0:
        return 1
    return ceil_div(bit_size(number), 8)
```

```
def ceil_div(num: int, div: int) -> int:
    """
    Returns the ceiling function of a division between `num` and `div`.

    Usage::

        >>> ceil_div(100, 7)
        15
        >>> ceil_div(100, 10)
        10
        >>> ceil_div(1, 4)
        1

    :param num: Division's numerator, a number
    :param div: Division's divisor, a number

    :return: Rounded up result of the division between the parameters.
    """
    quanta, mod = divmod(num, div)
    if mod:
        quanta += 1
    return quanta


def extended_gcd(a: int, b: int):
    """Returns a tuple (r, i, j) such that r = gcd(a, b) = ia + jb
    """
    # r = gcd(a,b) i = multiplicitive inverse of a mod b
    #      or      j = multiplicitive inverse of b mod a
    #  Neg return values for i or j are made positive mod b or a
respectively
    # Iterateive Version is faster and uses much less stack space
    x = 0
    y = 1
    lx = 1
    ly = 0
    oa = a  # Remember original a/b to remove
    ob = b  # negative values from return results
    while b != 0:
        q = a // b
        (a, b) = (b, a % b)
        (x, lx) = ((lx - (q * x)), x)
        (y, ly) = ((ly - (q * y)), y)
    if lx < 0:
        lx += ob  # If neg wrap modulo orignal b
    if ly < 0:
        ly += oa  # If neg wrap modulo orignal a
    return a, lx, ly  # Return only positive values


def inverse(x: int, n: int) -> int:
    """Returns the inverse of x % n under multiplication, a.k.a x^-1 (mod
n)

    >>> inverse(7, 4)
    3
    >>> (inverse(143, 4) * 143) % 4
    1
    """

    (divider, inv, _) = extended_gcd(x, n)

    if divider != 1:
        raise NotRelativePrimeError(x, n, divider)
```

```
    return inv

from functools import reduce


def encrypt(message: int, pubkey: int) -> int:
    (n, ekey) = pubkey
    if message < 0:
        raise ValueError('Only non-negative numbers are supported')

    if message > n:
        raise OverflowError("The message %i is too long for n=%i" %
(message, n))

    return pow(message, ekey, n)


def decrypt(cyphertext: int, privkey: int) -> int:
    (n, dkey) = privkey
    return pow(cyphertext, dkey, n)


def chinese_remainder_decrypt(cyphertext: int, privkey: int) -> int:
    (primes, dkey) = privkey
    messages = []
    for prime in primes:
        messages.append(pow(pow(cyphertext, 1, prime), pow(dkey, 1, prime -
1), prime))
    return chinese_remainder(primes, messages)


def chinese_remainder(modulo, cyphers):
    sum = 0
    prod = reduce(lambda acc, b: acc * b, modulo)
    for n_i, a_i in zip(modulo, cyphers):
        p = prod // n_i
        sum += a_i * mul_inv(p, n_i) * p
    return sum % prod


def mul_inv(a, b):
    b0 = b
    x0, x1 = 0, 1
    if b == 1:
        return 1
    while a > 1:
        q = a // b
        a, b = b, a % b
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += b0
    return x1

import random
import analyse

MIN_SIZE = 2
MAX_SIZE = 1000
DEFAULT_BITS = 1024
MESSAGE_COUNT = 100
```

```
class Demo:

    def __init__(self, scenario_number, nbits=DEFAULT_BITS,
message_count=MESSAGE_COUNT, min_size=MIN_SIZE,
                 max_size=MAX_SIZE):
        self.messages = self.message_generator(message_count, min_size,
max_size)
        if scenario_number == 1:
            self.scenario1(nbits)
        elif scenario_number == 2:
            self.scenario2()
        elif scenario_number == 3:
            self.scenario3()
        elif scenario_number == 4:
            self.scenario4()
        elif scenario_number == 5:
            self.scenario5()

    @staticmethod
    def message_generator(n, min_size, max_size):
        messages = []
        for i in range(1, n + 1):
            messages.append(random.randrange(min_size, max_size))
        return messages

    def scenario1(self, nbits):
        times = [analyse.all_prime_count_test(self.messages, nbits)]
        title = "Графік залежності часу від кількості множників. n = " +
str(nbits)
        analyse.draw_graphs(times, title)

    def scenario2(self):
        dict_list = []
        dict_list.append(MIN_TIME)
        speed_rsa_times = {}
        dict_list.append(speed_rsa_times)
        analyse.draw_graphs_by_dict(speed_rsa_times, MIN_TIME)

    def scenario3(self):
        speed_rsa_times = {}
        no_crt_times = {}
        for modulo in MODULO:
            speed_rsa_times.update({modulo:
analyse.average_time_speed_rsa(self.messages, modulo)})
            no_crt_times.update({modulo:
analyse.average_time_no_crt_rsa(self.messages, modulo)})
        analyse.draw_graphs_by_dict2(speed_rsa_times, no_crt_times)

    def scenario4(self):
        speed_rsa_times = {}
        classic_rsa_times = {}
        for modulo in MODULO:
            print(MODULO)
            speed_rsa_times.update({modulo:
analyse.average_time_speed_rsa(self.messages, modulo)})
            classic_rsa_times.update({modulo:
analyse.average_time_classic_rsa(self.messages, modulo)})
        print(speed_rsa_times)
        print(classic_rsa_times)
        analyse.draw_graphs_by_dict3(speed_rsa_times, classic_rsa_times)

    def scenario5(self):
        speed_rsa_times = {}
```

```
        for i in range(1, 2, 1):
            k = i * 1024
            print(k)
            speed_rsa_times.update({k:
analyse.average_time_speed_rsa(self.messages, k)})
        print(speed_rsa_times)
        analyse.draw_graphs_by_dict(speed_rsa_times)

import math
import random
from rsa import common, prime

DEFAULT_EXPONENT = 65537
MIN_LIMIT = 20
MAX_LIMIT = 30


def keys_crt(nbits: int,
             prime_num: int = 2,
             accurate: bool = False,
             exponent: int = DEFAULT_EXPONENT,
             prime_calculation: bool = True):
    if nbits < 16:
        raise ValueError('Key too small')
    if prime_calculation:
        prime_num = prime_number_generator(nbits)
    if prime_num < 2:
        raise ValueError('prime number too small')
    if prime_num > nbits / 4:
        raise ValueError('prime number too large')

    # Generate the key components
    (n, e, d, primes) = gen_keys(nbits, prime_num, accurate=accurate,
exponent=exponent)

    return (
        (n, e),
        (primes, d)
    )


def keys(nbits: int,
         prime_num: int,
         accurate: bool = False,
         exponent: int = DEFAULT_EXPONENT,
         prime_calculation: bool = True):
    if nbits < 16:
        raise ValueError('Key too small')
    if prime_calculation:
        prime_num = prime_number_generator(nbits)
    if prime_num < 2:
        raise ValueError('prime number too small')
    if prime_num > nbits / 4:
        raise ValueError('prime number too large')

    # Generate the key components
    (n, e, d, primes) = gen_keys(nbits, prime_num, accurate=accurate,
exponent=exponent)

    return (
        (n, e),
        (n, d)
    )
```

```
def gen_keys(nbits: int,
             prime_num: int,
             accurate: bool = False,
             exponent: int = DEFAULT_EXPONENT):
    # Regenerate p and q values, until calculate_keys doesn't raise a
ValueError.
    while True:
        primes = find_primes(nbits, prime_num, accurate)
        try:
            (e,      d)      =      calculate_keys_custom_exponent(primes,
exponent=exponent)
            break
        except ValueError:
            pass
    n = math.prod(primes)
    return n, e, d, primes


def find_primes(total_bits: int, counter: int = 2, accurate: bool = False):
    nbits = total_bits // counter
    primes = []


    # Keep choosing other primes until they match our requirements.
    while not is_acceptable(primes, total_bits, accurate):
        primes.clear()
        for i in range(counter):
            primes.append(prime.get_prime(nbits))

    primes.sort(reverse=True)
    return primes


def is_acceptable(primes, total_bits: int, accurate: bool) -> bool:
    if len(primes) != len(set(primes)) or len(primes) == 0:
        return False

    if not accurate:
        return True

    # Make sure we have just the right amount of bits
    found_size = common.bit_size(math.prod(primes))
    return total_bits == found_size


def prime_number_generator(nbits):
    return nbits // random.randint(MIN_LIMIT, MAX_LIMIT)


def calculate_keys_custom_exponent(primes, exponent: int):
    phi_n = 1
    for _prime in primes:
        phi_n *= (_prime - 1)

    try:
        d = common.inverse(exponent, phi_n)
    except common.NotRelativePrimeError as ex:
        raise common.NotRelativePrimeError(
            exponent, phi_n, ex.d,
            msg="e   (%d)   and   phi_n   (%d)   are   not   relatively   prime
(divider=%i)" %
                (exponent, phi_n, ex.d))
```

```
    if (exponent * d) % phi_n != 1:
        raise ValueError("e (%d) and d (%d) are not mult. inv. modulo "
                         "phi_n (%d)" % (exponent, d, phi_n))

    return exponent, d

from rsa import common, randnum


def get_prime(nbits):
    assert nbits > 3  # the loop wil hang on too small numbers

    while True:
        integer = randnum.read_random_odd_int(nbits)

        # Test for primeness
        if is_prime(integer):
            return integer


def is_prime(number):

    # Check for small numbers.
    if number < 10:
        return number in {2, 3, 5, 7}

    # Check for even numbers.
    if not (number & 1):
        return False

    # Calculate minimum number of rounds.
    k = get_primality_testing_rounds(number)

    # Run primality testing with (minimum + 1) rounds.
    return miller_rabin_primality_testing(number, k + 1)


def get_primality_testing_rounds(number):
    # Calculate number bitsize.
    bitsize = common.bit_size(number)
    # Set number of rounds.
    if bitsize >= 1536:
        return 3
    if bitsize >= 1024:
        return 4
    if bitsize >= 512:
        return 7
    # For smaller bitsizes, set arbitrary number of rounds.
    return 10


def miller_rabin_primality_testing(n, k):
    # prevent potential infinite loop when d = 0
    if n < 2:
        return False

    # Decompose (n - 1) to write it as (2 ** r) * d
    # While d is even, divide it by 2 and increase the exponent.
    d = n - 1
    r = 0

    while not (d & 1):
        r += 1
        d >>= 1
```

```
    # Test k witnesses.
    for _ in range(k):
        # Generate random integer a, where 2 <= a <= (n - 2)
        a = randnum.randint(n - 3) + 1

        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue

        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == 1:
                # n is composite.
                return False
            if x == n - 1:
                # Exit inner loop and continue with next witness.
                break
        else:
            # If loop doesn't break, n is composite.
            return False

    return True

import os
import struct
from rsa import common, transform


def read_random_bits(nbits: int) -> bytes:
    """Reads 'nbits' random bits.

    If nbits isn't a whole number of bytes, an extra byte will be appended
with
    only the lower bits set.
    """

    nbytes, rbits = divmod(nbits, 8)

    # Get the random bytes
    randomdata = os.urandom(nbytes)

    # Add the remaining random bits
    if rbits > 0:
        randomvalue = ord(os.urandom(1))
        randomvalue >>= (8 - rbits)
        randomdata = struct.pack("B", randomvalue) + randomdata

    return randomdata


def read_random_int(nbits: int) -> int:
    """Reads a random integer of approximately nbits bits.
    """

    randomdata = read_random_bits(nbits)
    value = transform.bytes2int(randomdata)

    # Ensure that the number is large enough to just fill out the required
    # number of bits.
    value |= 1 << (nbits - 1)

    return value
```

```
def read_random_odd_int(nbits: int) -> int:
    """Reads a random odd integer of approximately nbits bits.

    >>> read_random_odd_int(512) & 1
    1
    """

    value = read_random_int(nbits)

    # Make sure it's odd
    return value | 1


def randint(maxvalue: int) -> int:
    """Returns a random integer x with 1 <= x <= maxvalue

    May take a very long time in specific situations. If maxvalue needs N
bits
    to store, the closer maxvalue is to (2 ** N) - 1, the faster this
function
    is.
    """

    bit_size = common.bit_size(maxvalue)

    tries = 0
    while True:
        value = read_random_int(bit_size)
        if value <= maxvalue:
            break

        if tries % 10 == 0 and tries:
            # After a lot of tries to get the right number of bits but
still
            # smaller than maxvalue, decrease the number of bits by 1.
That'll
            # dramatically increase the chances to get a large enough
number.
            bit_size -= 1
        tries += 1

    return value

import math


def bytes2int(raw_bytes: bytes) -> int:
    r"""Converts a list of bytes or an 8-bit string to an integer.

    When using unicode strings, encode it to some encoding like UTF8 first.

    >>> (((128 * 256) + 64) * 256) + 15
    8405007
    >>> bytes2int(b'\x80@\x0f')
    8405007

    """
    return int.from_bytes(raw_bytes, 'big', signed=False)


def int2bytes(number: int, fill_size: int = 0) -> bytes:
    """
    Convert an unsigned integer to bytes (big-endian)::
```

```
    Does not preserve leading zeros if you don't specify a fill size.

    :param number:
        Integer value
    :param fill_size:
        If the optional fill size is given the length of the resulting
        byte string is expected to be the fill size and will be padded
        with prefix zero bytes to satisfy that length.
    :returns:
        Raw bytes (base-256 representation).
    :raises:
        ``OverflowError`` when fill_size is given and the number takes up
more
        bytes than fit into the block. This requires the ``overflow``
        argument to this function to be set to ``False`` otherwise, no
        error will be raised.
    """

    if number < 0:
        raise ValueError("Number must be an unsigned integer: %d" % number)

    bytes_required = max(1, math.ceil(number.bit_length() / 8))

    if fill_size > 0:
        return number.to_bytes(fill_size, 'big')

    return number.to_bytes(bytes_required, 'big')

import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from rsa import key_classic as kc, key_mod as km, cryptor


def run_classic_rsa(message, nbits, prime_count):
    (pubkey, privkey) = kc.newkeys(nbits)
    crypto = cryptor.encrypt(message, pubkey)
    return cryptor.decrypt(crypto, privkey)


def run_multiprime_rsa(message, nbits, prime_count):
    (pubkey, privkey) = km.keys(nbits, prime_count)
    crypto = cryptor.encrypt(message, pubkey)
    return cryptor.decrypt(crypto, privkey)


def run_multiprime_crt_rsa(message, nbits, prime_count):
    (pubkey, privkey) = km.keys_crt(nbits, prime_count)
    crypto = cryptor.encrypt(message, pubkey)
    return cryptor.chinese_remainder_decrypt(crypto, privkey)


def run_speed_rsa(message, nbits, prime_count):
    (pubkey, privkey) = km.keys_crt(nbits)
    crypto = cryptor.encrypt(message, pubkey)
    return cryptor.chinese_remainder_decrypt(crypto, privkey)


def time_buffer(messages, nbits, rsa, prime_count: int = 2):
    times = []
    for m in messages:
        start_time = datetime.now()
        rsa(m, nbits, prime_count)
```

```
        times.append((datetime.now() - start_time).total_seconds())
    return times


def average_time(times):
    return round(np.mean(times).item(), 3)


def average_time_speed_rsa(messages, modulo):
    return average_time(time_speed_rsa(messages, modulo))


def average_time_no_crt_rsa(messages, modulo):
    return average_time(time_multiprime_rsa(messages, modulo, 2))


def average_time_classic_rsa(messages, modulo):
    return average_time(time_classic_rsa(messages, modulo))


def time_classic_rsa(messages, nbits):
    return time_buffer(messages, nbits, run_classic_rsa)


def time_multiprime_rsa(messages, nbits, prime_count):
    return time_buffer(messages, nbits, run_multiprime_rsa, prime_count)


def time_multiprime_crt_rsa(messages, nbits, prime_count):
    return     time_buffer(messages,     nbits,     run_multiprime_crt_rsa,
prime_count)


def time_speed_rsa(messages, nbits):
    return time_buffer(messages, nbits, run_speed_rsa)


def all_prime_count_test(messages, nbits):
    times = []
    for i in range(100, 193):
        print(i)
        times.append(average_time(time_multiprime_crt_rsa(messages,   nbits,
i)))
    return times


def draw_graphs(times, title):

    x = np.arange(1, len(times[0]) + 1, 1)
    for time in times:
        plt.plot(x, time, '-o', markersize=3, label="Мінімальний час роботи
" + str(min(time)) +
                                                    "для        L="        +
str(time.index(min(time)) + 3))
    plt.title(title)
    plt.ylabel('час, с')
    plt.xlabel('кількість множників')
    plt.legend(loc="upper right")
    plt.show()


def draw_graphs_by_dict(dictionary1):
    x = dictionary1.keys()
```

```
    plt.plot(x,  dictionary1.values(),  '-o',  markersize=3,  label="Графік
мінімального часу виконання алгоритму")
    plt.plot(x,  dictionary1.values(),  '-o',  markersize=3,  label="Графік
часу при L = N / M, 20 < M < 30")

    plt.title("Графік часу роботи запропонованого методу")
    plt.ylabel('час, с')
    plt.xlabel('довжина модуля N, біт')
    plt.legend(loc="upper right")
    plt.show()
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

АЛГОРИТМІЧНО-ПРОГРАМНИЙ МЕТОД АСИМЕТРИЧНОГО ШИФРУВАННЯ ДАНИХ

Виконав: Квітка Олександр Вячеславович

Керівник: доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович

Київ – 2021

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

Розвиток інформаційно-телекомунікаційних технологій значно ускладнює завдання збереження надійності захисту даних. Для цього необхідно підвищувати ефективність криптографічних алгоритмів.

- Довжина ключів асиметричних алгоритмів, що можуть гарантувати безпечність використання, постійно зростає.
- Збільшення довжини ключів значно уповільнює роботу алгоритмів шифрування.

ПРЕДМЕТ ДОСЛІДЖЕННЯ

Методи асиметричного шифрування даних

ОБ'ЄКТ ДОСЛІДЖЕННЯ

Процес асиметричного шифрування даних

МЕТА ДОСЛІДЖЕННЯ

Вдосконалити процес асиметричного шифрування для роботи з великими ключами шифрування довжиною понад 10 тисяч біт, що забезпечить криптостійкість асиметричного шифрування на найближчі 20 років шляхом зменшення його обчислювальної складності та збільшення резистентності до криптографічних атак.

Наукове завдання: вдосконалити процес асиметричного шифрування шляхом зменшення його обчислювальної складності та збільшення резистентності до криптографічних атак.

Окремі завдання

1. Провести аналіз класичних методу асиметричного шифрування та існуючі модифікації, а також види криптоаналітичних атак, що до них застосовуються.
2. Створити модифікований метод асиметричного шифрування, що має меншу обчислювальну складність, використовуючи досліджені підходи внесення покращень.
3. Проаналізувати переваги та недоліки запропонованого методу асиметричного шифрування.
4. Розробити програмне забезпечення, що реалізує запропонований метод асиметричного шифрування.
5. Експериментально підтвердити доцільність використання розробленого методу.

АНАЛОГИ

Переважна більшість продуктів використовує класичний метод RSA, оскільки більшість модифікацій існує лише у науковому просторі та не підлягають сертифікації. Прикладами таких продуктів є протокол захисту SSL, мережевий протокол SSH та криптографічна програма PGP.

СХЕМА КЛАСИЧНОГО МЕТОДУ ШИФРУВАННЯ RSA

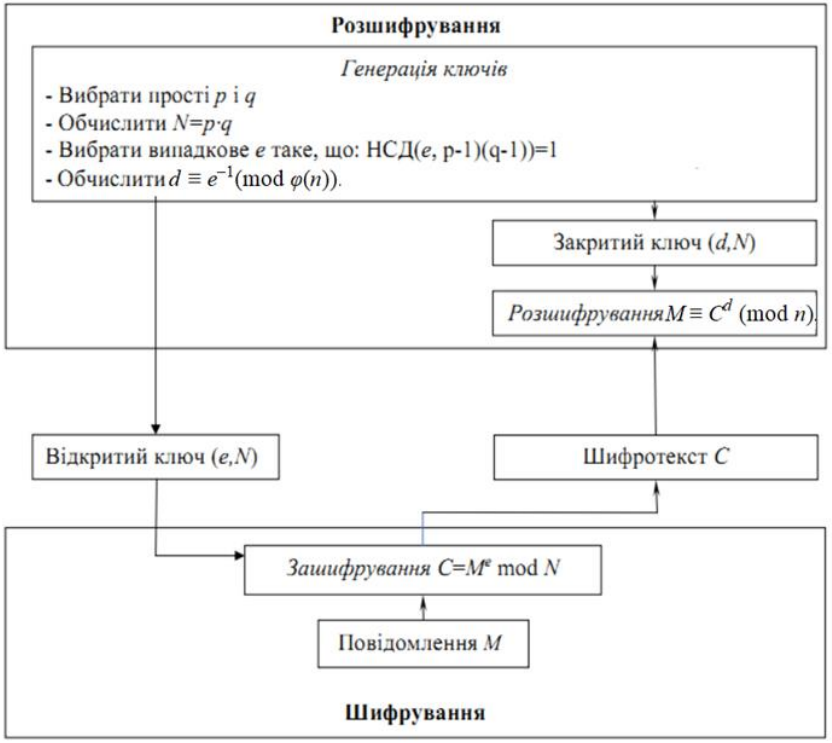
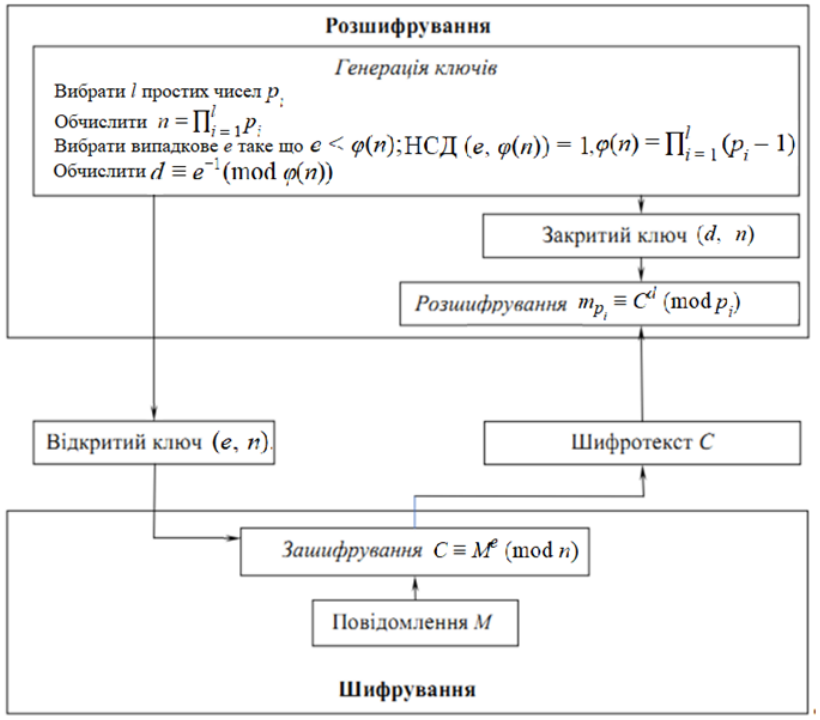


СХЕМА ЗАПРОПОНОВАНОГО МЕТОДУ АСИМТЕРИЧНОГО ШИФРУВАННЯ



ВИКОРИСТАНІ МЕТОДИ МОДИФІКАЦІЇ: СКЛАДЕНИЙ МОДУЛЬ

Для отримання числа n довільним чином генерується l великих простих чисел p_i та обчислюється їх добуток. Кількість великих простих чисел l обмежена як $3 \le l \le \frac{n}{4}$, тобто розрядність кожного окремого множника не менше 4 знаків.

$$n = \prod_{i=1}^{l} p_i$$

ВИКОРИСТАНІ МЕТОДИ МОДИФІКАЦІЇ: НАСЛІДОК З КИТАЙСЬКОЇ ТЕОРЕМИ ПРО ЛИШКИ

Дешифрування відбувається, згідно з теоремою Ейлера, за допомогою операції піднесення до степеня d по модулю p_i для всіх $i \in [1; l]$. Для підвищення швидкості обчислень використовується теорема Ферма.

$$m_{p_i} \equiv C^d \pmod{p_i} = (\mathrm{C} \bmod p_i)^{d \bmod (p_i - 1)} \bmod p_i$$

В кінці необхідно відновити повідомлення M, використовуючи m_{p_i} за допомогою Китайської теореми про лишки. Для цього потрібно вирішити систему рівнянь наступного вигляду:

$$\begin{cases} M \equiv m_{p_1} (\bmod p_1) \\ M \equiv m_{p_2} (\bmod p_2) \\ \dots \\ M \equiv m_{p_l} (\bmod p_l) \end{cases}$$

АНАЛІЗ КРИПТОСТІЙКОСТІ: АТАКА З ВИКОРИСТАННЯМ МЕТОДУ РЕШЕТА ЧИСЛОВОГО ПОЛЯ

Для атак із використанням методів факторизації типу решета числового поля для заданого розміру модуля n безпека однакова для будь-якої кількості множників, а складність факторизації не залежить від зміни їх кількості.

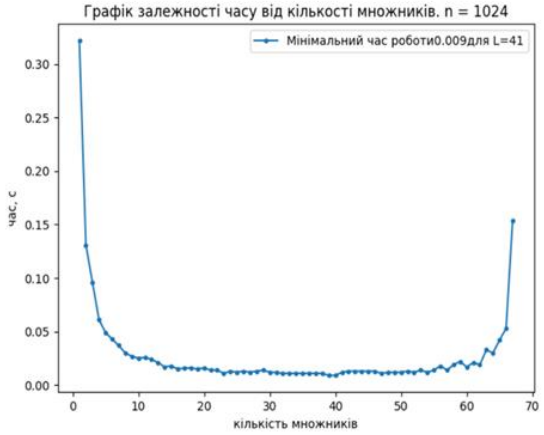
АНАЛІЗ КРИПТОСТІЙКОСТІ: АТАКА З МАЛОЮ ЕКСПОНЕНТОЮ ТА ВІДОМИМИ БІТАМИ

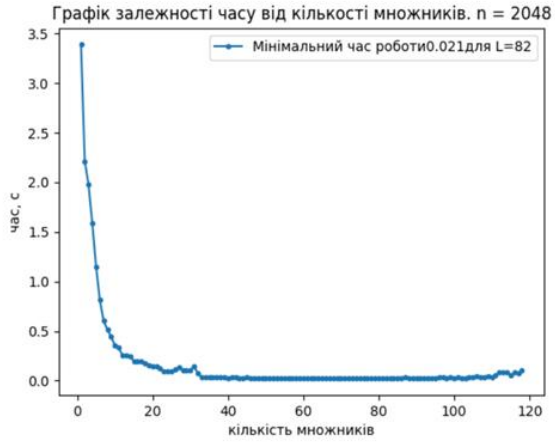
Атаки, що використовують математичні властивості схеми шифрування RSA, такі як метод Вінера, а також атаки з використанням відомих біт приватної або публічної експоненти можливі, навіть ефективні у певних випадках для модуля з трьох або чотирьох простих множників, проте для їх застосування необхідно виконання багатьох умов. Крім того, всі вони втрачають ефективність із збільшенням кількості простих множників у модулі. Однак, використання експонент невеликого розміру не рекомендується.

АНАЛІЗ КРИПТОСТІЙКОСТІ: АТАКА ГРУБОЮ СИЛОЮ

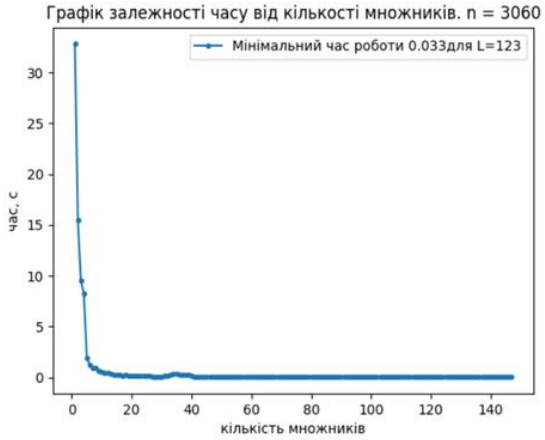
Для атаки грубою силою ймовірність знаходження одного з простих чисел p і q у класичному алгоритмі RSA становить $\frac{1}{2}$, тоді ймовірність факторизації n, що є добутком k простих множників, буде дорівнювати $1 - \frac{1}{2^k - 1}$. Отже із збільшенням кількості простих множників у модулі зростає у 2 рази із кожним новим множником, проте атаки такого типу в разі використання ключів актуальної довжини не ефективні.

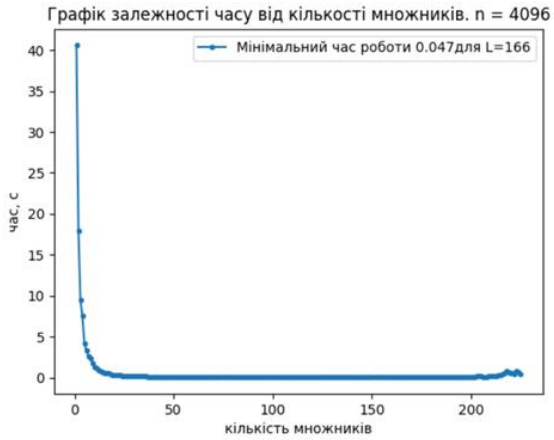
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ШВИДКОСТІ РОБОТИ МЕТОДУ ШИФРУВАННЯ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ КІЛЬКОСТІ ПРОСТИХ МНОЖНИКІВ У МОДУЛІ





ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ШВИДКОСТІ РОБОТИ МЕТОДУ ШИФРУВАННЯ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ КІЛЬКОСТІ ПРОСТИХ МНОЖНИКІВ У МОДУЛІ





ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ШВИДКОСТІ РОБОТИ МЕТОДУ ШИФРУВАННЯ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ КІЛЬКОСТІ ПРОСТИХ МНОЖНИКІВ У МОДУЛІ

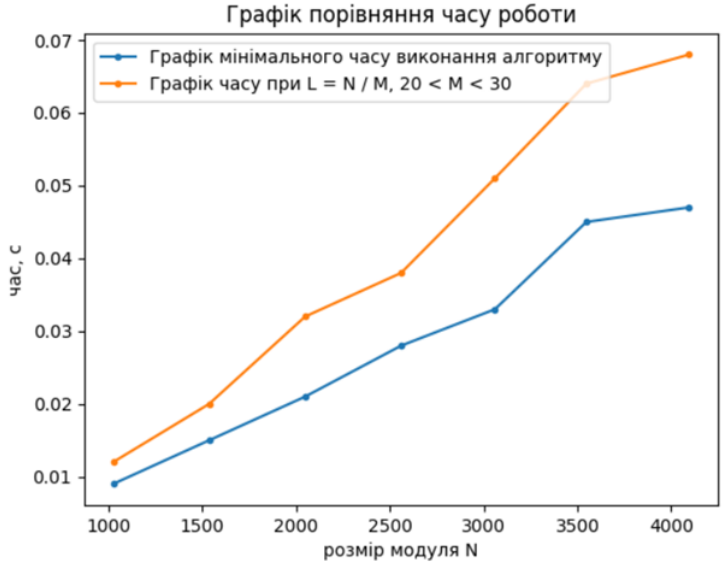
Аналізуючи графіки зміни швидкості роботи можна помітити тенденцію розміщення найменшого значення часу виконання алгоритму на одному інтервалі, пропорційно довжині модуля. Використовуючи отримані значення, можна побудувати наступне відношення:

$$\frac{n}{l} = \frac{1024}{41} \approx \frac{2048}{82} \approx \frac{3060}{123} \approx \frac{4096}{166} \approx 25$$

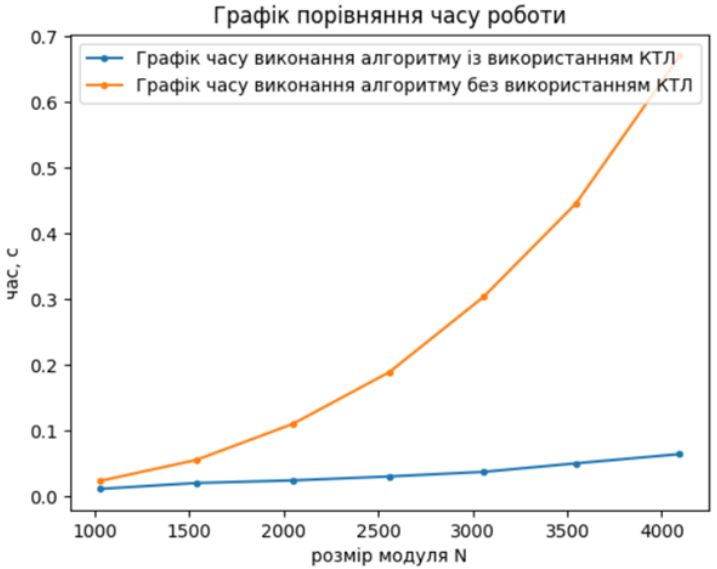
Отже, можна зробити висновок, що найменший час виконання алгоритму можна отримати, також використавши рандомізацію для підвищення складності атаки пропонується обирати кількість простих чисел l як

$$l = \frac{n}{m},\ m \in [20, 30].$$

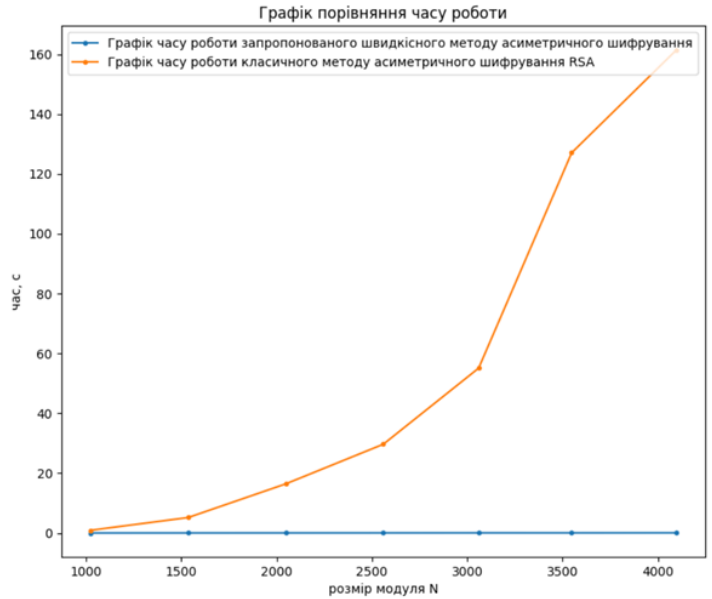
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ШВИДКОСТІ РОБОТИ МЕТОДУ ШИФРУВАННЯ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ КІЛЬКОСТІ ПРОСТИХ МНОЖНИКІВ У МОДУЛІ



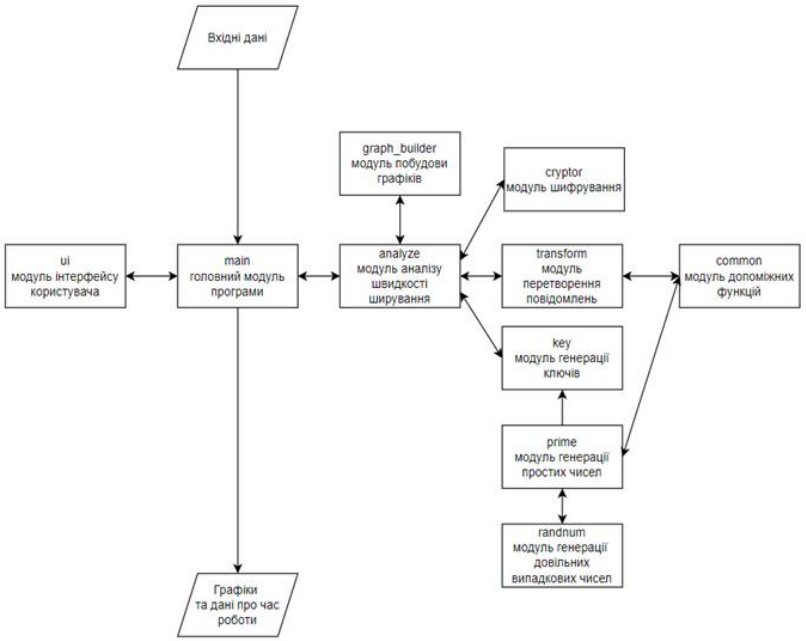
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ВПЛИВУ ВИКОРИСТАННЯ НАСЛІДКУ З КИТАЙСЬКОЇ ТЕОРЕМИ ПРО ЛИШКИ НА ШВИДКІСТЬ РОБОТИ МЕТОДУ ШИФРУВАННЯ



Дослідження ефективності використання запропонованого методу асиметричного шифрування у порівнянні із класичною реалізацією методу шифрування RSA



СТРУКТУРА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАПРОПОНОВАНОГО МЕТОДУ ШИФРУВАННЯ



ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАПРОПОНОВАНОГО МЕТОДУ АСИМЕТРИЧНОГО ШИФРУВАННЯ

Мова програмування: Python

Бібліотеки:

- NumPy – бібліотека високорівневих математичних функцій
- Matplotlib – бібліотека для візуалізації даних

КАНВА БІЗНЕС-МОДЕЛІ

Проблема
повільна генерація ключів шифрування;
недосконаліть стандартних реалізацій методів шифрування;
Розвиток методів криптоаналізу.

Рішення
програмний модуль, що реалізує модифікований метод шифрування RSA

Унікальна ціннісна пропозиція
модифікований метод асиметричного шифрування RSA, що при збереженні криптостійкості дає виграш у швидкості роботи

Прихована перевага
поєднуваність використання даного програмного продукту з будь-якими системами, які використовують інші шифрувальні програми

Споживачі
компанії, що працюють з даними клієнтів і користувачі, що хочуть посилити захист своїх систем

Ключові метрики
кількість проданих ліцензій

Потоки доходів
доходи від продажу ліцензій

доходи від підтримки програмного забезпечення

Канали
через рекламну кампанію продукту

Структура витрат
утримання персоналу для розробки та підтримки програмного забезпечення
податкові витрати
послуги юриста, бухгалтера
розповсюдження продукту, реклама

НАУКОВА НОВИЗНА

Запропоновано метод асиметричного шифрування, що відрізняється від існуючих застосуванням складеного модуля, що складається з простих чисел, кількість яких наближено дорівнює $\frac{1}{25}$ від довжини модуля, у поєднанні із застосуванням наслідку з Китайської теореми про лишки, що дозволяє зменшити обчислювальну складність експоненціально пропорційно до збільшення довжини модуля шифрування.

АПРОБАЦІЯ

Підготовлено тези доповіді «Модифікований метод генерації ключів для алгоритму асиметричного шифрування RSA» на XIV наукову конференцію магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021

ВИСНОВКИ ТА ПОДАЛЬША РОБОТА

1. Розроблено метод асиметричного шифрування на основі методу шифрування RSA, що використовує складений модуль та наслідок з Китайської теореми про лишки.
2. Розроблений метод дає зменшення обчислювальної складності роботи порівняно із класичним експоненціально пропорційно до збільшення довжини модуля шифрування, а також ускладнює відомі типи криптоаналітичних атак, однак втрачає складність атаки грубою силою.
3. Даний метод може бути використано як альтернативу класичному методу RSA у випадках, коли необхідно зменшити час роботи.

УНІКАЛЬНІСТЬ РОБОТИ

UNICHECK

User name:
Онай Микола Володимирович

Check ID:
1009690614

Check date:
16.12.2021 00:57:23 EET

Check type:
Doc vs Internet + Library

Report date:
16.12.2021 00:57:58 EET

User ID:
77218

File name: **Unicheck_Квітка (1)**

Page count: **60** Word count: **11597** Character count: **88703** File size: **94.47 KB** File ID: **1009689805**

4.3% Matches

Highest match: **1.66%** with Library source (File ID: **1009610640**)

2.04% Internet sources **16** ..... Page 62

Дякую за увагу!