МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Кваліфікаційна наукова
праця на правах рукопису

ЯКИМЕНКО ІГОР ЗІНОВІЙОВИЧ

УДК 004.056.53

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ НА ОСНОВІ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

Спеціальність 05.13.21 – «Системи захисту інформації»
Галузь знань: 12 – «Інформаційні технології»

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ Якименко І.З.

Київ – 2026

АНОТАЦІЯ

Якименко І.З. Методи та засоби криптографічного захисту інформації на основі системи залишкових класів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.21 – «Системи захисту інформації». – Західноукраїнський національний університет, Тернопіль, Державний університет інформаційно-комунікаційних технологій, Київ, 2025.

Сучасні задачі науки і техніки тісно пов’язані з потребою у надійному захисті інформаційних потоків. Традиційні криптографічні алгоритми, що базуються на симетричних та асиметричних шифрах, стають дедалі вразливішими через зростання обчислювальних можливостей зловмисників. Водночас вимоги до безпеки даних посилюються, що обумовлює потребу у використанні більш складних і ресурсомістких криптосистем. Це створює протиріччя між забезпеченням високої криптографічної стійкості та необхідністю швидкої обробки великих обсягів інформації.

У цьому контексті перспективним напрямом є використання системи залишкових класів (СЗК), особливо в її модифікованій досконалій формі (МДФ), а також поліноміальної арифметики та ієрархічних структур (ІСЗК). Вони дозволяють реалізувати розпаралелення обчислень, уникнути міжрозрядних переносів і зменшити часову складність криптографічних операцій. Поєднання цих підходів відкриває можливості для створення нових алгоритмів шифрування, які не лише забезпечують високий рівень захисту інформації, а й підвищують ефективність її обробки шляхом заміни традиційної операції множення на додавання.

Дисертаційна робота присвячена вирішенню актуальної науково-прикладної проблеми підвищення ефективності захисту інформаційних потоків на основі заміни в алгоритмах шифрування операції множення операцією додавання, використання цілочисельної, модифікованої досконалої форми,

поліноміальної та ієрархічної систем залишкових класів для розробки нових криптографічних алгоритмів.

Для цього необхідно розв’язати наступні задачі:

– проаналізувати сучасні методи, алгоритми та засоби захисту інформаційних потоків з метою визначення перспектив підвищення їх стійкості на основі використання різних форм СЗК та поліноміальних систем числення;

– удосконалити алгоритмічне забезпечення для реалізації асиметричних криптосистем Рабіна та Ель-Гамаля на основі операції додавання та векторно-модульного алгоритму модулярного множення та експоненціювання;

– удосконалити методи пошуку оберненого полінома за модулем та відновлення полінома за його залишками на основі методу невизначених коефіцієнтів;

– розробити симетричний та асиметричний криптографічні алгоритми в СЗК та дослідити їх стійкість до криптоаналітичних атак;

– розробити криптографічний алгоритм в поліноміальній СЗК та дослідити його стійкість;

– розробити двоключовий симетричний поліноміальний криптоалгоритм та провести дослідження його стійкості;

– розробити ієрархічний криптографічний алгоритм на основі СЗК та дослідити його стійкість;

– розробити поліноміальний ієрархічний криптографічний алгоритм в СЗК;

– розробити програмну реалізацію запропонованих криптоалгоритмів;

– розробити методологію криптографічного захисту інформації на основі заміни в алгоритмах шифрування операції множення операцією додавання, використання цілочисельної, модифікованої досконалої форми, поліноміальної та ієрархічної систем залишкових класів для розробки нових криптографічних алгоритмів.

Аналіз джерел показав, що підвищення стійкості криптографічних алгоритмів зазвичай забезпечується завдяки збільшенню довжини ключів і,

відповідно, розміру операндів математичних перетворень і призводить до зменшення швидкодії. Перспективним напрямом підвищення ефективності є розпаралелення обчислювальних процесів, зокрема із використанням непозиційної системи залишкових класів (СЗК), яка забезпечує обчислення з малорозрядними операндами без міжрозрядних переносів. За умов зростання обчислювальних можливостей сучасні симетричні та асиметричні криптосистеми стають дедалі більш вразливими до криптоаналітичних атак, що актуалізує потребу у створенні нових криптоалгоритмів із підвищеним рівнем стійкості. У цьому контексті значний науковий інтерес викликає використання поліноміальної арифметики в криптографії, зокрема в кільці поліномів Z[x], що дозволяє здійснювати базові операції (додавання, множення, ділення з остачею тощо) та використовувати поліноми як відкриті/зашифровані тексти і ключі. Поєднання СЗК і поліноміальної арифметики дає змогу реалізовувати ефективне паралельне обчислення, знижуючи часову складність криптографічних процедур. Особливої уваги заслуговує ієрархічна система залишкових класів (ІСЗК), яка сприяє оптимізації обчислень і водночас підвищує рівень криптографічної безпеки, відкриваючи нові перспективи для побудови надійних систем захисту даних.

Відповідно до цього, розроблено симетричний криптоалгоритм у системі залишкових класів та її модифікованій досконалій формі, в якому шифртекстом виступає набір залишків по відповідних модулях (ключах), що дозволило на основі побудованих аналітичних виразів оцінки стійкості встановити розрядності та кількості модулів системи залишкових класів для забезпечення такої ж стійкості, як і сучасний симетричний криптоалгоритм AES-256.

Також розроблено високопродуктивні симетричні та асиметричні криптоалгоритми на основі системи залишкових класів та її модифікованої досконалої форми шляхом довільної заміни базисних чисел, що дозволило підвищити рівень захисту даних та стійкість до криптоаналітичних атак.

Для збільшення швидкодії процесів криптографічних перетворень без втрати стійкості алгоритму розроблено криптографічний метод, в якому на

етапах шифрування/розшифрування застосовується Китайська теорема про залишки для блоків відкритого тексту, менших визначених модулів, та пошуку залишків за цими модулями відповідно.

Розроблено одно- та двоключові симетричні криптографічні методи, криптоаналіз яких за рахунок використання поліноміальної системи залишкових класів вимагає комбінаторної складності і може бути віднесений до класу NP-повних задач, а криптографічна стійкість істотно зростає при збільшенні степеня полінома та розмірності поля Галуа;

Запропоновано симетричні методи шифрування/розшифрування інформаційних потоків в ієрархічній цілочисельній та поліноміальній системі залишкових класів, які за рахунок представлення зашифрованого тексту наборами залишків за відповідними модулями (ключами) та розпаралення процесу обчислень дають змогу підвищити стійкість криптоалгоритму та збільшити його швидкодію;

Розроблено методологію криптографічного захисту інформації в системі залишкових класів, яка за рахунок використання векторно-модульних методів модулярного множення та експоненціювання, цілочисельної, модифікованої досконалої форми, поліноміальної та ієрархічної систем залишкових класів дає змогу забезпечити збільшення стійкості, зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного забезпечення та побудувати єдину стратегію захисту інформаційних потоків

Запропоновано поліноміальний, двох- та трьохмодульний цілочисельні асиметричні криптосистеми Рабіна, які за рахунок заміни операції множення операцією додавання та використання векторно-модульного методу характеризуються меншою часовою складністю в порівнянні з відомими;

Запропоновано метод пошуку оберненого полінома в кільці Z[x] на основі методу невизначених коефіцієнтів, який за рахунок усунення операції пошуку НСД поліномів дає змогу зменшити часову складність та підвищити його обчислювальну ефективність;

Удосконалено методи відновлення полінома по його залишках в кільці Z[x], які за рахунок операцій додавання добутку модулів та добутку залишків модулів дають змогу розпаралелити процес обчислення та уникнути процедури пошуку мультиплікативного оберненого полінома, що, в свою чергу, призводить до збільшення ефективності та зменшення часової складності в порівнянні з відомими.

Проведені експериментальні дослідження підтвердили достовірність теоретичних положень та практичних розробок дисертаційного дослідження. Результати досліджень впроваджені або плануються до впровадження (підтверджено відповідними актами) в Акціонерному товаристві «Тернопільобленерго» (№5291/24 від 17.12.2025 р.), ТзОВ НВФ «Інтеграл» (№03-07/2025 від 07.03.2025 р.), ТзОВ завод «Рембутехніка» (№ЦКБ/04-25 від 10.02.2025 р.), Управлінні кібербезпеки та цифрового розвитку відділу цифрової трансформації Міністерства енергетики України, Департаменті Бюро економічної безпеки України (від 12.01.2025 р.), використані при виконанні п’яти науково-дослідних робіт у Західноукраїнському національному університеті (ЗУНУ) (від 05.12.2025 р.), у навчальному та науковому процесах факультету комп’ютерних інформаційних технологій ЗУНУ (від 5.09.2025 р.).

Теоретичні результати роботи відображені в розроблених нових методах, методології та здійснених програмних реалізаціях.

Результати роботи дисертанта знайшли відображення у таких звітах науково-дослідних держбюджетних та госпдоговірних робіт кафедр комп’ютерної інженерії, спеціалізованих комп’ютерних систем та кібербезпеки Західноукраїнського національного університету: «Паралельні методи та засоби реалізації алгоритмів захисту інформації в комп’ютерних мережах з використанням математичного апарату еліптичних кривих» (Державний реєстраційний номер 0109U000035), «Опрацювання багаторозрядних чисел в системі залишкових класів» (Державний реєстраційний номер 0115U001607), «Розробка теоретичних засад методів формування та цифрового опрацювання даних у розподілених спеціалізованих комп’ютерних системах» (Державний

реєстраційний номер 0112U008458), «Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж» (Державний реєстраційний номер 0117U000414), «Виконання завдань Перспективного плану розвитку наукового напряму "Технічні науки" Західноукраїнського національного університету. Розробка методів та алгоритмів захищеного зберігання даних» (Державний реєстраційний номер 0121U114705), «Методи, алгоритми та засоби надійного захищеного зберігання даних на основі модулярних коригуючих кодів» (Державний реєстраційний номер 0118U003182), «Розробка алгоритмів надійного розподіленого зберігання даних на основі модулярних коригуючих кодів» (Державний реєстраційний номер 0118U100457), «Стійкі до криптоаналізу методи та засоби шифрування в поліноміальних системах» (Державний реєстраційний номер 0123U104713).

Ключові слова: система залишкових класів, китайська теорема про залишки, симетричний алгоритм шифрування, асиметричний алгоритм шифрування, криптостійкість алгоритму, модифікована досконала форма, векторно-модульний метод, обернений поліном за модулем, відновлення полінома за його залишками, поліноміальний симетричний криптоалгоритм, ієрархічний симетричний криптоалгоритм.

ABSTRACT

Yakymenko I.Z. Methods and means of cryptographic information protection based on residue number systems. – Qualifying scientific work on the rights of a manuscript.

Dissertation for the degree of Doctor of Technical Sciences in specialty 05.13.21 – "Information Security Systems". – West Ukrainian National University, Ternopil, State University of Information and Communication Technologies, Kyiv, 2025.

Modern problems of science and technology are closely related to the need for reliable protection of information flows. Traditional cryptographic algorithms based on symmetric and asymmetric ciphers are becoming increasingly vulnerable due to the growing computational capabilities of attackers. At the same time, data security requirements are intensifying, which necessitates the use of more complex and resource-intensive cryptosystems. This creates a contradiction between ensuring high cryptographic resistance and the need for rapid processing of large volumes of information.

In this context, a promising direction is the use of the residue number system (RNS), especially in its modified perfect form (MPF), as well as polynomial arithmetic and hierarchical structures (HRNS). They allow for the implementation of parallel computing, avoidance of inter-digit transfers, and reduction of the time complexity of cryptographic operations. The combination of these approaches opens possibilities for creating new encryption algorithms that not only ensure a high level of information protection but also increase the efficiency of its processing by replacing the traditional multiplication operation with addition.

The dissertation work is devoted to solving the actual scientific and technical problem of increasing the efficiency of information flow protection based on replacing multiplication operations with addition operations in encryption algorithms, using integer, modified perfect form, polynomial and hierarchical residue number systems for developing new cryptographic algorithms.

Research Objectives

To achieve this goal, it is necessary to solve the following tasks:

- analyze modern methods, algorithms and means of information flow protection in order to determine prospects for increasing their resistance based on the use of various forms of RNS and polynomial number systems;
- improve algorithmic support for implementing asymmetric cryptosystems of Rabin and ElGamal based on addition operations and vector-modular algorithm of modular multiplication and exponentiation;
- improve methods for finding the inverse polynomial modulo and polynomial restoration from its residues based on the method of undetermined coefficients;
- develop symmetric and asymmetric cryptographic algorithms in RNS and investigate their resistance to cryptanalytic attacks;
- develop a cryptographic algorithm in polynomial RNS and investigate its resistance;
- develop a two-key symmetric polynomial cryptoalgorithm and conduct research on its resistance;
- develop a hierarchical cryptographic algorithm based on RNS and investigate its resistance;
- develop a polynomial hierarchical cryptographic algorithm in RNS;
- develop software implementation of the proposed cryptoalgorithms;
- develop a methodology for cryptographic information protection based on replacing multiplication operations with addition operations in encryption algorithms, using integer, modified perfect form, polynomial and hierarchical residue number systems for developing new cryptographic algorithms.

Research Results

The analysis of sources showed that increasing the resistance of cryptographic algorithms is usually ensured by increasing the length of keys and, accordingly, the size of operands of mathematical transformations, which leads to a decrease in performance. A promising direction for increasing efficiency is the parallelization of computational processes, in particular using the non-positional residue number system (RNS), which provides calculations with small-digit operands without inter-digit

transfers. Under conditions of growing computational capabilities, modern symmetric and asymmetric cryptosystems are becoming increasingly vulnerable to cryptanalytic attacks, which actualizes the need to create new cryptoalgorithms with an increased level of resistance.

In this context, the use of polynomial arithmetic in cryptography, particularly in the ring of polynomials Z[x], is of significant scientific interest, allowing for basic operations (addition, multiplication, division with remainder, etc.) and using polynomials as plaintext/ciphertext and keys. The combination of RNS and polynomial arithmetic enables efficient parallel computation, reducing the time complexity of cryptographic procedures. Particular attention deserves the hierarchical residue number system (HRNS), which contributes to computation optimization and simultaneously increases the level of cryptographic security, opening new prospects for building reliable data protection systems.

Main Scientific Results

A symmetric cryptoalgorithm in the residue number system and its modified perfect form has been developed, in which the ciphertext is a set of residues by corresponding modules (keys), which allowed, based on the constructed analytical expressions for resistance assessment, to establish the bit lengths and number of modules of the residue number system to ensure the same resistance as the modern symmetric cryptoalgorithm AES-256.

High-performance symmetric and asymmetric cryptoalgorithms based on the residue number system and its modified perfect form have been developed through arbitrary replacement of basis numbers, which allowed for increasing the level of data protection and resistance to cryptanalytic attacks.

To increase the speed of cryptographic transformation processes without losing algorithm resistance, a cryptographic algorithm has been developed in which the Chinese remainder theorem is applied at the encryption/decryption stages for plaintext blocks smaller than defined modules and finding residues by these modules respectively.

One- and two-key symmetric cryptographic methods have been developed, the cryptanalysis of which, due to the use of polynomial residue number systems, requires combinatorial complexity and can be attributed to the class of NP-complete problems, and cryptographic resistance significantly increases with increasing polynomial degree and Galois field dimension.

Symmetric methods for encryption/decryption of information flows in hierarchical integer and polynomial residue number systems have been proposed, which, due to the representation of encrypted text by sets of residues by corresponding modules (keys) and parallelization of the computation process, allow for increasing cryptoalgorithm resistance and increasing its performance.

A methodology for cryptographic information protection in the residue number system has been developed, which, through the use of vector-modular methods of modular multiplication and exponentiation, integer, modified perfect form, polynomial and hierarchical residue number systems, allows for ensuring increased resistance, reduced time complexity, improved algorithm performance, specialized software, and building a unified strategy for information flow protection.

The proposed polynomial, two-module, and three-module integer asymmetric Rabin cryptosystems feature lower computational complexity than existing methods due to the substitution of multiplication with addition and the application of a vector-modular approach.

A method for finding the inverse polynomial in the ring Z[x] based on the method of undetermined coefficients has been proposed, which, by eliminating the operation of finding the GCD of polynomials, allows for reducing time complexity and increasing its computational efficiency.

Methods for polynomial restoration from its residues in the ring Z[x] have been improved, which, through addition operations of the product of modules and the product of module residues, allow for parallelizing the computation process and avoiding the procedure of finding the multiplicative inverse polynomial, which, in turn, leads to increased efficiency and reduced time complexity compared to known ones.

Practical Significance

The conducted experimental studies have confirmed the validity of the theoretical propositions and practical developments of the dissertation research. The research results have been implemented or are scheduled for implementation (confirmed by the relevant certificates) at JSC "Ternopiloblenergo" (No. 5291/24 dated Dec 17, 2025), Scientific-Production Firm "Integral" LLC (No. 03-07/2025 dated March 7, 2025), "Rempobuttekhnika" Plant LLC (No. TsKB/04-25 dated Feb 10, 2025), the Cybersecurity and Digital Development Directorate of the Digital Transformation Department of the Ministry of Energy of Ukraine, and the Department of the Bureau of Economic Security of Ukraine (dated Jan 12, 2025). Furthermore, the findings were utilized in the execution of five research and development projects at West Ukrainian National University (WUNU) (dated Dec 5, 2025), as well as in the educational and scientific processes of the Faculty of Computer Information Technologies of WUNU (dated Sept 5, 2025).

Research Projects

The theoretical results of the work are reflected in the developed new methods, methodology and implemented software implementations. The results of the dissertation work are reflected in the following reports of state-budget and contract research works of the departments of computer engineering, specialized computer systems and cybersecurity of West Ukrainian National University:

- "Parallel methods and means of implementing information protection algorithms in computer networks using the mathematical apparatus of elliptic curves" (State registration number 0109U000035)
- "Processing multi-digit numbers in the residue number system" (State registration number 0115U001607)
- "Development of theoretical foundations of methods for formation and digital data processing in distributed specialized computer systems" (State registration number 0112U008458)
- "Theoretical foundations and hardware means for improving the performance of wireless sensor networks" (State registration number 0117U000414)

- "Implementation of tasks of the Prospective Plan for the development of the scientific direction 'Technical Sciences' of West Ukrainian National University. Development of methods and algorithms for secure data storage" (State registration number 0121U114705)
- "Methods, algorithms and means of reliable secure data storage based on modular correcting codes" (State registration number 0118U003182)
- "Development of algorithms for reliable distributed data storage based on modular correcting codes" (State registration number 0118U100457)
- "Cryptanalysis-resistant methods and means of encryption in polynomial systems" (State registration number 0123U104713)

Keywords: residue number system, Chinese remainder theorem, symmetric encryption algorithm, asymmetric encryption algorithm, cryptographic resistance of algorithm, modified perfect form, vector-modular method, inverse polynomial modulo, polynomial restoration from its residues, polynomial symmetric cryptoalgorithm, hierarchical symmetric cryptoalgorithm.

Список основних публікацій здобувача

Статті:

1. Yakymenko I., Kasianchuk M., Martyniuk O., Martyniuk S. A Symmetric Cryptoalgorithm Based on a Hierarchical Residue Number System. *International Journal of Computing*, 2025. 24(1), pp. 92-101. https://doi.org/10.47839/ijc.24.1.3880 (Scopus).
2. Yakymenko I., Karpinski M., Shevchuk R., Kasianchuk M. Symmetric Encryption Algorithms in a Polynomial Residue Number System. *Journal of Applied Mathematics*., 2024, pp. 1-12. DOI:10.1155/2024/4894415 (Scopus).
3. Nykolaychuk Ya., Yakymenko I., Vozna N., Kasianchuk M. Residue Number System Asymmetric Cryptoalgorithms. *Cybernetics and Systems Analysis*. 2022, Vol. 58, No. 4, P.611-618. http://jnas.nbuv.gov.ua/article/UJRN-0001335526. https://doi.org/10.1007/s10559-022-00494-7. (Scopus).

Shevchuk R., Yakymenko I., Karpinski M., Shylinska I., Kasianchuk M. Finding the inverse of a polynomial modulo in the ring Z[x] based on the method of undetermined coefficients. *Computer Science*. vol. 25. no. 2. 2024. pp.1-14. DOI:10.7494/csci.2024.25.2.5740 (Scopus).

4. M. M. Kasianchuk, I. Z. Yakymenko, Ya. M. Nykolaychuk Symmetric Cryptoalgorithms in the Residue Number System. *Cybernetics and Systems Analysis*, 2021, Vol 57, Issue 2, p.184-189. https://doi.org/10.1007/s10559-021-00358-6
5. Nykolaychuk Ya., Kasianchuk M., Yakymenko I. Theoretical Foundations of the Modified Perfect form of Residue Number System. *Cybernetics and Systems Analysis*. 2016. Vol. 52, №2. pp. 219-223 (Scopus).
6. Nykolaychuk Ya., Kasianchuk M., Yakymenko I. Theoretical Foundations for the Analytical Computation of Coefficients of Basic Numbers of Krestenson's Transformation. *Cybernetics and Systems Analysis*. 2014. Vol. 50, № 5. pp. 649-654 (Scopus).
7. Iakymenko I., Kasianchuk M., Kinakh I., Karpinski M. Construction of distributed thermal or piezoelectric sensor based on residue systems. Przeglad Elektrotechniczny. 2017. №1. pp. 290-294 (Scopus). DOI: 10.15199/48.2017.01.69
8. Kasianchuk M., Yakymenko I., Yatskiv S., Gomotiuk O., Bilovus L. The Method of Joint Execution of the Basic Operations of the Rabin Cryptosystem. *CEUR Workshop Proceedings*, 2023, 3373, pp. 425–436. (Scopus).
9. Kasianchuk M., Yakymenko I., Yatskiv V., Karpinski M., Yatskiv S. Method of Multi-Bit Numbers Multiplication in Residue Number System for Asymmetric Cryptosystems. *CEUR Workshop Proceedings*, 2022, 3156, pp. 365–377. https://ceur-ws.org/Vol-3156/paper27.pdf . (Scopus).
10. Stanislaw Zawislak, Mykhailo Kasianchuk, Igor Iakymenko, Daniel Jancarczyk Methods of Crypto-stable Symmetric Encryption in the Residual Number System. *Procedia Computer Science.* Volume 207, 2022, pp. 128-137. 10.1016/j.procs.2022.09.045 (Scopus)
11. Yakymenko I., Kasianchuk M., Shylinska I. A Method for Polynomial Recovery from its Resudues Based on Addition in Z[x] Ring. *Informatics and*

Mathematical Methods in Simulation Vol.14 (2024), No. 4, pp. 305-313. DOI 10.15276/imms.v14.no4.305.

12. Якименко І. З., Касянчук М. М., Івасьєв С. В. Криптосистема Рабіна на основі операції додавання. *Математичне та комп'ютерне моделювання*. Серія: Технічні науки № 19, 2019. 145–150 с. DOI: https://doi.org/10.32626/2308-5916.2019-19.145-150

13. Якименко І. З. Удосконалення реалізації криптоалгоритму Ель-Гамаля на основі системи залишкових класів. *Інформатика та математичні методи в моделюванні*, 2018, 8, № 1. С. 69-77

14. Касянчук М.М., Якименко І.З., Івасьєв С.В., Мандебура Н.М., Неміш В.М. Дослідження часових характеристик апаратної реалізації методів пошуку оберненого елемента за модулем. *Вісник Хмельницького національного університету. Технічні науки*. 2017. №6 (255). С. 191-197.

15. Касянчук М.М., Якименко І.З., Івасьєв С.В., Момотюк О.В. Експериментальне дослідження програмної реалізації методів пошуку оберненого елемента за модулем. *Інформатика та математичні методи в моделюванні*. 2017. Т.7, №3. С. 178–186.

16. Касянчук М.М., Якименко І.З., Івасьєв С.В., Маслияк Б.О. Метод розширення набору модулів модифікованої досконалої форми системи залишкових класів. *Математичне та комп'ютерне моделювання: Технічні науки.* 2017. В.15. С.73-78.

17. Касянчук М.М., Якименко І.З., Паздрій І.Р., Івасьєв С.В. Експериментальне дослідження програмної реалізації сумісного виконання алгоритму Евкліда та множення. *Інформатика та математичні методи в моделюванні.* 2017. Т.7, №1-2. С. 29–36.

18. Касянчук М.М., Якименко І.З., Дубчак Л.О., Рендзеняк Н.А., Мандебура Н.М. Модифікований метод шифрування Рабіна з використанням різних форм системи залишкових класів. *Вісник Хмельницького національного університету. Технічні науки.* 2017. №1(245). С. 127-131.

19. Касянчук М.М., Якименко І.З., Долинюк Т.М., Рендзеняк Н.А. Експериментальне дослідження програмної реалізації методів модулярного експоненціювання. *Інформатика та математичні методи в моделюванні*. 2015. Т.5, №4. С. 376–382.

20. Івасьєв С.В., Якименко І.З., Касянчук М.М. Вдосконалений алгоритм пошуку символів Якобі. *Оптико-електронні інформаційно-енергетичні технології*. 2015. Том 29, № 1. С. 45-50.

21. Касянчук М.М., Якименко І.З., Паздрій І.Р., Николайчук Я.М. Аналітичний пошук модулів досконалої форми системи залишкових класів та їх застосування в китайській теоремі про залишки. *Вісник Хмельницького національного університету. Технічні науки*. 2015. №1(221). С. 170-176.

22. Николайчук Я. М., Касянчук М.М., Якименко І.З., Івасьєв С.В. Ефективний метод модулярного множення в теоретико-числовому базисі Радемахера–Крестенсона. *Вісник Національного університету «Львівська політехніка». Комп'ютерні системи та мережі*. 2014. № 806. С. 195-199.

23. Якименко І.З., Касянчук М.М., Тимошенко Л.М., Гребень Н.Є. Алгоритми опрацювання інформаційних потоків в комп'ютерних системах. *Інформатика та математичні методи в моделюванні*. 2013. Т.3, №3. С. 266–274.

24. Якименко І.З., Касянчук М.М., Кімак В.Л. Теоретичні основи зменшення часової та апаратної складності систем захисту інформаційних потоків на основі еліптичних кривих з використанням теоретико-числового базису Радемахера-Крестенсона. *Вісник Національного університету «Львівська політехніка» «Комп'ютерні системи та мережі».* 2012. №745. С. 190–197.

25. Николайчук Я.М., Касянчук М.М., Якименко І.З., Долинюк Т.М. Теоретичні основи виконання модулярних операцій множення та експоненціювання в теоретико–числовому базисі Крестенсона–Радемахера. *Інформатика та математичні методи в моделюванні*. 2011. №2. С. 123–130.

26. Zadiraka V., Yakymenko I., Kasianchuk M., Ivasiev S. Theoretical and numerical Krestenson's basis and its application to problems of cryptographic protection and factorization of multidigit numbers, *Computer technologies in information security: collective monograph*, By edited V.Zadiraka, Ya.Nykolaichuk, Ternopil: Kart-blansh, 2015. P. 216-260. Ch. 5.

27. Yakymenko I., Kasyanchuk M., Volynskyi O. Fundamental application-oriented tasks in Krestenson base, *Methods of effective protection of information flows: collective monograph*, By edited V.Zadiraka, Ya.Nykolaichuk, Ternopil: Terno-graf, 2014. P. 149-185. Ch.6.

28. Kasianchuk M., Yakymenko I., Ivasiev S. High-Productivity Methods of Finding Residues Multidigital Numbers By Modulo, in *Inżynier XXI Wieku: VI Międzynarodowa Konferencja studentow oraz doktorantow, 02.12.2016: monografia*, 1st ed., Bielsko – Biała (Poland): Akademia Techniczno-Humanistyczna w Bielsku-Białej. 2016. pp. 123-130. Chapter in monograph.

29. Якименко І. Алгоритми побудови модифікованої досконалої форми системи залишкових класів. *Спеціалізовані комп'ютерні технології в інформатиці: Колективна монографія*. Під ред. В.Задіраки, Я.Николайчука. Тернопіль: Бескиди, 2017. С. 580-604.

30. Kasianchuk M., Yakymenko I., Ivasiev S. Theoretical foundations for creating five modular modified perfect form of the system of residual classes, in *Inżynier XXI Wieku: VII Międzynarodowa Konferencja studentow oraz doktorantow, 08.12.2017: monografia*, 1st ed., Vol.2., Bielsko – Biała (Poland): Akademia Techniczno-Humanistyczna w Bielsku-Białej, 2017, pp. 123-130. Chapter in monograph.

Матеріали конференцій:

31. Yakymenko I., Kasianchuk M., Martyniuk O., Martyniuk S., Martyniuk A., Yakymenko Y. A Symmetric Cryptoalgorithm in a Polynomial Hierarchical Residual Number System. *Proceedings International Conference on Advanced Computer Information Technologies*, ACIT., 2025, pp. 501-504. DOI: 10.1109/ACIT65614.2025.11185808

32. Yakymenko I., Martyniuk O., Martyniuk S., Yakymenko Y., Kasianchuk M. Hierarchical Encryption in a Residual Number System. *Proceedings International Conference on Advanced Computer Information Technologies*, ACIT., 2024, pp. 496–499 (Scopus). DOI: 10.1109/ACIT62333.2024.10712567

33. Shevchuk R., Yakymenko I., Kasianchuk M. Encryption Using Residue Number System: Research Trends and Future Challenges. *Proceedings International Conference on Advanced Computer Information Technologies*, ACIT2024, 2024, pp. 552–559 (Scopus). DOI: 10.1109/ACIT62333.2024.10712566

34. Shevchuk R., Karpinski M., Kasianchuk Yakymenko I., Melnyk A., Tykhyi R. Software for Improve the Security of Kubernetes-based CI/CD Pipeline. *Proceedings International Conference on Advanced Computer Information Technologies*, ACIT2023, 2023, pp. 420–425. (Scopus). DOI: 10.1109/ACIT58437.2023.10275654

35. Yakymenko I., Kasianchuk M., Shylinska I., Shevchuk R., Yatskiv V., Karpinski M. Polynomial Rabin Cryptosystem Based on the Operation of Addition. *12th International Conference on Advanced Computer Information Technologies*, ACIT 2022, 2022, pp. 345–350. DOI:10.1109/ACIT54803.2022.9913089 (Scopus).

36. Yakymenko I., Kasianchuk M., Yatskiv V., Shevchuk R., Koval V., Yatskiv S. Sustainability and Time Complexity Estimation of Cryptographic Algorithms Main Operations on Elliptic Curves. *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*. pp. 494-498 (Scopus). DOI: 10.1109/ACIT52158.2021.9548534

37. Mykhailo Kasianchuk, Ihor Yakymenko, Vasyl Yatskiv, Stepan Ivasiev, Andriy Sverstiuk. Same Bit-Size Moduli Formation of Residue Number System for Application in Asymmetric Cryptography. *IntelITSIS 2021*. pp. 301-308.

38. Yakymenko I., Shylinska I., Kasianchuk M., Bilovus L., Gomotiuk O. Algorithmic Support for Rabin Three-Modular Cryptosystem Based on the Operation of Addition. *IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT).* 2020. pp. 328-331 (Scopus).

39. Kasianchuk M., Yakymenko I., Karpinski M., Shevchuk R., Karpinskyi V., Shylinska I. Theoretical Bases for Reducing the Time Complexity of the Rabin Cryptosystem. *Conference on Computer Science and Information Technologies*. 2020. pp. 628-639. (Scopus). DOI: 10.1007/978-3-030-63270-0_43

40. Ivasiev S., Kasyanchuk M., Yakymenko I., Gomotiuk O., Shylinska I., Bilovus L. Algorithmic support for Rabin cryptosystem implementation based on addition.*10th International Conference on Advanced Computer Information Technologies (ACIT)*. 2020. pp. 779-782. (Scopus). DOI: 10.1109/ACIT49673.2020.9208923

41. Yakymenko I., Kasianchuk M., Ivasiev S., Shevchuk R., Batko Y., Vasylkiv V. Method for Determining Prime and Relatively Prime Numbers of 2^n+k Type Based on the Periodicity Property. *10th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany. 2020. pp. 751–754. https://doi.org/10.1109/ACIT49673.2020.9208812 (Scopus).

42. Yakymenko I., Kasianchuk M., Gomotiuk O., Tereshchuk G., Ivasiev S., Basistyi P. Elgamal cryptoalgorithm on the basis of the vector-module method of modular exponentiation and multiplication. *IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*. 2020. pp. 926-929. (Scopus). DOI: 10.1109/TCSET49122.2020.235572

43. Karpinski M., Rajba S., Zawislak S., Warwas K., Kasianchuk M., Ivasiev S., Yakymenko I. A method for decimal number recovery from its residues based on the addition of the product modules, *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Metz, France. 2019. pp.13-17. https://doi.org/10.1109/IDAACS.2019.8924395. (Scopus).

44. Kasianchuk M., Yakymenko I., Ivasiev S., Shevchuk R., Tymoshenko L. The method of factorizing multi-digit numbers based on the operation of adding odd numbers. *CEUR Workshop Proceedings 8th International Conference Advanced Computer Information Technologies ACIT*. June 2018. 2018, pp. 232-235, (Scopus).

45. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Karpinski M., Gomotiuk O. Effective algorithms for finding the remainder of multi-digit numbers. *Advanced Computer Information Technology (ACIT–2019): Proceedings of the International Conference*. 2019, pp. 175-178. (Scopus). DOI: 10.1109/ACITT.2019.8779899

46. Yakymenko I., Kasianchuk M., Ivasiev S., Melnyk A., Nykolaichuk Y. Realization of RSA cryptographic algorithm based on vector-module method of modular exponention. *In Proceedings of the 2018 14th International Conference on Advanced Trends in Radioelecrtronics*. Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 20–24 February 2018. 2018. pp. 550–554.

47. Якименко І.З., Касянчук М.М., Кінах Я.І., Власюк І.М., Суслін В.В. Удосконалення реалізації асиметричних криптоалгоритмів на основі системи залишкових класів. *Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології» (ACIT)*. 2018. с. 79.

48. Карпінський, М. П., Кінах, Я. І., Яциковська, У. О., Якименко, І. З., Касянчук, М. М. Удосконалення архітектури комп'ютерної мережі для програмної реалізації криптоаналітичних алгоритмів. *Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології".* 2018. С. 93.

49. Карпінський М. П., Кінах Я. І., Войтенко, О. С., Паславський, В. Р., Якименко, І. З., Касянчук, М. М. Теоретичний аналіз інформаційної безпеки в комп'ютерних мережах. *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій". 2*, 2017. С.81-82.

50. Rajba T., Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of time characteristics of search methods of inverse element by the module in: Proc. *IEEE 9th Intern. Conf. on Intelligent Data Acquisition and Advanced*

Computing Systems: Technology and Applications (IDAACS-2017) (Bucharest, Romania. 21–23 Sept, 2017), 2017. pp. 82–85. https://doi.org/10.1109/IDAACS.2017.8095054. (Scopus).

51. Kasianchuk M., Yakymenko I., Pazdriy I., Melnyk A., Ivasiev S., Rabin's modified method of encryption using various forms of system of residual classes. *14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, 2017. pp. 222-224. (Scopus). DOI: 10.1109/CADSM.2017.7916120

52. Якименко І.З. Методологія криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої та поліноміальної систем залишкових класів, *Матеріали XIV Міжнародної науково-технічної конференції ITSec-2025 Безпека інформаційних технологій*, 22-24 травня 2025, м. Тернопіль (Україна). 2025. С. 224-228.

53. Karpiński M., Ivasiev S., Yakymenko I., Kasianchuk M., Gancarczyk T. Advanced method of factorization of multi-bit numbers based on Fermat's theorem in the system of residual classes. *International Conference on Control, Automation and Systems* (*ICCAS–2016*): Proceedings. Gyeongju, Korea. V.1. 2016. pp.1484–1486 (Scopus). DOI: 10.1109/ICCAS.2016.7832500

54. Nykolaychuk Ya., Ivas'ev S., Yakymenko I., Kasianchuk M. Test of verification of multidigit numbers on simplicity on the basis of method of vector and modular multiplication. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2016*): *Proceedings of the XIII–th International Conference.* L'viv–Slavske. 2016. pp.534-536 (Scopus). DOI: 10.1109/TCSET.2016.7452107

55. Kozaczko D., Ivasiev S., Yakymenko I., Kasianchuk M. Vector Module Exponential in the Remaining Classes System. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (*IDAACS–2015*): Proceedings of the 2015 IEEE 8th International Conference. Warsaw, Poland. V.1. 2015. pp.161–163 (Scopus). DOI: 10.1109/IDAACS.2015.7340720

56. Kasianchuk M., Yakymenko I., Pazdriy I., Zastavnyy O. Algorithms of findings of perfect shape modules of remaining classes system. *The Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2015*): Proceedings of the XIII International Conference. Polyana-Svalyava. 2015. pp.168-171 (Scopus). DOI: 10.1109/CADSM.2015.7230866

57. Ivas'ev S., Kasyanchuk M., Yakymenko I., Nykolaychuk Ya. Fundamental Backgrounds of the Discrete Logarithms Theory in the Rademacher–Krestenson's Basis. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2012*): Proceedings of the XI–th International Conference. L'viv–Slavske. 2012. 93 P. (Scopus).

58. Касянчук М.М., Якименко І.З., Тимошенко Л.М., Івасьєв С.В., Николайчук Я.М. Векторно-модульний метод модулярного множення. *Сучасні інформаційні та електронні технології: Матеріали Міжнародної науково-практичної конференції.* Одеса. 2014, С. 152.

Патенти:

59. Пат. 159225 Україна МПК G06F 7/00 (2025.01). Накопичуючий синхронізований двійковий суматор / Николайчук Я.М., Грига В.М., Якименко І.З., Грига Л.П., № u 2024 04320 заявл. 03.09.2024; опубл. 08.05.2025, Бюл. №19/2025.

Пат. 160091 Україна МПК G06F7/04. Пристрій порівняння даних, представлених у непозиційній системі залишкових класів / Николайчук Я.М., Якименко І.З., Івасьєв С.В, Грига В.М. № u202404328 заявл. 03.09.2024; опубл. 06.08.2025, Бюл. № 32/2025.

ЗМІСТ

ВСТУП

Актуальність теми. Розв’язання переважної більшості сучасних задач науки і техніки безумовно пов’язані з опрацюванням та захищеною передачею інформаційних потоків. Традиційні технології забезпечення захисту інформації зазвичай передбачають застосування симетричних та/або асиметричних криптоалгоритмів. Значний вклад у їх розвиток внесли такі вітчизняні та зарубіжні вчені: В.К. Задірака, І.Д. Горбенко, О.Г. Корченко, А.А. Хорошко, О.П. Скринник, О.В. Потій, М. Є. Шелест, О.О. Кузнецов, М.П. Карпінський, В. П. Сіденко, Ю.О. Дрейс, С.О. Гнатюк, В.М. Кінзерявий, Н. Фергюссон, М. Рабін, Т. Ель-Гамаль, Н. Кобліц, В. Діффі, М. Хелман та інші.

Слід зазначити, що вимоги, які ставляться до сучасних систем захисту інформації, як правило, досить строгі та, як наслідок, громіздкі в реалізації та затратні в експлуатації. Крім того, для підвищення стійкості криптоалгоритмів потрібно збільшувати довжину ключів і, відповідно, значення операндів математичних перетворень. Це приводить до зменшення швидкодії криптоалгоритмів. Найперспективнішим шляхом її підвищення є розпаралелення процесу обчислень. Цією властивістю володіє непозиційна система залишкових класів (СЗК).

Значний теоретичний та прикладний внесок у розвиток СЗК та її застосування зробили такі вчені: М. Валах (M. Valach), А. Свобода (A. Svoboda), І. Акушський, Д. Юдицький, В. Амербаєв, В. Торгашев, В. Краснобаєв, Я. Николайчук, В. Яцків, М. Касянчук, А. Омонді (A. Omondi), Б. Премкумар (B. Premkumar), А. Молахоссеіні (A. Molahosseini), А.Мохан (A.Mohan) та інші.

До інших основних переваг СЗК можна віднести можливість виконання операцій над малорозрядними операндами та відсутність міжрозрядних переносів. Такі властивості дозволяють ефективно використовувати СЗК, зокрема, її модифіковану досконалу форму (МДФ), в криптографії.

Крім того, при значних зростаннях обчислювальних можливостей, сучасні симетричні та асиметричні криптосистеми стають надзвичайно вразливими. Це

означає, що їх криптоаналіз може здійснюватися за прийнятний для зловмисника час. Тому для забезпечення цілісності, конфіденційності та доступності інформації виникає потреба в розробці нових криптографічних алгоритмів, які будуть стійкішими до криптоаналітичних атак. В цьому напрямку поліноміальні криптоалгоритми відкривають нові перспективи. Аналогічно до цілочисельних застосувань, у кільці поліномів Z[x] можливі базові операції додавання, множення, піднесення до степеня, ділення з остачею тощо. Поліноми успішно можна використовувати у вигляді відкритого і зашифрованого тексту, відкритого і таємного ключів при застосуванні в криптографічних алгоритмах.

Слід ще відзначити, що поєднання поліноміальної арифметики та СЗК дає змогу використати переваги останньої, зокрема, розпаралелення процесу обчислень. Це призводить до зменшення часової складності та спрощення процесу обчислень при шифруванні/розшифровуванні. А використання двох різнотипних ключів забезпечить підвищення стійкості запропонованої системи до криптоаналізу.

У цьому контексті особливий інтерес викликає використання цілочисельних та поліноміальних багаторівневих або ієрархічних СЗК (ІСЗК). Такий підхід дає змогу оптимізувати арифметичні операції та зменшити обсяг необхідних обчислень, забезпечуючи при цьому високий рівень безпеки даних. Завдяки своїм унікальним властивостям та високому потенціалу оптимізації ІСЗК відкриває нові перспективи у сфері криптографічного захисту даних.

Отже, розробка нових методів шифрування в СЗК та дослідження їх стійкості до криптоаналізу на даний час є надзвичайно актуальною задачею.

Враховуючи викладене, актуальною науково-прикладною проблемою є розробка методів, засобів та методології криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої форми, поліноміальної та ієрархічної систем залишкових класів, яка виникає в результаті об’єктивного *протиріччя* між потребою забезпечення високої криптографічної стійкості та передавання великих обсягів конфіденційної інформації, з одного боку, та забезпечення підвищення швидкодії процесу шифрування/розшифрування.

Наукова концепція дисертації полягає у побудові та обґрунтуванні єдиної стратегії криптографічного захисту інформаційних потоків, в якій криптографічні перетворення виконуються в системі залишкових класів (СЗК) та її похідних формах (цілочисельній, модифікованій досконалій, поліноміальній та ієрархічній), а обчислювально затратні операції множення/піднесення до степеня реалізуються через операції додавання із застосуванням векторно-модульних алгоритмів модулярного множення та експоненціювання.

Зв'язок роботи з науковими програмами, планами і темами. Дисертаційна робота виконувалася у рамках таких науково-дослідних держбюджетних та госпдоговірних робіт кафедр комп'ютерної інженерії, спеціалізованих комп'ютерних систем та кібербезпеки Західноукраїнського національного університету: «Паралельні методи та засоби реалізації алгоритмів захисту інформації в комп'ютерних мережах з використанням математичного апарату еліптичних кривих» (Державний реєстраційний номер 0109U000035), «Опрацювання багаторозрядних чисел в системі залишкових класів» (Державний реєстраційний номер 0115U001607), «Розробка теоретичних засад методів формування та цифрового опрацювання даних у розподілених спеціалізованих комп'ютерних системах» (Державний реєстраційний номер 0112U008458), «Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж» (Державний реєстраційний номер 0117U000414), «Виконання завдань Перспективного плану розвитку наукового напряму "Технічні науки" Західноукраїнського національного університету. Розробка методів та алгоритмів захищеного зберігання даних» (Державний реєстраційний номер 0121U114705), «Методи, алгоритми та засоби надійного захищеного зберігання даних на основі модулярних коригуючих кодів» (Державний реєстраційний номер 0118U003182), «Розробка алгоритмів надійного розподіленого зберігання даних на основі модулярних коригуючих кодів» (Державний реєстраційний номер 0118U100457), «Стійкі до криптоаналізу методи та засоби шифрування в поліноміальних системах» (Державний реєстраційний номер 0123U104713).

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення рівня стійкості та ефективності криптосистем шляхом створення нових методів, засобів та методології криптографічного захисту інформації в системі залишкових класів.

Для досягнення поставленої мети у дисертаційній роботі необхідно розв'язати низку взаємопов'язаних задач:

1) проаналізувати сучасні методи, алгоритми та засоби захисту інформаційних потоків з метою визначення перспектив підвищення їх стійкості на основі використання різних форм СЗК та поліноміальних систем числення;

2) удосконалити та підвищити ефективність алгоритмічного забезпечення для реалізації асиметричних криптосистем Рабіна та Ель-Гамаля на основі операції додавання та векторно-модульного алгоритму модулярного множення та експоненціювання;

3) удосконалити та підвищити ефективність методів пошуку оберненого полінома за модулем та відновлення полінома за його залишками на основі методу невизначених коефіцієнтів;

4) розробити симетричний та асиметричний криптографічні алгоритми в СЗК та дослідити їх стійкість до криптоаналітичних атак;

5) розробити криптографічний одноключовий та двоключовий алгоритми в поліноміальній СЗК та дослідити їх стійкість;

6) розробити ієрархічний цілочисельний та поліноміальний криптографічні алгоритми на основі СЗК та дослідити їх стійкість;

7) здійснити програмну реалізацію запропонованих криптоалгоритмів;

8) розробити методологію криптографічного захисту інформації на основі заміни в алгоритмах шифрування операції множення операцією додавання, з використанням цілочисельної, модифікованої досконалої форми, поліноміальної та ієрархічної систем залишкових класів для розробки нових криптографічних алгоритмів.

Об'єкт дослідження – процеси шифрування інформаційних потоків на основі системи залишкових класів.

Предмет дослідження – методи, алгоритми та засоби для підвищення стійкості криптоалгоритмів на основі цілочисельної та поліноміальної систем залишкових класів.

Методи дослідження. Проведені дослідження ґрунтуються на математичних основах алгебри і теорії чисел (для розробки методів пошуку оберненого полінома, відновлення полінома за його залишками, реалізації китайської теореми про залишки (КТЗ)), теорії алгоритмів (для оцінки стійкості відомих та розроблених методів), методах криптографії (для розробки тримодульної криптосистеми Рабіна, векторно-модульної криптосистеми Ель-Гамаля), програмуванні (для реалізації симетричних, асиметричних цілочисельних криптоалгоритмів в СЗК та симетричних криптоалгоритмів в поліноміальній СЗК), теорії множин (для побудови методології захисту інформаційних потоків на основі цілочисельної та поліноміальної СЗК), статистиках (для обробки експериментальних результатів).

Наукова новизна отриманих результатів полягає в наступному:

1) вперше розроблено симетричний криптоалгоритм у системі залишкових класів, який за рахунок розбиття відкритого повідомлення на залишки по відповідних попарно взаємнопростих модулях (ключах) та використання китайської теореми про залишки дозволяє розпаралелити обчислювальний процес, зменшити розмірність операндів та на основі побудованих аналітичних виразів встановити розрядність та кількість модулів системи залишкових класів для забезпечення такої ж стійкості, як і сучасний симетричний криптоалгоритм AES-256;

2) вперше розроблено високопродуктивні симетричні та асиметричні криптоалгоритми на основі системи залишкових класів та її модифікованої досконалої форми, які за рахунок довільної заміни базисних чисел в процесі шифрування на попарно взаємнопрості з відповідними модулями додаткові ключі дозволяють підвищити криптостійкість та забезпечити необхідний рівень захисту інформаційних потоків;

3) вперше розроблено криптографічний метод, в якому за рахунок

шифрування відкритого тексту у вигляді залишків за допомогою китайської теореми про залишки і розшифрування на основі операції пошуку залишків за відповідними модулями забезпечується підвищення швидкості розшифрування інформації без втрати стійкості алгоритму;

4) отримав подальший розвиток метод пошуку оберненого полінома в кільці Z[x] на основі методу невизначених коефіцієнтів, який за рахунок усунення операції пошуку найбільшого спільного дільника двох поліномів дозволив зменшити часову складність та підвищити швидкодію алгоритму при його використанні в поліноміальних криптосистемах;

5) удосконалено методи відновлення полінома за його залишками в кільці Z[x], які за рахунок використання операції додавання добутку модулів або їх залишків за відповідними модулями дозволяють уникнути обчислювально громіздкої процедури пошуку мультиплікативного оберненого полінома, що, в свою чергу, призводить до збільшення швидкодії та зменшення часової складності поліноміальних алгоритмів шифрування;

6) вперше розроблено одно- та двоключові симетричні криптографічні методи в поліноміальній системі залишкових класів, які за рахунок заміни в процесі шифрування базисних поліномів на довільно вибрані попарно взаємнопрості з модулями поліноми дозволяють створити додаткову структурну неоднозначність, ускладнити криптоаналіз через необхідність розв’язання NP-повної задачі та збільшити криптографічну стійкість;

7) вперше розроблено симетричні методи шифрування/розшифрування інформаційних потоків в ієрархічній цілочисельній та поліноміальній системах залишкових класів, які за рахунок представлення зашифрованого тексту наборами залишків за відповідними модулями (ключами) та розпаралелення процесу обчислень дозволяють підвищити стійкість криптоалгоритму та збільшити його швидкодію;

8) отримали подальший розвиток поліноміальний, дво- та тримодульний цілочисельні асиметричні криптосистеми Рабіна, які за рахунок заміни операції множення на операцію додавання та використання векторно-модульного методу

модулярного множення дозволяють зменшити часову складність криптографічних перетворень і підвищити швидкодію реалізації алгоритмів;

9) вперше розроблено методологію криптографічного захисту інформації в системі залишкових класів, яка за рахунок застосування векторно-модульних методів модулярного множення та експоненціювання, цілочисельної, модифікованої досконалої форми, поліноміальної та ієрархічної систем залишкових класів дає змогу забезпечити збільшення стійкості, зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного забезпечення та побудувати єдину стратегію криптографічного захисту інформаційних потоків на основі системи залишкових класів.

Практичне значення одержаних результатів.

1) розроблено алгоритмічне забезпечення для дво- та тримодульної криптосистеми Рабіна та криптосистеми Ель-Гамаля на основі СЗК, а також з використанням векторно-модульного методу модулярного множення та експоненціювання, що дозволило значно зменшити часову складність криптографічних перетворень у порівнянні з традиційними підходами;

2) здійснено програмну реалізацію симетричних та асиметричних криптоалгоритмів у СЗК, завдяки чому емпірично підтверджено переваги запропонованих методів, що є підґрунтям для впровадження розроблених алгоритмів у сучасні інформаційні системи;

3) розроблено програмне забезпечення для реалізації ієрархічного симетричного криптоалгоритму в СЗК, яке завдяки розпаралеленню процесу обчислення забезпечує високу швидкодію процесів шифрування/дешифрування, що надає переваг при застосуванні у багаторівневих системах захисту даних;

4) розроблено програмне забезпечення симетричного шифрування в поліноміальній СЗК, яке забезпечує зручність використання та дозволяє адаптивно налаштовувати параметри відповідно до задач користувача, а також спрощує пояснення складних процесів шифрування.

Результати досліджень впроваджені або плануються до впровадження (підтверджено відповідними актами) в Акціонерному товаристві

«Тернопільобленерго» (№5291/24 від 17.12.2025 р.), ТзОВ НВФ «Інтеграл» (№03-07/2025 від 07.03.2025 р.), ТзОВ завод «Рempобуттехніка» (№ЦКБ/04-25 від 10.02.2025 р.), Управлінні кібербезпеки та цифрового розвитку відділу цифрової трансформації Міністерства енергетики України, Департаменті Бюро економічної безпеки України (від 12.01.2025 р.), використані при виконанні п'яти науково-дослідних робіт у Західноукраїнському національному університеті (ЗУНУ) (від 05.12.2025 р.), факультету комп'ютерних інформаційних технологій ЗУНУ (від 5.09.2025 р.).

Особистий внесок здобувача. Наукові положення, які містяться в дисертації, отримані здобувачем особисто. У друкованих працях, опублікованих у співавторстві, автору належить: [1, 32] – запропоновано симетричний криптоалгоритм в ІПСЗК та проведено дослідження криптостійкості, [2] – запропоновано симетричний криптоалгоритм в ПСЗК, [3] – запропоновано асиметричний криптоалгоритм в СЗК, [4] – удосконалено метод пошуку оберненого поліному в кільці Z[x] на основі методу невизначених коефіцієнтів, [5] – запропоновано симетричні криптоалгоритми у СЗК, [6, 11, 17, 30, 31, 38, 57] – розроблена метод побудови МДФ СЗК та МДФ СЗК для модулів однакової розрядності, [7, 28] – представлено теоретичні основи аналітичного обчислення коефіцієнтів базисних чисел перетворення Крестенсона, [9] – представлено метод пониження складності виконання основних операцій криптосистеми Рабіна, [10] – запропоновано метод множення багаторозрядних чисел у системі залишкових класів для асиметричних криптосистем, [12] – удосконалено метод відновлення полінома по його залишках на основі операції додавання, [13, 19, 31, 40, 41, 52] – розроблено алгоритмічне забезпечення криптосистеми Рабіна з використанням операції додавання та на основі МДФ СЗК, [14, 43] – удосконалено реалізацію криптоалгоритму Ель-Гамаля на основі векторно-модульного методу модулярного множення, експоненціювання та на основі СЗК, [19] – запропоновано модифікований метод шифрування Рабіна з використанням різних форм СЗК, [21] – удосконалено алгоритм пошуку символів Якобі, [22] – представлено аналітичний пошук модулів досконалої форми системи

залишкових класів та їх застосування в китайській теоремі про залишки, [26] – запропоновано теоретичні основи виконання модулярних операцій множення та експоненціювання в теоретико–числовому базисі Крестенсона–Радемахера, [33] – запропоновано симетричний криптографічний алгоритм в ІСЗК, [34] – проведено аналіз тенденцій досліджень та майбутні виклики алгоритмів шифрування з використанням СЗК, [36] – удосконалено реалізацію поліноміального криптоалгоритму Рабіна на основі операції додавання, [39] удосконалено реалізацію трьохмодульного криптоалгоритму Рабіна на основі операції додавання, [42] представлено метод визначення простих та взаємно простих чисел типу 2^n+k на основі властивості періодичності, [44] – запропоновано метод відновлення десяткового числа з його залишків на основі додавання модулів добутку, [53] – розроблено методологію криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої та поліноміальної систем залишкових класів, [58] – Розроблено фундаментальні основи теорії дискретного логарифму у базисі Радемахера-Крестенсона; [59] – розроблено векторно-модульний метод модулярного множення.

Апробація результатів дисертації. Основні результати дисертаційної роботи доповідались і обговорювались на таких міжнародних та вітчизняних конференціях, школах, семінарах, як: International Conference on Advanced Computer Information Technologies (2024, 2023, 2022, 2021, 2020, 2018 роки), Intelligent Information Technologies & Systems of Information Security (2021 рік), International Conference on Computer Sciences and Information Technologies (CSIT) (2020 рік), Conference on Computer Science and Information Technologies (2020 рік), International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (2020, 2018, 2016 роки), IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (2019, 2017, 2015, 2012 роки), VI Всеукраїнська школа-семінар молодих вчених і студентів «Сучасні комп’ютерні інформаційні технології» (ACIT) (2018 рік), Науково-технічна конференція «Інформаційні моделі, системи та технології» (2018 рік), Міжнародна науково-

технічна конференція молодих учених та студентів «Актуальні задачі сучасних технологій» (2018 рік), International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM) (2017, 2015 роки), Міжнародна науково-технічна конференція ITSec-2025 Безпека інформаційних технологій (2025, 2024 роки), International Conference on Control, Automation and Systems (ICCAS–2016) (2016 рік), Міжнародна науково-практична конференція «Сучасні інформаційні та електронні технології» (2014 рік).

Публікації. За результатами досліджень, які викладені в дисертації, опубліковано 61 наукову працю, серед яких 5 колективних монографій, 26 публікацій у наукових фахових виданнях України та закордонних виданнях, в тому числі 11 статей включено в наукометричні бази Scopus та/або Web of Science (з них, відповідно до класифікації SCImago Journal and Country Rank або Journal Citation Reports, три статті віднесено до квартилю Q2, чотири – до квартилю Q3 та одна – до квартилю Q4) та 28 публікацій у матеріалах міжнародних та всеукраїнських конференцій (з них 22 публікацій включено в наукометричні бази Scopus та/або Web of Science), 2 патенти на корисну модель. Одна стаття, розділ монографії і одні тези написані автором одноосібно.

Обсяг і структура дисертації. Дисертація складається з анотації, змісту, вступу, шістьох розділів, загальних висновків, списку використаних джерел і додатків. Основний текст роботи викладено на 347 сторінках. Список використаних джерел нараховує 331 найменувань на 38 сторінках. Робота містить 65 таблиць та 99 рисунків (з них 10 таблиць і 6 рисунків займають повну сторінку), 2 додатків на 35 сторінках. Загальний обсяг роботи 420 сторінок.

РОЗДІЛ 1. АНАЛІЗ ПОБУДОВИ СИМЕТРИЧНИХ ТА АСИМЕТРИЧНИХ КРИПТОАЛГОРИТМІВ ТА УМОВИ ЗАСТОСУВАННЯ СИСТЕМИ ЧИСЛЕННЯ ЗАЛИШКОВИХ КЛАСІВ

1.1 Симетричні алгоритми шифрування

1.1.1 Стандарт шифрування даних DES

Стандарт шифрування даних DES (Data Encryption Standard) було розроблено в 1970-х роках спеціалістами компанії IBM і офіційно затверджено в 1976 році як федеральний стандарт США для захисту урядової та комерційної інформації, що не стосується питань національної безпеки [17].

Подібно до шифру одноразового блокнота, DES працює з даними, представленими у двійковій формі. Розмір блоку даних і довжина ключа встановлені рівними 64 бітам. Це означає, що двійкове повідомлення M ділиться на блоки по 64 біти, кожен з яких шифрується окремо за допомогою одного і того ж 64-бітового ключа K. Таким чином, процес шифрування здійснюється блоково, забезпечуючи незалежну обробку кожного блоку даних $M = M_1 M_2 M_3 ...$ перетворюється у шифртекст $C = C_1 C_2 C_3 ...$, де $C_i = E_K(M_i)$.

У стандарті DES кожен блок зашифрованих даних C є двійковою послідовністю довжиною 64 біти [18, 19]. Алгоритм шифрування E повинен відповідати трьом основним вимогам:

Можливість розшифрування: для будь-якого ключа Kрізним блокам відкритого тексту M' і M'' відповідають різні блоки криптотексту C' і C''. Це означає, що алгоритм E з використанням будь-якого ключа виконує перестановку всіх можливих двійкових послідовностей довжини 64 біти.

Ефективність: процеси шифрування і розшифрування повинні виконуватися з високою швидкістю, що забезпечує практичну застосовність алгоритму.

Надійність: у разі, якщо ключ K невідомий, алгоритм повинен забезпечувати стійкість до спроб розшифрування.

Процес шифрування в алгоритмі DES починається з обробки 64-бітового блоку відкритого тексту (рисунок 1.1). Після початкової перестановки цей блок розділяється на дві рівні частини: праву (R) та ліву (L) половини, кожна з яких має довжину 32 біти. Далі виконується 16 ітерацій (раундів), у межах яких функція f поєднує дані з ключем (рисунок 1.2).

На кожному раунді ключ K піддається зсуву, після чого з 56-бітового ключа обираються 48 бітів. Права половина даних розширюється з 32 до 48 бітів за допомогою спеціальної перестановки. Розширена права половина поєднується з 48 бітами ключа за допомогою операції XOR. Результат проходить через вісім S - блоків, які виконують нелінійне перетворення, утворюючи нову 32-бітову послідовність. Ця послідовність піддається черговій перестановці. Усі ці операції реалізуються з застосуванням функції f. Результат роботи f поєднується з лівою половиною даних за допомогою операції XOR, після чого попередня права половина стає новою лівою половиною, а новий результат стає правою половиною.

Ці дії повторюються у кожному з 16 раундів. У результаті формується вихідний 64-бітовий блок. На завершальному етапі виконується фінальна перестановка, яка є оберненою до початкової.

Раунд i шифрування можна формалізувати таким чином: нехай B_i - стан блоку після i-ї ітерації, L_i і R_i - ліва і права половини блоку B_i, K_i - 48-бітовий ключ, сформований для i-ї ітерації, f - функція, яка виконує всі підстановки, перестановки і операції XOR з ключем.

Тоді раунд описується формулами:

$$L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i). \quad (1.1)$$

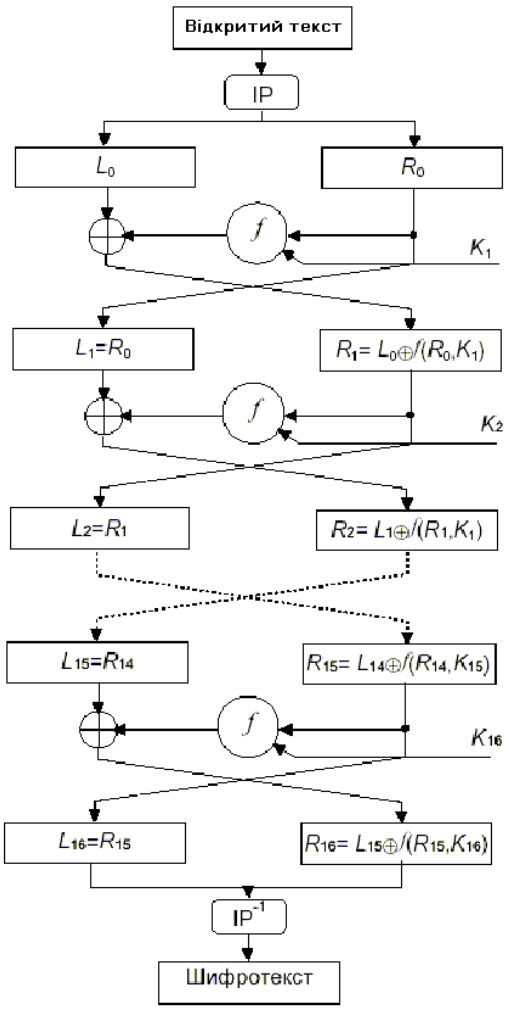


Рисунок 1.1 – Алгоритм DES

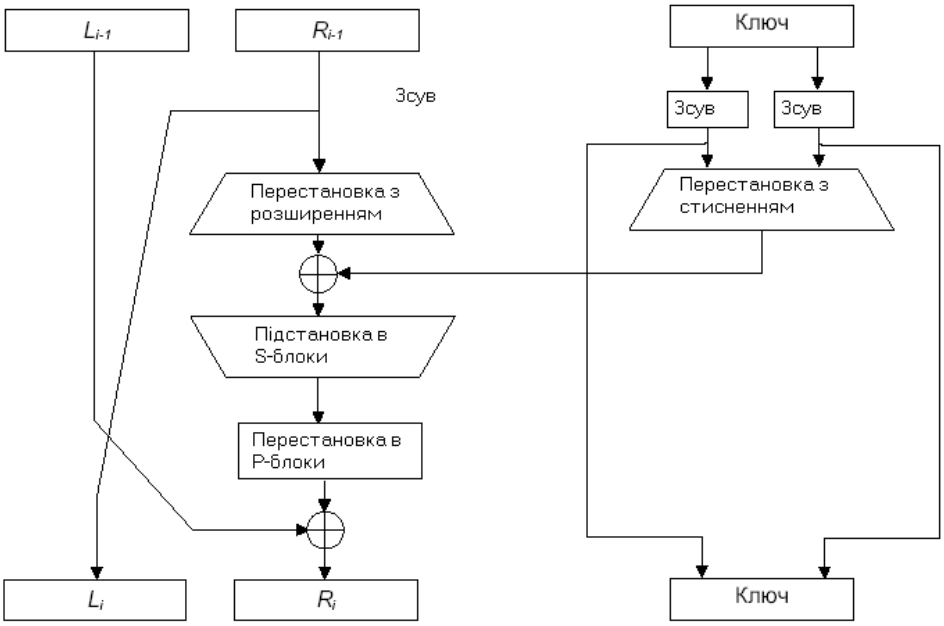


Рисунок 1.2 - Один етап DES

Початкова перестановка виконується до першого раунду шифрування. Вона передбачає реорганізацію позицій бітів у 64-бітовому блоці відповідно до заздалегідь визначеної схеми.

Наприклад, 58-й біт вихідного блоку переміщується на першу позицію, 50-й біт — на другу, 42-й — на третю і так далі, згідно з правилом, зазначеним у таблиці 1.1.

Початкова перестановка та відповідна кінцева перестановка не мають впливу на криптографічну стійкість алгоритму DES. Проте, оскільки багатобітова перестановка є складною для реалізації в програмному забезпеченні (на відміну від її апаратної реалізації, яка є значно простішою), у деяких програмних реалізаціях DES ці етапи опускаються. Однак виключення початкової та завершальної перестановок порушує відповідність алгоритму офіційному стандарту DES.

Таблиця 1.1 - Початкова перестановка

58,	50,	42,	34,	26,	18,	10,	2,	60,	52,	44,	36,	28,	20,	12,	4,
62,	54,	46,	38,	30,	22,	14,	6,	64,	56,	48,	40,	32,	24,	16,	8,
57,	49,	41,	33,	25,	17,	9,	1,	59,	51,	43,	35,	27,	19,	11,	3,
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

У алгоритмі DES початковий 64-бітовий ключ скорочується до 56 бітів шляхом видалення кожного восьмого біта, як це зазначено в таблиці 1.2. Відкинуті біти служать виключно для перевірки парності, що забезпечує можливість контролю коректності ключа.

Таблиця 1.2 - Перестановка ключа

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18,
10,	2,	59,	51,	43,	35,	27,	19,	11,	3,	60,	52,	44,	36,
63,	55,	47,	39,	31,	23,	15,	7,	62,	54,	46,	38,	30,	22,
14,	6,	61,	53,	45,	37,	29,	21,	13,	5,	28,	20,	12,	4

Після отримання 56-бітового ключа для кожного з 16 раундів DES генерується новий 48-бітовий підключ. Ці підключі K_i формуються за наступним

принципом. Спочатку 56-бітовий ключ розділяється на дві частини по 28 бітів. Далі кожна з цих половинок піддається циклічному зсуву вліво на один або два біти в залежності від поточного раунду, що зазначено в таблиці 1.3.

Таблиця 1.3 - Число бітів зсуву в залежності від етапу

Этап	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після циклічного зсуву здійснюється вибірка 48 бітів з початкових 56. Оскільки під час цього процесу не тільки вибирається підмножина бітів, але й змінюється їхній порядок, операція отримує назву перестановка із стисненням, що описана в таблиці 1.4. В результаті цієї операції утворюється нова послідовність з 48 бітів. Наприклад, біт, що перебуває на 33-й позиції зсунутого ключа, переміщається на 35-ту позицію у результаті, а 18-й біт з цього ж ключа буде відкинуто.

Таблиця 1.4 - Перестановка зі стисненням

14,	17,	11,	2,4,	1,	5,	3,	28,	15,	6,	21,	10,
23,	19,	11,	4,	26,	8,	16,	7,	27,	20,	13,	2,
41,	52,	31,	37,	47,	55,	30,	40,	51,	45,	33,	48,
44,	49,	39,	56,	34,	53,	46,	42,	50,	36,	29,	32

Унаслідок циклічних зсувів для кожного підключа вибирається різна підмножина бітів початкового ключа. Кожен біт ключа використовується приблизно в 14 з 16 можливих підключів. Права половина даних R_i піддається розширенню з 32 до 48 бітів. Оскільки в цьому процесі не просто відбувається дублювання певних бітів, але й змінюється їх порядок, операція отримала назву перестановка з розширенням, або ж *E*-блок. У цьому процесі для кожного 4-бітового блоку на вхідному етапі перший і четвертий біти стають відповідно першим і четвертим бітом вихідного блоку, тоді як другий і третій біти перетворюються на один біт у результаті. Наприклад, біт на 3-й позиції вхідного блоку переміщається на 4-ту позицію вхідного блоку, а біт на 21-й позиції

переміщається на 30-ту та 32-ту позиції вихідного блоку (як показано на рисунку 1.3).

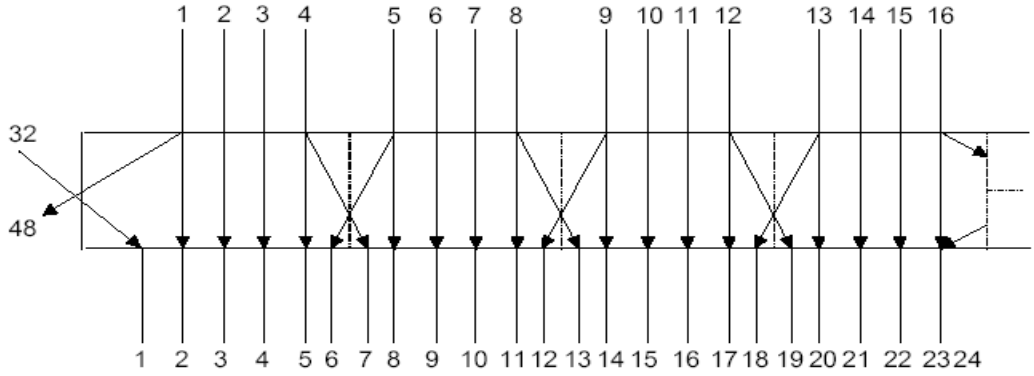


Рисунок 1.3 - Перестановка з розширенням

Хоча розмір вихідного блоку перевищує розмір вхідного, кожен вхідний блок формує унікальний вихідний блок (таблиця 1.5).

Таблиця 1.5 - Перестановка з розширенням

32,	1,	2,	3,	4,	5,	4,	5,	6,	7,	8,	9,
8,	9,	10,	11,	12.,	13,	12,	13,	14,	15,	16,	17,
16,	17,	18,	19,	20,	21,	20,	21,	22,	23,	24,	25,
24,	25,	26,	27,	28,	29,	28,	29,	30,	31,	32,	1

Після об'єднання стиснутого блоку з розширеним шляхом виконання операції XOR над 48-бітовим результатом, здійснюється операція підстановки. Підстановки виконуються у межах восьми окремих S-блоків, кожен з яких має 6 біт на вхід і 4 біти на вихід. 48 бітів розподіляються на вісім 6-бітових підблоків, кожен з яких обробляється окремим S-блоком: перший підблок обробляється S-блоком 1, другий — S-блоком 2, і так далі (рисунок 1.4).

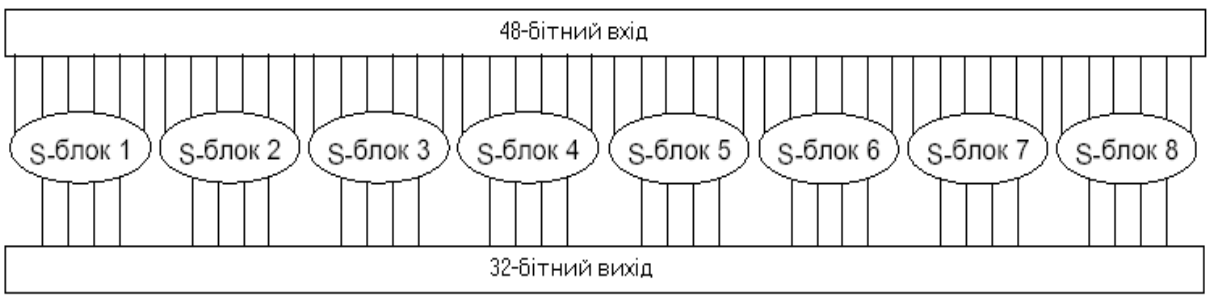


Рисунок 1.4 - Підстановка - S-блоки.

Кожен S-блок представляється таблицею з 2 рядків і 16 стовпців. Причому всі елементи у блоці є 4-бітовими числами. Вихідне значення S-блоку визначається на основі 6 вхідних біт, які вказують на відповідні номери стовпців та рядків, де слід шукати результат (таблиця 1.6).

Таблиця 1.6

S-блоки

S-блок 1:															
14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12.,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12.,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,
S-блок 2:															
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10,
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5,
0,	14,	7,	11,	10,	4,	13,	1,	5,	8,	12,	6,	9,	3,	2,	15,
13,	8,	10,	1,	3,	15,	4,	2,	11,	6,	7,	12,	0,	5,	14,	9,
S-блок 3:															
10,	0,	9,	14,	6,	3,	15,	5,	1,	13,	12,	7,	11,	4,	2,	8,
13,	7,	0,	9,	3,	4,	6,	10,	2,	8,	5,	14,	12,	11,	15,	1,
13,	6,	4,	9,	8,	15,	3,	0,	11,	1,	2,	12,	5,	10,	14,	7,
1,	10,	13,	0,	6,	9,	8,	7,	4,	15,	14,	3,	11,	5,	2,	12,
S-блок 4:															
7,	13,	14,	3,	0,	6,	9,	10,	1,	2,	8,	5,	11,	12,	4,	15,
13,	8,	11,	5,	6.	15,	0,	3,	4,	7,	2,	12,	1,	10,	14,	9,
10,	6,	9,	0,	12,	11,	7,	13,	15,	1,	3,	14,	5,	2,	8,	4,
3,	15,	0,	6,	10,	1,	13,	8,	9,	4,	5,	11,	12,	7,	2,	14,
S-блок 5:															
2,	12,	4,	1,	7,	10,	11,	6,	8,	5,	3,	15,	13,	0,	14,	9,
14,	11,	2,	12,	4,	7,	13,	1,	5,	0,	15,	10,	3,	9,	8,	6,
4,	2,	1,	11,	10,	13,	7,	8,	15,	9,	12,	5,	6,	3,	0,	14,
11,	8,	12,	7,	1,	14,	2,	13,	6,	15,	0,	9,	10,	4,	5,	3,
S-блок 6:															
12,	1,	10,	15,	9,	2,	6,	8,	0,	13,	3,	4,	14,	7,	5,	11,
10,	15,	4,	2,	7,	12,	9,	5,	6,	1,	13,	14,	0,	11,	3,	8,
9,	14,	15,	5,	2,	8,	12,	3,	7,	0,	4,	10,	1,	13,	11,	6,
4,	3,	2,	12,	9,	5,	15,	10,	11,	14,	1,	7,	6,	0,	8,	13,

Продовження таблиці 1.6

S-блок 7:															
4,	11,	2,	14,	15,	0,	8,	13,	3,	12,	9,	7,	5,	10,	6,	1,
13,	0,	11,	7,	4,	9,	1,	10,	14,	3,	5,	12,	2,	15,	8,	6,
1,	4,	11,	13,	12,	3,	7,	14,	10,	15,	6,	8,	0,	5,	9,	2,
6,	11,	13,	8,	1,	4,	10,	7,	9,	5,	0,	15,	14,	2,	3,	12,
S-блок 8:															
13,	2,	8,	4,	6,	15,	11,	1,	10,	9,	3,	14,	5,	0,	12,	7,
1,	15,	13,	8,	10,	3,	7,	4,	12,	5,	6,	11,	0,	14,	9,	2,
7,	11,	4,	1,	9,	12,	14,	2,	0,	6,	10,	13,	15,	3,	5,	8,
2,	1,	14,	7,	4,	10,	8,	13,	15,	12,	9,	0,	3,	5,	6,	11

Вхідні біти визначають відповідний елемент S-блоку за певним принципом. Наприклад розглянемо 6-бітовий вхід S-блоку: b_1, b_2, b_3, b_4, b_5 і b_6. Біти b_1 і b_6 комбінуються для формування 2-бітового числа в діапазоні від 0 до 3, що вказує на рядок таблиці. Середні 4 біти, з b_2 по b_5, утворюють 4-бітове число від 0 до 15, яке відповідає стовпцю таблиці. Наприклад, припустимо, що на вхід шостого S-блоку (біти функції XOR з 31 по 36) подаються як 110011. Перший і останній біти, поєднуючись, утворюють число 11, що вказує на рядок 3 в шостому S-блоці. Середні 4 біти утворюють число 1001, що відповідає стовпцю 9 того ж S-блоку. Елемент S-блоку 6, який знаходиться на перетині рядка 3 і стовпця 9, дорівнює 14 (нумерація рядків і стовпців починається з 0.) Таким чином, на виході замість 110011 буде отримано значення 1110.

Значно спрощує програмну реалізацію S-блоків у вигляді масивів з 64 елементами. Підстановка за допомогою S-блоків є критично важливим етапом у DES, оскільки інші операції алгоритму є лінійними і відносно простими для аналізу. Натомість S-блоки є нелінійними, і саме вони значною мірою забезпечують криптографічну стійкість DES.

Після підстановки через S-блоки, 32-бітовий вихід (вісім 4-бітових блоків) піддається перестановці за допомогою P-блоку. Ця перестановка змінює розташування кожного біта, переміщаючи їх на нові позиції. Жоден біт не дублюється і не залишається без уваги. Такий процес називається прямою або простотою перестановкою (таблиця 1.7). Наприклад, біт у позиції 21 переміщається на позицію 4, а біт у позиції 4 — на позицію 31.

Таблиця 1.7

Перестановка за допомогою P-блоків

16,	7,	20,	21,	29,	12,	28,	17,	1,	15,	23,	26,	5,	18,	31,	10,
2,	8,	24,	14,	32,	27,	3,	9,	19,	13,	30,	6,	22,	11,	4,	25

На завершальному етапі результат перестановки за допомогою P-блоків об'єднується з лівою половиною початкового 64-бітового блоку з використанням XOR. Після цього ліва та права половини міняються місцями.

Кінцева перестановка є зворотною до початкової перестановки. Після останнього етапу DES ліва і права половини не міняються місцями, а замість цього об'єднаний блок $R_{16}L_{16}$ використовується як вхід для заключної перестановки (таблиця 1.8). Перестановка половинок із подальшим циклічним зсувом дала б той самий результат. Така реалізація дозволяє використовувати алгоритм як для шифрування, так і для розшифрування.

Таблиця 1.8

Заключна перестановка

40,	8,	48,	16,	56,	24,	64,	32,	39,	7,	47,	15,	55,	23,	63,	31,
38,	6,	46,	14,	54,	22,	62,	30,	37,	5,	45,	13,	53,	21,	61,	29,
36,	4,	44,	12,	52,	20,	60,	28,	35,	3,	43,	11,	51,	19,	59,	27,
34,	2,	42,	10,	50,	18,	58,	26,	33,	1,	41,	9,	49,	17,	57,	25

Для шифрування та розшифрування застосовується один і той самий алгоритм. DES дозволяє використовувати однакову функцію як для шифрування, так і для розшифрування блоку. Основною відмінністю є порядок використання ключів: для розшифрування ключі використовуються в зворотному порядку. Тобто, якщо на етапах шифрування використовувалися певні ключі $K_1, K_2, K_3, ..., K_{16}$, то для розшифрування ці ключі будуть використовуватися в зворотному порядку $K_{16}, K_{15}, K_{14}, ..., K_1$.

Алгоритм генерування ключів для кожного етапу також має циклічний характер. Ключ зсувається праворуч, при цьому кількість позицій зсуву змінюється за певною схемою: 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Існує чотири основні режими роботи алгоритму DES: електронна кодова книга (ECB), зчеплення блоків шифру (CBC), зворотний зв’язок по шифртексту (CFB) та зворотний зв’язок по виходу (OFB).

Алгоритм DES має історичну важливість і був ефективним на момент свого створення. Проте короткий ключ і вразливість до сучасних атак роблять його непридатним для використання в сучасних системах, що вимагають високого рівня безпеки. Для заміни DES рекомендується використовувати AES або 3DES у випадках, коли необхідна сумісність із застарілими системами.

1.1.2 Стандарт потрійного шифрування даних (3DES)

Triple-DES [20] — це алгоритм шифрування блочним шифром. Як видно з назви, 3DES застосовує DES тричі до кожного блоку даних, щоб підвищити безпеку зашифрованих даних. Оскільки безпека 3DES втричі вища, ніж DES, тепер він вважається кращим, ніж DES [21-23]. Однак він займає значну кількість часу в порівнянні з попередником. 3DES працює так само, як і DES, у циклі довжиною 3.

Схема шифрування на основі Triple DES показані на рисунку 1.5.

Спочатку вихідний звичайний текст шифрується одним ключем, отриманий зашифрований текст знову шифрується іншим ключем, і, нарешті, виконується знову третім ключем.

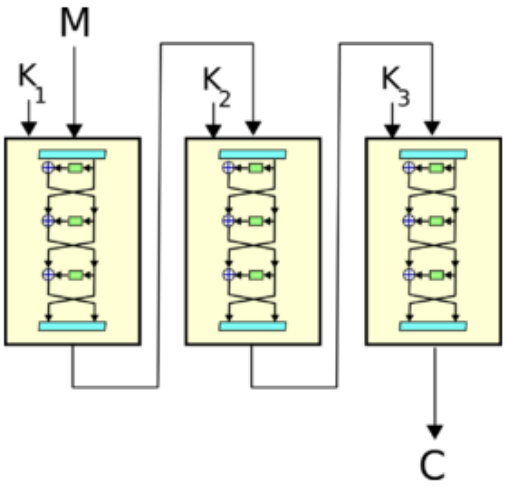


Рисунок 1.5 – Схема шифрування на основі Triple DES

1.1.3 Симетричний алгоритм шифрування AES

AES [24-26] розділяє 128 біт звичайного тексту на 16 байтів вхідних даних. Кожен байт, тобто від A_0 до A_{15}, складається з 8 біт даних. 8-розрядне двійкове число кожного байта передається до *S*-блоків рівня заміщення байтів (рис. 1.6).

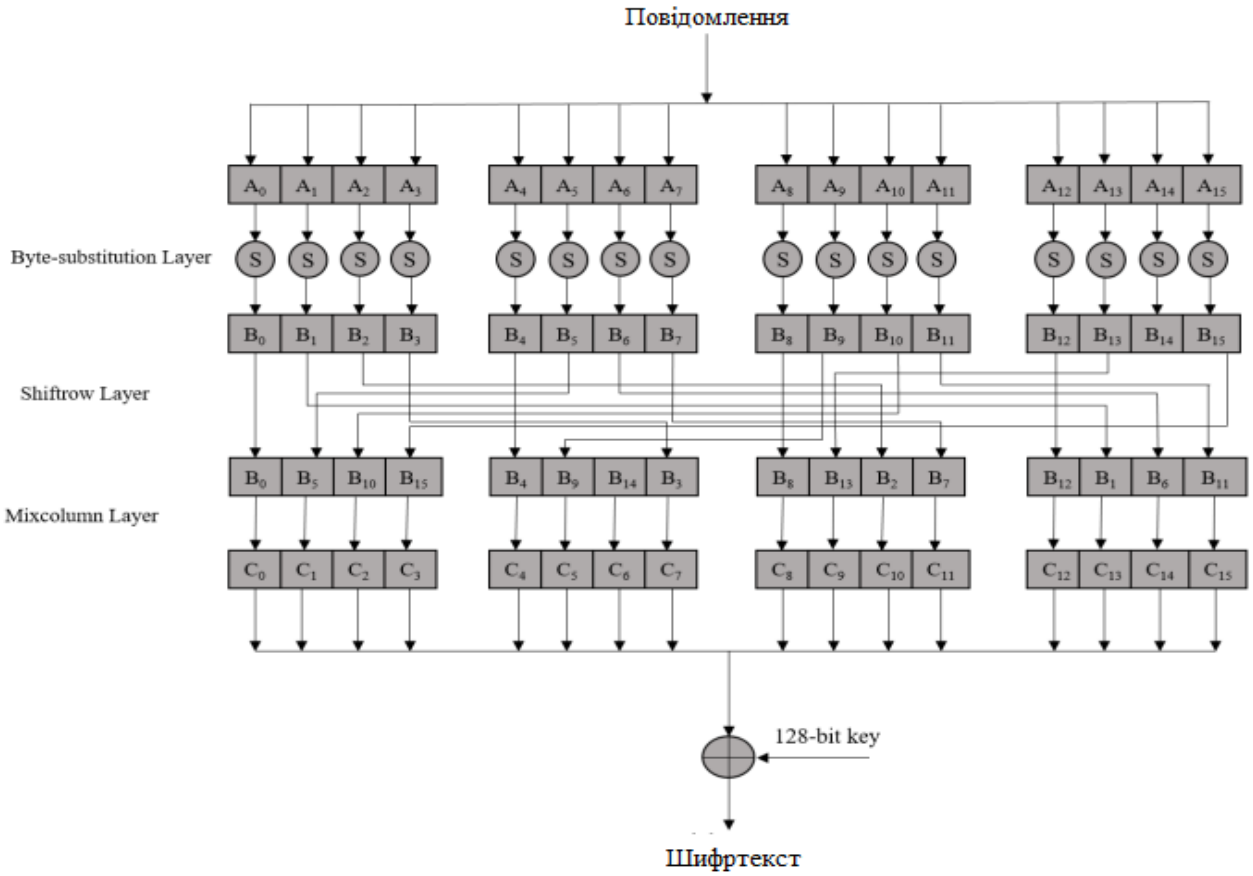


Рисунок 1.6 – Потік процесу AES

Рівень заміни байтів – це ряд із шістнадцяти паралельних *S*-блоків, кожен із яких містить вісім вхідних і вісім вихідних бітів. Усі шістнадцять *S*-Box ідентичні, і кожен байт даних A_i замінюється іншим байтом B_i в результаті *S*–Box [23, 27]. Тому операцію *S*-блоку можна позначити як $S(A_i) = B_i$.

Схему симетричного шифрування представлено на рисунку 1.7.

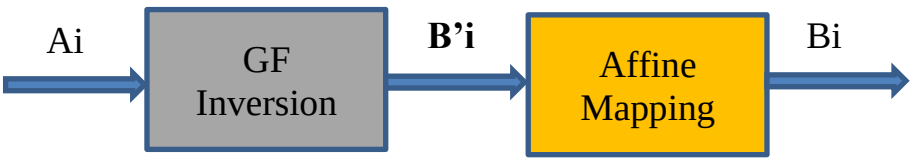


Рисунок 1.7 – Схема симетричного шифрування

Заміна S-Box – це однозначне відображення 256 вхідних елементів на один вихід [28, 29]. Вони володіють сильною алгебраїчною структурою.

Побудова передбачає двоетапне математичне перетворення, як показано на рисунку 1.8. У першій частині для кожного вхідного елемента A_i обчислюється обернений елемент $A_i^{-1} = B_i$, який можна безпосередньо замінити за допомогою мультиплікативно зворотної таблиці.

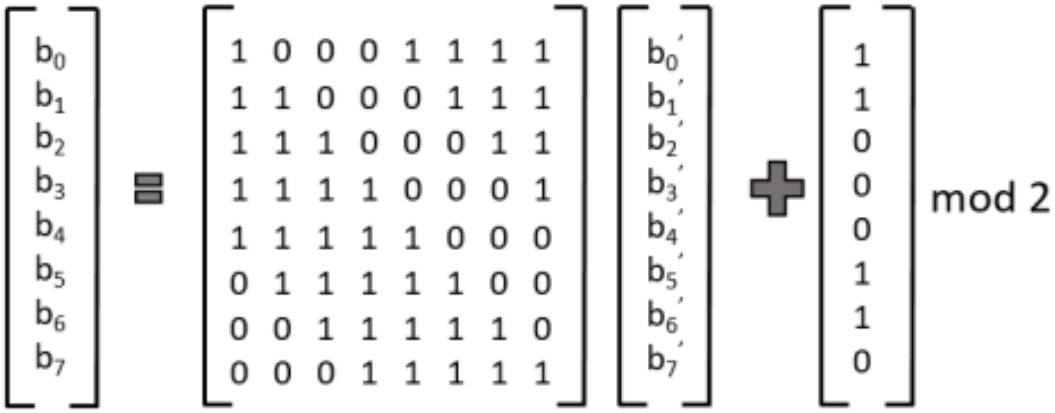


Рисунок 1.8 – Афінне відображення

Тому це перетворення відоме як інверсія поля Галуа. У другій частині рівня кожен із вихідних байтів інверсії поля Галуа, тобто B_i, множиться на постійну бітову матрицю, після чого додається постійний 8-бітний вектор як показано на рисунок 1.8. Це відомо як афінне відображення [30-32].

На рисунку 1.8 $B_i' = (b_7^1, b_6^{1\prime}, \dots, b_0^{1\prime})$ є порозрядним векторним представленням кінцевого виходу *S*-box. Наприклад, припустимо A_i=11001100 як вхідний біт для *S*-блоку. Спочатку він перетворюється в шістнадцяткову систему запису, тобто *CC*. Потім еквівалентний обернений витягується з зворотної таблиці, тобто 1B. 1B потім перетворюється на двійкове представлення (00011011) і передається до B_i' (зліва направо), і виконується матрична операція. Остаточний вихід B_i=01001011 (4B) замінюється як вихід для *CC* у *S*-блоку. Обчислюючи обидва кроки для всіх 256 елементів введення *S*-Box, можна побудувати на основі таблиці 1.9.

Таблиця 1.9

Таблиця S-box

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
10	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
11	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
12	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
13	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
14	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
15	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

На рівні рядків зсуву шістнадцять вихідних бітів ($B_0...B_{15}$) із рівня заміщення байтів організовані як матриця 4x4 [33]. Кожні 4 байти беруться як вектор-стовпець у вхідній матриці. Під час перетворення байти з другого рядка циклічно зсуваються вправо на три байти. Подібним чином третій рядок зміщується на два байти, а четвертий – на один байт. Перший рядок вихідної матриці залишається незмінним, як показано на рисунку 1.9.

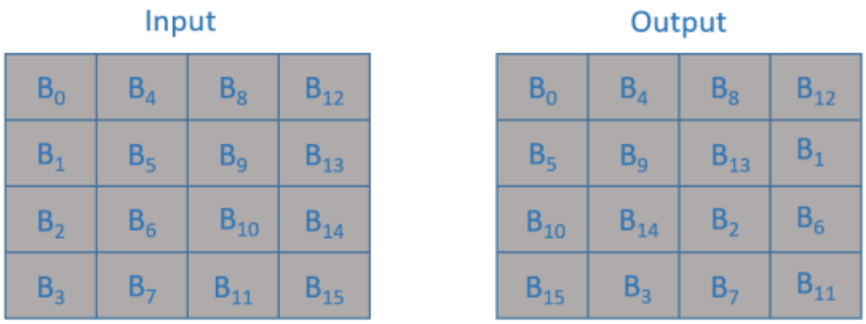


Рисунок 1.9 – Матриці входу та виходу

Шар Mixcolumn об’єднує кожен стовпець матриці стану. Чотири байти з кожного стовпця вихідної матриці стану (шару Shiftrows) розглядаються як вхідні дані для рівня mixcolumn. Цей вектор-стовпець множиться на фіксовану матрицю 4x4, яка складається з постійних значень, як показано на рисунку 1.10.

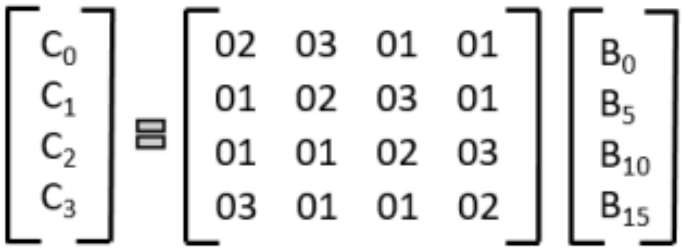


Рисунок 1.10 - Шар Mixcolumn

Вихідні байти $C_0....C_{15}$ є результатом цього множення матриці. Усі арифметичні дії виконуються в полі Галуа. Фіксовані константи в матриці представлені в шістнадцятковій формі.

У AES із 128-бітним розміром ключа спочатку ключ генерується генератором випадкових чисел (ГВЧ) і обробляється через 10 циклів. AES має рівень додавання ключів у кожному раунді (підключ є в кожному раунді AES). У першому циклі згенеровані 128 бітів розбиваються на чотири 32 біти, і кожні 32 біти зберігаються в масиві, позначеному як *W*, як показано на рисунку 1.11.

Від другого до останнього раунду інші елементи масиву обчислюються згідно співвідношення: $W[4i] = W[4(i_1)] + g(W[4i_1])$, де $i = 1, \ldots, 10$ – кількість раундів, g – нелінійна функція з 32-бітовим входом і виходом. Решту три масиви підключа обчислюються рекурсивно як: $W[4i + j] = W[4i + j_1] + W[4(i_1) + j]$, де $i = 1, \ldots, 10$ і $j = 1,2,3$. Функція інвертує вхідні біти та виконує побайтову підстановку S-Box, а також додає (XOR) округлений коефіцієнт RC. Коефіцієнт округлення є елементом поля Галуа $GF(2^8)$, тобто 8-бітним значенням.

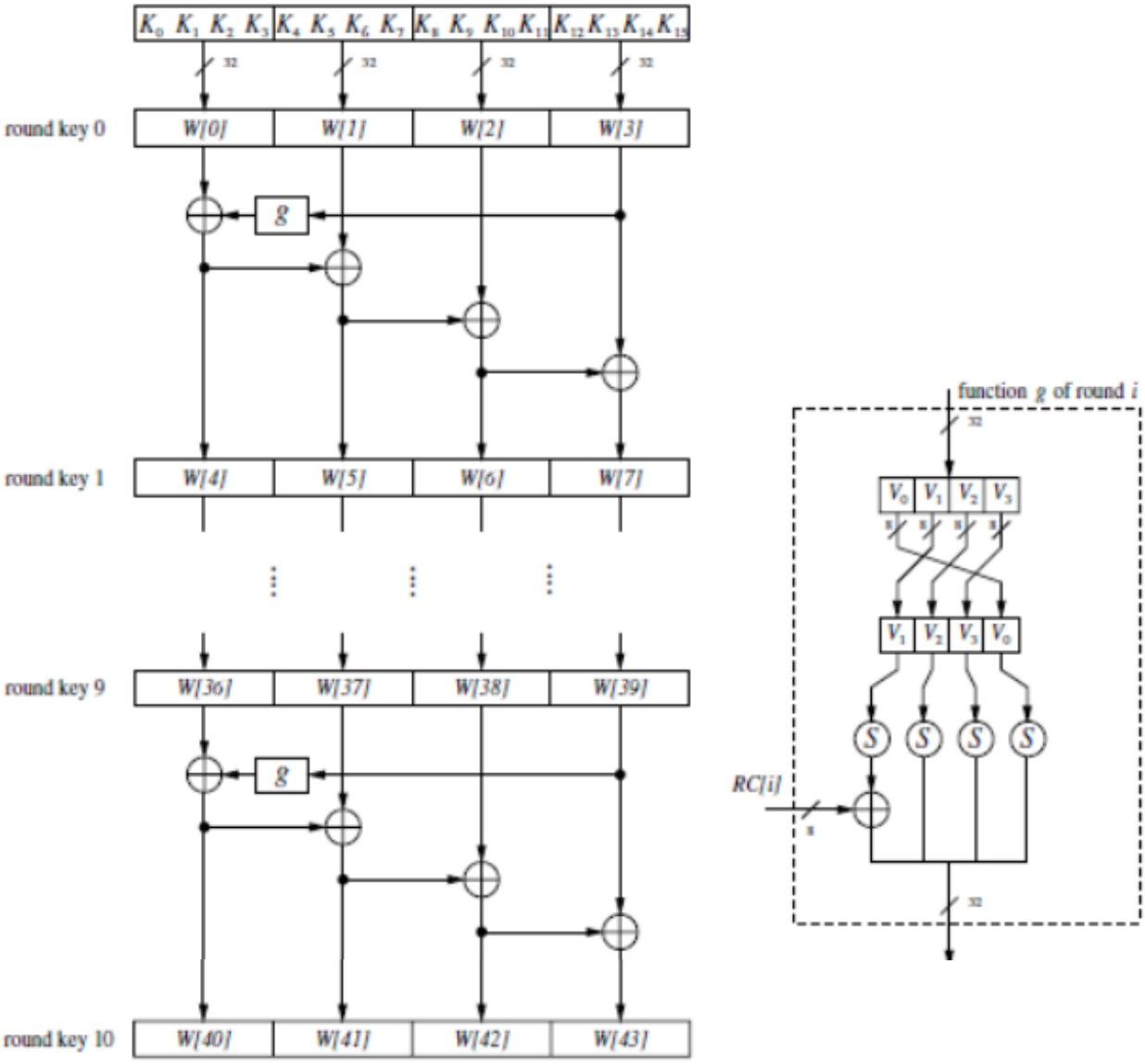


Рисунок 1.11 – Розподіл ключів AES для 128-бітного розміру ключа [27]

Він додається лише до крайнього лівого байта у функції g(). Коефіцієнти раунду для кожного раунду є стандартними значеннями, вказаними нижче:

$RC[1] = x_0 = (00000001)$, $RC[2] = x_1 = (00000010)$, $RC[3] = x_2 = (00000100)$, $RC[4] = x_3 = (00001000)$,…,$RC[10] = x_9 = (00110110)$, де i = 1, . . . , 10.

Операції, що виконуються на кожному з рівнів розшифрування AES [34], є зворотними до операцій шифрування, як показано на рисунку 1.12.

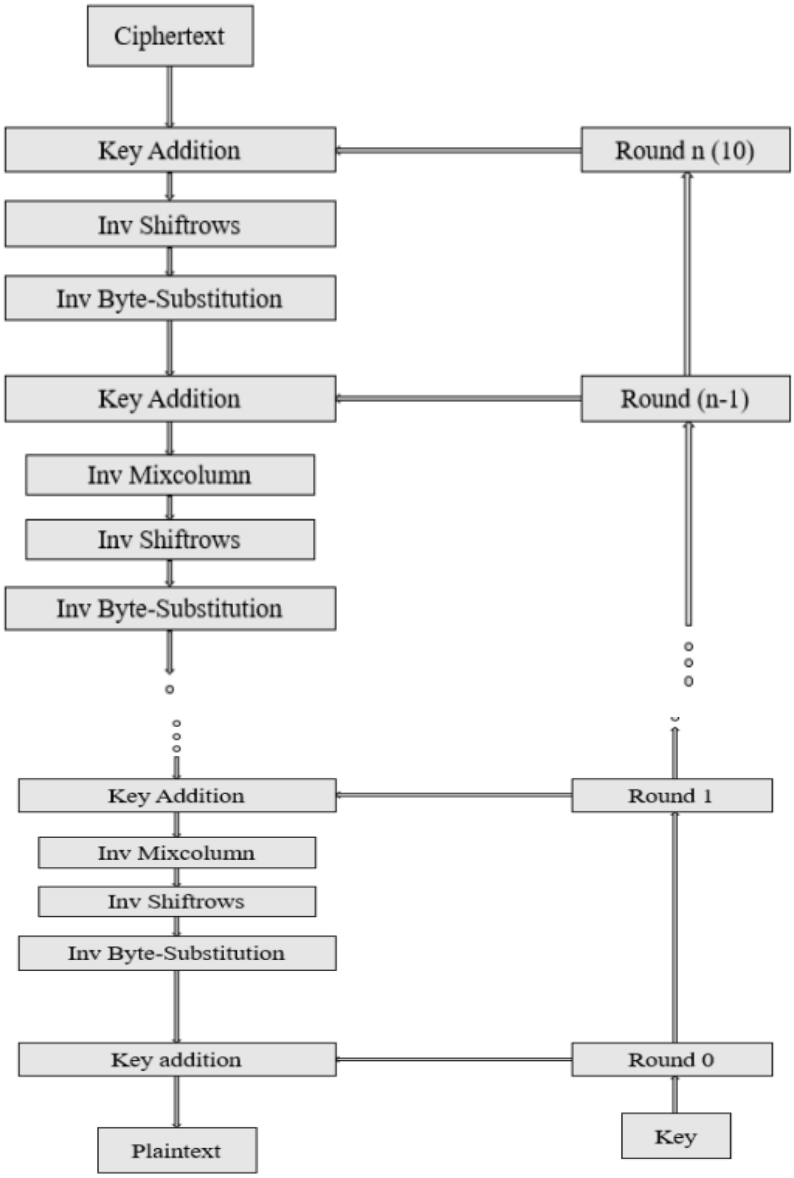


Рисунок 1.12 – Схема розшифрування згідно алгоритму AES

Проце розшифрування починається із зашифрованого тексту з останнього рівня додавання ключа. Вихідні дані потім обробляються за допомогою рівня Inverse mixcolumn, рівня Inverse shiftrow і, нарешті, до рівня зворотної заміни байтів. Перший раунд розшифрування не містить шар mixcolumn, оскільки його немає в останньому раунді шифрування AES. Таким чином, розшифрування AES починається з виведення рівня додавання ключа, надісланого до рівня інверсного зсуву рядків.

Зворотний крок MixColumn застосовується до вхідних байтів після додавання підключа (перший раунд розшифрування є винятком). Операція

змінюється шляхом множення вхідних байтів на обернену матрицю з постійними записами. Приклад перших чотирьох вхідних байтів (C_0, C_1, C_2, C_3), помножених на постійну матрицю, показано на рисунку 1.13.

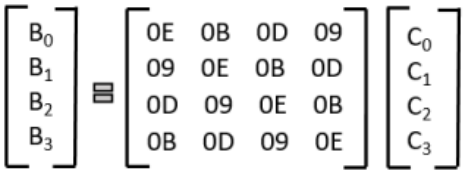


Рисунок 1.13 – Зворотний шар Mixcolumn

Множення та додавання коефіцієнтів виконується в GF(2^8), так само, як шар mixcolumn у шифруванні.

На підрівні Inverse ShiftRows виконується операція зсуву рядків вхідної матриці в протилежному напрямку. Вихідні байти (B_0...B_{15}) інверсного рівня стовпця змішування впорядковані в матриці 4x4. Перший рядок матриці залишається незмінним; другий рядок циклічно зміщується вправо на три байти. Подібним чином третій рядок зсувається на два байти, а четвертий – на один байт вправо. Результат шару інверсного зсуву рядків показано на рисунку 1.14.

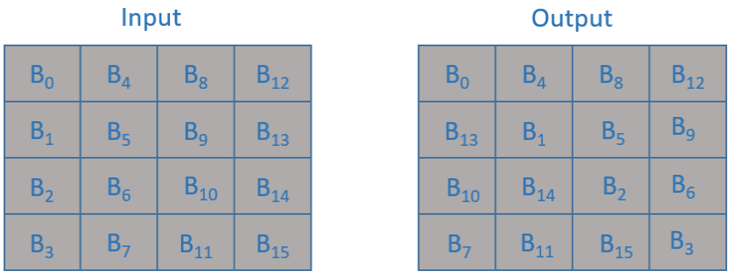


Рисунок 1.14 –Матриці в інверсних Shiftrows

На рівні інверсної заміни байтів всі операції змінюються, щоб отримати інверсні S-блоки. Вихідні байти з шару Inverse shiftrows проходять два перетворення: афінне відображення та інверсія Галуа. Подібно до шифрування, вхідні біти множаться на обернену матрицю в афінному відображенні з подальшим додаванням постійного 8-бітового вектора, показаного на рисунку 1.15.

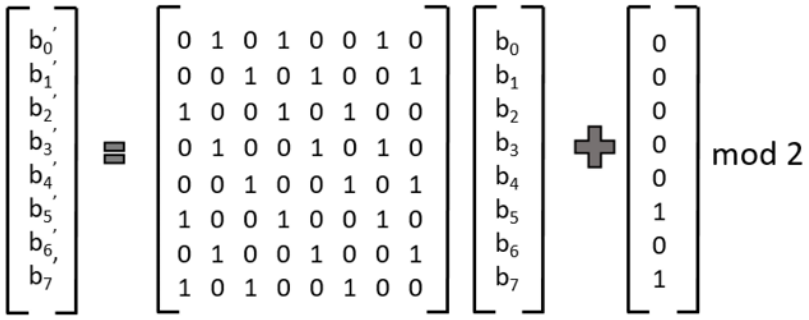


Рисунок 1.15 – Обернене афінне відображення

На другому кроці біти у векторі B_i^1 піддаються інверсії Галуа. Усі множення та інверсії обчислюються за допомогою фіксованого незвідного полінома $P(x) = x^8+x^4+x^3+x+1$.

Інверсія відповідного біта замінюється з мультиплікативної зворотної таблиці як $A_i=(B_i^1)^{-1}$. Усі 256 елементів обчислюються за допомогою виразу $A_i=S^{-1}(B_i)$ і утворюють інверсну таблицю S-блоку з їх шістнадцятковими нотаціями, як показано в таблиці 1.10. Тому до кінця інверсного шару заміни байтів зашифрований текст розшифровується у відкритий текст.

Таблиця 1.10

Зворотний S-блок

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
10	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
11	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
12	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
13	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
14	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
15	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

де B_i – вхід, а B_i^1 – вихід афінного відображення.

У [35-37] показано результати, отримані під час запуску програми моделювання з використанням різних даних, таких як текстові файли, файли PDF, документ Word і зображення різних розмірів, які використовуються для проведення 35 експериментів, де порівнюється три алгоритми DES, 3DES і AES.

Результати для криптографічних алгоритмів DES, 3DES і AES наведені в таблиці 1.11.

Таблиця 1.11

Порівняння криптографічних алгоритмів DES, 3DES і AES

Алгоритми Фактори	AES	DES	3DES
Тип шифру	Симетричний блоковий шифр	Симетричний блоковий шифр	Симетричний блоковий шифр
Розмір ключа	128, 192 або 256 бітів	56 бітів	168, 112 або 56 бітів
Розмір блоку	128, 192 або 256 бітів	64 бітів	64 бітів
Розроблений, роки	2000	1977	1978
Раунди	10 (128 бітів), 12 (192 біти), 14 (256 бітів)	16	48
Час шифрування (мс) (Розмір файлу 25 Кб)	0.89	1.391	0.48, 0.67,0.95

На рисунку 1.16 показано часові затрати алгоритмів шифрування за допомогою DES, 3DES і AES.

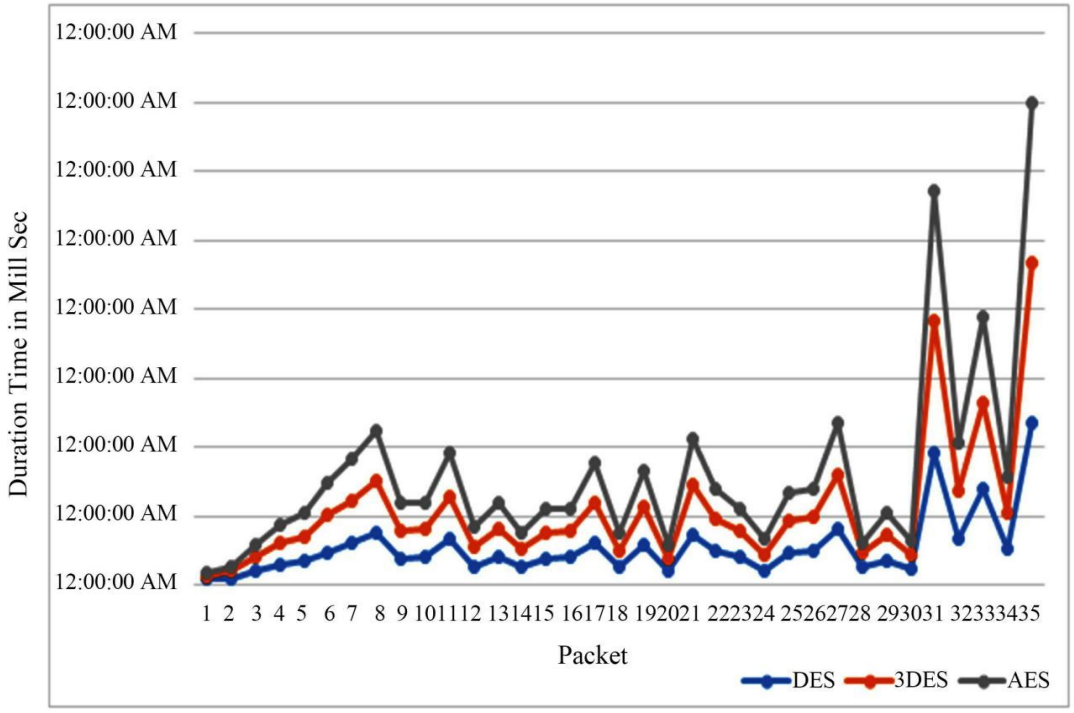


Рисунок 1.16 – Час роботи алгоритмів шифрування DES, 3DES і AES

Помічено, що кращим показником швидкодії процесу шифрування характеризується алгоритм AES для всіх розмірів вхідних параметрів.

Пропускна здатність схеми шифрування обчислюється шляхом ділення загальної кількості зашифрованого відкритого тексту в мегабайтах на загальний час шифрування для кожного алгоритму [38].

На рисунку 1.17 показано пропускну здатність DES, 3DES і AES (мегабайт/с) в експерименті.

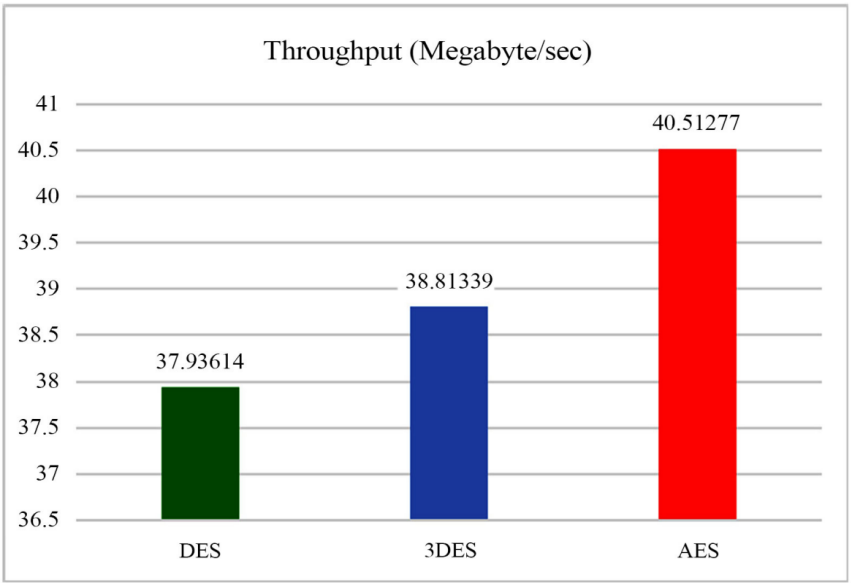


Рисунок 1.17 - Пропускна здатність DES, 3DES і AES (мегабайт/с)

Однак існуючі симетричні криптоалгоритми не володіють високою швидкодією при їх реалізації в процесі шифрування/розшифрування і не забезпечують належного рівня захисту інформаційних потоків.

Результат показує, що пропускна здатність алгоритму AES найвища для всіх розмірів.

На рисунку 1.18 показано середній час кожного алгоритму шифрування в експерименті.

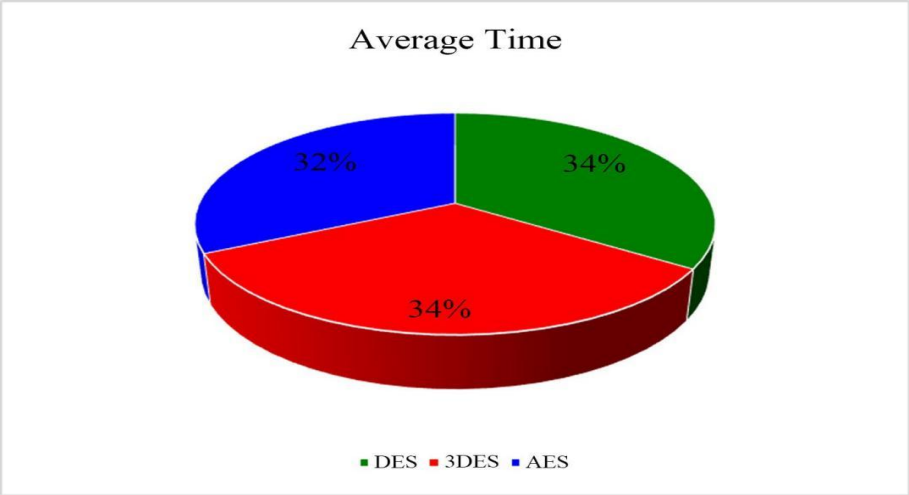


Рисунок 1.18 - Середній час кожного алгоритму шифрування

Результати показують перевагу алгоритму 3DES над іншими алгоритмами з точки зору часу обробки.

Існує значна кількість криптографічних алгоритмів, призначених для забезпечення цілісності та конфіденційності даних. Будь-який із них потенційно здатний гарантувати захист інформації, проте вибір конкретного алгоритму визначається низкою чинників, серед яких - показники продуктивності, обчислювальні можливості системи, рівень алгоритмічної складності та ступінь криптографічної стійкості [1]. У наукових дослідженнях різними авторами здійснювалися порівняльні оцінки ефективності алгоритмів шифрування на основі аналізу низки технічних та експлуатаційних параметрів.

Abood і Guirguis [2] провели порівняльне дослідження наявних на даний момент алгоритмів шифрування, таких як AES, DES, TDES, DSA, RSA, ECC, EEE та CR4, на основі їхньої безпеки, розміру ключа, складності та часу.

На основі цього дослідження AES, BlowFish [3, 4], RC4, E-DES і TDES [5] є швидкими алгоритмами з точки зору шифрування/розшифрування. Вони дійшли висновку, що AES є найнадійнішим алгоритмом з точки зору стійкості, довжини ключа, безпеки, а також гнучкості.

Ріман і Ебі-Чар [6] проаналізували продуктивність чотирьох алгоритмів блокового шифрування, таких як AES, DES, 3DES і E-DES, на основі швидкості, розміру блоку та розміру ключа. Вони прийшли до висновку, що E-DES

перевершує всі інші три моделі на основі вхідних файлів і експериментальних результатів. Він шифрує/дешифрує дані швидше, ніж інші тестовані алгоритми. У порівнянні з DES E-DES продемонстрував покращення у двох сферах; більш проста реалізація та більш важливі ключові та вхідні блоки для забезпечення безпеки.

Діксіт та ін. [7] в своїй праці провів аналіз існуючих симетричних методів шифрування\розшифрування та здійснив порівняння їх з точки зору розробки, кількості раундів, довжини ключа, розміру блоку, знайдених атак, рівня безпеки, можливих ключів, часу, необхідного для перевірки всіх можливих ключів. Також здійснили порівняння традиційних і гібридних методів шифрування, таких як DSA-RSA, AES-RC4, RC4-AES-SERPENT, SERPENT-RC4 і AES-ECC. Вони дійшли висновку, що AES-ECC зменшує часову та просторову складність, а гібридний алгоритм DSA-RSA має кращу продуктивність і пропускну здатність.

Бханот і Ганс [8] порівняли та проаналізували різні алгоритми шифрування даних як у симетричних, так і в асиметричних категоріях, щоб знайти найкращий алгоритм. Вони порівнювали алгоритми на основі розробки, довжини ключа, кількості раундів, необхідних для шифрування та розшифрування, розміру блоку, різних типів знайдених атак, рівня безпеки та швидкості шифрування. У своєму дослідженні вони помітили, що стійкість кожного алгоритму може визначатися керуванням ключами, типом криптографії (симетрична і асиметрична), кількістю ключів, кількістю бітів, які використовуються в ключі тощо. Вони дійшли висновку, що BlowFish і ECC мають кращі результати продуктивності.

Вахід та інші [9] провели аналіз різних алгоритмів шифрування, таких як DES, 3DES, AES, RSA та BlowFish, на основі їх продуктивності, слабких і сильних сторін.

У своєму дослідженні вони дійшли висновку, що BlowFish кращий з точки зору використання пам’яті, часу та пом’якшення атак. Однак, якщо

конфіденційність і цілісність є основними проблемами, тоді AES стає кращим вибором.

Каур і Махаджан [10] провели порівняння симетричних ключових алгоритмів як у локальній системі, так і в хмарній системі. Для двох систем вони реалізували чотири симетричні ключові алгоритми: AES, DES, BlowFish і DESede. Локальна реалізація була виконана за допомогою Java на eclipse, а хмарна реалізація використовувала eclipse-SDK і Google App Engine. Для оцінки використовувалися вхідні дані розміром 10 КБ, 13 КБ, 39 КБ і 56 КБ. Вони виявили, що коефіцієнт прискорення DES і BlowFish відображає невелику зміну зі збільшенням розміру вхідних даних, тоді як для AES він зменшується.

Вони відзначили, що DESede вимагав більше часу, а BlowFish був найнижчим за часом.

У їхньому висновку зазначено, що з точки зору продуктивності BlowFish, AES і DES є кращими алгоритмами, а AES продемонстрував високу безпеку з найменшими витратами часу.

Хенді та ін. [11] розробив легку криптосистему, яку називають простим і високонадійним алгоритмом шифрування/розшифрування (SHSED) для зберігання даних у хмарних обчисленнях.

Їхня система заснована на алгоритмі шифрування IDEA, і її продуктивність порівнювалася з AES, DES і LED.

Запропонований алгоритм працює краще, ніж AES і LED, однак, його продуктивність була трохи повільнішою, ніж DES.

Тьягі та Ганпаті [12] провели теоретичне дослідження чотирьох популярних симетричних алгоритмів, таких як DES, 3DES, AES та BlowFish. Вони порівнювали ці алгоритми на основі різних факторів, таких як швидкість, розмір блоку, захист від атак, конфіденційність, пропускна спроможність, енергоспоживання, розмір ключа тощо. Згідно з їхнім дослідженням, BlowFish мав кращу продуктивність, враховуючи час шифрування, час розшифрування та пропускну здатність. Вони також дійшли висновку, що 3DES був найнижчим з точки зору продуктивності.

Princy [5] проаналізував різні симетричні ключові алгоритми, такі як AES, DES, 3DES, BlowFish, RC4 і RC6, щодо безпеки, продуктивності, часу обробки та кількості раундів. Результати показали, що BlowFish забезпечив більшу конфіденційність і безпеку передачі даних через небезпечний канал, збільшивши розмір ключа зі 128 до 448.

Mathur і Kesarwani [3] провели порівняння продуктивності DES, 3DES, AES, RC2, RC6 і BlowFish. Здійснювалася оцінка цих алгоритмів з врахуванням основних параметрів: довжини ключа, методу кодування, типу даних і розміру пакета. Було встановлено, що методи кодування не впливають на процеси шифрування чи розшифрування цих алгоритмів. BlowFish перевершив усі інші алгоритми при зміні розміру пакету. Крім того, автори показали, що RC2 мав низьку продуктивність і пропускну здатність порівняно з іншими алгоритмами. RC2, RC6 і BlowFish зіткнулися зі значним недоліком порівняно з іншими алгоритмами, коли тип вхідних даних змінився з тексту на зображення. Проведені дослідження показали, що збільшення розміру ключа призведе до зростання витрат на енергоспоживання та часу.

Nema та Rizvi [13] проаналізували DES, 3DES, AES, BlowFish, Twofish, Threefish, RC2, RC4, RC5 та RC6 з точки зору пропускної здатності, масштабованості, безпеки, використання пам’яті, енергоспоживання, швидкості та гнучкості. Їх результати показують, що BlowFish був найефективнішим алгоритмом з точки зору безпеки, гнучкості, використання пам’яті, а також продуктивності.

Марваха та ін. [14] проаналізували DES, Triple DES і RSA рівень безпеки, час шифрування/розшифрування, пропускну здатність. Продуктивність алгоритмів залежить від розміру вхідних даних. В результаті проведених досліджень зробили висновок, що швидкість і пропускна здатність DES кращі, ніж у 3DES. Крім того, DES показав меншу стійкість у порівнянні з 3DES і RSA. 3DES забезпечив більшу конфіденційність і масштабованість у цілому, але порівняно з DES і RSA він споживав більше енергії з меншою пропускною здатністю.

Надім і Джавед [15] розробили та порівняли DES, 3DES, AES і BlowFish за допомогою Java та оцінили їх продуктивність на основі різних типів і розмірів вхідних даних, швидкості виконання та різних апаратних платформ. Виходячи з їх порівняння, BlowFish мав кращу продуктивність, ніж інші. Вони ранжували ці алгоритми нпо часових затратах: BlowFish (найшвидший), DES, AES (середні щодо швидкодії), Triple DES (найповільніший). Швидкість виконання алгоритмів на основі блочного шифру зростала при збільшенні розміру блоків і зменшенні розміру ключа. Однак в алгоритмах потокового шифрування швидкість зменшується при збільшенні розміру блоку. Вони також прийшли до висновку, що безпека, яку забезпечує алгоритм, зростає з кількістю раундів шифрування, хоча це сповільнює швидкість алгоритму.

Sun [16] представив нещодавнє дослідження більшості методів захисту конфіденційності, запропонованих у літературі для хмарних систем. У роботі систематизовано різні методи, доступні в літературі для хмарних систем. Існує кілька методів, які підпадають під шифрування на основі атрибутів (ABE), шифрування на основі атрибутів ключової політики (KP-ABE), (KP-ABE), шифрування на основі атрибутів політики шифрованого тексту (CP-ABE) та багато інших методів. Автори приділили увагу поточним проблемам, пов’язаним з кількома запропонованими технологіями захисту для хмари. Перелічені основні проблеми: довіра, контроль доступу та шифрування. Таким чином, шифрування для хмарних систем залишається актуальною проблемою для дослідників.

Принципово новий підхід до симетричного шифрування забезпечується використанням СЗК, де замість єдиного монолітного ключа використовується розподілена система модулів. Це кардинально змінює парадигму криптоаналізу - замість атаки на один ключ зловмисник повинен одночасно компрометувати всі модулі системи, що комбінаторно збільшує складність зламу. Результати порівняльного аналізу симетричних криптоалгоритмів представлено в таблиці 1.12.

Таблиця 1.12

Порівняльна характеристика симетричних криптоалгоритмів

Параметр порівняння	AES-256	DES	3DES	Криптосистеми			
				СЗК	МДФ СЗК	СЗК зі зміною базисних чисел	КТЗ
				k – кількість модулів, l – їх розмірність			
Довжина ключа	256 біт	56 біт	168 біт	$k \times l$ біт	$k \times l$ біт	$2k \times l$ біт	$k \times l$ біт
Розмір блоку	128 біт	64 біт	64 біт	Змінний ($N < P$)	Змінний ($N < P$)	Змінний ($N < P$)	Підблоки < p_i
Криптостійкість	2^{255}	2^{55}	2^{112}	$\left(\left(\frac{2^{l+1}}{l}\right)^k l^2\right)$	$\left(\left(\frac{2^{l+1}}{l}\right)^k l^2\right)$	$O\left(\begin{matrix}log_2(k-1)) \\ \cdot k \cdot l^6\end{matrix}\right)$	$O\left(\left(\frac{2^{l+1}}{l}\right)^{k\ (l \cdot \log 2l)^2}\right)$
Паралелізація	Обмежена	Ні	Ні	Повна	Повна	Повна	Часткова
Швидкість шифрування	Висока	Середня	Низька	Висока	Найвища	Висока	Середня
Швидкість розшифрування	Висока	Середня	Низька	Низька	Висока	Низька	Висока
Математична основа	Підстановки та перестановки	Feistel мережа	Triple DES	КТЗ	МДФ + КТЗ	Модифікована КТЗ	КТЗ для підблоків
Стійкість до квантових атак	Вразливий	Вразливий	Вразливий	Підвищена	Підвищена	Найвища	Підвищена

Порівняльна таблиця симетричних криптографічних алгоритмів демонструє суттєві відмінності між класичними стандартами (AES-256, DES, 3DES) та сучасними розробками, заснованими на СЗК**,** включаючи модифікації із векторно-модульним методом, зміною базисних чисел та КТЗ для підблоків.

Класичні алгоритми, зокрема AES-256, залишаються загальновизнаними через високу ефективність та стандартизацію, проте їх вразливість до квантових атак є критичною проблемою з огляду на зростаючі обчислювальні можливості. DES і 3DES мають недостатню довжину ключа і низьку стійкість, що вже призвело до їхнього поступового виведення з експлуатації.

Алгоритми на основі СЗК демонструють вищий рівень безпеки**,** перш за все завдяки використанню паралельної обробки по модулях, що забезпечує високу або навіть найвищу швидкість шифрування. Особливо ефективними є

алгоритми з МДФ СЗК, які поєднують високу швидкість із підвищеною криптостійкістю. Варіанти із зміною базисних чисел забезпечують найвищу теоретичну стійкість, завдяки залученню більш складних обчислювальних структур.

Важливо зазначити, що паралелізація є ключовою перевагою сучасних СЗК-алгоритмів, що дозволяє знижувати затрати часу на шифрування та масштабувати їх для високопродуктивних систем. При цьому можливе деяке зниження швидкості розшифрування в окремих варіантах, проте воно компенсується зростанням загальної безпеки.

Отже, симетричні криптоалгоритми, побудовані на основі СЗК та її модифікацій, забезпечують високу продуктивність, криптостійкість і квантову стійкість**,** що робить їх перспективними для впровадження в сучасних та майбутніх криптографічних протоколах. Їхня архітектура дозволяє подолати обмеження класичних підходів і ефективно протидіяти як класичним, так і квантовим атакам.

1.2 Аналіз найбільш поширених асиметричних криптосистем

Стійкість асиметричних систем захисту інформаційних потоків ґрунтується на одому з наступних типів незворотних перетворень: факторизація, дискретне логарифмування, пошук коренів алгебраїчних рівнянь [39-49]. Однак збільшення розмірності вхідних параметрів призводить до втрати продуктивності сучасних систем захисту. Тому постає задача пошуку високопродуктивних криптоалгоритмів, стійких до різного роду атак [50].

При оцінці складності факторизації і дискретного логарифмування було показано, що ці операції мають приблизно однакову обчислювальну складність [51, 52]. Звідси випливає, що ключі одинакової довжини в асиметричних криптосистемах RSA, Ель-Гамаля, Рабіна будуть мати приблизно однакову стійкість [53-61].

1.2.1 Криптосистема Рабіна

Криптосистема Рабіна грунтується на складності знаходження квадратичного лишку [62]. Схема шифрування починається з генерування ключів. Для цього вибираються два випадкових багаторозрядних простих числа p і q, для яких виконується умова: $p \equiv q \equiv 3 \bmod 4$. Її виконання суттєво спрощує та прискорює пошук коренів за модулями p і q. Далі обчислюється значення $t=p \cdot q$. Число t виступає відкритим ключем, а p і q — закритим.

Шифрування відкритого повідомлення M відбувається за такою формулою:

$$C \equiv M^2 \bmod t. \tag{1.2}$$

При розшифруванні криптограми C вводяться додаткові допоміжні величини v і m:

$$v=C \bmod p; \qquad m=C \bmod q. \tag{1.3}$$

Для знаходження M необхідно знайти квадратичні лишки v, m відповідно за модулями p і q:

$$x^2 \equiv v(mod\, p), \qquad y^2 \equiv m(mod\, q), \tag{1.4}$$

В результаті утворюється чотири системи порівнянь (i=1…4):

$$\begin{cases} M_i \equiv \pm x(mod\, p); \\ M_i \equiv \pm y(mod\, q). \end{cases} \tag{1.5}$$

Одне з рішень (1.5), яке грунтується на використанні китайської теореми про залишки (КТЗ) [53, 63, 64] буде шуканим повідомленням M.

Аналіз літературних джерел дозволив виявити недоліки класичної криптосистеми Рабіна. Зокрема, у [65] авторам вдалося досягнути більшої

криптостійкості без збільшення обчислювальної складності шляхом заміни квадратичної конгруенції на кубічне рівняння.

В [66] зазначено переваги використання гаусівських цілих чисел для генерації відкритого ключа криптосистеми Рабіна. Це дозволило розробити відповідну арифметику для теореми Вільсона і КТЗ, а також для обчислення символів Лежандра і квадратичних залишків.

1.2.2 Криптографічний алгоритм Ель-Гамаля

Основними операціями асиметричних криптосистем є модулярне експоненціювання та модулярне множення багаторозрядних чисел [67-70]. На сьогоднішний день для забезпечення належного рівня захисту вхідні параметри (ключі, блок шифрування та модуль криптоперетворення) повинні бути не менші 2048 біт з перспективою зростання в найближчі роки до 4096 біт [71, 72]. Однак найбільш поширена в сучасних обчислювальних системах двійково-десяткова система числення має певні функціональні обмеження, що неминуче призведе до погіршення часових характеристик роботи алгоритмів.

Одним із шляхів підвищення швидкодії асиметричних криптоалгоритмів є застосування різних форм системи залишкових класів (СЗК) та векторно-модульних методів модулярного множення та експоненціювання [73]. Зокрема, в [74] розроблено трьохмодульний криптоалгоритм Рабіна з використанням звичайної цілочисельної та модифікованої досконалої форм СЗК [75-77], який має перевагу у стійкості перед класичною криптосистемою Рабіна за рахунок збільшення блоку відкритого тексту для шифрування. В [51] представлено реалізацію криптоалгоритму RSA на основі використання векторно-модульного методу модулярного експоненціювання. Це дало можливість замінити обчислювально складні арифметичні операції операцією додавання, що збільшує швидкодію роботи криптосистеми RSA.

Тому актуальною є задача застосування векторно-модульного методу модулярного експоненціювання та множення для реалізації криптоалгоритму

Ель-Гамаля, що дозволить підвищити його швидкодію та зменшити, відповідно, часову складність.

Для шифрування блоку відкритого тексту M в криптосистемі Ель-Гамаля вибирається велике просте число p і розглядаються всі операції в полі (або в мультиплікативній групі) по модулю числа p. Вибирається випадкове число $1<q<p$, яке є генератором мультиплікативної групи (елемент, який при модулярному експоненціюванні по всіх степенях утворює всі елементи групи). В даному випадку всі числа, які взаємно прості з p, будуть генераторами. Далі, вибравши показник степеня $2<x<p-1$, обчислюється число y:

$$y = q^x \bmod p. \tag{1.6}$$

Тоді відкритим ключем буде набір (y, q, p), а закритим – число x. Складність відновлення закритого ключа з відкритого пов’язана із задачею дискретного логарифма [78]. Зазначена задача є настільки ж складною, наскільки і факторизація. На даний час не існує ефективних поліноміальних алгоритмів для обчислення числа x, хоча існують субекспоненціальні алгоритми і алгоритми для квантового комп’ютера, які для певної розрядності вхідних параметрів здатні вирішувати цю задачу [79].

Для шифрування у схемі Ель – Гамаля вводиться допоміжне випадкове число $1<k<p-1$ і обчислюється два числа a і b, які є блоками зашифрованого тексту:

$$a = q^k \bmod p,\ b = y^k \cdot M \bmod p. \tag{1.7}$$

Отже, у схемі Ель – Гамаля розмір зашифрованого повідомлення завжди вдвічі більший, ніж розмір відкритого тексту.

Розшифровування відбувається згідно такого виразу:

$$M = b \cdot (a^x)^{-1} \bmod p. \quad (1.8)$$

Слід зазначити, що для практичних обчислень більше підходить наступна формула:

$$M = b(a^x)^{-1} \bmod p = ba^{p-1-x} \bmod p. \quad (1.9)$$

Так як в схему Ель-Гамаля вводиться випадкова величина, то вона є імовірнісною або її ще називають шифром багатозначної заміни. У них спостерігається більша стійкість до криптоаналізу в порівнянні зі схемами з певним процесом шифрування . Недоліком криптоалгоритму Ель-Гамаля є подвоєння довжини зашифрованого тексту в порівнянні з початковим.

1.2.3 Криптографічний алгоритм RSA

Криптосистема RSA (Rivest-Shamir-Adleman) одна з найвідоміших і широко використовуваних криптосистем з відкритим ключем, розроблена в 1977 році Рональдом Рівестом, Аді Шаміром і Леонардом Адлеманом [80]. Стійкість алгоритму базується на математичній складності факторизації великих чисел, що робить його одним із найбезпечніших методів шифрування.

RSA використовує пару ключів: відкритий ключ (public key) для шифрування і закритий ключ (private key) для розшифрування. Основний принцип полягає в тому, що знання відкритого ключа не дозволяє легко отримати закритий ключ. Приведемо основні етапи шифрування/розшифрування згідно криптоалгоритму RSA.

1. Генерування ключів

1.1 Вибір простих чисел: Обираються два великі випадкові прості числа p і q.

1.2 Обчислення модуля криптоперетворення: $n = p \times q$. Значення n використовується як модуль у всіх операціях.

1.3 Обчислення функції Ейлера: $\varphi(n) = (p-1)(q-1)$.

1.4 Вибір відкритого ключа, який використовується для шифрування: обирається число e, яке є взаємно простим із $\varphi(n)$, тобто НСД(e, $\varphi(n)$)=1.

1.5 Обчислення таємного ключа, який використовується для розшифрування: знаходиться d, яке задовольняє рівняння $d \times e \equiv 1 \pmod{\varphi(n)}$.

2.Шифрування

Для шифрування повідомлення M, яке подане у цифровій формі використовується формула:

$$C = M^e \bmod n, \tag{1.10}$$

де C — зашифроване повідомлення (шифротекст).

3. Розшифрування

Розшифрування виконується за допомогою закритого ключа за формулою:

$$M = C^d \bmod n, \tag{1.11}$$

де M — вихідне повідомлення (відкритий текст).

До основних переваг криптосистеми RSA можна віднести: високий рівень безпеки завдяки математичній складності факторизації великих чисел; RSA підтримує шифрування даних і створення цифрових підписів; використовується в багатьох протоколах, таких як SSL/TLS, PGP тощо; відкритий ключ можна передавати через канал зв’язку, що полегшує створення захищених з'єднань.

Однак, окрім переваєє ще певні недоліки криптосистеми RSA: шифрування та розшифрування займають значно більше часу порівняно з симетричними алгоритмами; для забезпечення високого рівня безпеки потрібні ключі розмірності не менше 2048, а краще 4096 біт; вразливість до квантових обчислень значно знижує ефективність RSA; неефективний для великого вхідного повідомлення, краще підходить для шифрування невеликих даних або ключів.

Слід зазначити, що криптосистема RSA широко використовується для створення цифрових підписів, що забезпечують автентичність і цілісність даних, шифрування симетричних ключів у протоколах, таких як HTTPS, в платіжних системах для забезпечення захищених транзакцій, для шифрування електронних листів в протоколах PGP.

RSA є основним стандартом у сучасній криптографії. Його надійність і універсальність зробили його ключовою частиною безпечної передачі даних у різних сферах. Незважаючи на можливі виклики з боку квантових обчислень, RSA залишається важливим інструментом у забезпеченні інформаційної безпеки.

1.2.4 Криптографія еліптичних кривих

Системи захисту інформаційних потоків з використанням математичного апарату еліптичних кривих (ЕК) запропоновані В. Міллером і Н. Кобліцем й вже більше 55 років інтенсивно досліджуються науковцями світу [81-90]. Особливий інтерес до них обумовлений такими перевагами, як швидкодія та порівняно невелика довжина ключа.

Вся сукупність відомих теоретичних результатів переконливо доводить, що ЕК дають унікальні можливості для побудови надійних і ефективних криптографічних алгоритмів. Основними арифметичними операціями, відносно яких точки ЕК утворюють абелеву групу, є додавання та експоненціювання (множення на скаляр). До істотних переваг таких алгебраїчних структур відносяться:

– відносна простота обчислення параметрів цих груп;

– відсутність ефективних алгоритмів розв’язання задачі дискретного логарифмування в цих групах і малоймовірність їх появи в майбутньому, що дозволяє використовувати ключі невеликої довжини з одночасною гарантією високої стійкості [91];

– наявність нескладних умов, дотримання яких зводить нанівець використання відомих специфічних методів криптоаналізу.

На сьогодні ЕК застосовують для реалізації різноманітних класів систем захисту інформації, зокрема їх можна використовувати для побудови асиметричних криптосистем, систем електронного цифрового підпису (ЕЦП), електронних платежів, побудови генераторів псевдовипадкових послідовностей тощо [92].

Використання ЕК дозволяє забезпечити суттєво менший час обчислень, необхідний для зашифрування інформації на основі операції експоненціювання точок на ЕК, в порівнянні з криптоалгоритмами, що використовують задачу дискретного логарифму, та криптосистеми RSA [73, 93], причому без зниження криптографічної стійкості [94]. Незважаючи на вагомі переваги, наявні і певні труднощі, які стримують практичне застосування ЕК в криптографічних системах захисту інформації. Вони зумовлені тим, що потребують вирішення таких актуальних класів задач [95, 96]:

- задання простого числа p – модуля перетворення груп точок ЕК;
- генерування параметрів ЕК;
- генерування базових точок на ЕК;
- знаходження порядку ЕК;
- вибір алгоритму експоненціювання точок ЕК [97].

Нехай задано просте число $p > 3$. Тоді еліптичною кривою E, що визначена над скінченим простим полем F_p, називається множина пар чисел (x, y), $x, y \in F_p$, що задовольняють наступну тотожність:

$$y^2 \equiv x^3 + ax + b \ (mod \ \ p), \tag{1.12}$$

де $a, b \in F_p$ і виконана умова $4a^3 + 27b^2 \neq 0 \ (\text{mod} \ p)$.

Інваріантом еліптичної кривої називається величина $J(E)$, що задовольняє наступну тотожність:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} \ (mod \ \ p) \ . \tag{1.13}$$

Коефіцієнти *a*, *b* еліптичної кривої *E*, по відомому інваріанту *J(E)*, визначаються наступним чином:

$$\begin{cases} a \equiv 3k \ (mod \ \ p), \\ b \equiv 2k \ (mod \ \ p), \text{ де } k \equiv \frac{J(E)}{1728-J(E)} \ (mod \ \ p), \quad J(E) \neq 0{,}1728 \end{cases} \quad (1.14)$$

Пари (*x*, *y*), що задовольняють тотожність 1.12, називаються точками еліптичної кривої *E*, значення *x* та *y* - координатами точки, наприклад *P*. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні *x*- та *y*-координати.

На множині описаних точок задають операцію додавання, що має просту геометричну інтерпретацію. Для отримання суми двох точок необхідно провести через них пряму. Ця пряма перетинає ЕК в додатковій третій точці. Третя точка і є результатом додавання перших двох. Наведемо математичні співвідношення, які описують дану процедуру (будемо позначати знаком "+") на еліптичній кривій Е. Спочатку визначимо нульову точку О рівностями:

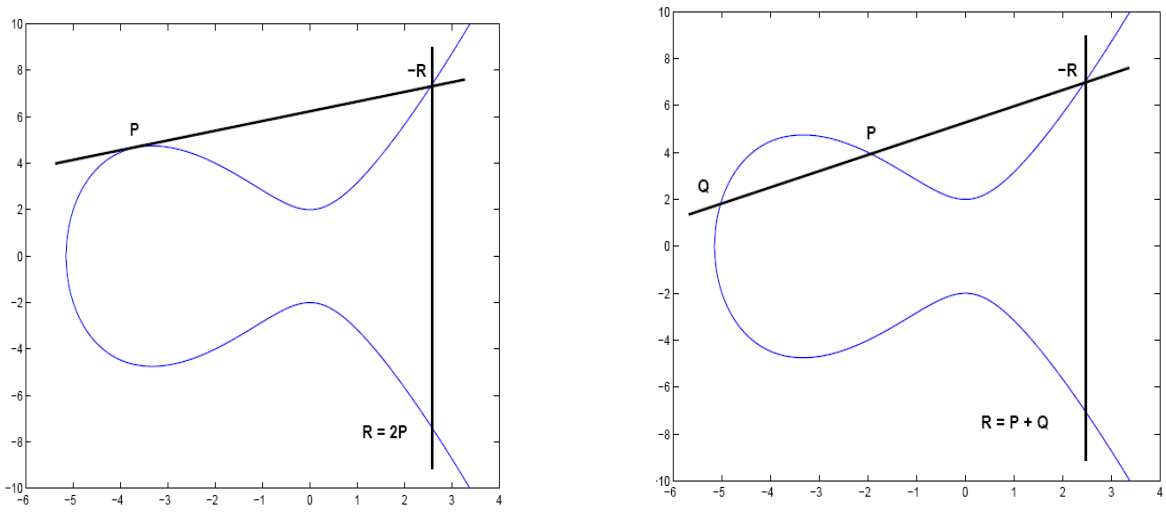
$$Q + O = O + Q = Q\,, \quad (1.15)$$

де Q – випадкова точка еліптичної кривої Е. Для випадкової точки $Q(x, y)$ еліптичної кривої Е, точка $P(x,-y)$ задовольняє рівності $P + Q = O$ і називається її запереченням.

В загальному випадку, додавання двох точок $Q_1(x_1, y_1)$ и $Q_2(x_2, y_2)$ еліптичної кривої Е визначається наступним чином:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}, \text{ де } \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & x_1 \neq x_2 \\ \dfrac{(3x_1^2 + a)}{2y_1}, & x_1 = x_2 \end{cases} \quad (1.16)$$

де x_3, y_3, відповідно, *x*- та *y*-координати суми – точки $Q_3(x_3, y_3)$ еліптичної кривої E. Відповідно, коли $x_1 == x_2$, другий випадок обчислення коефіцієнту λ, то операція додавання називається операцією дублювання очки на еліптичній кривій E. Графічне представлення додавання двох точок та знаходження кратних точок на ЕК зображено на рисунку 1.19.



а б

Рисунок 1.19 - Геометрична інтерпретація подвоєння точки *P* (а) і додавання точок *P* и *Q* (б)

Відносно введеної операції додавання множина всіх точок еліптичної кривої *E*, разом з нульовою точкою, утворюють скінченну адитивну групу порядку *m*, для якої виконується наступна нерівність:

$$p+1-2\sqrt{p} \le m \le p+1+2\sqrt{p} \qquad (1.17)$$

Точка *Q* називається точкою кратності *k*, або просто – кратною точкою еліптичної кривої *E*, якщо для деякої точки *P* виконується наступна рівність:

$$Q = \underbrace{P + \cdots + P}_{k} = kP \qquad (1.18)$$

Слід зауважити той факт, якої розмірності мають бути коефіцієнти базової ЕК, для того щоб задавався визначений (певний) рівень криптосистеми на такій кривій.

Проведений комплексний аналіз найбільш поширених асиметричних криптосистем, зокрема системи Рабіна, Ель-Гамаля, RSA та криптографії на основі математичного апарату еліптичних кривих, виявив їх критичну залежність від складних арифметичних операцій. Встановлено, що основними операціями в цих системах є модулярне множення, модулярне експоненціювання та пошук оберненого елемента за модулем, які характеризуються значною часовою складністю. Порівняльні характеристики асиметричних криптосистем наведено в таблиці 1.13.

Проведений аналіз свідчить про поступову еволюцію асиметричних криптосистем від класичних моделей (RSA, Рабіна, Ель-Гамаля, ECC) до вдосконалених конструкцій, заснованих на системі залишкових класів (СЗК) та гібридних архітектурах.

Класичні алгоритми характеризуються високим рівнем математичної обґрунтованості та широким впровадженням у стандартах безпеки, проте мають обмежену стійкість до квантових атак через залежність від задач факторизації або дискретного логарифмування.

Таблиця 1.13

Порівняльна характеристика асиметричних криптосистем

Криптосистема	Математична основа	Розмір ключа (біт)	Стійкість до квантових атак	Часова складність, k- кількість модулів, l – їх розрядність	Переваги
1	2	3	4	5	6
RSA	Факторизація великих чисел	2048–4096	Низька	$O(l^3)$	Стандартизований, надійний
Рабіна	Квадратичні лишки	1024–2048	Низька	$O(l^3)$	Простота реалізації
Ель-Гамаля	Дискретний логарифм	2048–4096	Низька	$O(l^2(\log\log l)^2)$	Семантична безпека

Продовження таблиці 1.13

1	2	3	4	5	6
ECC	Еліптичні криві	256–512	Низька	O(l^3)	Короткі ключі
Асиметричний у СЗК	СЗК + гібридна архітектура	k×l + k×$\log_2$p	Висока	$O\left(k^2 \cdot l^{k+7}\right)$	Подвійний захист, паралелізація, протидія криптоаналізу
Удосконалений Рабіна (СЗК)	Квадратичні лишки + СЗК	1024–2048	Підвищена	$O(l^2\left(\frac{k^2+k}{2}\right) + \frac{l}{2} log\ l)$	Паралелізація, зниження складності
Удосконалений Ель-Гамаля (СЗК)	Дискретний логарифм + СЗК + векторно-модульний метод	2048–4096	Підвищена	$O\left(l^2 + \frac{3}{2} llogl\right)$	8-кратне прискорення при l=256
Поліноміальний алгоритм Рабіна	Квадратичні лишки в Z[x]	Змінний	Висока	$O\left(5l\ log\ l + l^{\frac{3}{2}} + l\right)$	Квантова стійкість
Двомодульний Рабіна (СЗК)	Квадратичні лишки + 2 модулі	1024	Підвищена	$O(l^2\left(\frac{k^2+k}{2}\right) + \frac{l}{2} log\ l)$	Простота, швидкість
Трьохмодульний Рабіна (СЗК)	Квадратичні лишки + 3 модулі	1536	Підвищена	O(n^2/3)	Оптимальний компроміс

Зокрема, алгоритми RSA, Рабіна та Ель-Гамаля є вразливими до квантових обчислень, що обґрунтовує потребу в переході до постквантових підходів.

1.3 Система залишкових класів в алгоритмах шифрування

У 1955 р. чеський інженер М. Валах першим висунув ідею, яку активно підтримував чеський математик А. Свобода, використовувати систему залишкових класів для операцій над числами ЕОМ [76, 77, 98].

В обчислювальній практиці це була видатна ідея з використанням відомої китайської теореми про залишки. Ця ідея привернула увагу великої групи вчених. Виник новий науковий напрям - модульна арифметика. Роботи М.Валаха та А.Слободи зацікавили американців і через кілька років вони переїхали до США.

Одним із напрямків розвитку модульної арифметики є науково-дослідна робота наукового керівника проекту Р.Г.Біяшева зі створення, аналізу та використання непозиційних поліноміальних записів чисел [99].

Він говорить, що алгебра поліномів над полем за модулем полінома, незвідного над цим полем, є полем, аналогом китайської теореми про залишки (КТЗ) для поліномів, системою, і доведено відновлення полінома з його залишків.

У [100, 101] запропоновано модель алгоритму шифрування, розробленого на основі нотацій чисел непозиційного полінома. Розглянуто можливість модифікації розробленої моделі з використанням мережі Фейстеля та режимів шифрування. Запропонована модель криптографічного алгоритму значно підвищить статистичні характеристики отриманих шифртекстів.

При захисті інформації при створенні системи формування та розподілу ключів в якості критерію шифрування та криптостійкості цифрових систем використовується криптографічна стійкість алгоритму шифрування, а не довжина ключа. Ця система заснована на використанні арифметики непозиційних систем числення, тобто на використанні системи розрахунку залишкових класів [102-104].

У класичній системі залишкових класів за основу системи числення беруться взаємно прості числа, тоді будь-яке число представляється у вигляді залишку від ділення на систему. У порівнянні з класичною системою запропоновані криптографічні процедури розглядаються в СЗК, де за основу беруться прості числа [105]. В алгоритмі шифрування за повний ключ береться послідовність ключів і прості числа, а за основу системи вибираються всі прості числа, що не перевищують довжину зашифрованого повідомлення з обраних баз у порядку розподілу [101, 106, 107].

Якщо задано набір цілих натуральних чисел $p_1, p_2,..., p_l$, які далі називають базисом системи [63], то під системою числення в залишкових класах розуміється система, в якій представлено натуральне число N як набір залишків

за вибраними ознаками $N = (\alpha_1, \alpha_2, \dots, \alpha_l)$, а формування цифр α_i здійснюється наступним згідно формули:

$$\alpha_i = N - \left[\frac{N}{p_i}\right] \cdot p_i, \tag{1.19}$$

тобто цифра i-го порядку α_i числа N є найменшим позитивним залишком від ділення N на p_i і $\alpha_i < p_i$. Тут і далі [*] позначає цілу частину числа.

На відміну від позиційного запису числа, формування розряду кожного розряду в цьому випадку проводиться незалежно один від одного [108].

Згідно з китайською теоремою про залишки, представлення числа N у вигляді послідовності цифр $\alpha_1, \alpha_2, \dots, \alpha_l$ буде унікальним, якщо числа p_i попарно прості.

Діапазон об’ємів представлених чисел у цьому випадку буде рівний:

$$P = p_1 \cdot p_2 \cdot \dots \cdot p_n$$

Тут, подібно до позиційного запису чисел, діапазон представлених чисел зростає як добуток основ, а розрядність чисел N зростає як сума цифр тих самих основ.

Таким чином, у класичній системі залишкових класів [109] попарно прості числа вибираються як система основ, і будь-яке число в ній однозначно представляється своїми залишками (відрахуваннями) від ділення на цю систему основ.

Приклад 1. Нехай основами СЗК є числа p_1=3, p_2=5, p_3=7. Діапазон СЗК визначається як: $P = p_1 \cdot p_2 \cdot p_3 = 3 \cdot 5 \cdot 7 = 105$.

Представимо числа A = 17 і B = 63 в системі залишкових класів для виділених базисів (2): $A_{p_1p_2p_3}$=$17_{3\,5\,7}$=(2,2.3), $B_{p_1p_2p_3}$=$63_{3\,5\,7}$=(0,3,0).

Розглянемо правила виконання операцій додавання і множення в системі залишкових класів, якщо і числа, і результат операції знаходяться в діапазоні [0, P).

Припустимо, що числа A і B представлені відрахуваннями α_i і β_i за основами p_i відповідно:

$$A = (\alpha_1, \alpha_2, \dots, \alpha_n),\ B = (\beta_1, \beta_2, \dots, \beta_n).$$

Припустимо також, що результати додавання A+B і множення AB представлені відповідно залишками [110]:

$$A + B = (\gamma_1, \gamma_2, \dots, \gamma_l),\ A \cdot B = (\delta_1, \delta_2, \dots, \delta_l).$$

Тут мають місце такі співвідношення:

$$A < P, B < P, A + B < P, A \cdot B < P.$$

Доведено, що результат додавання і множення буде відповідно:

$$\gamma_i = \alpha_i + \beta_i - \left[\frac{\alpha_i + \beta_i}{p_i}\right] \cdot p_i, \tag{1.20}$$

$$\delta_i = \alpha_i \cdot \beta_i - \left[\frac{\alpha_i \cdot \beta_i}{p_i}\right] \cdot p_i. \tag{1.21}$$

тобто γ_i порівнюється з $\alpha_i + \beta_i$ за модулем p_i, але δ_i порівнюється з $\alpha_i \cdot \beta_i$ того самого модуля p_i:

$$\gamma_i = (\alpha_i + \beta_i)\, mod\, p_i, \tag{1.22}$$

$$\delta_i = (\alpha_i \cdot \beta_i)\, mod\, p_i. \tag{1.23}$$

Примітка. У модульній арифметиці або в теорії конгруенцій цілі числа a і b називають порівнянними за модулем t, якщо їх залишки при діленні t збігаються, і це позначається як $a = b(mod\, t)$. Число t називається модулем.

Для позначення залишку використовується запис без дужок $b = a\, mod\, t$, що означає, що $a = kt + b$, де b — залишок від ділення a на t.

Приклад 2. Для прикладу 1 отримуємо. Відповідно до формули (1.22) A+B=(2,0,3).

Перевірка: сума чисел A = 17 і B = 63 дорівнює 80, якщо записати її в непозиційній формі (тобто в системі залишкових класів) для підстав прикладу 1, то вона матиме вигляд 80 = (2,0,3).

Приклад 3. Знайдемо добуток чисел A = 17 і B = 6. У системі залишкових класів (для прикладу 1) вони будуть представлені у вигляді: A=17=(2,2.3), B=6=(0,1,6).

Відповідно до формули (1.5) AB = (0,2,4) і дорівнює 102.

1.3.1 Алгоритм шифрування на основі поліномів

Нетрадиційний алгоритм шифрування електронного повідомлення заданої довжини N включає два етапи:

- вибір системи поліноміальних основ (формування NPNS) і порядку їх розташування;

- генерація ключової (псевдовипадкової) послідовності.

Ці два етапи описують вибір одного чи другого варіанта базової системи. Суть їх полягає в наступному.

Етап 1. Нехай $p_1, p_2, \dots, p_s$ – незвідні поліноми з двійковими коефіцієнтами, які використовуються як робочі бази [111, 112].

Поліном $P = p_1 p_2 \dots p_s$ є основним робочим діапазоном. У цій системі будь-яке окреме представлення у вигляді його відрахувань за обочими основами (або залишками від ділення на) $p_1, p_2, \dots, p_s$ відповідно.

Тоді повідомлення довжиною N біт можна інтерпретувати як послідовність залишків $\alpha_1, \alpha_2, \dots, \alpha_s$ від поділу полінома $F(x)$ на робочі основи $p_1, p_2, \dots, p_s$ відповідно:

$$F(x) = \big(\alpha_1(x), \alpha_2(x), \dots, \alpha_s(x)\big). \qquad (1.24)$$

У виразі (1.24) залишки $F_i(x) = \alpha_i(mod\, p_i), i = \overline{1,s}$ вибираються таким чином, що перші l_1 бітів повідомлення відображаються на двійкові залишкові

коефіцієнти α_1, наступні біти l_2 є залишкові двійкові коефіцієнти α_2 і так далі, двійкові коефіцієнти α_s залишку відображаються на останні двійкові біти l_s.

Подання (або реконструкція) в позиційній формі полінома F виконується його непозиційною формою (1.24). У разі зберігання, передачі та обробки інформації вона здійснюється за такою формулою:

$$F = \sum_{i=1}^{s} \alpha_i B_i, \; B_i = \frac{\prod_{i=1}^{s} p_i}{p_i} \cdot M_i \equiv 1 (mod\, p_i), \tag{1.25}$$

де $i = \overline{1,s}$ і значення поліномів M_i також вибираються для порівняння, вказані у формулі.

Для збереженої та переданої інформації відновлення відбувається за виразом [113]:

$$F = \sum_{i=1}^{s} \alpha_i P_i, \text{ де } P_i = \frac{P}{p_i}, i = \overline{1,s}. \tag{1.26}$$

Етап 2. Для шифрування використовується ключова послідовність довжиною N біт, яка також інтерпретується як послідовність залишків, але від ділення деяких інших $G(x)$ за тими ж робочими основами системи:

$$G(x) = \big(\beta_1(x), \beta_2(x), \dots, \beta_s(x)\big), \tag{1.27}$$

де $G_i(x) = \beta_i (mod\, p_i), i = \overline{1,s}$.

Відповідно до числового запису операції у функціях $F(x)$, $G(x)$, $H(x)$ виконуються паралельно відносно тих, що обрані як робочі бази [114].

Комп'ютерна реалізація розробленого алгоритму шифрування дає різні криптографічні моделі.

У цій моделі шифрування криптограма електронного повідомлення $H(x) = \big(\omega_1(x), \omega_2(x), \dots, \omega_s(x)\big)$ отримується в результаті множення поліномів (1.24) і (1.27) відповідно до властивостей порівнянь.

Потім елементи послідовності залишків $\omega_1(x), \omega_2(x), \dots, \omega_s(x)$ – найменші залишки від розбиття добутків $\alpha_i(x) \cdot \beta_i(x)$ на відповідні підстави $p_i(x)$:

$$\alpha_i(x) \cdot \beta_i(x) \equiv \omega_i(x)\big(mod\, p_i\,(x)\big). \tag{1.28}$$

У загальному вигляді криптограма $H(x)$ буде виглядати так: $\omega_1(x)$ залишкові коефіцієнти пов'язані з першим бітом l_1 криптограми $H\,(x)$. Наступні біти l_2 криптограми ставляться у відповідно l_2 коефіцієнти залишку $\omega_2(x)$ і так далі. Коефіцієнти останнього залишку $\omega_s(x)$ ставляться у відповідність останнім криптограмам l_S.

При розшифруванні криптограми $H(x)$ з відомого ключа $G(x)$ для кожного значення $\beta_i(x)$ розраховується обернений поліном, як випливає з (1.28), з умови виконання порівняння:

$$\beta_i(x) \cdot {\beta_i}^{-1}(x) \equiv 1\big(mod\, p_i\,(x)\big),\ i = \overline{1,s}. \tag{1.29}$$

В результаті проведених розрахунків отримується обернений многочлен $G^{-1}(x) = \left({\beta_1}^{-1}(x), {\beta_2}^{-1}(x), \dots, {\beta_s}^{-1}(x)\right)$, до многочлена $G(x)$. Тоді вихідне повідомлення згідно з (1.28) і (1.29) відновлюється за допомогою дедукцій шляхом наступних порівнянь:

$$\alpha_i(x) \equiv {\beta_i}^{-1}(x)\omega_i(x)\big(mod\, p_i\,(x)\big). \tag{1.30}$$

Таким чином, у моделі алгоритму шифрування електронного повідомлення заданої довжини N біт у NPNS повним ключем є обрана поліноміальна базова система $p_1, p_2, \dots, p_s$ отримана шляхом генерації ключа $G^{-1}(x) = \left({\beta_1}^{-1}(x), {\beta_2}^{-1}(x), \dots, {\beta_s}^{-1}(x)\right)$ псевдовипадкової послідовності до нього, який обчислюється згідно з виразом (1.30).

Розглянемо приклади переведення з позиційної системи в систему залишкових класів і назад:

1. Задаємо робочі основи $\rho_1, \rho_2, \dots, \rho_l$

2. Обчислюємо $p = \rho_1 \cdot \rho_2 \cdot \dots \cdot \rho_l$

3. Вибираємо $A \in [0, \rho)$ і $B \in [0, \rho)$.

4. Переводимо в систему залишкових класів A і B

$A = A(mod\, \rho_1) = (\alpha_1, \alpha_2, \dots, \alpha_l)$ і $B = B(mod\, \rho_1) = (\beta_1, \beta_2, \dots, \beta_l)$

$A + B = A + B(mod\, \rho_1) = (j_1, j_2, \dots, j_l)$.

5. Використовуючи $\delta = \frac{\rho}{\rho_i}(mod\, \rho_i), i = \overline{1, l}$ визначається δ_i.

6. Вибираємо m_i так, щоб виконувалося порівняння $\frac{m_i * \rho}{\rho_i}(mod\, \rho_i) \equiv 1(mod\, \rho_i), i = \overline{1, l}$.

7. $B_i = \frac{m_i * \rho}{\rho_i}(mod\, \rho_i), i = \overline{1, l}$.

8. Повертається ρ_i шифрування до $A + B = \sum_{i=1}^{l} B_i j_i\, (mod \rho)$.

Хоча СЗК не позбавлена недоліків (труднощі при виконанні операцій ділення, порівняння, визначення переповнення розрядної сітки [115, 116], її успішно можна застосовувати для додавання, віднімання, піднесення до степеня [117-125], множення цілих багаторозрядних чисел [126, 127], корекції даних в процесі передачі по каналу зв'язку [128, 129], що є важливо для асиметричної криптографії [130-133].

До переваг СЗК можна віднести [134-138]:

1) можливість виконання операцій над числами, які менші за вибрані модулі [139];

2) розпаралелення процесу обчислень, що є найбільш перспективним шляхом підвищення швидкодії обчислювальних систем [140-142];

3) відсутність міжрозрядних переносів.

Такі властивості СЗК дозволяють істотно зменшити споживання енергії у відповідних цифрових пристроях [143]. Крім криптографії [144, 145], переваги СЗК можуть бути використані для обробки цифрових сигналів [146-148] та

зображень [149, 150], для хмарних обчислень [151, 152], для Інтернет-речей [153], для завадостійкого кодування [154-156] тощо.

Найбільш принциповим недоліком СЗК, який загальмував її розвиток, є складність при переведенні чисел із СЗК в десяткову або двійкову систему числення [157, 158]. Слід відмітити, що в переважній більшості робіт [159, 160] використовуються модулі, які є Ферма- або Мерсено-подібними числами та степенями двійки. Відомі підходи відновлення десяткового числа, що ґрунтуються на основі китайської теореми про залишки (КТЗ) або алгоритму Гарнера, передбачають пошук оберненого елемента за модулем та множення на нього, що може привести до переповнення розрядної сітки [161]. Уникнути цих недоліків дозволяє досконала [162-164] та модифікована досконала форми (МДФ) СЗК [165, 166]. В таблиці 1.14 представлені характеристики різних форм СЗК.

Таблиця 1.14

Характеристики форм СЗК

Форма СЗК	Відновлення десяткового числа	m_i	Пошук оберне-ного елемента	Множен-ня на m_i	Перевищен-ня модуля P
Звичайна	$A = \left(\sum_{i=1}^{l} b_i P_i m_i\right) mod\ P$	$P_i^{-1}\ mod\ p_i$	+	+	Значне
ДФ	$A = \left(\sum_{i=1}^{l} b_i P_i\right) mod\ P$	1	-	-	Значне
МДФ	$A = \left(\sum_{i=1}^{l} b_i P_i m_i\right) mod\ P$	±1	-	-	Незначне

Крім того, перспективним кроком у розвитку криптографії є розробка методів шифрування у СЗК, зокрема у її МДФ. Зокрема, в [167] запропоновано ієрархічний підхід для реалізації асиметричних криптосистем та КТЗ в

асиметричній криптографії. В [168] представлено сучасні методи використання паралелізму СЗК для генерації ключів в асиметричних криптосистемах.

Незважаючи на існуючі ефективні алгоритми шифрування інформаційних потоків у СЗК [169, 170], на основі яких розроблено різні програмно-апаратні засоби, які дають змогу забезпечити захист від помилок в процесі передавання інформації і певний рівень захисту інформації.

Отже, основним завдання є розробка більш ефективних криптоалгоритмів з використанням різних форм СЗК.

1.4 Перспективи застосування поліноміальних систем числення в системах захисту інформації

На сьогоднішній день вимоги до симетричних методів шифрування стали вищими з підвищенням швидкодії та стійкості, які є результатом стрімкого розвитку обчислювальних засобів. Альтернативою сучасних числових криптоалгоритмів є поліноміальні [114]. У кільці Z[x], як і у будь-якому іншому кільці поліномів, виконуються базові операції додавання, множення та ділення з остачею, пошук оберненого поліному за модулем, відновлення полінома за його залишками тощо [113, 133, 171-179]. Основна ідея застосування поліномів у криптографії полягає в тому, що їх можна використовувати як відкритий текст і ключі для шифрування та розшифрування повідомлень, побудови електронних цифрових підписів та інших криптографічних протоколів [180].

Використання системи залишкових класів (СЗК) при реалізації криптографічних алгоритмів захисту інформації на основі поліноміальної арифметики у кільці Z[x] [181], по аналогії з цілочисельною СЗК, приводить до розпаралелення процесу обчислень та зменшення об’єму даних, які необхідно обробляти в ході криптографічних операцій. Це, в свою чергу, забезпечує зменшення часових характеристик реалізації та підвищення ефективності методу шифрування. Базовими операціями у ПСЗК є пошуку мультиплікативного оберненого полінома в кільці $Z[x]$ та відновлення полінома за його залишками.

Одним з найбільш поширених методів пошуку мультиплікативного оберненого полінома в кільці $Z[x]$ є підхід, що грунтується на розширеному алгоритмі Евкліда [182]. Слід зазначити, що його застосування потребує великої кількості арифметичних операцій над поліномами: ділень, пошуку залишків [183], експоненціювань [73], множень і підстановок [184, 185]. Однак він володіє найменшою часовою складністю порівняно з іншими відомими методами [186, 187].

При застосуванні алгоритму Евкліда обчислення оберненого полінома зводиться до вирішення двох задач: знаходження найбільшого спільного дільника поліномів [188] та розв'язання діофантових рінянь [189] на основі розширеного алгоритму Евкліда.

Нехай $f(x)$ і $g(x)$ - поліноми в кільці Z[x] і $deg(f(x)) > deg(g(x))$, де функція deg позначає степінь полінома. Згідно основної теореми алгебри для поліномів [190], існує пара $q(x)$ і $r(x)$ з кільця Z[x], для якої $f(x) = q(x)g(x) + r(x)$, $0 < deg\, r\,(x) < deg\, g\,(x)$. Тоді при умові, що $g(x)$ не ділиться на $r(x)$, для $g(x)$ виконується рівність:

$$g(x) = r(x)q_1(x) + r_1(x),\ 0 < deg\, r_1\,(x) < deg\, r\,(x).$$

Далі якщо $r(x)$ не ділиться на $r_1(x)$, то:

$$r(x) = r_1(x)q_1(x) + r_2(x),\ 0 < deg\, r_2\,(x) < deg\, r_2\,(x)$$ і т.д.

Цей процес є скінченний, тобто існує таке n, при якому $r_{n-1}(x)$ буде ділитися на $r_n(x)$.

В результаті отримується система рівнянь, на основі якої здійснюється пошук найбільшого спільного дільника (НСД) двох поліномів:

$$\begin{gathered} f(x) = q(x)g(x) + r(x),\ \ 0 < deg\, r\,(x) < deg\, g\,(x) \\ g(x) = r(x)q_1(x) + r_1(x),\ 0 < deg\, r_1\,(x) < deg\, r\,(x) \\ r(x) = r_1(x)q_2(x) + r_2(x),\ 0 < deg\, r_2\,(x) < deg\, r_1\,(x) \\ \ldots\ldots\ldots\ldots \end{gathered} \tag{1.31}$$

$$r_{n-2}(x) = r_{n-1}(x)q_{n-1}(x) + r_n(x),\ 0 < deg\, r_n(x) < deg\, r_{n-1}(x)$$
$$r_{n-1}(x) = r_n(x)q_n(x).$$

Послідовність (1.11) визначає хід алгоритму Евкліда. Для нього справедливе співвідношення щодо НСД поліномів:

$$НСД\big(f(x), g(x)\big) = НСД\big(g(x), r(x)\big) = НСД\big(r(x), r_1(x)\big) = \cdots$$
$$= НСД\big(r_{n-1}(x), r_n(x)\big) = r_n(x).$$

Обчислення оберненого полінома в кільці $Z[x]$ зводиться до розв'язування діофантового рівняння, оскільки для двох взаємнопростих поліномів $f(x)$ і $g(x)$ існують такі поліноми $l(x)$, $s(x) \in Z[x]$, для яких виконується рівність $f(x) \cdot l(x) + g(x) \cdot s(x) = \text{НСД}\big(f(x), g(x)\big) = w$. Якщо $f(x)$ і $g(x)$ не є взаємнопрості, то, згідно визначення кільця, оберненого полінома не існує.

Для спрощення процедури пошуку оберненого полінома формули (1.31) необхідно переписати наступним чином:

$$r(x) = f(x) - q(x)g(x)$$
$$r_1(x) = g(x) - r(x)q_1(x) = g(x) - \big(f(x) - q(x)g(x)\big)q_1(x) = g(x)\big(1 +$$
$$q(x)q_1(x)\big) - f(x)q_1(x)$$
$$\dots$$
$$\text{НСД}\big(f(x), g(x)\big) = r_n(x) = f(x)l(x) + g(x)s(x),$$

Це представлення $\text{НСД}\big(f(x), g(x)\big)$ називається співвідношенням Безу для поліномів, а поліноми $l(x)$ і $s(x)$ – поліномами Безу. Тоді $f(x)^{-1}\ mod\, g\,(x) = l(x)\, mod\, g\,(x)$.

У роботі [191] представлені математичні основи поліномів перестановки, степені яких не перевищували 6, та їх обернених в кінцевих полях. Зазначено,

що результати проведених досліджень знаходять застосування в криптографії, теорії кодування та теорії комбінаторного проектування.

Стаття [192] присвячена пошуку обернених елементів в кільці усічених поліномів N-го степеня (N^{th} Degree Truncated Polynomial Ring). Запропонований метод грунтується на обчисленні оберненого поліному за допомогою адаптаційного зворотного алгоритму, визначеного в полі поліномів із двійковими та трійковими коефіцієнтами. Також відмічено, що даний алгоритм можна розширити для поліномів із коефіцієнтами різних полів, в тому числі і в $Z(x)$. Проведено дослідження продуктивності алгоритму у порівнянні з інверсним алгоритмом Чжао та Су.

Для побудови оптимальних вузлів інтерполяції на дискретних інтервалах в [193] використано метод зображення оберненого полінома. Показано, що цей метод надзвичайно ефективний для Т-поліномів. Проведені дослідження в [194] вказують, що для оберненого полінома існує обмеженння довжини, яка залежить від величини прийнятної помилки. Крім того, представлено п’ять прикладів застосування інверсії полінома для вирішення задач фізики та математики. В [195] представлено алгоритм, що генерує обернені елементи над скінченими полями GF (3^m). Обчислення базуються на множеннях, піднесеннях до квадрату та кубу.

1.4.1 Методи відновлення полінома за його залишками

Теоретичною основою при відновленні полінома за його залишками у відповідному кільці є алгебра і теорія чисел [196], зокрема китайська теорема про залишки (КТЗ) в поліноміальній формі [197].

Будь-який поліном $N(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ можна представити у вигляді залишків $b_i(x) = c_l x^l + c_{l-1}x^{l-1} + \cdots c_1 x + c_0$від ділення на незвідні многочлени $p_i(x) = s_z x^z + s_{z-1}x^{z-1} + \cdots + s_1 x + s_0$, які називаються поліноміальними модулями:

$$b_i(x) = N(x)\, mod\, p_i\,(x). \qquad (1.32)$$

де $deg\ b_i = deg\ N - deg\ p_i$.

При цьому необхідною і достатньою умовою є нерівність $N(x) < P(x) = \prod_{i=1}^{k} p_i\ (x)$, де k - кількість незвідних поліномів. Тоді вихідний поліном можна однозначно відновити на основі КТЗ:

$$N(x) = \left(\sum_{i=1}^{k} m_i(x) P_i(x) b_i\ (x)\right) mod\ \ P(x), \tag{1.33}$$

де $P_i(x) = \frac{P(x)}{p_i(x)}$, $m_i(x) = P_i^{-1}(x)\ mod\ p_i\ (x)$.

Ще одним методом відновлення полінома у відповідному кільці за його залишками є алгоритм Гарнера [198], основою якого є співвідношення:

$$N(x) = n_0(x) + n_1(x)p_1(x) + n_2(x)p_1(x)p_2(x) + \ldots + n_{n-1}(x)p_1(x)p_2(x)\ldots p_{i-1}(x)$$

де $0 \le n_i(x) < p_{i+1}(x)$, i=0, 1, …, k-1.

$$n_i(x) = \frac{b_{i+1}(x) - (n_0(x) + n_1(x)p_1(x) + \ldots + n_{i-1}(x)p_1(x)p_2(x)\ldots p_{i-1}(x))}{p_1(x)p_2(x)\ldots p_i(x)}\ mod\ p_{i+1}\ (x). \tag{1.34}$$

В цьому випадку поліноми $n_i(x)$ обчислюються послідовно один за одним на основі рекурентної формули (1.34). Крім того, і алгоритм Гарнера, і КТЗ придатні для аналогічних операцій у цілочисельній арифметиці [199].

Основними недоліками розглянутих вище методів відновлення полінома за його залишками є їх строго послідовна структура, що приводить до унеможливлення процесу розпаралелення обчислень, виконання операцій над поліномами вищих порядків (зокрема, обчислення залишку за модулем $P(x)$) необхідності пошуку мультиплікативного оберненого елемента за модулем у кільці поліномів [200].

Для пошуку останнього найбільш поширеними є такі методи [74]: перебір всіх можливих варіантів, за допомогою розширеного алгоритму Евкліда, на основі функції Ейлера. Дані підходи характеризуються значною обчислювальною складністю.

1.4.2 Застосування поліноміальних алгоритмів в системах захисту інформації

Дослідженнями схем шифрування на основі системи Рабіна більш активно розпочали займатися науковці на початку 21 ст. [201, 202]. Зокрема, в [203] відмічається, що така схема шифрування з відкритим ключем є однією з найбільш поширених та безпечних.

Тому у циклі праць [203-205] автори модифікували схему шифрування ElGamal на основі відкритого ключа із множини цілих чисел до двох основних ідеальних множин $Z[i]$ та $F[x]$, розширивши відповідну арифметику. Крім того, було показано, як модифіковані обчислювальні методи можна використовувати для розширення криптосистеми Рабіна на область $F[x]/\langle f(x)\rangle$, де $f(x)$ - поліном в $F[x]$, з кільця поліномів скінченного поля F. Автори довели, що розширений алгоритм вимагає незначного збільшення обчислювальних ресурсів у порівнянні з класичним, але забезпечує суттєво вищий рівень безпеки.

Сутнісна цінність розробок полягає ще у тому, що, використовуючи арифметику в кільцях поліномів над скінченними полями $Z_p[x]/\langle h(x)\rangle$, де p - просте число, $h(x)$ - незвідний поліном у $Z_p[x]$ степеня n, були внесені зміни до класичної схеми шифрування Рабіна та детально описані обчислювальні процедури в новій системі, вказані її переваги, основні з яких є такі:

– генерування непарного простого p як у класичному, так і в модифікованому методах вимагає однакових часових затрат;

– повна система залишків Z_n має елементи p і q, тоді як у $Z_p[x]/\langle h(x)\rangle$ є елементи $p \cdot n$, через що модифікований метод забезпечує розширення діапазону вибраних повідомлень й ускладнює процес шифрування;

– обчислення, задіяні в модифікованому методі, не вимагають обчислювальних процедур, які значно відрізняються від тих, що використовуються в класичному.

До вагомих недоліків запропонованої системи автори відносять:

– завдання вибору правильного повідомлення серед чотирьох можливих;

– специфіку відтворення останніх кількох цифр повідомлення та вибір одного квадратного кореня з чотирьох;

– визначення двох незвідних поліномів, що ускладнює арифметику та вимагає додаткових затрат часу для обчислень.

Однак додатковий захист, що забезпечується вказаним методом, виправдовує прикладені зусилля.

На потребу розвитку й вдосконалення системи шифрування вказано у роботі [206], де детально досліджено криптосистему з відкритим ключем RSA. Дослідниця вважає, що на сьогоднішній день ця криптосистема продовжує бути досить безпечною завдяки тому, що стійкість ґрунтується на задачі цілочисельного розкладання на множники з субекспоненційною складністю. Але одночасно зазначено, що при відкритті ефективного алгоритму факторизації цілих чисел криптосистема RSA перестане бути безпечною. Вагомим результатом дослідження вважається запропонований у роботі [206] додаток для інтерактивного комп'ютера, який дозволяє формувати поліноми для RSA, здійснювати тестування шифрування та розшифрування, а також має інструкцію й приклади щодо користування.

В [207-209] для створення асиметричної криптосистеми було обрано модифіковані поліноми Чебишева, які утворюють напівгрупу відносно операції композиції. Наявні переваги розроблених підходів створюють передумови для їх

застосування на практиці, однак отримані результати щодо безпеки представлених криптосистем показують, що вони пов'язані з проблемою дискретних логарифмів.

У монографії [210] репрезентовано результати, пов'язані з перестановочними поліномами над скінченними кільцями та скінченними полями та їх застосуванням у багатовимірній криптографії з відкритими ключами. Автор критично оцінив криптосистеми, які базуються на вирішенні нелінійних рівнянь над скінченними полями та запропонував розробку й аналіз деяких ефективних багатовимірних криптосистем із відкритими ключами.

Більшість сучасних симетричних криптоалгоритмів є блочними і ця особливість обмежує функціональні можливості їх реалізації. Зокрема, розмір ключа повинен бути рівний або більший від розміру блоку, що для великого повідомлення призводить до кількаразового використання алгоритму шифрування. Така процедура зменшує стійкість алгоритму, збільшує часову складність і, в той же час, ускладнює реалізацію. Дослідження симетричних алгоритмів шифрування у десятковій системі числення проводяться багатьма авторами. Наприклад, у роботі [211] пропонується 8-бітовий алгоритм шифрування, який грунтується на ідеях відомих симетричних криптоалгоритмів. При генеруванні псевдовипадкових ключів автори застосовують дивергентні поліноми із змінними коефіцієнтами, побітові операції з даними та двохпарольну ідентифікацію. Здійснено апаратну реалізацію запропонованого підходу та порівняння часових характеристик з алгоритмом AES 8-бітної архітектури на основі мікроконтролера Arduino Uno (ATmega328).

Праця [212] присвячена розробці та дослідженню апаратно реалізованих методів швидкої поліноміальної арифметики для деяких операцій гомоморфного шифрування на основі алгоритму Карацуби. Крім того, в [213] розглядаються можливості пришвидшення виконання операції поліноміального множення для гомоморфного шифрування при реалізації на FPGA. У [214] представлено характеристику реалізацій поліноміального множення для гомоморфного шифрування на основі GPU.

У [215] розроблено високоефективний метод шифрування зображення, який грунтується на поліномах перестановки в скінченних полях та є стійким до різного роду атак. Крім того, у запропонованому алгоритмі шифрування відсутні помилки округлення, тому шифрування здійснюється без втрат.

У [216] пропонується новий метод побудови S-блоків алгоритму AES, який грунтується на заміні незвідного полінома та афінного відображення. Стійкість створеного S-блоку оцінюється кількома стандартними тестами (бієктивності, нелінійності, строгого лавинного критерію (SAC), критерію бітонезалежності) і перевершує стійкість відомих *S*-box.

У [217] пропонується метод побудови S-блоку алгоритму AES на основі найменшої кількості вибраних незвідних поліномів, які відповідають певним критеріям. Таких поліномів є 17 і їх використання спрощує апаратну реалізацію S-блоку. Проведено дослідження SAC і відмічено, що поліном $p(x) = x^8 + x^7 + x^6 + x + 1$ є найкращим з ідеальним значенням SAC=0,5, що вказує на стійкість та надійність побудованого S-блоку.

У [218] запропоновано удосконалення симетричного алгоритму шифрування AES з використанням динамічних S-блоків, параметри яких залежать від ключа, динамічних незвідних поліномів та афінних констант.

У [219, 220] представлена найпоширеніша на сьогоднішній день симетрична криптосистема AES в кільці поліномів. Основна ідея полягає у виборі незвідного поліному, на основі якого будується алгоритм шифрування. Запропонований підхід був реалізований у середовищі MATLAB для 30 різних незвідних поліномів. У результаті проведених чисельних експериментів вдалося встановити незначний вплив зміни незвідних поліномів на рівень лавини.

Поліноміальна арифметика знайшла також своє застосування і для асиметричних криптосистем. Зокрема, в [219, 221] було розроблено модифіковану арифметику, необхідну для криптосистеми RSA з цілими числами Гауса та поліномами над скінченними полями. Аналіз описаних обчислювальних процедур дав можливість визначити їх переваги над класичними. В [221]

запропоновано алгоритмічне забезпечення криптосистеми Рабіна в поліноміальній системі числення.

Висновки до першого розділу

1. Проведено аналіз теоретичних основ симетричних криптоалгоритмів, визначено, що AES має кращий захист, володіє більшою швидкодією та пропускною здатністю. Однак існуючі симетричні криптоалгоритми не забезпечують належного рівня захисту інформаційних потоків.

2. Проведено аналіз найбільш поширених асиметричних криптосистем (Рабіна та на основі математичного апарату еліптичних кривих (ЕК)), основними операціями у яких є модулярне множення, модулярне експоненціювання та пошук оберненого елемента за модулем. Встановлено, що вони характеризуються значною часовою складністю і при вхідних параметрах розмірності 1024 біт і більше проявляються недоліки двійково-десяткової арифметики (наявність міжрозрядних переносів, строга послідовність при виконанні обчислень), що, в свою чергу, сповільнює виконання арифметичних операцій.

3. Проаналізовано особливості застосування СЗК в криптографічних методах захисту інформації. Встановлено основні переваги СЗК, а саме: можливість виконання операцій над числами, які менші за вибрані модулі, розпаралелення процесу обчислень, що є найбільш перспективним шляхом підвищення швидкодії обчислювальних систем та відсутність міжрозрядних переносів. Обгрунтовано можливість застосування методів шифрування у СЗК, зокрема у її МДФ.

4. Проведений аналіз літературних джерел свідчить про актуальність та важливість поліноміальних алгоритмів захисту інформаційних потоків. Відповідно, розробка нових методів, які стійкі до атак різного типу, є важливим напрямком розвитку сучасних криптосистем. Зокрема, поєднання поліноміальної арифметики та СЗК у кільці поліномів дозволить розпаралелити

процес виконання базових операцій в кільці поліномів [16], що, в свою чергу, призведе до підвищення швидкодії програмної реалізації та зменшення часової складності алгоритму [17, 20, 21], забезпечивши необхідний рівень захищеності.

Основні результати першого розділу відображені у роботах [1, 35, 39, 51, 53-56, 62, 67, 69, 70, 73, 74, 94, 96, 97, 105, 164, 165, 183, 187, 199, 200, 202].

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ЗМЕНШЕННЯ ЧАСОВОЇ СКЛАДНОСТІ ТА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ

2.1 Теоретичні основи визначення простих та взаємно простих чисел на основі властивості періодичності

Одним з основних аспектів розвитку прикладної математики є теорія простих та взаємно простих чисел. Їх важливим класом, який можна використати в криптографії, системі залишкових класів, а також для побудови високопродуктивних процесорів при оптимізації обчислень є числа виду $2^l + k$.

Крім того, прості і взаємнопрості числа є фундаментальними елементами в багатьох галузях математики та її застосувань. Їх унікальні властивості дозволяють вирішувати різноманітні завдання в науці, техніці та інформаційних технологіях.

Числа виду $2^l + k$ є особливим класом простих і взаємнопростих чисел, які знаходять застосування в кількох ключових напрямах:

1. Криптографія: завдяки своїй структурі числа виду використовуються для побудови стійких алгоритмів шифрування та генерації ключів. Їх використання в криптосистемах дає змогу досягати високого рівня безпеки і ефективності, особливо в умовах, де потрібна робота з великими числами.

2. Системи залишкових класів: числа виду широко застосовуються у залишкових системах для оптимізації обчислень. Вони дозволяють розробляти високопродуктивні алгоритми, які ефективно працюють із великими обсягами даних, що є важливим для сучасних інформаційних систем.

3. Комп'ютерна архітектура: у проєктуванні процесорів і обчислювальних систем ці числа використовуються для прискорення

арифметичних операцій, таких як множення і ділення. Їх застосування сприяє зменшенню енергоспоживання та підвищенню швидкості виконання обчислень.

4. Математичне моделювання: простота і передбачуваність властивостей чисел виду дозволяють використовувати їх у числовому аналізі та моделюванні фізичних і технічних процесів, де необхідні надійні та ефективні обчислення.

5. Теорія чисел: Цей клас чисел має важливе значення для розв'язання задач теорії чисел, включаючи дослідження розподілу простих чисел, властивостей модульної арифметики та багатьох інших фундаментальних проблем.

Основою розробленого методу є побудова матриці, елементи якої є залишками степеня двійки по простих модулях:

$$b_{i+1f} = 2 \cdot b_{if}(mod\, p_f). \tag{2.1}$$

де $b_{if} = 2^i(mod\, p_f)$, i=0,…,l-1, p_f - прості числа.

В результаті проведених досліджень було встановлено, що числа виду $2^n(mod\, p_f)$ володіють властивістю періодичності. В таблиці 2.1 представлено результат обчислень згідно формули (2.1) для перших r простих чисел для визначення періоду π_i. Слід відмітити, що значення b_{i+1f} повторюються з певним періодом π_{i+1}, тобто $b_{\pi_i+2\ \ f} = b_{1f}$. Для кожного простого модуля p_f період π_i буде різним. Наприклад, для $p_3 = 7$ період $\pi_3 = 3$, для $p_{10} = 31$ період $\pi_{10} = 5$ і т.д.

Визначення періодичності є ключовим моментом при вирішенні задачі подільності та встановлення взаємної простоти чисел виду $2^l + k$ на основі таких співвідношень [222, 223]:

$$(2^{\pi_i n+s_i} + k) \equiv 0(mod\, p_f). \tag{2.2}$$

$$(2^{\pi_i l+s_i} + k) \neq 0(mod\, p_f), \tag{2.3}$$

де π_i - період модуля p_f, s_i - значення, при якому $(2^{s_i}+k)(mod\, p_f)=0$.

Таблиця 2.1

Визначення періодичності чисел виду 2^l по простих модулях p_f

2^i / p_f	2^{l-1}	2^{l-2}	…….	2^i	……..	2^2	2^1	2^0
p_1=3	b_{l-11}	b_{l-21}	……	b_{i1}	…….	b_{21}	b_{11}	b_{01}
$p_2=5$	b_{l-12}	b_{l-22}	……	b_{i2}	…….	b_{22}	b_{12}	b_{02}
$p_3=7$	b_{l-13}	b_{l-23}	……	b_{i3}	…….	b_{23}	b_{13}	b_{03}
…….	……..	…….	……	……	…….	……	…….	…….
p_f	b_{l-1f}	b_{l-2f}	……	b_{if}	…….	b_{2f}	b_{1f}	b_{0f}
…….	……..	…….	……	……	…….	……	…….	…….
p_{r-1}	b_{l-1r-1}	b_{l-2r-1}	……	b_{ir-1}	…….	b_{2r-1}	b_{1r-1}	b_{0r-1}
p_r	b_{l-1r}	b_{l-2r}	……	b_{ir}	…….	b_{2r}	b_{1r}	b_{0r}

Формула (2.2) відповідає за встановлення подільності чисел, а (2.3) - за взаємно простоту. При цьому стартова позиція рекурентної перевірки подільності числа на прості множники визначається згідно виразу [219]:

$$res(2^{s_i}+k)\big(mod\, p_f\big)+res\sum_{j=0}^{l}2^{\pi_j}(mod\, p_f)\equiv 0(mod\, p_f). \qquad (2.4)$$

Пошук залишків по простих модулях в формулах (2.3) і (2. 4) здійснюється на основі векторно-модульного методу. На першому етапі формується таблиця 2.2 зі значеннями $C_j=2^j \bmod p_f$, $C_{j+1}=2\cdot C_j \bmod p_f$, де $j=0\ldots l-1$, та $q_i=(2^{\pi_i l+s_i}+k)=\sum_{z=0}^{l-1}a_z 2^z$, причому $a_z=0{,}1$ для l – розрядного числа q_i.

Таблиця 2. 2

Векторно-модульний метод пошуку залишку

C_i		C_{l-1}	C_{l-2}		C_i		C_2	C_1	C_0
q_i		q_{l-1}	q_{l-2}		q_i		q_2	q_1	q_o

Значення залишку обчислюється згідно такого співвідношення:

$$(2^{\pi_i l+s_i}+k)\left(mod\, p_f\right)=\left(\sum_{i=0}^{l-1} C_i\, q_i\right) mod\, p_f \qquad (2.5)$$

Даний метод дозволяє зменшити часову складність пошуку залишків.

2.1.1 Приклад визначення простих та взаємно прости чисел виду 2^l+k

На основі формули (2.4) для чисельного експерименту будується таблиця 2.3, яка дає можливість ефективно вирішувати задачі не тільки пошуку простих та взаємно простих чисел виду 2^l+3, але й факторизації. Ця таблиця є потужним інструментом для аналізу властивостей чисел та оптимізації обчислень у системах з великими даними.

Для прикладу розглядаються числа виду з метою перевірки їх подільності на прості числа в діапазоні від 3 до 147. Такий підхід дозволяє швидко ідентифікувати прості числа або знаходити їх дільники, що є важливим для задач криптографії та теорії чисел. Процедуру можна застосувати до чисел будь-якої розрядності, що робить її універсальною. Проте варто зазначити, що зберігання подібних таблиць вимагає значних обсягів пам’яті, особливо для великих значень .

Додатково, умови подільності чисел виду 2^n+k (k=1, 3, 5, 11, 13) наведені в таблиці 2.4. Ці умови дозволяють проводити швидкі перевірки та аналізувати властивості чисел для подальшого їх застосування у чисельних методах, факторизації та побудові систем залишкових класів. Наприклад, такі таблиці можуть використовуватись для оптимізації алгоритмів розкладу чисел на прості множники або генерації ключів у криптографічних системах.

Таблиця 2.3

Пошук простих чисел виду 2^l+3, розклад на прості множники

131072		65536		32768		16384		8192		4096		2048		1024		512		256		128		64		32		16		8		4		2		1	
17		16		15		14		13		12		11		10		9		8		7		6		5		4		3		2		**1**		**0**	
5,7	1	10 7	1		1	7	1	5,11, 147	1		1	7	1	13, 79	1	5, 103	1	7, 37	1	131	1	67	1	5,7	1	19	1	11	1	7	1		**2**		**1**
7	1	13, 109	1	5,11	1	7, 131	1	83	1		1	5,7	1		1		1	7	1	5	1		1	7,11	1	13, 19	1	5	1	7, 59	1	29, 101	1		1
5,7, 11	1		1		1	7	1	79,5	1		1	7, 29	1	13	1	5	1	7, 37	1	11, 53	1		1	5,7	1	19, 97	1		1	7	1	5	1	61	1
7	1	13	1	5, 109	1	7	1		1		1	5,7	1		1	11	1	7, 137	1	5	1	103	1	7	1	13, 19	1	5, 131	1	7	1		1		1
5,7	1	79	1		1	7,	1	5	1		1	7, 11, 139	1	13, 131	1	5	1	7, 37	1	59	1		1	5,7,	1	19	1	29	1	7	1	5,11	1	67	1
7, 131	1	13	1	5	1	7, 109	1	11,29 53,97	1		1	5,7	1		1		1	7	1	5,83	1	61	1	7	1	13, 19	1	5,11	1	7	1		1		1
5,7	1	79	1	11	1	7	1	5	1		1	7, 101	1	13	1	5	1	7, 37	1		1		1	5,7, 11	1	19	1	103	1	7	1	5	1		1
7,11	1	13	1	5	1	7	1	109	1	67	1	5,7, 59	1		1		1	7	1	5, 11, 107	1	131	1	7,29	1	13, 19	1	5	1	7	1	79	1		1
5,7	1		1	29	1	7	1	5, 131	1	61	1	7	1	13	1	5,11	1	7, 37	1		1		1	5,7	1	19, 97	1		1	7	1	5	1		1
7	1	13	1	5	1	7	1		1	109	1	5,7, 11	1		1		1	7	1	5	1		1	7,53	1	13, 19, 79	1	5	1	7	1	11, 83	1	103	1
5,7, 139	1		1	59	1	7	1	5,11	1		1	7	1	13	1	5	1	7, 37	1	29	1		1	5,7	1	19	1	11	1	7, 131	1	5	1		1
7,29	1	13, 107	1	5,11, 103	1	7	1	97	1		1	5,7, 109, 137	1		1	131	1	7, 37	1	5, 79	1	67	1	7,11	1	13, 19	1	5	1	7	1		1		1
5,7, 11	1	131	1		1	7	1	5,83	1		1	7	1	13	1	5, 53	1	7	1	11	1		1	5,7	1	19	1	101	1	7	1	5	1	61	1
7	1	13	1	5	1	7	1		1		1	5,7	1	79, 109	1	11,29	1	7, 37	1	5	1		1	7	1	13, 19	1	5	1	7	1		1		1

Таблиця 2.4

Умови подільності чисел виду $2^l + k$ (k=1, 3, 5, 11, 13)

Прості числа	Вирази виду $2^l + 1$, які діляться на прості числа	Вирази виду $2^l + 3$, які діляться на прості числа	Вирази виду $2^l + 5$, які діляться на прості числа	Вирази виду $2^l + 11$, які діляться на прості числа	Вирази виду $2^l + 13$, які діляться на прості числа
3	$2^{2l+1} + 1$	-	-	-	-
5	$2^{4l+2} + 1$	$2^{4l+1} + 3$	-	-	-
7	-	$2^{3l+2} + 3$	$2^l + 5$, при l=1	-	-
11	$2^{10l+5} + 1$	$2^{10l+3} + 3$	$2^{10l+9} + 5$	-	-
13	$2^{12l+6} + 1$	$2^{12l+10} + 3$	$2^{12l+3} + 5$	$2^{12l+2} + 11$	-
17	$2^{8l+4} + 1$	-	-	-	$2^{8l+3} + 13$
19	$2^{18l+9} + 1$	$2^{18l+4} + 3$	$2^{18l+7} + 5$	$2^{18l+4} + 11$	$2^{18l+15} + 13$
23	-	-	$2^{11l+6} + 5$	$2^{11(l+1)} + 11$	-
29	$2^{28l+14} + 1$	$2^{28l+19} + 3$	$2^{28l+8} + 5$	$2^{28l+12} + 11$	$2^{28l+5} + 13$
31	-	-	-	-	-
37	$2^{36l+18} + 1$	$2^{36l+8} + 3$	$2^{36l+5} + 5$	$2^{36l+13} + 11$	$2^{36l+30} + 13$
41	$2^{20l+10} + 1$	-	$2^{20l+17} + 5$	-	-
43	$2^{14l+7} + 1$	-	-	$2^{14l+6} + 11$	-
47	-	-	$2^{23l+9} + 5$	$2^{46l+18} + 11$	$2^{23l+8} + 13$
53	$2^{52l+26} + 1$	$2^{52l+43} + 3$	$2^{52l+21} + 5$	$2^{52l+32} + 11$	$2^{52l+51} + 13$
59	$2^{58l+29} + 1$	$2^{58l+21} + 3$	$2^{58l+35} + 5$	$2^{58l+55} + 11$	$2^{58l+17} + 13$
61	$2^{60l+30} + 1$	$2^{60l+36} + 3$	$2^{60l+52} + 5$	$2^{60l+46} + 11$	$2^{60l+11} + 13$
67	$2^{66l+33} + 1$	$2^{66l+6} + 3$	$2^{66l+48} + 5$	$2^{66l+27} + 11$	$2^{66l+53} + 13$
71	-	-	-	$2^{35l+12} + 11$	-
73	-	-	-	-	-
79	-	$2^{39l+10} + 3$	-	-	-
83	-	$2^{82l+31} + 3$	$2^{82l+51} + 5$	$2^{82l+49} + 11$-	-
89	-	-	-	$2^{11l+9} + 11$	-
97	$2^{45l+24} + 1$	$2^{45l+40} + 3$	-	-	-
101	$2^{100l+50} + 1$	$2^{100l+19} + 3$	$2^{100l+74} + 5$	$2^{100l+65} + 11$	$2^{100l+17} + 13$

Слід зазначити, що комірки таблиці 2.4, які залишилися незаповненими, відповідають взаємнопростим числам виду $2^l + k$ з відповідними простими, а всі інші числа є складеними.

2.2 Теоретичні основи зменшення часової складності та підвищення ефективності при реалізації криптосистеми Рабіна на основі операції додавання

Останнім часом зі збільшенням обсягу інформації, що передається через телекомунікаційні канали, зростають вимоги до її безпеки [224, 225], а також до ефективності алгоритмів шифрування та розшифрування [226-228]. Вирішення цих проблем є неможливим без застосування асиметричних криптографічних систем, які усувають ключовий недолік симетричних алгоритмів - необхідність створення надійного каналу для обміну ключами [229, 230].

Більшість поширених асиметричних криптосистем, зокрема RSA і криптосистема Ель-Гамаля, базуються на модулярному експоненціюванні [51, 231, 232]. Ця операція характеризується високою обчислювальною складністю, що негативно впливає на швидкодію відповідних алгоритмів [233, 234]. На відміну від них, криптосистема Рабіна використовує операцію піднесення до квадрату за модулем для створення зашифрованого тексту. Такий підхід дозволяє суттєво підвищити швидкість шифрування, зберігаючи при цьому високий рівень криптостійкості [201, 235]. Стійкість цієї системи до атак ґрунтується як на складності факторизації [236], так і на проблемі визначення квадратичних лишків. Обидві задачі належать до класу проблем із субекспоненційною часовою складністю [237].

Проте недоліком криптосистеми Рабіна є необхідність відновлення початкового повідомлення на основі залишків, що виконується із застосуванням китайської теореми про залишки (КТЗ). Базові обчислення, пов’язані з цим методом, є ресурсоємними, що призводить до зниження швидкодії під час розшифрування [238, 239]. У зв’язку з цим актуальною є розробка алгоритмів для реалізації криптосистеми Рабіна, які дозволяють

зменшити обчислювальну складність за рахунок заміни ресурсомістких арифметичних операцій більш ефективними [196, 240].

З метою оптимізації криптосистеми Рабіна для генерування ключів та шифрування інформації пропонується використання векторно-модульного підходу [51, 241].

Одне з обраних чисел p або q (наприклад, p) подається у двійковій формі, згідно виразу: $p = \sum_{i=0}^{l-1} p_i \cdot 2^i$, де l – його розрядність, p_i=0 або1. На наступному етапі формується векторний рядок q_i, який обчислюється за правилом рекурентного співвідношення: $q_i=2\cdot q_{i-1}=2^i q_0$, $q_0=q$ (таблиця 2.5).

Таблиця 2.5

Представлення вектор-рядків для операції множення

i	l-1	…	2	1	0
p_i	p_{l-1}	…	p_2	p_1	p_0
$q_i=2\cdot q_{i-1}$	q_{l-1}	…	q_2	q_1	$q_0=q$

Результат множення $h=p\cdot q$ обчислюється за формулою:

$$h = p \cdot q = \sum_{i=0}^{l-1} p_i \cdot q_i, \qquad (2.6)$$

Таким чином, операція множення зводиться до додавання елементів q_i, для яких відповідні p_i=1. Побудовані значення можна представити у вигляді таблиці 2.6, що дозволяє полегшити виконання операцій у процесі шифрування.

Для шифрування початкове число M представляється у двійковій формі: $M = \sum_{i=0}^{l-1} d_i \cdot 2^i$ (d_i=0 або 1) На основі цього запису формується векторний рядок m_i , який визначається за рекурентним співвідношенням: $m_i = 2 \cdot m_{i-1} \, mod \, h$, $m_0=M$.

Таблиця 2.6

Представлення вектор-рядків для виконання операції множення за модулем

i	l-1	…	2	1	0
d_i	d_{l-1}	…	d_2	d_1	d_0
$m_i=2\cdot m_{i-1} \bmod h$	m_{l-1}	…	m_2	m_1	$m_0=M$

Шифрування відбувається згідно співвідношення:

$$C = M^2 \, mod \, h = \left(\sum_{i=0}^{l-1} d_i \cdot m_i\right) mod \, h, \qquad (2.7)$$

Відповідно, операція множення (або піднесення до квадрату) за модулем може бути замінена на операцію модулярного додавання тих елементів m_i, для яких відповідні d_i дорівнюють 1.

Для обчислення залишку $a \bmod h$ [183], де розрядність a рівне l, необхідно представити у двійковій системі числення: $a = \sum_{i=0}^{l-1} a_i \cdot 2^i$ (a_i=0 або 1). На основі цього запису формується вектор-рядок $a_{1i}=2^i \bmod h$, i=0, …, l, який представлено в таблиці 2.7, а результат обчислюється за формулою:

$$a \, mod \, h = \left(\sum_{i=0}^{l-1}\left(a_i 2^i \, mod \, h\right)\right) mod \, h = \left(\sum_{i=0}^{l-1}(a_i a_{1i})\right) mod \, h. \qquad (2.8)$$

Таблиця 2.7

Таблиця знаходження залишку $a \bmod p$

i	l-1	l-2		2	1	0
a_i	a_{l-1}	a_{l-2}	…	a_2	a_1	a_0
$2^i \bmod h$	$2^{l-1} \bmod h$	$2^{l-2} \bmod h$		$2^2 \bmod h$	$2^1 \bmod h$	$2^0 \bmod h$
a_{1i}	$a_{1\,l-1}$	$a_{1\,l-2}$	…	a_{12}	a_{11}	a_{10}

Згідно з таблицею 2.7 та виразом (2.8), залишок $a \, mod \, h$ обчислюється як сума тих степенів двійки (або елементів a_{1i}), для яких відповідні a_i=1.

Також варто зауважити, що два сусідні значення a_{1i} та $a_{1\ i+1}$ взаємопов'язані за допомогою рекурентного співвідношення:

$$a_{1i+1} = \begin{cases} 2 \cdot a_{1\ i},\ 2 \cdot a_{1\ i} < n \\ 2 \cdot a_{1\ i} - p,\ 2 \cdot a_{1\ i} \geq n. \end{cases} \quad (2.9)$$

Для пошуку залишку за модулем немає необхідності виконувати обчислювально витратну операцію ділення з остачею. Цей процес може бути значно спрощений за рахунок використання послідовного віднімання. Крім того, множення на 2 легко реалізується апаратними засобами шляхом додавання нуля в кінці двійкового запису числа [242, 243].

Для обчислення значень *x* та *y* необхідно визначити квадратний корінь за модулем. Класичні методи, такі як використання символів Якобі або Лежандра, є складними та потребують значних обчислювальних ресурсів [244, 245]. У зв'язку з цим пропонується альтернативний підхід, що базується на операціях додавання та перевірки числа на повноту квадрата. Це значно знижує часову складність криптографічних обчислень у криптосистемі Рабіна.

Для знаходження $x\ mod\ p = \sqrt{f}$, слід виконати послідовність операцій: $f+p$, $f+2p$, … , $f+i \cdot p$, де i – найменше значення, при якому $f+i \cdot p$ є повним квадратом. Аналогічно виконується обчислення $y^2\ mod\ q = s$.

Для розв'язання систем порівнянь пропонується використовувати метод додавання модуля. Розглянемо, наприклад, першу систему порівнянь (1.4). Оскільки будь-яку конгруенцію $M_1 \bmod p = x$ можна представити у вигляді $M_1 = \lambda p + x$, де $\lambda = 0, 1, 2, \ldots$, до залишку x необхідно додавати модуль p доти, доки не буде виконано умову $(x+\lambda p) \bmod q = M_1 \bmod q = y$.

На рисунку 2.1 зображена блок-схема відновлення десяткового числа за його залишками з використанням операції додавання, а на рисунку 2.2 — схема реалізації криптосистеми Рабіна із застосуванням запропонованого методу.

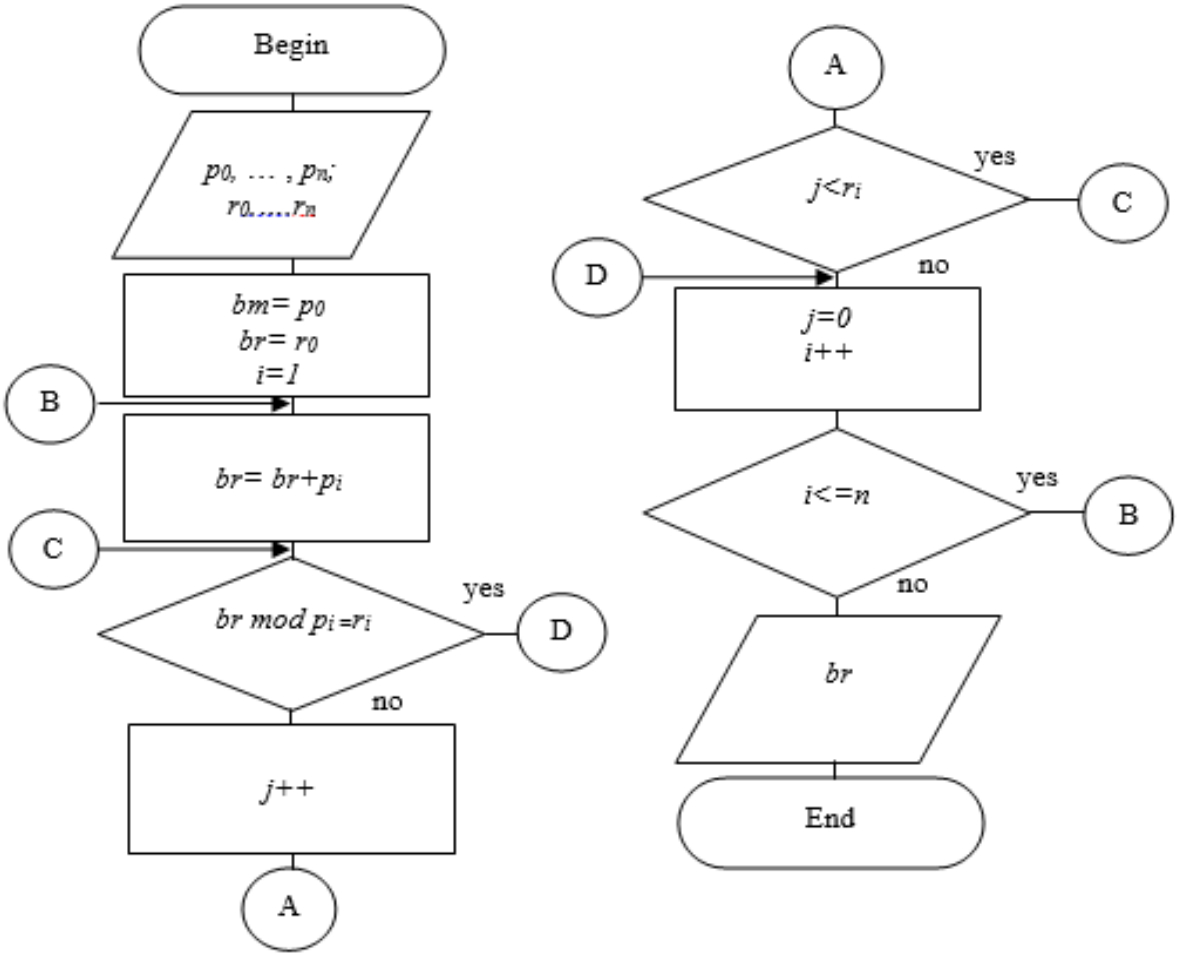


Рисунок 2.1 – Блок-схема відновлення десяткового числа за його залишками на основі операції додавання

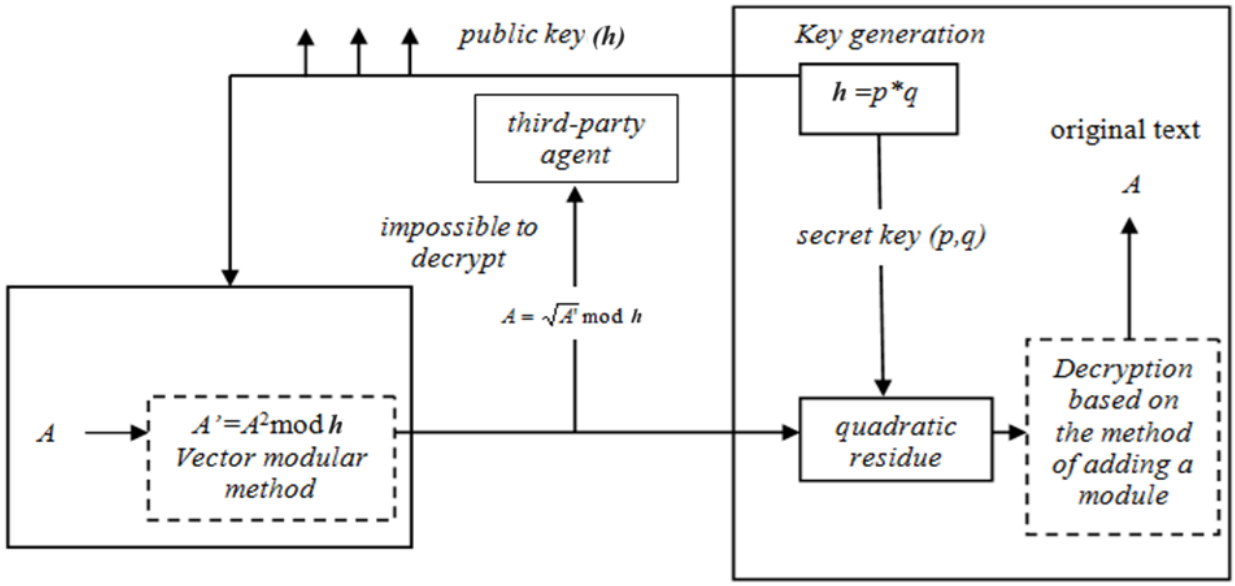


Рисунок 2.2 – Схема реалізації криптосистеми Рабіна з використанням методу додавання модуля

Слід зауважити, що класичні методи, такі як використання китайської теореми залишків (КТЗ) та алгоритму Гарнера, передбачають обчислення оберненого елемента за модулем. Ця операція є досить трудомісткою та характеризується високою обчислювальною складністю [246–250]. Як наслідок, значно погіршуються часові характеристики під час реалізації криптоалгоритму Рабіна.

Зменшення обчислювальної складності та оптимізація процесу є важливими задачами, які дозволяють підвищити ефективність алгоритмів, зокрема шляхом виключення або мінімізації операцій з оберненими елементами [251].

2.2.1 Приклад реалізації криптосистеми Рабіна на основі операції додавання

Вибираються таємні ключі, наприклад p=37, q= 43, тоді з використанням (2.6) і таблиці 2.5 отримується значення: $n=p\cdot q=37\cdot 43=1376+172+43=1591$ (табл. 2.8).

Таблиця 2.8

Обчислення добутку $h=p\cdot q$ =31·47

i	5	4	3	2	1	0
p_i	1	0	0	1	0	1
$q_i=2\cdot q_{i-1}$	1376	688	344	172	86	43

Нехай відкритий текст M=155. Використовуючи формулу (2.2) та дані, наведені в таблиці 2.2, формується відповідна таблиця 2.9, яка відображає процес шифрування.

Таблиця 2.9

Процес шифрування

i	7	6	5	4
d_i	1	0	0	1
$m_i=2\cdot m_{i-1} \bmod h$	748	374	187	889
i	3	2	1	0
d_i	1	0	1	1
$m_i=2\cdot m_{i-1} \bmod h$	1240	620	310	155

В результаті отримується значення $C = 155^2 \bmod 1591 =$ $=(748+889+1240+310+155) \bmod 1591=3342 \bmod 1591$. Результат пошуку залишку визначається згідно таблиці 2.7 та 2.8 (таблиця 2.10).

Таблиця 2.10

Таблиця пошуку залишку 3342 mod 1591

i	11	10	9	8	7	6
a_i	1	1	0	1	0	0
a_{1i}	457	1024	512	256	128	64
i	5	4	3	2	1	0
a_i	0	0	1	1	1	0
a_{1i}	32	16	8	4	2	1

Отриманий результат: $3342 \bmod 1591=(457+1024+256+8+4+$ $+2) \bmod 1591=1751 \bmod 1591=1751-1591=160$.

Для розшифрування криптограми C знаходяться відповідні значення: $f=160 \bmod 43=31$, $s=160 \bmod 37=12$, причому залишки знаходяться з використанням таблиці 2.3 та виразу (2.3). На наступному етапі формуються послідовності для знаходження повного квадрату:

31, 31+43=74, 74+43=117, 117+43=160, 160+43=203, 203+43=246, 246+43=289; 12, 12+37=49.

В результаті проведених обчислень, отримуються значення $\sqrt{31}(mod\ 43) = 17$ та 43-17=26; $\sqrt{12}(mod\ 37) = 7$ та 37-7=30. Для

отримання результату розшифрування формуються чотири пари чисел: (26, 7), (26, 30), (17, 7), (17, 30), які визначають чотири системи конгруенцій:

$$\begin{cases} M_1 \, mod \, 43 = 26; \\ M_1 \, mod \, 37 = 7; \end{cases} \begin{cases} M_2 \, mod \, 43 = 26; \\ M_2 \, mod \, 37 = 30; \end{cases}$$
$$\begin{cases} M_3 \, mod \, 43 = 17; \\ M_3 \, mod \, 37 = 7; \end{cases} \begin{cases} M_4 \, mod \, 43 = 17; \\ M_4 \, mod \, 37 = 30. \end{cases} \quad (2.10)$$

Відповідно до методу додавання модуля, розв'язки перших двох систем подаються у вигляді однієї таблиці. Зокрема, до залишку 26 послідовно додається модуль 43 доти, поки залишок від знайденої суми за модулем 37 не стане рівним 7 та 30 (таблиця 2.11).

Таблиця 2.11

Процес розшифрування

λ	0	1	2	3	4
26+43·λ	26	69	112	155	198
(26+43·λ)mod 37	26	32	1	7	13
λ	5	6	7	8	9
26+43·λ	241	284	327	370	413
(26+43·λ)mod 37	19	25	31	0	6
λ	10	11	12	13	
26+43·λ	456	499	542	585	
(26+43·λ)mod 37	12	18	24	30	

Отже, згідно таблиці 2.11, розв'язками систем (2.10) є значення M_1=155 (відкритий текст), M_2=585, M_3=1591-155=1436, M_4=1591-585=1006, які отримані без використання громіздких процедур модулярного множення та експоненціювання, КТЗ, пошуку квадратичного лишку та оберненого

елемента на основі розширеного алгоритму Евкліда, а також необхідності контролю переповнення розрядної сітки при виконанні проміжних обчислень.

Отже, відповідно до таблиці 2.11, розв'язками системи (2.10) є значення M_1=155 (відкритий текст), M_2=585, M_3=1591-155=1436, M_4=1591-585=1006. Вони були отримані без застосування обчислювально складних процедур, таких як: модулярне множення, експоненціювання, використання китайської теореми залишків (КТЗ), пошук квадратичних лишків або обчислення оберненого елемента через розширений алгоритм Евкліда. Крім того, підхід дозволяє уникнути проблем з переповненням розрядної сітки під час виконання проміжних обчислень, що є важливою перевагою в контексті криптографічних операцій.

Це свідчить про значну оптимізацію процесу обчислень, яка знижує обчислювальну складність і підвищує ефективність криптосистем, роблячи їх більш практичними для реального застосування без втрати криптостійкості.

2.2.2 Аналіз часових складностей

Для аналізу часової складності криптосистеми Рабіна слід враховувати найбільш ресурсоємні операції, які виконуються під час шифрування та розшифрування. Серед них найсуттєвіший внесок у загальну складність мають модулярне множення, обчислення квадратичних лишків та процедура відновлення числа за залишками.

Часова складність модулярного множення залежить від обраного алгоритму [157]: класичний метод множення має квадратичну складність $O(l^2)$ (l – розрядність множників), алгоритм Шонхаге-Штрасена - лінійно-логарифмічною $O(l \cdot log\, l \cdot log(log\, l))$, алгоритми Карацуби та Тома-Кука - $O(l^{1,585})$ та $O(l^{1,465})$ відповідно, матрично-модульний метод - $O(2l\, log\, l)$, векторно-модульний метод, в якому операція множення замінюється додаванням - $O(\frac{l}{2} log\, l)$ [237].

Розшифрування у криптосистемі Рабіна зводиться до відновлення позиційного представлення числа за його залишками. У розробленому методі використовуються операції множення та додавання з складностями $O(l^2)$ і $O(l)$ відповідно. При використанні КТЗ для багатомодульної криптосистеми Рабіна необхідно k – разів знайти обернений елемент за модулем з часовою складністю $O(l^3)$ [237], у класичному алгоритмі Гарнера присутні ті ж самі операції, що і в КТЗ.

Тому використання тільки операції додавання за модулем [252] у криптосистемі Рабіна дозволяє зменшити часову складність розшифрування з $O2(kl^3 + 2l^2k + lk)$ (КТЗ) або $O1\left(kl^3 + l^2\left(\frac{k^2+k}{2}\right)\right)$ (класичний алгоритм Гарнера) до $O\left(l^2\left(\frac{k^2+k}{2}\right)\right)$ згідно запропонованого методу, де $\left(\frac{k^2+k}{2}\right)$ - кількість операцій множення. Відповідно, загальна часова складність криптосистеми Рабіна зменшується з $O1(kl^3 + 2l^2k + lk + l^{1,465})$ до $O\left(l^2\left(\frac{k^2+k}{2}\right) + \frac{l\,log_2\,l}{2}\right)$ за рахунок заміни операції множення додаванням. Згідно розглянутого у п.4 прикладу для k=2 останні вирази для класичного та запропонованого методів набудуть відповідно такого вигляду: $O1(2l^3 + 4l^2 + 2l + l^{1,465})$ та $O\left(3l^2 + \frac{l\,log_2\,l}{2}\right)$.

Ефективність запропонованого удосконалення реалізації криптосистеми Рабіна визначається як співвідношення часових складностей:

$$\mathrm{E}(l,k) = \frac{(2l^3 + 4l^2 + 2l + l^{1,465})}{l^2\left(\frac{k^2+k}{2}\right) + \frac{l\,log_2\,l}{2}} = \frac{4l^2 + 8l + 2l^{0,465} + 4}{l(k^2+k) + log_2\,l}$$

Графічне співвідношення часових складностей у логарифмічній шкалі представлено на рисунку 2.3.

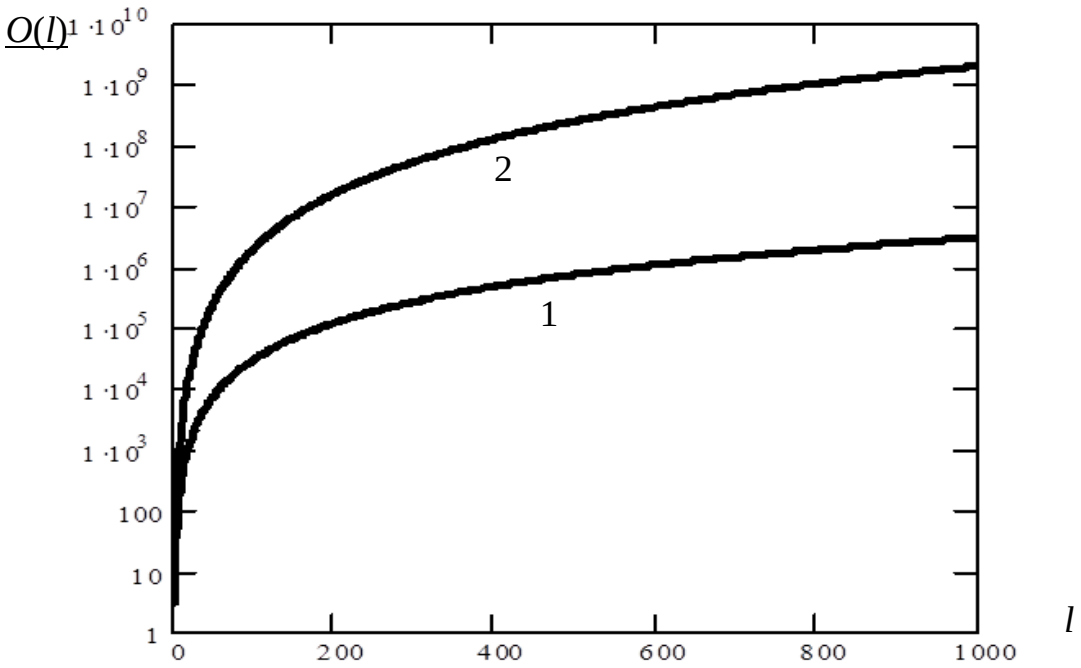


Рисунок 2.3 – Часова складність криптосистеми Рабіна на основі запропонованого (1) та класичного (2) методів

Отже, запропонований підхід складає теоретичну основу для зменшення часової складності та підвищення ефективності приблизно у 170 разів для вхідних значень k=2, l=256 та, в свою чергу, підвищення ефективності при реалізації криптосистеми Рабіна.

2.3 Теоретичні основи трьохмодульної криптосистеми Рабіна на основі операції додавання

Аналіз літературних джерел дозволив виявити недоліки класичної криптосистеми Рабіна. Зокрема, у [253] авторам вдалося досягнути більшої криптостійкості без збільшення обчислювальної складності шляхом заміни квадратичної конгруенції на кубічне рівняння.

В [66] зазначено переваги використання гаусівських цілих чисел для генерації відкритого ключа криптосистеми Рабіна. Це дозволило розробити відповідну арифметику для теореми Вільсона і КТЗ, а також для обчислення символів Якобі, Лежандра і квадратичних залишків. У [201, 254] розглянута

трьохмодульна криптосистема Рабіна (або H-Rabin), стійкість якої базується на розкладі добутку трьох чисел $t=pqr$ де p, q і r - прості числа. Це дозволяє збільшити величину блоку відкритого тексту без втрати криптостійкості, що забезпечує більшу захищеність криптосистеми від відповідних видів атак. Крім того, обґрунтовано використання у криптосистемі Рабіна МДФ СЗК.

Однак у зазначених удосконаленнях використовуються обчислювально громіздкі операції модулярного множення багаторозрядних чисел, піднесення до квадрату, пошуку мультиплікативного оберненого елемента за модулем тощо. Тому розробка алгоритмічного забезпечення для реалізації криптосистеми Рабіна на основі елементарної операції додавання є актуальною задачею.

Для зменшення часової складності трьохмодульної криптосистеми Рабіна [254] при шифруванні пропонується використовувати векторно-модульний метод модулярного множення [67]. На першому етапі вибираються три великих простих числа p, q і r і обчислюється значення $t= p \cdot q \cdot r$, де число t - відкритий ключ, а p, q і r – таємний.

Шифрування відкритого повідомлення M відбувається за допомогою відкритого ключа t за формулою (2.6).

З використанням векторно-модульного методу модулярного множення значення $M \cdot M \bmod t$ шукаються таким чином. Відкритий блок M представляється у двійковій формі: $M = \sum_{i=0}^{l-1} a_i \cdot 2^i$, де $a_i = 0, 1$, l– розрядність модуля. Далі будуються два вектор-рядки, в першому з яких записуються елементи M_i, в другому - $m_0 = 2^0 M \bmod t$, $m_i = 2 \cdot h_{i-1} \bmod t$ (таблиця 2.12).

Результат модулярного множення $M \cdot M \bmod t$ знаходиться згідно формули:

$$C = M^2 \bmod t = \left(\sum_{i=0}^{l-1} a_i \cdot m_i\right) \bmod t, \qquad (2.11)$$

Таблиця 2.12

Представлення вектор-рядків модулярного множення

i	l-1	…	2	1	0
a_i	a_{l-1}	…	a_2	a_1	a_0
$m_i = 2 \cdot m_{i-1} \, mod \, t$	m_{l-1}	…	m_2	m_1	$m_0 = 2^0 \cdot M \, mod \, t$

Отже, операція модулярного множення замінюється операцією додавання тих m_i, для яких відповідні a_i рівні 1. Даний метод характеризується меншою часовою складністю порівняно з класичними.

При розшифруванні криптограми C, аналогічно (2.7), вводяться додаткові допоміжні величини s, w і u:

$$s = C \, mod \, p; w = C \, mod \, q; u = C \, mod \, r. \qquad (2.12)$$

Відповідно до (2.11), значення x, y і z шукаються з таких порівнянь:

$$x^2 \equiv s(mod \, p), y^2 \equiv w(mod \, q), z^2 \equiv u(mod \, r). \qquad (2.13)$$

Для знаходження x, y і z необхідно обчислити значення кореня квадратного за модулем. Класичні підходи з використання символів Якобі або Лежандра є трудомісткими [255]. Тому доцільно використати метод, який вимагає тільки операції додавання та перевірки, чи є число повним квадратом. Така процедура дозволяє суттєво зменшити обчислювальну складову методу Рабіна. Отже, для пошуку значення $\sqrt{s} \, mod \, t$ необхідно виконати наступну послідовність дій: $s + t$, $s + 2t$, …, $s + i \cdot t$, де i - значення, при якому $s + i \cdot t$ буде повним квадратом. Аналогічно шукається $y^2 \equiv w(mod \, q)$, $z^2 \equiv u(mod \, r)$. Оскільки розв'язками порівнянь (2.6)-(2.8) буде 6 значень, то для розшифрування потрібно розв'язати вісім систем порівнянь, що утворюються як комбінації усіх можливих варіантів пошуку відкритого повідомлення

(i=1…8):

$$\begin{cases} M_i \equiv \pm x(mod\, p); \\ M_i \equiv \pm y(mod\, q); \\ M_i \equiv \pm z(mod\, r); \end{cases} \quad (2.14)$$

Шуканим повідомленням M буде один із розв'язків систем порівнянь (2.9). Для їх вирішення доцільно використати метод на основі додавання добутку модулів, який характеризується меншою обчислювальною складністю в порівнянні з класичними: використання КТЗ та алгоритму Гарнера [256].

Для прикладу розглянемо одну із систем порівнянь (2.14), в якій параметри x, y, z додатні. Оскільки будь-яку конгруенцію $x \bmod p = M_1$ можна представити у вигляді $x = \gamma p + M_1$, де $\gamma = 0, 1, 2, \ldots$, то до залишку $M_1^{(1)} = x$ потрібно додавати модуль p стільки разів, поки не буде виконуватись конгруенція $M_1^{(2)} \equiv y(mod\, q)$, де $M_1^{(2)} = \mathrm{M}_1^{(1)} + \gamma_1 p$. Далі необхідно додавати добуток pq, поки не буде виконуватись конгруенція $M_1^{(3)} \equiv z(mod\, r)$, де $M_1^{(3)} = M_1^{(2)} + \gamma_2 pq = M_1$. Аналогічним чином шукаються розв'язки інших систем порівнянь (2.9).

Слід відмітити, що в КТЗ та алгоритмі Гарнера необхідно шукати мультиплікативний обернений елемент за модулем. При використанні багаторозрядних чисел класичні методи (повний перебір всіх можливих варіантів, використання теореми Ейлера або алгоритму Евкліда [34]) характеризуються великою обчислювальною складністю, що призводить до збільшення часових характеристик при реалізації криптоалгоритму Рабіна.

2.3.1 Приклад застосування розробленого методу

Нехай таємним ключем криптосистеми Рабіна буде три простих числа p=31, q= 23 *і* r =19. Далі обчислюється значення відкритого ключа:

$t = p \cdot q \cdot r = 31 \cdot 23 \cdot 19 = 13547$.

В якості відкритого тексту для шифрування вибирається повідомлення $M = 215$. Тоді згідно формули (2.11) на основі векторно-модульного методу модулярного множення шукається значення $C \equiv 215^2 \, mod \, 1\,3547$.

Для цього повідомлення M представляється у двійковій системі числення: $M = 215 = 11010111_2$.

Далі будуються два вектор-рядки, в першому з яких записуються елементи a_i, які можуть дорівнювати 0 або 1, в другому - елементи $m_0 = 2^0 215 \, mod \, 1\,3547 = 215$, $m_1 = 2 \cdot 215 \, mod \, 1\,3547 = 430$, $m_2 = 2 \cdot 430 \, mod \, 1\,3547 = 860$, $m_3 = 2 \cdot 860 \, mod \, 1\,3547 = 1720$, $m_4 = 2 \cdot 1720 \, mod \, 1\,3547 = 3440$, $m_5 = 2 \cdot 3440 \, mod \, 1\,3547 = 6880$, $m_6 = 2 \cdot 6880 \, mod \, 1\,3547 = 13740 \, mod \, 1\,3547 = 213$, $m_7 = 2 \cdot 213 \, mod \, 1\,3547 = 426$ (табл. 2.13).

Таблиця 2.13

Представлення вектор-рядків модулярного множення M^2

i	7	6	5	4	3	2	1	0
M_i	1	1	0	1	0	1	1	1
m_i	426	213	6880	3440	1720	860	430	215

Результат модулярного множення $215 \cdot 215 \, mod \, 1\,3547$ знаходиться згідно формули (2.2):

$$C = 215 \cdot 215 \, mod \, 1\,3547 = (215 + 430 + 860 + 3440 + 213 + \\ + 426) \, mod \, 1\,3547 = 5584.$$

Отже, операція модулярного множення замінюється операцією модулярного додавання тих m_i, для яких відповідні a_i рівні 1.

При розшифруванні криптограми *C*, згідно (2.12), визначаються додаткові допоміжні величини s, w і u:

$$s = 5584\, mod\, 31 = 4;\ w = 5584\, mod\, 23 = 18;\ u = 5584\, mod\, 19 = 17.$$

Відповідно (2.13), значення x, y і z шукаються з таких порівнянь: $x^2 \equiv 4(mod\, 31)$,$y^2 \equiv 18(mod\, 23)$, $z^2 \equiv 17(mod\, 19)$.

Отже, значення $\sqrt{4}\, mod\, 31 = \pm 2$ (-2mod31=29). Щоб знайти $\sqrt{18}\, mod\, 23$ згідно вище описаного методу необхідно виконати наступну послідовність дій: $18 + 23 = 41$, $18 + 46 = 64$. Оскільки 64 є повним квадратом, то $\sqrt{18}\, mod\, 23 = \pm 8$ (-8 mod 23=15). Аналогічним чином шукається $z \equiv \sqrt{17}(mod\, 19) = \pm 6$ (-6 mod 19=13).

Оскільки розв'язками трьох порівнянь є 6 значень, то, відповідно до (2.14), для розшифрування потрібно розв'язати вісім систем порівнянь, що утворюються як комбінації можливих варіантів пошуку відкритого повідомлення (*i*=1…8):

$$\begin{cases} M_i \equiv \pm 2 (\mathrm{mod}\, 31); \\ M_i \equiv \pm 8 (\mathrm{mod}\, 23); \\ M_i \equiv \pm 6 (\mathrm{mod}\, 19). \end{cases}$$

$$\begin{cases} M \equiv 29 (\mathrm{mod}\, 31); \\ M \equiv 8 (\mathrm{mod}\, 23); \\ M \equiv 6 (\mathrm{mod}\, 19). \end{cases} \qquad (2.15)$$

Розв'язок однієї з восьми систем буде шуканим значення *M*. В таблиці 2.14 наведено приклад розв'язання системи за допомогою додавання добутку модулів. Даний розв'язок визначає правильне розшифрування зашифрованого тексту.

Таблиця 2.14

Розв’язок системи конгруенцій, який визначає правильне розшифрування зашифрованого тексту

i	29+(i-1)×31	(29+(i -1)*31)mod23
1	29	6
2	60	14
3	91	22
4	122	7
5	153	15
6	184	0
7	215	8
$p_1 \times p_2$=713		
	215+(i-1)×713	(215+(i-1)×713)mod19
1	215	6

Отже, розшифрованим повідомленням є M=215. Аналогічно, без використання громіздких операцій та необхідності контролю переповнення розрядної сітки при виконанні проміжних обчислень, можна отримати розв’язки інших семи систем з (2.15), які не будуть відповідати правильному розшифруванню повідомлення.

2.3.2 Порівняння часових складностей класичного криптоалгоритму Рабіна на основі векторно-модульного та класичного методів

Для аналізу часової складності криптосистеми Рабіна слід враховувати найбільш ресурсоємні операції, які виконуються під час шифрування та розшифрування. Серед них найсуттєвіший внесок у загальну складність мають модулярне множення, обчислення квадратичних лишків та процедура відновлення числа за залишками.

Часова складність модулярного множення залежить від обраного алгоритму [237]:

- Класичний метод множення має квадратичну складність $O(l^2)$;

- Алгоритм Шонхаге-Штрассена забезпечує лінійно-логарифмічну складність $O(l\ log\ log l)$;

- Методи Карацуби $O(l^{1.585})$ та Тома-Кука $O(l^{1.465})$ дозволяють прискорити операції множення;

- Матрично-модульний метод характеризується складністю $O\left(\frac{l}{2} log_2\right)$.

Розшифрування в криптосистемі Рабіна базується на відновленні числа за залишками. У запропонованому методі [257] використовуються лише операції додавання та множення з часовими складностями $O(l)$ і $O(l^2)$. Для багатомодульних криптосистем, реалізованих на основі китайської теореми залишків (КТЗ), розв'язання вимагає k - разового обчислення обернених елементів за модулем зі складністю $O(l^3)$. Аналогічно, у класичному алгоритмі Гарнера присутні ті самі операції, що значно збільшують загальну обчислювальну складність.

Запропонований метод суттєво оптимізує криптосистему Рабіна, замінюючи витратні операції множення та модулярного експоненціювання на додавання. Це знижує часову складність з $O(2kl^3 + 2l^2k + lk)$ до $O1\left(l^2\left(\frac{k^2+k}{2}\right) + \frac{l}{2} log_2 l\right)$, $\left(\frac{k^2+k}{2}\right)$ – кількість операцій множення. Для прикладу, якщо $k = 3$, то складність становить $O\left(6l^2 + \frac{l}{2} log_2 l\right) \approx O(l^2)$. Загальна складність криптосистеми скорочується з $O(kl^3 + 2l^2k + lk + l^{1.465})$ до $O1(l^2\left(\frac{k^2+k}{2}\right) + \frac{l}{2} log_2 l)$.

Запропонований метод оптимізації криптосистеми Рабіна демонструє значне зменшення обчислювальної складності завдяки заміні ресурсозатратних операцій. Це не лише підвищує ефективність роботи алгоритму, але й робить його більш придатним для використання у системах із обмеженими обчислювальними ресурсами, таких як пристрої Інтернету речей (IoT). Графічний аналіз часових характеристик (рисунок 2.4)

підтверджує доцільність застосування запропонованого підходу для підвищення продуктивності криптографічних алгоритмів.

Ефективність запропонованого удосконалення реалізації криптосистеми Рабіна визначається як співвідношення часових складностей:

$$\mathrm{E}(l,k) = \frac{(2l^3 + 4l^2 + 2l + l^{1,465})}{l^2\left(\frac{k^2+k}{2}\right) + \frac{l\,log_2\,l}{2}} = \frac{4l^2 + 8l + 2l^{0,465} + 4}{l(k^2+k) + log_2\,l}$$

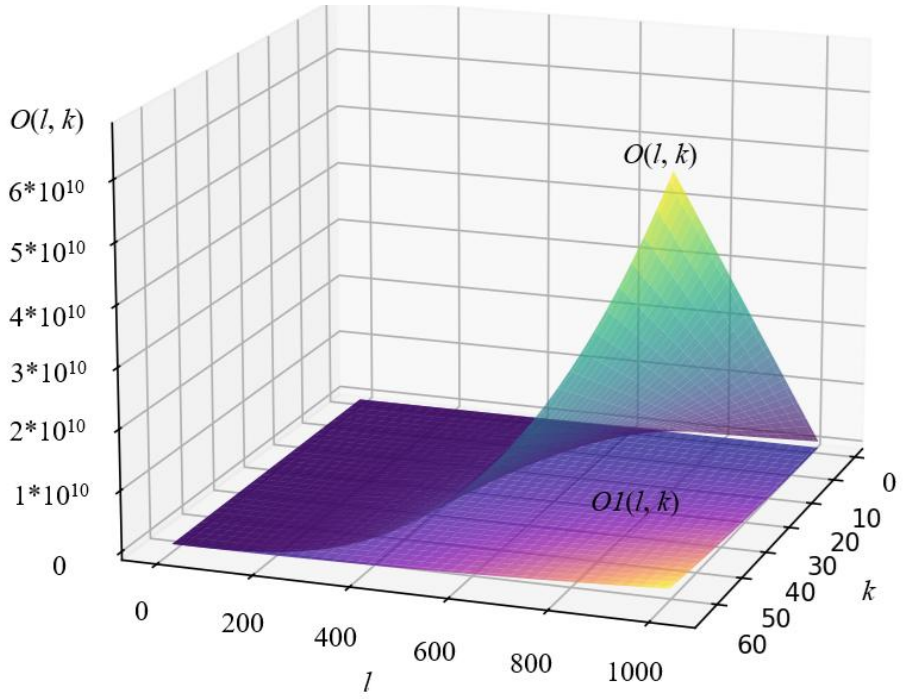


Рисунок 2.4 - Графічна залежність часової складності запропонованого методу від кількості модулів та їх розрядності

Результати проведених досліджень свідчать, що впровадження розробленого алгоритмічного забезпечення для тримодульного криптоалгоритму Рабіна, заснованого на операції додавання, дозволяє значно знизити часову складність базових операцій. Зокрема, при параметрах k=3 та l=256 складність зменшується у 128 разів, переходячи з кубічної до квадратичної. Такий підхід також сприяє спрощенню апаратної реалізації та зниженню структурної складності під час схемотехнічного проєктування

тримодульної криптосистеми Рабіна. Це досягається завдяки використанню однотипних суматорів замість помножувачів та мультиплексорів.

2.4 Удосконалення реалізації алгоритму шифрування Ель-Гамаля з використанням системи залишкових класів та векторно-модульного алгоритму модулярного експоненціювання

На сьогоднішній день для забезпечення високого рівня захисту інформаційних потоків використовують асиметричні криптоалгоритми RSA, Ель-Гамаля [258], Рабіна [253] з параметрами – ключі, блок шифрування та модуль криптоперетворення не менше 1024 біт з перспективою їх зростання в найближчі роки до 2048 та 4096 біт, що призводить зменшення часових характеристик. При шифруванні/дешифруванні основними операціями зазначених асиметричних алгоритмів є модулярне експоненціювання та піднесення до квадрату за модулем багаторозрядних чисел. Багатьма вченими розроблялися методи модулярного множення та експоненціювання, які мають певні функціональні обмеження, а саме використовують двійково-десяткову систему числення, яка характеризується високою обчислювальною складністю.

Слід відмітити, що існують набори модулів, які утворюють досконалу форму СЗК (базисні числа рівні 1) [259] та МДФ СЗК (базисні числа рівні ±1) [34], що суттєво зменшує часову складність переведення.

Тому постає задача підвищення швидкодії та зменшення складності базових операцій асиметричних криптоалгоритмів RSA, Ель-Гамаля на основі сумісного застосування СЗК та алгоритму векторно-модульного множення [73, 260].

На першому етапі відбувається генерування ключів, а саме:

1. Вибираються два простих число p і q, та випадкове ціле число x для якого справджується нерівність: $1<x<p$.

Обчислюється значення $y = q^x \, mod \, p$ на основі сумісного використання СЗК [261] і векторно-модульного алгоритму модулярного експоненціювання [64, 262], тобто згідно формули, що дозволяє розпаралелити процес на l – потоків, які відповідають кількості модулів:

$$y = q^x \, mod \, p = \left(\sum_{i=0}^{l-1} b_i B_i q_i\right) mod \, p, \tag{2.16}$$

де $q_i = (q \, mod \, p_i)^x \, mod \, p_i = \left(\left(\sum_{i=0}^{l-1} b_i B_i q_i\right) mod P\right) mod p$, $P = \prod_{i=1}^{k} p_i$, $B_i = \frac{p}{p_i}$, $b_i = B_i^{-1} \, mod \, p_i$, k – кількість модулів та потоків. Пошук значення $q_i = (q \, mod \, p_i)^x \, mod \, p_i$ здійснюється на основі використання векторно-модульного методу модулярного множення, представивши $x = \sum_{j=0}^{l-1} x_j \cdot 2^j$, де $x_j = 0{,}1$ і згідно формули обчислити q_i:

$$q_i = (q \, mod \, p_i)^{\sum_{j=0}^{l-1} x_j \cdot 2^j} \, mod \, p_i = \prod_{j=0}^{l-1} (q \, mod \, p_i)^{x_j \cdot 2^j} = \prod_{j=0}^{l-1} s_j \, mod \, p_i, \tag{2.17}$$

де $s_j = (q \, mod \, p_i)^{2^i} \, mod \, p_i = \left(s_{j-1}\right)^2 mod \, p_i$.

Тоді будь–який степінь x можна записати за степенями 2 і шуканий результат можна отримати, перемноживши відповідну кількість стовбців за допомогою таблиці 2.15.

Таблиця 2.15

Вектор піднесення до степеня в базисі Радемахера–Крестенсона

$x_{l\text{-}1}$		x_i	…	x_1	x_0
$s_{l\text{-}1}$		s_i	…	s_1	s_0
$q^{2^{l-1}} \, mod \, p_i$	…	$q^{2^i} \, mod \, p_i$	…	$q^{2^1} \, mod \, p_i$	$q^{2^0} \, mod \, p_i$

Для пошуку значення q_i перемножаються ці значення s_i для яких $x_j = 1$. Основними перевагами такого методу є здійснення операцій над числами

значно менших розмірів в порівнянні з класичним підходом, що дозволяє пришвидшити алгоритм модулярного експоненціювання.

При знаходженні значення $s_i s_{i-1} \, mod \, p_i$ представимо $s_{i-1} = \sum_{z=0}^{l-1} w_z \cdot 2^z$, де $w_z = 0,1$, n–розрядність модуля p_i. На основі використання векторно-модульного методу будуються два вектор-рядки, в першому з яких записуються елементи:

$$h_0 = 2^0 s_i \, mod \, p_i, \; h_i = 2 \cdot s_{i-1} \, mod \, p_i, \tag{2.18}$$

в другому w_i, як показано в таблиці 2.16.

Таблиця 2.16

Представлення вектор-рядків модульного множення

h_{l-1}	…	h_i	…	h_1	h_0
w_{l-1}		w_i	…	w_1	w_0

Результатом модулярного множення двох l – розрядних чисел знаходиться згідно формули:

$$s_i s_{i-1} \, mod \, p_i = \left(\sum_{i=0}^{l-1} w_i \cdot h_i \right) mod \, p_i. \tag{2.19}$$

Розроблений метод характеризується меншою часовою складністю порівняно з класичними. Отже, відкритим ключем буде (p, q, y), а закритим буде x.

2. Шифрування. Вибирається випадкове ціле число v таке, що $1<v<p-1$. Обчислюється значення $a = q^v \, mod \, p$. Для цього скористаємося СЗК та векторно-модульним алгоритмом модулярного експоненціювання на основі формули:

$$y = q^v \, mod \, p = (\sum_{i=1}^{v} b_i B_i q_i) \, mod \, p, \quad (2.20)$$

де $q_i = (q \, mod \, p_i)^v \, mod \, p_i = ((\sum_{i=1}^{v} b_i B_i q_i) \, mod \, P) mod p_i$, $P = \prod_{i=1}^{k} p_i$, $B_i = \frac{p}{p_i}$, $b_i = B_i^{-1} \, mod \, p_i$, k – кількість модулів та потоків.

Пошук значення $q_i = (q \, mod \, p_i)^v \, mod \, p_i$ здійснюється на основі використання векторно-модульного методу модулярного множення, представивши $v = \sum_{j=0}^{l-1} v_j \cdot 2^j$, де $v_j = 0,1$ і згідно формули обчислити q_i:

$$q_i = (q \, mod \, p_i)^{\sum_{j=0}^{l-1} v_j \cdot 2^j} \, mod \, p_i = \prod_{j=0}^{l-1} (q \, mod \, p_i)^{v_j \cdot 2^j} = \prod_{j=0}^{l-1} y_j \, mod \, p_i. \quad (2.21)$$

де $y_i = q^{2^i} \, mod \, p_i = (y_{i-1})^2 \, mod \, p_i$.

Тоді будь–який степінь v можна записати за степенями 2 і шуканий результат можна отримати, перемноживши відповідну кількість стовбців за допомогою таблиці 2.17.

Таблиця 2.17

Вектор піднесення до степеня в базисі Радемахера–Крестенсона

v_{l-1}		v_i	…	v_1	v_0
y_{l-1}		y_i	…	y_1	y_0
$q^{2^{l-1}} \, mod \, p_i$	…	$q^{2^i} \, mod \, p_i$	…	$q^{2^1} \, mod \, p_i$	$q^{2^0} \, mod \, p_i$

Для пошуку значення q_i перемножаються ці значення y_i для яких $v_j = 1$. При знаходженні значення $y_i y_{i-1} \, mod \, p_i$ представимо $y_{i-1} = \sum_{m=0}^{l-1} a_m \cdot 2^m$, де $a_m = 0,1$, n–розрядність модуля p_i. На основі використання векторно-модульного методу будуються два вектор-рядки, в першому з яких записуються елементи:

$$r_0 = 2^0 y_i \, mod \, p, r_i = 2 \cdot y_{i-1} \, mod \, p, \quad (2.22)$$

в другому a_m, як показано в таблиці 2.18.

Таблиця 2.18

Представлення вектор-рядків модульного множення

$r_{l\text{-}1}$	…	r_i	…	r_1	r_0
$a_{l\text{-}1}$		a_i	…	a_1	a_0

Результат модулярного множення двох l – розрядних чисел знаходиться згідно формули:

$$y_i y_{i-1} \, mod \, p_i = \left(\sum_{i=0}^{l-1} a_i \cdot r_i\right) mod \, p_i, \tag{2.23}$$

Розроблений метод характеризується меншою часовою складністю порівняно з класичними за рахунок заміни операції множення операцією додавання.

Для отримання шифротексту для повідомлення M знаходиться значення $b = y^v \cdot M \, mod \, p$.

Спочатку обчислюється значення $\alpha = y^v \, mod \, p$ на основі використання векторно-модульного алгоритму модулярного експоненціювання, тобто згідно формули:

$$\alpha = y^v \, mod \, p = \left(\sum_{i=0}^{k-1} b_i B_i y_i\right) mod \, p, \tag{2.24}$$

де $y_i = (y \, mod \, p_i)^v \, mod \, p_i = \left(\left(\sum_{i=0}^{k-1} b_i B_i q_i\right) mod \, P\right) mod p_i$, $P = \prod_{i=1}^{k} p_i$, $B_i = \frac{p}{p_i}$, $b_i = B_i^{-1} \, mod \, p_i$, k – кількість модулів та потоків.

Пошук значення $y_i = (y \, mod \, p_i)^v \, mod \, p_i$ здійснюється на основі використання векторно-модульного методу модулярного множення, представивши $v = \sum_{j=0}^{l-1} v_j \cdot 2^j$, де $v_j = 0{,}1$ і згідно формули обчислити y_i:

$$y_i = (y \bmod p_i)^{\sum_{j=0}^{l-1} k_j \cdot 2^j} \bmod p_i = \prod_{j=1}^{l-1} (y \bmod p_i)^{k_j \cdot 2^j} = \prod_{j=0}^{l-1} \beta_j \bmod p_i. \quad (2.25)$$

де $\beta_i = y^{2^i} \bmod p_i = (\beta_{i-1})^2 \bmod p_i$.

Тоді будь–який степінь k можна записати за степенями 2 і шуканий результат можна отримати, перемноживши відповідну кількість стовбців за допомогою таблиці 2.19.

Таблиця 2.19

Вектор піднесення до степеня в базисі Радемахера–Крестенсона

v_{l-1}		v_i	…	v_1	v_0
β_{l-1}		β_i	…	β_1	β_0
$y^{2^{n-1}} \bmod p_i$	…	$y^{2^i} \bmod p_i$	…	$y^{2^1} \bmod p_i$	$y^{2^0} \bmod p_i$

Для пошуку значення y_i перемножаються ці значення β_i для яких $k_j = 1$. При знаходженні значення $\beta_i \beta_{i-1} \bmod p_i$ представимо $\beta_{i-1} = \sum_{g=0}^{l-1} f_g \cdot 2^g$, де $f_g = 0,1$, l–розрядність модуля p_i. На основі використання векторно-модульного методу будуються два вектор-рядки, в першому з яких записуються елементи:

$$c_0 = 2^0 \beta_i \bmod p_i, c_i = 2 \cdot \beta_{i-1} \bmod p_i, \quad (2.26)$$

в другому f_g, як показано в таблиці 2.20.

Таблиця 2.20

Представлення вектор-рядків модульного множення

c_{l-1}	…	c_i	…	c_1	c_0

f_{l-1}		f_i	…	f_1	f_0

Результат модулярного множення знаходиться згідно формули:

$$\beta_i\beta_{i-1}\, mod\, p_i = \left(\sum_{i=0}^{l-1} c_i \cdot f_i\right) mod\, p_i. \quad (2.27)$$

На наступному кроці знайдемо значення $b = y^v \cdot M\, mod\, p = \alpha \cdot M\, mod\, p$ на основі векторно-модульного методу модулярного множення. При знаходженні значення $\alpha \cdot M\, mod\, p$ представимо $M = \sum_{k=0}^{l-1} M_k \cdot 2^k$, де $\alpha_j, M_k = 0{,}1$, n–розрядність модуля p. Будуються два вектор-рядки, в першому з яких записуються елементи:

$$\delta_0 = 2^0\alpha\, mod\, p_i,\ \delta_i = 2 \cdot \delta_{i-1}\, mod\, p_i, \quad (2.28)$$

в другому M_i, як показано в таблиці 2.21.

Таблиця 2.21

Представлення вектор-рядків модульного множення

δ_{l-1}	…	δ_i	…	δ_1	δ_0
M_{l-1}		M_i	…	M_1	M_0

Результат модулярного множення знаходиться зі співвідношення:

$$b = \alpha \cdot M\, mod\, p = \left(\sum_{i=0}^{l-1} \delta_i \cdot M_i\right) mod\, p. \quad (2.29)$$

В результаті проведених обчислень, отримується пара (a,b), яка є шифротекстом.

3. Розшифрування відбувається згідно формули $M = b \cdot (a^x)^{-1}\, mod\, p$. Спочатку знаходиться значення $\varepsilon = a^x\, mod\, p$, для цього

використовується векторно-модульний метод модулярного експоненціювання:

$$\varepsilon = a^x \bmod p = \left(\sum_{i=1}^{k} b_i B_i a_i\right) \bmod p, \quad (2.30)$$

де $a_i = (a \bmod p_i)^v \bmod p_i$, $p = \prod_{i=1}^{l} p_i$, $B_i = \frac{p}{p_i}$, $b_i = B_i^{-1} \bmod p_i$, k – кількість модулів та потоків.

Пошук значення $a_i = (a \bmod p_i)^x \bmod p_i$ здійснюється на основі використання векторно-модульного методу модулярного множення, представивши $x = \sum_{j=0}^{l-1} x_j \cdot 2^j$, де $x_j = 0{,}1$ і згідно формули обчислити a_i:

$$a_i = (a \bmod p_i)^{\sum_{j=0}^{l-1} x_j \cdot 2^j} \bmod p_i = \prod_{j=1}^{l-1} (a \bmod p_i)^{x_j \cdot 2^j} = \prod_{j=0}^{l-1} \varphi_j \bmod p_i. \quad (2.31)$$

де $\varphi_i = a^{2^i} \bmod p_i$, при чому $\varphi_i = (\varphi_{i-1})^2 \bmod p_i$.

Тоді будь–який степінь x можна записати за степенями 2 і шуканий результат можна отримати, перемноживши відповідну кількість стовбців за допомогою таблиці 2.22. Для пошуку значення a_i перемножаються ці значення φ_i для яких $x_j = 1$. При обчисленні $\varphi_i \varphi_{i-1} \bmod p_i$ представимо $\varphi_{i-1} = \sum_{j=0}^{l-1} \phi_j \cdot 2^j$, де $\phi_j = 0{,}1$, n–розрядність модуля p_i.

Таблиця 2.22

Вектор піднесення до степеня в базисі Радемахера–Крестенсона

x_{l-1}		x_i	…	x_1	x_0
φ_{l-1}		φ_i	…	φ_1	φ_0
$a^{2^{l-1}} \bmod p_i$	…	$a^{2^i} \bmod p_i$	…	$a^{2^1} \bmod p_i$	$a^{2^0} \bmod p_i$

На основі використання векторно-модульного методу будуються два вектор-рядки, в першому з яких записуються елементи:

$$g_0 = 2^0\varphi_i \, mod \, p_\text{i}, \, g_i = 2 \cdot g_{i-1} \, mod \, p_i, \tag{2.32}$$

в другому φ_j, як показано в таблиці 2.23.

Таблиця 2.23

Представлення вектор-рядків модульного множення

g_{l-1}	$\cdots$	g_i	$\cdots$	g_1	g_0
φ_{l-1}		ϕ_i	$\cdots$	φ_1	φ_0

Результат модулярного множення обчислюється згідно формули:

$$\varphi_i\varphi_{i-1} \, mod \, p_i = \left(\sum_{i=0}^{l-1} g_i \cdot \phi_i\right) mod \, p_i. \tag{2.33}$$

В подальшому необхідно знайти $\eta = \varepsilon^{-1} \, mod \, p$ на основі методу з додаванням залишку, що дозволить зменшити часову складність в порівнянні з класичними методами, які грунтуються на розширеному алгоритмі Евкліда, методі повного перебору та з використанням функції Ейлера.

Спочатку обчислюється $\eta_0 = p \, mod \, \varepsilon \neq 0$, після чого послідовно виконується операція додавання: $\eta_1 = (\eta_0 + 1) \, mod \, \varepsilon$, $\eta_2 = (\eta_1 + \eta_0) \, mod \, \varepsilon = (2\eta_0 + 1) \, mod \, \varepsilon, \ldots, \eta_i = (\eta_{i-1} + \eta_0) \, mod \, \varepsilon = (i\eta_0 + 1) \, mod \, \varepsilon.$

Описана процедура продовжується до тих пір, поки деяке число η_i не стане рівним нулю. Тоді обернений елемент визначається за формулою:

$$\eta = \varepsilon^{-1} \, mod \, p = \frac{i \cdot p + 1}{\varepsilon}. \tag{2.34}$$

Векторно-модульний метод модулярного множення доволить знайти значення $M = b \cdot \eta \, mod \, p$, що і є результатом розшифрування. Записуємо та $\eta = \sum_{i=0}^{l-1} \eta_i \cdot 2^i$, де $b_j, \eta_i = 0{,}1$, l –розрядність модуля p. На основі

використання векторно-модульного методу будуються два вектор-рядки, в першому з яких записуються елементи:

$$\lambda_0 = 2^0 b \, mod \, p, \lambda_i = 2 \cdot \lambda_{i-1} \, mod \, p, \quad (2.35)$$

в другому η_i, як показано в таблиці 2.24.

Таблиця 2.24

Представлення вектор-рядків модульного множення

λ_{l-1}	…	λ_i	…	λ_1	λ_0
η_{l-1}		η_i	…	η_1	η_0

Результат модулярного множення $M = b \cdot \varphi \, mod \, p$ знаходиться згідно співвідношення:

$$M = b \cdot \eta \, mod \, p = \left(\sum_{i=0}^{l-1} \lambda_i \cdot \eta_i\right) mod \, p. \quad (2.36)$$

Отже, запропонований підхід шифрування/розшифрування асиметричної криптосистеми Ель-Гамаля дозволяє розпаралелити процес та проводити обчислення над числами меншої розрядності в порівнянні з класичним підходом, що, в свою чергу, призводить до зменшення часової складності базових операцій: модулярного множення та експоненціювання, пошуку оберненого елемента за модулем.

2.4.1 Порівняння часових складностей класичного криптоалгоритму Ель-Гамаля та на основі векторно-модульного методу модулярного експоненціювання

При аналізі часової складності криптографічного алгоритму Ель-Гамаля необхідно враховувати найбільш трудомісткі операції в процесі шифрування/розшифрування, а саме: модулярне множення, модулярне

експоненціювання і пошуку оберненого елемента за модулем. В таблиці 2.25 представлено складності базових операції класичного криптоалгоритму Ель-Гамаля на основі векторно-модульного методу модулярного експоненціювання.

Таблиця 2.25

Складність базових операції класичного криптоалгоритму Ель-Гамаля на основі векторно-модульного методу модулярного експоненціювання

№ п\п	Базові операції	Класичний криптоалгоритм Ель-Гамаля	Криптоалгоритму Ель-Гамаля на основі векторно-модульного методу модулярного експоненціювання
1.	Модулярне множення	Алгоритм Шонхаге-Штрассена з лінійно-логарифмічною складністю $O(l\,log\,logl)$; методи Карацуби $O(l^{1.585})$ та Тома-Кука $O(l^{1.465})$ [239]	Векторно-модульний метод $O\left(\frac{l}{2} log\, l\right)$.
2.	Обернений елемент за модулем	На основі розширеного алгоритму Евкліда $O(l^3)$ [239]	На основі додавання залишку $O\left(\frac{l * log_2 l}{2} + log_2 l * log_2\left(\frac{l}{2}\right)\right)$ [263, 264]
3.	Модулярне експоненціювання	Алгоритм Шонхаге-Штрассена: $O(l^2\,(\,loglogl)^2)$, метод Карацуби $O(l^{3.17})$,а Тома-Кука $O(l^{2.93})$ [266-269]	Векторно-модульний метод модулярного експоненціювання $O((l^2/4)\,(\,logl)^2)$ [265]

На першому етапі застосування криптосистеми Ель-Гамаля відбувається генерування ключів, при цьому обчислюється значення $y = q^x\, mod\, p$ з

часовою складністю $O((l^2/4)\,(log l)^2)$ в запропонованому підході з використанням вектоно-модульного методу і в класичному на основі алгоритму Шонхаге-Штрассена $O(l^2\,(loglogl)^2)$.

На етапі шифрування знаходяться значення $a = q^v \bmod p$ і $b = y^v \cdot M \bmod p$ з базовими операціями модулярного множення та експоненціювання, загальна часова складність яких становить: $O\left((l^2/2)\,(log_2 l)^2 + \frac{l}{2} log_2\right)$ – запропонованим підходом і $O(2(l^2\,(loglogl)^2) + l\,log\,logl)$.

Розшифрування в криптосистемі Ель-Гамаля базується на операціях модулярного множення та експоненціювання, пошуку оберненого елемента за модулем: $M = b \cdot (a^x)^{-1} \bmod p$. Часова складність відновлення вхідного повідомлення складатиме: $O\left((l^2/4)\,(log_2 l)^2 + \frac{l * log_2 l}{2} + log_2 l * log_2\left(\frac{l}{2}\right) + \frac{l}{2} log_2 l\right)$ – запропонованим методом, $O(l^2\,(loglogl)^2 + l\,log\,logl + l^3)$ – класичним підходом.

З врахуванням усіх етапів шифрування/розшифрування криптосистеми Ель-Гамаля загальна часова складність запропонованим підходом з використанням векторно-модульного методу модулярного множення та експоненціювання знижується в порівнянні з класичним методом з $O(l^3 + 4l^2\,(loglogl)^2 + 2l\,log\,logl)$ до $O\left(l^2\,(log_2 l)^2 + log_2 l * log_2\left(\frac{l}{2}\right) + \frac{3}{2} l log_2 l\right)$.

Результати проведених досліджень свідчать, що використання векторно-модульних методів модулярного множення та експоненціювання, дозволяє значно знизити часову складність базових операцій. Зокрема, при параметрах l=256 складність зменшується у 8 разів, що дає змогу спростити апаратну реалізацію та знизити структурну складність під час схемотехнічного проєктування криптосистеми Ель-Гамаля. Це досягається завдяки використанню однотипних суматорів замість помножувачів та мультиплексорів.

На рисунку 2.5 представлено графіки залежностей часових складностей запропонованого підходу та класичного від розмірності вхідних параметрів.

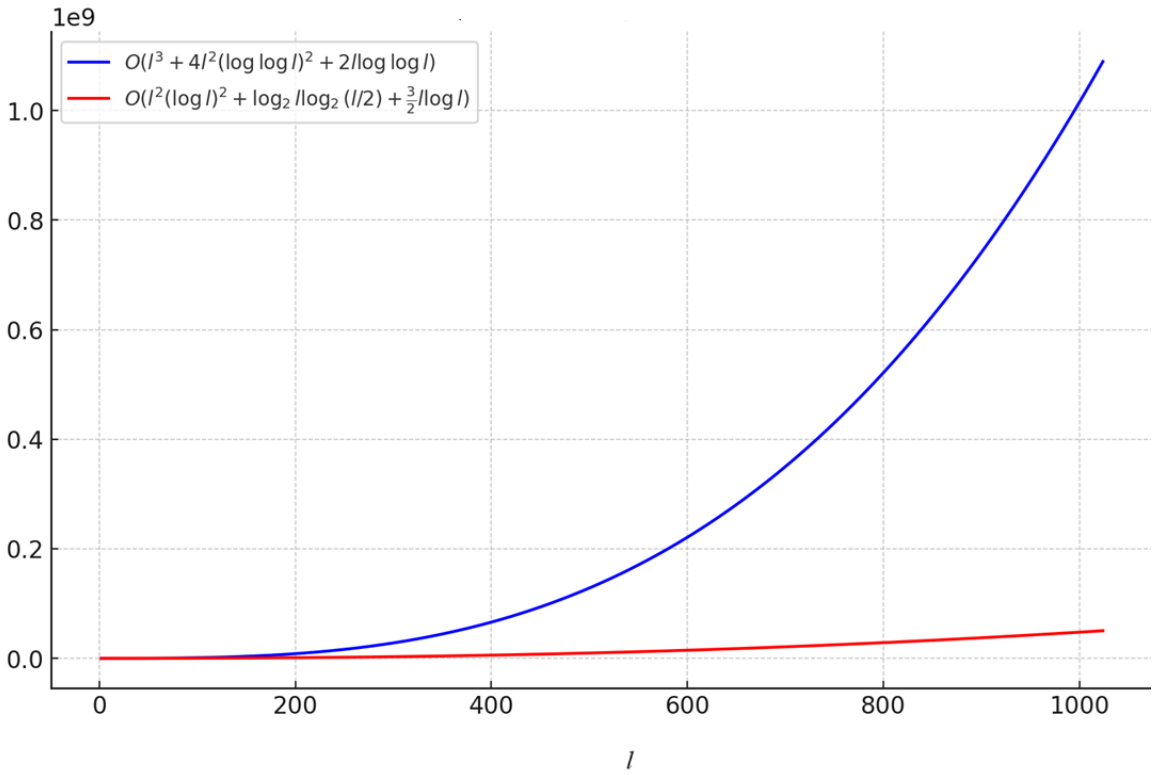


Рисунок 2.5 - Залежність часових складностей запропонованого підходу та класичного від розмірності вхідних параметрів

Ефективність обчислюється як співвідношення часових складностей класичного і запропонованого підходу:

$$\mathrm{E}(l) = \frac{(l^3+4l^2\,(loglogl)^2+2l\,log\,logl)}{l^2\,(log_2 l)^2+log_2 l*log_2\left(\frac{l}{2}\right)+\frac{3}{2}llog_2 l} \tag{2.37}$$

В результаті проведених досліджень слід відмітити, що при малих значеннях обидві складності мають близькі значення, але зростання значно прискорюється при збільшенні розмірності вхідних параметрів. Крім того ефективність запропонованого підходу збільшується у 8 разів при вхідних параметрах 256 біт. Отримані результати підтверджують, що в довгостроковій перспективі класичний алгоритм Ель-Гамаля менш ефективний у порівнянні з запропонованим підходом щодо реалізації з векторно-модульних методів модулярного множення і експоненціювання. Таким чином, для великих

розмірностей задачі вибір алгоритму з нижчою асимптотичною складністю може суттєво зменшити обчислювальні витрати.

Висновки до другого розділу

1. Запропоновано теоретичні основи визначення простих та взаємно простих чисел на основі властивості періодичності, що дає змогу ефективно вирішувати задачі оптимізації обчислень пошуку такого роду чисел. Встановлено умови подільності чисел виду $2^n + k$ згідно модульних операцій по простих модулях. Наведено приклад застосування розробленого методу для визначення подільності чисел виду 2^n+3 на прості числа в діапазоні від 3 до 147.

2. Розроблено теоретичні основи для реалізації криптоалгоритму Рабіна за допомогою використання тільки операції додавання. Це дозволяє збільшити швидкодію процесу шифрування/дешифрування інформаційних потоків шляхом уникнення виконання обчислювально громіздких операцій (множення, пошуку квадратного кореня за модулем, пошуку оберненого елемента тощо) та призводить до підвищення стійкості та зменшення часової складності криптосистеми Рабіна.

3. Розроблено алгоритмічне забезпечення для реалізації трьохмодульної криптосистеми Рабіна з використанням тільки операції додавання, що дозволлило пришвидшити процес шифрування/розшифровування інформаційних потоків шляхом заміни операції множення операцією додавання та уникнути процедури пошуку мультиплікативного оберненого елемента за модулем. Здійснено аналітичне порівняння часових складностей запропонованого та відомого підходів. Показано, що використання розробленого методу дозволяє зменшити часову складність з кубічної до квадратичної.

4. Удосконалено реалізацію асиметричного алгоритму шифрування Ель-Гамаля, а саме базові операції – модулярного експоненціювання багаторозрядних чисел, на основі СЗК та векторно-модульного алгоритму

модулярного множення, що дозволило суттєво зменшити часову складність та підвищити ефективність виконання процесу шифрування/дешифрування.

Основні результати другого розділу відображені у роботах [51, 67, 73, 183, 223, 233, 236, 240-241, 251, 254, 257-260, 261-265].

РОЗДІЛ 3. ТЕОРЕТИЧНІ ОСНОВИ СИМЕТРИЧНИХ ТА АСИМЕТРИЧНИХ КРИПТОАЛГОРИТМІВ СИСТЕМІ ЗАЛИШКОВИХ КЛАСІВ

3.1 Побудова наборів модулів системи залишкових класів однакової розрядності для застосування в криптографії

Можливість застосування СЗК в обчислювальних алгоритмах обумовлюється наявністю певного ізоморфізму між математичними операціями над цілими числами та відповідними операціями над системою цілих невід'ємних залишків по окремих модулях. Причому додавання, множення та піднесення до цілого додатного степеня будь–яких цілих невід'ємних чисел повністю ідентичні відповідним операціям, що виконуються над системою залишків [106, 270].

Очевидно, що одним із шляхів підвищення швидкодії обчислювачів, які працюють у СЗК, є вибір спеціалізованих наборів модулів. В переважній більшості робіт розглядаються модулі виду 2^m, $2^m \pm 1$, що дає можливість раціонального використання регістрів розрядної сітки [146, 259]. Крім того, в СЗК запропоновано багато різних наборів модулів різного виду та різної кількості для визначених застосувань, які істотно впливають на всі частини апаратної реалізації [271]. Зокрема, в [84] представлено безпечні та ефективні підходи для застосування СЗК в криптографії на еліптичних кривих. У [272] розроблено ефективні алгоритми реалізації RSA-криптографічної системи на основі СЗК, експериментальні дослідження яких показали, що вони володіють більшою швидкодією та стійкістю до атак грубої сили в порівнянні з класичним. В роботі [273] розроблено методи швидкого виконання арифметичних операцій додавання, множення та піднесення до степеня за модулем в модулярній системі числення при реалізації криптографічних перетворень, в яких показано істотне зменшення часу виконання основних базових операцій криптоалгоритмів.

Але в усіх розглянутих підходах виникає необхідність обчислення оберненого елемента та множення на нього, що істотно зменшує швидкість відновлення десяткового числа із СЗК [246, 247, 274]. Це є, поряд з труднощами при виконанні ділення та порівняння чисел, основним недоліком, який стримував розвиток СЗК. Одним із шляхів для спрощення цієї процедури є застосування різних форм СЗК. Зокрема, у [162] розроблено та досліджено методи побудови ДФ СЗК, у якій система модулів p_i вибирається таким чином, що значення всіх коефіцієнтів m_i=1. Це дозволяє усунути громіздку процедуру пошуку мультиплікативного оберненого елемента за модулем та множення на нього при використанні КТЗ. Однак ДФ СЗК має обмежене застосування, оскільки розрядності відповідних їй модулів, і відповідно залишків, суттєво відрізняються, що призводить до нераціонального використання регістрів розрядної сітки.

В роботі [248, 275] була розроблена модифікована МДФ СЗК, у якій виконується умова:

$$M_i \, mod \, p_i=1,\ p_i\text{-}1 \quad \text{або} \quad M_i \, mod \, p_i=\pm 1. \tag{3.1}$$

В цьому випадку коефіцієнти m_i=±1 і відповідно при відновленні числа по його залишках згідно КТЗ сума стає знакозмінною. Крім усунення вказаного недоліку ДФ СЗК, це істотно зменшує величину суми $\sum_{i=1}^{k} b_i M_i m_i$, яка в багатьох випадках не буде перевищувати значення діапазону обчислень. Це дозволить спростити пошук залишку за модулем P.

Однак на даний час відсутні методи побудови систем модулів, які задовольняють умовам МДФ СЗК і мають однакову розрядність, що дозволить раціонально використовувати регістри розрядної сітки при вирішенні задач асиметричної криптографії та завадостійкого кодування [276, 277].

Математичні розрахунки, аналогічні проведеним у [278], дозволяють звести (3.1) до такого виразу:

$$\sum_{i=1}^{k} \frac{1}{p_i} = \gamma \pm \frac{1}{\prod_{i=1}^{z} p_i}, \quad (3.2)$$

де γ=0, ±1, ±2, ±3, ...

Для спрощення коефіцієнт γ, на відміну від ДФ СЗК, можна вибрати рівним 0, що при заданій кількості модулів відповідає найбільшому значенню P. Тоді останню рівність можна подати у такому вигляді:

$$\sum_{i=1}^{k} M_i = \pm 1. \quad (3.3)$$

Нехай невідомими будуть два останні модулі p_{k-1} та p_k. Тоді (3.3) можна представити у такому вигляді:

$$p_{k-1}p_k(p_2p_3 \dots p_{k-2} + p_1p_3 \dots p_{k-2} + \dots + p_1p_2 \dots p_{k-3}) \quad + p_1p_2 \dots p_{k-2}(p_{k-1} + p_k) = \pm 1. \quad (3.4)$$

Вводиться позначення:

$$p_{k-1,k} = \frac{a,b - p_1p_2 \dots p_{k-2}}{p_2p_3 \dots p_{k-2} + p_1p_3 \dots p_{k-2} + \dots + p_1p_2 \dots p_{k-3}}. \quad (3.5)$$

Підставивши (3.5) в (3.4), після відповідних математичних перетворень будемо мати умову, яка повинна виконуватися для визначення набору будь-якої кількості модулів МДФ СЗК, два з яких невідомі:

$$\pm(p_2p_3 \dots p_{k-2} + p_1p_3 \dots p_{k-2} + \dots + p_1p_2 \dots p_{k-3}) + (p_1p_2p_{k-2})^2 = ab. \quad (3.6)$$

Отже, ліву частину (3.6) потрібно факторизувати, на основі чого визначаються параметри a та b. Крім того, модулі p_k та p_{k-1} мають бути цілими числами, тобто:

$$(a, b - p_1p_2 \dots p_{k-2}) \, mod(|p_2p_3 \dots p_{k-2} + p_1p_3 \dots p_{k-2} + \dots + p_1p_2 \dots p_{k-3}|) = 0. \quad (3.7)$$

Вирази (3.6) та (3.7) визначають умови для знаходження будь-якої кількості модулів МДФ СЗК, два з яких невідомі.

Для прикладу розглянемо МДФ СЗК, яка складається з чотирьох модулів. Умови (3.5) - (3.7) набудуть такого вигляду:

$$p_{3,4} = \frac{a,b-p_1p_2}{p_1+p_2}; \pm(p_1 + p_2) + (p_1p_2)^2 = ab; \left(a, b - p_{1,2}\right) mod(|p_1 + p_2|) = 0. \quad (3.8)$$

Нехай розрядність модулів становить 4 біти і p_1=8, p_2=-9. Тоді з (3.8) отримаємо: $p_{3,4} = -(a, b + 72)$ і $ab = \pm 1 + 5184 = \begin{cases} 5183 = 71 \cdot 73 \\ 5185 = 5 \cdot 17 \cdot 61 \end{cases}$. Усі можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=8, p_2=-9 та діапазона обчислень представлені в таблиці 3.1.

Таблиця 3.1

Можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=8, p_2=-9 та діапазона обчислень

№	p_1, p_2	ab	a	b	p_3	p_4	P
1	8, -9	5185	1	5185	-73	-5257	27630792
2			-1	-5185	-71	5113	26137656
3			5	1037	-77	-1109	6148296
4			-5	-1037	-67	965	4655160
5			17	305	-89	-377	2415816
6			-17	-305	-55	233	922680
7			61	85	-133	-157	1503432
8			-61	-85	-11	13	10296
9		5183	1	5183	-73	-5255	27620280
10			-1	-5183	-71	5111	26127432
11			71	73	-143	-145	1492920
12			-71	-73	-1	1	72

Слід зазначити, що дана таблиця вимагає уточнення знаків модулів згідно виразу (3.1). Отже, система з чотирьох 4-бітних модулів однакової розрядності, яка задовольняє умовам МДФ СЗК, матиме такий вигляд: -8, 9, 11, -13.

3.1.1 Метод повного перебору

Для використання методу повного перебору доцільно вибрати формулу (3.3). Для чотирьох модулів вона набуде такого вигляду:

$$p_1p_2p_3 + p_1p_2p_4 + p_1p_3p_4 + p_2p_3p_4 = \pm 1. \quad (3.9)$$

Цілочисельні значення параметрів p_2, p_3 та p_4 утворюють систему модулів, що задовольняють умовам МДФ СЗК.

В таблиці 3.2 представлено можливі варіанти наборів з чотирьох 8-бітних модулів для побудови МДФ СЗК

Таблиця 3.2

Можливі варіанти наборів з чотирьох 8-бітних модулів для побудови МДФ СЗК

№	p_1	p_2	p_3	p_4
1	-131	134	151	-155
2	-134	141	143	-151
3	-151	178	203	-255
4	-169	177	179	-188
5	-197	199	241	-244
6	-208	217	219	-229

Очевидно, що даний метод непридатний для пошуку великої кількості модулів великої розрядності, оскільки вимагає значних часових затрат.

3.1.2 Метод заміни

Для чотирьохмодульної МДФ СЗК після заміни p_2, p_3, p_4=a, b, c-p_1, та відповідних математичних перетворень формула (3.9) перетворюється до такого виразу:

$$abc - p_1^2(a + b + c) = \pm 1 - 2p_1^2. \tag{3.10}$$

Зворотня заміна дозволяє повернутися до початкових параметрів, а ділення лівої та правої частини на p_1^2, дозволяє отримати умови, які повинні виконуватися для побудови МДФ СЗК:

$$(p_2 + p_1)(p_3 + p_1)(p_4 + p_1)\, mod\, p_1^2 = \pm 1, \tag{3.11}$$
$$\frac{(p_2 + p_1)(p_3 + p_1)(p_4 + p_1) \pm 1}{p_1^2} = p_2 + p_3 + p_4 + p_1.$$

Перший вираз (3.11) вказує, що $(p_2 + p_1)(p_3 + p_1)(p_4 + p_1) = k \cdot p_1^2 \pm 1$, де k=±1, ±2, … . Це означає, що $k \cdot p_1^2 \pm 1$ має розкладатися як мінімум на два множники (тоді третій буде ±1). Числові розрахунки показують, що у переважній більшості випадків для виконання умов (3.11) потрібно використовувати рівність:

$$(p_2 + p_1)(p_3 + p_1)(p_4 + p_1) = -p_1^2 + 1. \tag{3.12}$$

В таблиці 3.3 представлено процедуру пошуку модулів однакової розрядності (5-7 біт) для побудови МДФ СЗК.

Таблиця 3.3

Процедура пошуку модулів однакової розрядності (5-7 біт) для побудови МДФ СЗК.

n, біт	p_1	$-p_1^2+1$	Факторизація	p_2+p_1	p_3+p_1	p_4+p_1	p_2	p_3	p_4
5	-19	-360	$-2^3\cdot3^2\cdot5$	2	4	-45	21	23	-26
6	-34	-1155	$-3\cdot5\cdot7\cdot11$	3	5	-77	37	39	-43
7	-76	-5775	$-3\cdot5^2\cdot7\cdot11$	5	7	-165	81	83	-89
	-103	-10608	$-2^4\cdot3\cdot13\cdot17$	6	8	-221	109	111	-118

З таблиць 3.1-3.3 слідує, що при побудові чотирьохмодульних МДФ СЗК, які зручно використовувати в асиметричних криптосистемах та в задачах завадостійкого кодування, для усіх модулів однакової розрядності виконуються такі співвідношення: p_1, p_4 <0, p_2, p_3 >0.

3.2 Криптосистема на основі звичайної цілочисельної СЗК

Перспективним кроком у розвитку криптографії є розробка методів шифрування у СЗК, зокрема у її МДФ, які забезпечують можливість розпаралелення обчислень і, як наслідок, підвищення швидкодії.

У цьому розділі буде детально розглянуто симетричні методи шифрування, побудовані на основі СЗК, їх переваги, особливості реалізації, а також аналіз їхньої криптостійкості, що є важливим етапом для розробки ефективних і захищених криптосистем нового покоління.

3.2.1 Теоретичні основи СЗК та правила кодування тексту

Відомо [106, 279], що будь-яке десяткове число N в СЗК представляється у вигляді невід'ємних залишків b_i (i-номер модуля) від ділення N на кожен із системи натуральних попарно взаємно простих модулів p_i:

$$b_i = N \bmod p_i. \quad (3.13)$$

Відновлення числа N із СЗК в десяткову систему числення в переважній більшості випадків відбувається згідно китайської теореми про залишки (КТЗ):

$$N = \left(\sum_{i=1}^{k} b_i M_i m_i\right) mod\, P, \quad (3.14)$$

де $P = \prod_{i=1}^{k} p_i$, $M_i = \frac{P}{p_i}$, m_i шукається з виразу $m_i = M_i^{-1} mod\, p_i = 1$, k – кількість модулів. При цьому повинна виконуватися нерівність $N<P$. Спростити відновлення десяткового числа за його залишками дозволяє використання МДФ СЗК, в якій модулі підібрані таким чином, що $m_i=\pm 1$ [165, 278]. Такі особливості СЗК та КТЗ зумовлюють їх використання для побудови симетричних криптосистем. В першу чергу потрібно визначитися з правилами перекодування текстової інформації в числову. Для прикладу, це може бути класичний варіант, коли букві відповідає число, що є її порядковим номером в алфавіті (крім того, ще може бути впорядкування букв по частоті, з якою вони зустрічаються в досить довгих текстах, використання ключових слів, подібно до шифруючих таблиць Трисемуса, довільна нумерація букв тощо). Нехай для англійської абетки, не враховуючи великих та малих символів, буде встановлена така відповідність між буквами та числами, яка представлена в таблиці 3.4.

Таблиця 3.4

Відповідність між буквами та числами англійського алфавіту

Буква	a	b	c	D	e	f	g	h	i	J	k	L	m
Номер	00	01	02	03	04	05	06	07	08	09	10	11	12
Буква	n	o	p	Q	r	s	t	u	v	W	x	Y	z
Номер	13	14	15	16	17	18	19	20	21	22	23	24	25

3.2.1 Симетричне шифрування на основі пошуку залишків

В симетричній криптосистемі обом абонентам повинні бути відомі модулі p_i, які виступають ключами. Відкрите повідомлення розбивається на блоки N, для кожного з яких повинна виконуватись умова $N<P$. Якщо першою цифрою блоку відкритого тексту буде 0, то тоді потрібна відповідна домовленість між абонентами. Однак доцільно, щоб блок відкритого тексту поміщав у собі цілу кількість букв.

Шифртекстом виступає набір залишків b_i, які отримуються з формули (3.13). Розшифрування або відновлення десяткового числа за його залишками відбувається згідно виразу (3.14) [280].

На рисунку 3.1 представлена загальна схема шифрування запропонованим методом на основі СЗК. Введено такі позначення: А – відправник, В – одержувач, N – відкритий текст; b_i – криптограма. Ключі p_i , i=1..m заздалегідь передаються по закритому каналу (вважається, що канал надійний) [281].

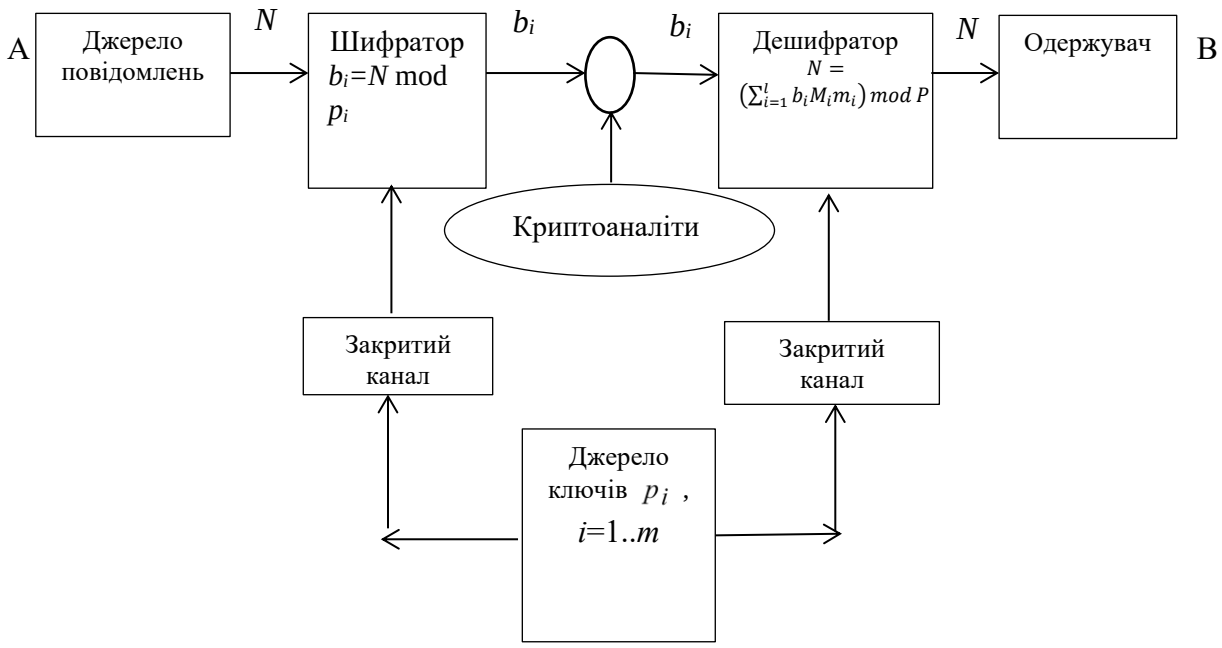


Рисунок 3.1 – Схема шифрування запропонованим алгоритмом на основі СЗК

Наведемо приклад. Нехай кількість модулів k=4, p_1=47, p_2=53, p_3=67, p_4=73. Тоді відповідно P=12183481. Відкритим текстом нехай буде слово keys, якому згідно таблиці 3.4 відповідає число 10042418. Результат шифрування та попередні розрахунки для розшифрування при використанні звичайної цілочисельної СЗК наведені в таблиці 3.5.

Отже, в результаті шифрування отримується послідовність 22315627, яка є шифртекстом. Перехопивши це повідомлення, зловмиснику, не знаючи ключів (модулів), дуже важко відновити відкритий текст, який являє собою суперпозицію декількох параметрів.

Таблиця 3.5

Результат шифрування та попередні розрахунки для розшифрування при використанні звичайної цілочисельної СЗК

i	1	2	3	4
p_i	47	53	67	73
b_i	22	31	56	27
M_i	259223	229877	181843	166897
$M_i \bmod p_i$	18	16	5	19
m_i	34	10	27	50

Для спрощення обчислень при розшифруванні формулу (3.14) запишемо таким чином, щоб шукати залишок від ділення на P кожного доданка (3.14):

$$N = \left(\sum_{i=1}^{k}\left((b_i M_i m_i)\ mod\ P\right)\right) mod\ P. \qquad (3.15)$$

Тоді N=((259223·34·22)mod 12183481+(229877·10·31)mod 12183481+ +(181843·27·56)mod 12183481+(166897·50·27)mod 12183481)mod 12183481= =(11146589+10344465+6910034+6008292) mod 12183481=10042418.

Видно, що чисельне значення відкритого тексту та результат розшифрування збігаються.

3.2.2 Побудова системи модулів для МДФ СЗК

Для порівняння розглянемо МДФ СЗК, яка складається із чотирьох модулів.

Покладемо p_1=49, тоді p_2=-50 і із (3.11) отримується: $p_{3,4} = -(a, b + 2450)$ і $ab = \pm 1 + 6002500 = \begin{cases} 6002499 = 3 \cdot 19 \cdot 31 \cdot 43 \cdot 79 \\ 6002501 = 2381 \cdot 2521. \end{cases}$. В таблиці 3.6 представлено усі можливі варіанти значень модулів p_3, p_4 при p_1=49, p_2=-50

Таблиця 3.6

Можливі варіанти значень модулів p_3, p_4 при p_1=49, p_2=-50

№	ab	a	b	p_3	p_4
1	2	3	4	5	6
1	3·19·31·43·79	1	3·19·31·43·79	-2451	-6004949
2		-1	-3·19·31·43·79	-2449	6000049
3		3	19·31·43·79	-2453	-2003283
4		-3	-19·31·43·79	-2447	1998383
5		19	3·31·43·79	-2469	-318371
6		-19	-3·31·43·79	-2431	313471
7		31	3·19·43·79	-2481	-196079
8		-31	-3·19·43·79	-2419	191179
9		43	3·19·31·79	-2493	-142043
10		-43	-3·19·31·79	-2407	137143
11		3·19	31·43·79	-2507	-107757
12		-3·19	-31·43·79	-2393	102857
13		79	3·19·31·43	-2529	-78431
14		-79	-3·19·31·43	-2371	73531
15		3·31	19·43·79	-2543	-66993
16		-3·31	-19·43·79	-2357	62093
17		3·43	19·31·79	-2579	-48981
18		-3·43	-19·31·79	-2321	44081

Продовження таблиці 3.6

19		3·79	19·31·43	-2687	-27777
20		-3·79	-19·31·43	-2213	22877
21		19·31	3·43·79	-3039	-12641
22		-19·31	-3·43·79	-1861	7741
23		19·43	3·31·79	-3267	-9797
24		-19·43	-3·31·79	-1633	4897
25		31·43	3·19·79	-3783	-6953
26		-31·43	-3·19·79	-1117	2053
27		19·79	3·31·43	-3951	-6449
28		-19·79	-3·31·43	-949	1549
29		3·19·31	43·79	-4217	-5847
30		-3·19·31	-43·79	-683	947
31		31·79	3·19·43	-4899	-4901
32		-31·79	-3·19·43	-1	1
33	2381·2521	1	2381·2521	-2451	-6004951
34		-1	-2381·2521	-2449	6000051
35		2381	2521	-4831	-4971
36		-2381	-2521	-69	71

3.2.3 Шифрування на основі МДФ СЗК

З усіх можливих варіантів, представлених в таблиці 3.6, доцільно вибрати p_3=-69, p_4=71, оскільки ці значення найменше відрізняються від перших двох вибраних модулів. Додаткові дослідження показують, що для кожного модуля потрібно змінити його знак на протилежний. Слід зазначити, що при розрахунках використовується абсолютна величина кожного модуля, а його знак враховується параметром m_i=±1. Діапазон обчислень P=12002550.

В якості відкритого тексту виберемо теж саме слово keys, яке відповідає числу 10042418. Результат шифрування згідно виразу (3.14) та попередні розрахунки, аналогічні таблиці 3.5, для розшифрування при використанні МДФ СЗК наведені в таблиці 3.7.

Отже, в результаті шифрування отримується шифртекст 15182036.

В зв’язку із знакозмінністю m_i, яке може набувати значень ±1, та тим, що $b_i<p_i$, то відновлення десяткового числа при використанні МДФ СЗК доцільно виконувати згідно формули (3.14): N=(-244950·1·15 + 240051·1·18 + 173950·1·20-169050·1·36)mod12002550=(-3674250+4320918+3479000-6085800)mod12002550= =(-1960132)mod12002550=10042418.

Таблиця 3.7

Результат шифрування та попередні розрахунки для розшифрування при використанні МДФ СЗК

i	1	2	3	4
p_i	49	50	69	71
b_i	15	18	20	36
M_i	244950	240051	173950	169050
$M_i \bmod p_i$	48	1	1	70
m_i	48mod49=-1mod49	1	1	70mod71=-1mod71

Видно, що використання МДФ СЗК приводить до зменшення кількості арифметичних операцій (зокрема, уникнення пошуку оберненого елемента та множення на нього в (3.14) [247], спрощення процедури пошуку залишку за модулем P [183]), які виконуються над операндами, що мають меншу розрядність в порівнянні із звичайною цілочисельною формою СЗК. В свою чергу, це зумовлює підвищення швидкодії процесу розшифрування інформації. Представлений алгоритм шифрування доцільно використовувати тоді, коли під час обміну інформацією необхідно швидко зашифрувати повідомленнями, а розшифрування може тривати більше часу.

3.2.4 Оцінка криптостійкості запропонованого алгоритму шифрування

Для забезпечення необхідного рівня захисту інформаційних потоків необхідно здійснити оцінку стійкості запропонованого методу шифрування, дослідивши основні вразливості та можливі варіанти криптоаналізу з точки зору зловмисника.

Оскільки роль ключів відіграють значення модулів p_i , то для математичної атаки необхідно перебрати всі можливі варіанти наборів p_i. Зазначені модулі повинні бути попарно взаємно простими, тобто задовольняти умову:

$$НСД(p_i, p_j) = 1,\ i=1, \ldots, m;\ j=1, \ldots, m;.i \neq j. \quad (3.16)$$

Оскільки, щоб отримати відкритий текст, для кожного набору необхідно застосувати КТЗ, то часові затрати (і, відповідно, стійкість алгоритму) [239, 282] на криптоаналіз повним перебором оцінюватиметься кількістю всіх можливих наборів p_i, помноженим на часову складність КТЗ. Однак слід відмітити, що оцінити теоретичну криптостійкість даного алгоритму можна тільки наближено.

3.2.4.1 Оцінка криптостійкості на основі закону асимптотичного розподілу простих чисел

Вважаємо, що модулі будуть простими числами. Згідно асимптотичного розподілу простих чисел, їх кількість на інтервалі від 0 до деякого q наближено визначається за формулою $\pi(q) = \frac{q}{ln\, q}$. Нехай для l-розрядного числа $q=2^l$. Тоді $\pi(q) = \frac{2^l}{l\, ln\, 2} \approx \frac{2^{l+1}}{l}$. Оскільки l можна вважати достатньо великим, то приблизна кількість варіантів для вибору системи з k модулів $\left(\frac{2^{l+1}}{l}\right)^k$. Часова складність КТЗ знову ж наближено становить l^2 [255], тому загальна часова складність для криптоаналізу запропонованого криптоалгоритму становить $O\left(\left(\frac{2^{l+1}}{l}\right)^k l^2\right)$.

На рисунку 3.2 зображена поверхня, яка характеризує залежність складності $\ln(O(l,k))$ (в логарифмічній шкалі) від розрядності l та кількості модулів k.

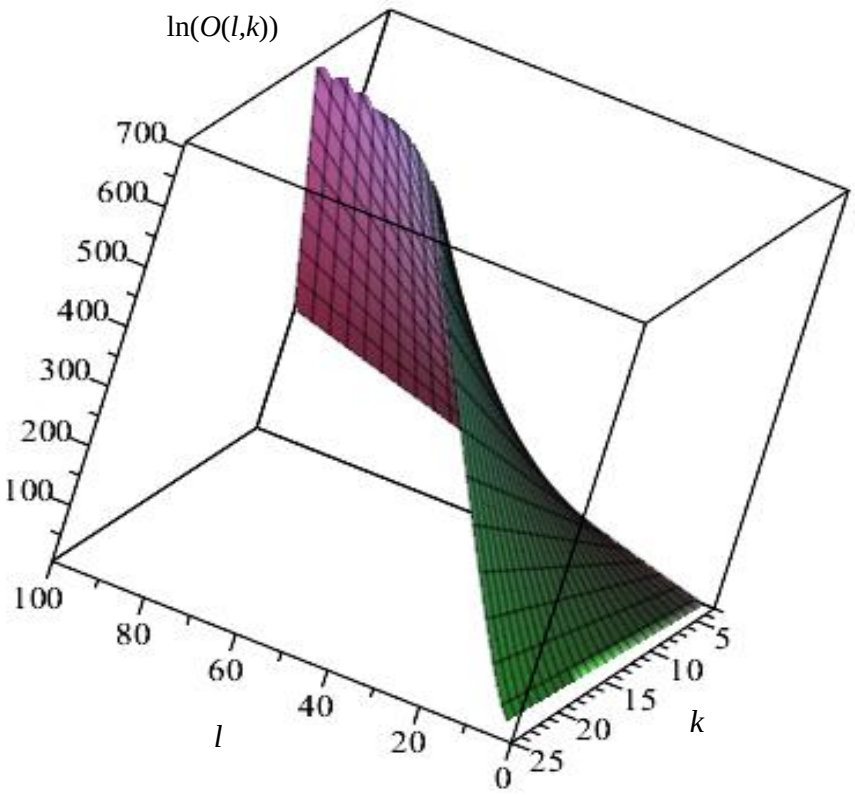


Рисунок 3.2 - Залежність складності ln($O(l)$) від розрядності l та кількості модулів k

Видно, що із збільшенням вказаних параметрів складність криптоаналізу зростає. З [34, 283] відомо, що для криптоаналізу сучасного симетричного шифру AES з l-бітовим ключем необхідно 2^{l-1} бітових операцій (максимальний ключ алгоритму AES становить 256 біт). З рівності $\left(\frac{2^{l+1}}{l}\right)^k l^2 = 2^{255}$ можна знайти розрядності та кількість модулів СЗК, які забезпечують таку ж стійкість, як і ключ найбільшої довжини алгоритму AES (табл. 3.8).

Таблиця 3.8

Розрядності та кількість модулів СЗК, які забезпечують таку ж стійкість, як і ключ найбільшої довжини алгоритму AES

К-сть модулів	3	4	5	6	7	8	9	10	11	12	13	14	15
Розрядність	87	66	54	46	40	35	32	29	27	25	23	21	20

Таблиця 3.7 демонструє, що із збільшенням кількості модулів їх розрядність зменшується. Слід відзначити, що стійкість запропонованого симетричного алгоритму у СЗК підвищується у 2 рази у порівнянні з класичним AES-256 при кількості модулів l=5, 6, 7, 8 з розрядностями k=139, 115, 98, 85 біт відповідно.

3.2.4.2 Оцінка криптостійкості на основі функції Ейлера

Кількість взаємно простих чисел з заданим обчислюється з використанням функції Ейлера $\varphi(p_i)$. Причому максимальне значення $\varphi_{\max}(p_i)$ приймає, коли $p_{imax}^{(l)}$ - максимальне просте число із заданої розрядності n тому, що $\varphi_{max}\left(p_{imax}^{(l)}\right) = p_{imax}^{(l)} - 1$. Наприклад для l=8 - розрядних чисел $p_{imax}^{(8)} = 251$, $\varphi_{max}\left(p_{imax}^{(8)}\right) = 250$.

Мінімальне значення функції Ейлера $\varphi_{\min}(p_i)$ буде в тому випадку, коли p_i розкладається в добуток простих чисел, тобто $p_i = \prod_{j=1}^{k} p_j$, де p_j - прості числа. Для l=8 розрядних чисел $p_i^{(8)}$ мінімальне значення $\varphi_{\min}(p_i^{(8)})$ буде, коли $p_i^{(8)} = 2 \cdot 3 \cdot 5 \cdot 7 = 210$, тобто $\varphi_{\min}(p_i^{(8)}) = \varphi(2) \cdot \varphi(3) \cdot \varphi(5) \cdot \varphi(7) = 1 \cdot 2 \cdot 4 \cdot 6 = 48$.

Оскільки перебір всіх можливих наборів попарно взаємно простих чисел пов'язаний з функцією Ейлера, то при максимальному фіксованому простому p_1 для p_2 кількість можливих варіантів вибору буде $\phi(p_1)$, для p_3 – відповідно $\phi(p_2)$, …, для p_k – $\phi(p_{k-1})$. Не зменшуючи загальності, вважаємо, що $p_1 > p_2 > \cdots > p_{k-1} > p_k$, причому тільки один з модулів може бути складеним. Тоді загальна кількість всіх наборів модулів оцінюватиметься згідно такої формули:

$$K = \prod_{i=1}^{k-1} \phi(p_i) \qquad (3.17)$$

Даний метод дозволяє здійснити оцінку, коли значення K буде максимальним, мінімальним та приймати значення в середині цього діапазону. Слід відмітити, що при максимальному значенні K буде забезпечена найбільша криптостійкість алгоритму шифрування, а, відповідно, при мінімальному – найменша.

Загальний час розшифрування на основі математичної атаки з врахуванням часової складності КТЗ буде обчислюватися згідно співвідношення $O\left(k \cdot l^2 \cdot \prod_{i=1}^{k-1} \phi(p_i)\right)$. Аналогічно до попереднього випадку, збільшення криптостійкості можна досягнути завдяки збільшенню кількості модулів (ключів), їх розрядності, а також вибором таких модулів, для яких значення $\phi(p_i)$ буде максимальним.

В таблиці 3.9 наведено приклад знаходження максимальної криптостійкості запропонованим методом для 4-ьох модулів розрядністю n=32, 64 та 128 біт.

Таблиця 3.9

Оцінка максимальної криптостійкості для 4-ьох модулів розрядністю l=32, 64 та 128 біт

№	l, біт	p_i	Максимальна оцінка криптостій-кості
1	32	p_1=4294967291,p_2=4294967279, p_3=4294967231, p_4=4294967197	$1{,}6 \cdot 10^{32}$
2	64	p_1=18446744073709551557, p_2=18446744073709551533, p_3=18446744073709551521, p_4=18446744073709551437	$5{,}1 \cdot 10^{61}$
3	128	p_1=340282366920938463463374607431768211297, p_2=340282366920938463463374607431768211283, p_3=340282366920938463463374607431768211223, p_4=340282366920938463463374607431768211219	$1{,}3 \cdot 10^{120}$

Оскільки в оцінці стійкості значення функції Ейлера є l - розрядними числами, то складність пошуку кількості варіантів ключів обчислюється згідно

співвідношення $O(log_2(k-1)\cdot l^2)$. В результаті загальна оцінка стійкості становить $O(log_2(k-1)\cdot l^4)$, графік залежності якої від розрядності та кількості модулів представлено на рисунку 3.3.

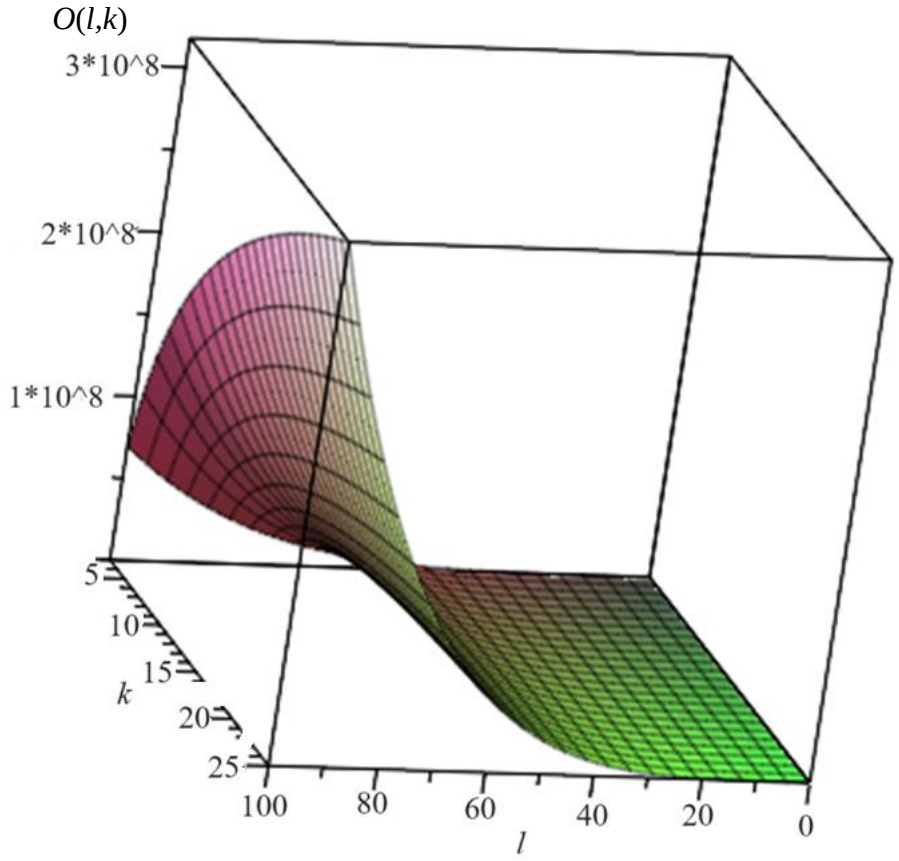


Рисунок 3.3 – Графік залежності стійкості від розрядності та кількості модулів

З рисунку 3.3 видно, що із збільшенням цих параметрів криптостійкість алгоритму різко збільшується.

3.3 Симетричне шифрування на основі КТЗ

У випадку, коли потрібне швидке розшифровування, доцільно використати інший симетричний криптоалгоритм. Однак при його використанні усі модулі СЗК мають перевищувати максимальне числове значення букви відкритого тексту. На рисунку 3.4 представлена загальна схема шифрування запропонованим методом.

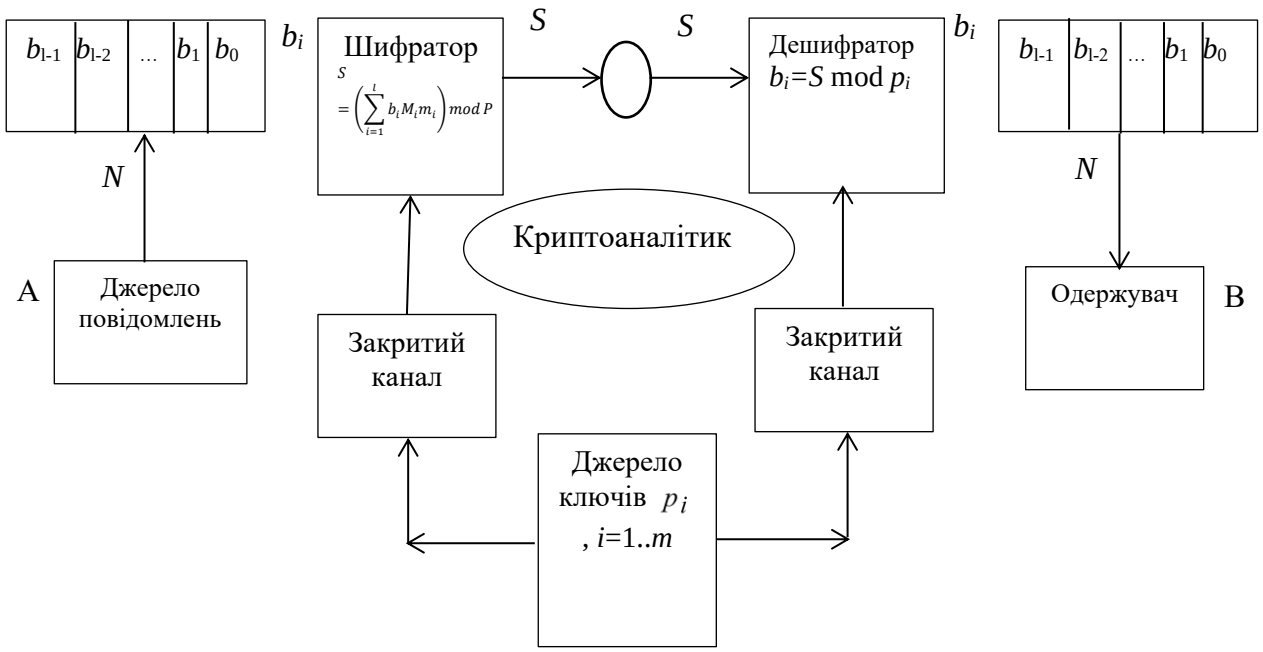


Рисунок 3.4 - Загальна схема симетричного шифрування на основі КТЗ

Блок відкритого тексту розбивається на підблоки, які менші відповідного модуля і виступають залишками від ділення деякого числа, яке є шифртекстом, на ці модулі. Шифртекст отримується згідно виразу (3.9).

Попередні розрахунки для шифрування при використанні звичайної цілочисельної СЗК (p_1=47, p_2=53, p_3=67, p_4=73, P=12183481) та відкритого тексту keys (10042418) наведені в таблиці 3.10.

Таблиця 3.11

Попередні розрахунки для шифрування при використанні звичайної цілочисельної СЗК

i	1	2	3	4
p_i	47	53	67	73
b_i	10	04	24	18
M_i	259223	229877	181843	166897
$M_i \bmod p_i$	18	16	5	19
m_i	34	10	27	50

Шифртекст:

S=((259223·34·10)mod12183481+(229877·10·4)mod12183481+

+(181843·27·24)mod12183481+(166897·50·18)mod12183481)mod12183481=(2851

453+ +9195080+8182935+4005528)mod12183481=12051515.

Відповідно, процес розшифрування полягає в пошуку залишків 12051515mod p_i: 12051515mod47=10, 12051515mod53=04, 12051515mod67=24, 12051515mod73=18. Після конкатенації числове значення відкритого тексту матиме вигляд 10042418, що відповідає слову keys.

Слід відмітити, що криптостійкість даного методу буде менша порівняно з попереднім, оскільки складність пошуку залишків $O(l \cdot \log_2 l)$ [283] менша, ніж для множення – $O(l^2)$ і оцінюватиметься згідно виразу $O\left(\left(\frac{2^{l+1}}{l}\right)^k (l \cdot log_2 2l)^2\right)$. Слід зазначити, що запропонований підхід дозволив підвищити криптографічну стійкість до криптоаналітичних атак у 3 рази при кількості модулів k=5 та їх розрядності l=150 біт в порівнянні з AES-255 та пришвидшити процес розшифрування/

При використанні МДФ СЗК отримується таблиця 3.12 для попередніх розрахунків.

Таблиця 3.12

Попередні розрахунки для шифрування при використанні МДФ СЗК

i	1	2	3	4
p_i	49	50	69	71
b_i	10	04	24	18
M_i	244950	240051	173950	169050
$M_i \bmod p_i$	48	1	1	70
m_i	48mod49=-1mod49	1	1	70mod71=-1mod71

Результат шифрування при використанні формули (3.14):

N=(-244950·1·10 +240051·1·4+173950·1·24-169050·1·18)mod12002550=

(-449500+960204+4174800-3042900)mod12002550=(-357396)mod12002550=

=11645154.

Знову ж, для розшифрування потрібно знайти залишки 11645154mod p_i: 11645154mod49=10, 11645154mod50=04, 11645154mod69=24, 11645154mod71=18. Отже, числове значення відкритого тексту має вигляд 10042418, що відповідає слову keys.

Аналогічно до попереднього випадку, підбір модулів, які задовольняють умовам МДФ СЗК, значно спрощують процес відновлення десяткового числа за його залишками.

3.4 Теоретичні основи симетричних методів шифрування в системі залишкових класів з допомогою зміни базисних чисел

Відновлення числа N по його залишках (формула 3.1) відбувається, як правило, на основі китайської теореми про залишки (КТЗ) [157]:

$$N = \left(\sum_{i=1}^{k} b_i M_i m_i\right) mod\, P, \tag{3.18}$$

де $P = \prod_{i=1}^{k} p_i$, $M_i = \frac{P}{p_i}$, m_i шукається з виразу $m_i = M_i^{-1}\, mod\, p_i$, k – кількість модулів [160, 161]. При цьому повинна виконуватися нерівність $N<P$.

Суть одного з методів симетричного шифрування в СЗК полягає в тому, що при відновленні числа в позиційну систему числення за його залишками у сумі (3.14) множення відбувається не на параметри $m_i = M_i^{-1}\, mod\, p_i$, а на довільно вибрані коефіцієнти k_i.

Отже, для генерації ключів обидва абоненти повинні вибрати відомі тільки їм обом системи модулів p_i та відповідні коефіцієнти k_i, для яких виконуються такі умови: $1<k_i<p_i$ та НСД (k_i, p_i)=1. Якщо p_i є простим числом, то друга умова виконується завжди. Відповідно, і відправнику, і отримувачу відомі параметри M_i та m_i.

Для шифрування текстову інформацію необхідно записати у числовій формі. Найпоширенішим класичним методом є заміна букви на її номер в

алфавіті, причому нумерація починається з 0. Тоді на етапі шифрування спочатку вибирається блок відкритого тексту $N<P$, який потім записується в СЗК згідно виразу (3.13). Шифрування відбувається при відновленні числа в позиційну систему числення згідно такого виразу:

$$N' = \left(\sum_{i=1}^{k} b_i M_i k_i\right) mod\, P. \tag{3.19}$$

Знайдене число являється шифртекстом, яке передається від одного абонента до іншого.

При розшифруванні спочатку обчислюються такі величини:

$$q_i = \left(m_i (k_i^{-1} mod\, p_i)\right) mod\, p_i;\ b_i' = N' mod\, p_i \tag{3.20}$$

Для отримання істинних залишків b_i необхідно виконати перетворення згідно співвідношення:

$$b_i = \left(b_i' q_i\right) mod\, p_i = \left(b_i' m_i k_i^{-1}\right) mod\, p_i. \tag{3.21}$$

Відповідно, відновлення числа N, яке є відкритим текстом, здійснюється за формулою (3.18) або можна використати вираз, який з неї випливає:

$$N = \left(\sum_{i=1}^{k} M_i m_i \left(\left(b_i' m_i k_i^{-1}\right) mod\, p_i\right)\right) mod\, P =$$

$$= \left(\sum_{i=1}^{k} M_i m_i \left(\left(b_i' q_i\right) mod\, p_i\right)\right) mod\, P. \tag{3.22}$$

Коректність запропонованої криптосистеми встановлюється з властивостей конгруенцій, врахувавши, що p_i є дільником числа P, та рівність $m_i = M_i^{-1} mod\, p_i$. Звідси отримуємо:

$$b_i = \left(b_i' q_i\right) mod\, p_i = \left((N' mod\, p_i) \cdot (m_i k_i^{-1})\, mod\, p_i\right) mod\, p_i =$$

$$= \left(\left(\left(\sum_{j=1}^{k} b_j k_j M_j\right) mod\, P\right) mod\, p_i \cdot (m_i k_i^{-1})\, mod\, p_i\right) mod\, p_i = \qquad (3.23)$$

$$= \left((b_i k_i M_i)\, mod\, P \cdot (m_i k_i^{-1})\, mod\, p_i\right) mod\, p_i = (b_i m_i M_i)\, mod\, p_i = b_i.$$

В таблиці 3.13 наведено приклад використання запропонованої симетричної криптосистеми для трьох модулів (k=3), відкритого тексту RNS=(171318), модулів p_1=47, p_2=59, p_3=71 та вибраних коефіцієнтів k_1=19, k_2=23, k_3=31.

Таблиця 3.13

Приклад використання запропонованої симетричної криптосистеми на основі СЗК

i	N	p_i	P	M_i	m_i	b_i	k_i	N'	b_i'	$k_i^{-1} mod\, p_i$	q_i
1	171318	47	196883	4189	8	3	19	2504	13	5	40
2		59		3337	34	41	23		26	18	22
3		71		2773	18	66	31		19	55	67

Отже, згідно виразу (3.19) шифртекст отримується таким чином:

N'=(4189·3·19+3337·41·23+2773·66·31)mod 196883=2504.

Після пошуку параметрів b_i' , $k_i^{-1} mod\, p_i$ та q_i розшифрування відбувається згідно виразу (3.22): N=(4189·8·((13·40)mod47)+3337·34·((26·22)mod59)+2773·18·((19·67)mod71))mod 196883= =171318.

У випадку, коли абоненти обмежені у часі, то доцільно прийняти, що k_i=1. Це зменшує стійкість до криптоаналізу, однак шифрування відбувається за спрощеною формулою:

$$N' = \left(\sum_{i=1}^{k} b_i M_i\right) mod\ P. \quad (3.24)$$

Крім того, при розшифруванні усувається необхідність виконання процедури пошуку оберненого елемента за модулем та множення на нього, оскільки q_i=m_i і розшифрування відбувається згідно таких виразів:

$$b_i = \left(b_i' m_i\right) mod\ p_i;\ N = \left(\sum_{i=1}^{k} M_i m_i \left(\left(b_i' m_i\right) mod\ p_i\right)\right) mod\ P. \quad (3.25)$$

Наприклад, для заданих у таблиці 1 вхідних параметрів за формулами (3.24)-(3.25) можна отримати: *N'*=(4189·3+3337·41+2773·66)mod 196883=135519; $b_1' = 18$, $b_2' = 55$, $b_3' = 51$; *N*=(4189·8·((18·8)mod47)+3337·34·((55·34)mod59)+2773·18·((51·18)mod71))mod 196883=171318.

Слід відмітити, що для зменшення операндів при відновленні числа за його залишками деякі з параметрів k_i можна вибрати від'ємними. Зокрема, для вхідних даних з таблиці 3.11 та k_1=-19, k_2=-23, k_3=31 суми (3.15), (3.18) стають знакозмінними:

N'=(-4189·3·19-3337·41·23+2773·66·31)mod 196883=122281, $b_1' = 34$, $b_2' = 33$, $b_3' = 19$, q_1=-40, q_2=-22, q_3=67, *N*=(4189·8·((-34·40)mod47)+3337·34·((-33·22)mod59)+2773·18·((19·67)mod71))mod196883= 171318.

Інший метод симетричного шифрування в СЗК полягає в тому, що відкритий текст розбивається на блоки, які менші від вибраних модулів і виступають залишками b_i по цих модулях. Після вибору параметрів k_i шифрування відбувається згідно з виразом (3.19). Шифртекстом буде значення *N'*.

Розшифрування відбувається за формулами (3.20), (3.21), згідно яких шукаються параметри q_i, $b_i^{'}$, та b_i. Конкатенація значень b_i утворює відкритий текст. Слід зазначити, що у випадку, коли потрібне швидке розшифрування, шифртекстом можуть виступати також параметри $b_i^{'}$.

Для вибраного вище відкритого тексту RNS=(171318), модулів p_1=47, p_2=59, p_3=71 та коефіцієнтів k_1=19, k_2=23, k_3=31, врахувавши дані таблиці 3.11, згідно (3.19) будемо мати: *N'*=(4189·17·19+3337·13·23+2773·18·31)mod 196883=157367.

Тоді за формулами (3.20), (3.21) отримуємо такі результати: $b_1^{'} = 11$, $b_2^{'} = 14$, $b_3^{'} = 31$; b_1=(11·8·5) mod 47=17; b_2=(14·34·18) mod 59=13; b_3=(31·18·55) mod 71=18. Отримані значення b_i відповідають відкритому тексту RNS=(171318). Згідно домовленостей між абонентами шифртекстом може виступати або параметр *N'*=157367, або конкатенація значень $b_i^{'}$: 111431.

При k=1 згідно (3.24), другої формули (3.20) та першої формули (3.25) маємо: *N'*=(4189·17+3337·13+2773·18)mod 196883=164508; $b_1^{'} = 8$, $b_2^{'} = 16$, $b_3^{'} = 1$; b_1=(8·8) mod 47=17; b_2=(16·34) mod 59=13; b_3=(1·18) mod 71=18.

Для знакозмінної суми у (3.15) (k_1=-19, k_2=-23, k_3=31) буде такий результат: *N'*=(-4189·17·19--3337·13·23+2773·18·31)mod 196883=180939, $b_1^{'} = 36$, $b_2^{'} = 45$, $b_3^{'} = 31$, q_1=-40, q_2=-22, q_3=67, b_1=(-36·40) mod 47=17; b_2=(-45·22) mod 59=13; b_3=(31·67) mod 71=18.

Іншим підходом для зменшення часової складності при симетричному шифруванні в СЗК може бути використання різних форм СЗК, зокрема її ДФ та МДФ. У них модулі підібрані таким чином, що відповідно виконуються співвідношення $m_i = M_i^{-1} \, mod \, p_i = 1$ та $m_i = M_i^{-1} \, mod \, p_i = \pm 1$. Зокрема, умовам МДФ СЗК для трьох модулів задовольняють такі залежності між модулями: $p_{2,3}$=2·p_1±1. Шифрування відбувається за формулою (3.19), розшифрування – (3.20)-(3.22), взявши до увагу що m_i може набувати значень 1 або -1.

В таблиці 3.14 наведено приклад використання запропонованої криптосистеми на основі МДФ СЗК для трьох модулів k=3), відкритого тексту RNS=(171318), модулів p_1=37, p_2=73, p_3=75 та коефіцієнтів k_1=19, k_2=23, k_3=31.

Таблиця 3.14

Приклад використання запропонованої симетричної криптосистеми на основі МДФ СЗК

i	N	p_i	P	M_i	m_i	b_i	k_i	N'	b_i'	$k_i^{-1} \bmod p_i$	q_i
1	171318	37	202575	5475	-1	8	19	91608	33	2	-2
2		73		2775	1	60	23		66	54	54
3		75		2701	1	18	31		33	46	46

Залишки b_i шукаються згідно формули (3.13), тоді з виразу (3.19) отримується шифртекст:

$$N'=(5475\cdot8\cdot19+2775\cdot60\cdot23+2701\cdot18\cdot31)\bmod 202575=91608.$$

Після пошуку відповідних параметрів b_i', $k_i^{-1} \bmod p_i$ та q_i розшифрування відбувається згідно виразу: N=(-5475·((-33·2)mod37)+2775·((66·54)mod73)+2701·((33·46)mod71))mod 202575=171318.

При k_i=1 та заданих в таблиці 3.14 вхідних параметрів за формулами (3.24)-(3.25) можна отримати: N'=(5475·8+2775·60+2701·18)mod 202575=56343; $b_1' = 29$, $b_2' = 60$, $b_3' = 18$; N=(-5475·(-29)+2775·60+ +2701·18)mod 202575=171318.

Слід зазначити, що в цьому випадку $b_1' = p_1 - b_1$, $b_{2,3}' = b_{2,3}$, тому в якості шифртексту доцільно вибрати значення N'=56343. Крім того, при k_i=m_i (k_1=m_1=-1; $k_{2,3}$=$m_{2,3}$=1) відкритий та зашифрований тексти будуть однаковими, тобто шифрування не відбуватиметься.

При виборі деяких k_i від'ємними (нехай, як і в попередніх випадках, k_1=-19, k_2=-23, k_3=31) та заданих в таблиці 3.14 вхідних параметрах згідно (3.19), (3.18) отримуються такі результати: *N'*=(5475·8·(-19)+2775·60·(-23)+ +2701·18·31)mod202575=86658, $b_1' = 4$, $b_2' = 7$, $b_3' = 33$, q_1=2, q_2=-54, q_3=46, *N*=(-5475·((2·4)mod37)+2775·((-54·7)mod73)+2701·((46·33)mod75))mod202575= =171318.

У випадку, коли блоки відкритого тексту виступають залишками від ділення на вибрані модулі при вхідних даних таблиці 3.12 згідно виразів (3.19)-(3.21) маємо: *N'*=(5475·17·19+2775·13·23+2701·18·31)mod 202575=53808; $b_1' = 10$, $b_2' = 7$, $b_3' = 33$; b_1=(11·8·5) mod 47=17; b_2=(14·34·18) mod 59=13; b_3=(31·18·55) mod 71=18. Шифртекстом може виступати або параметр *N'*=53808, або конкатенація значень b_i': 100733.

При виборі k_i=1 обчислення спростяться: *N'*=(5475·17+2775·13+2701·18)mod 202575= =177768; $b_1' = 20$, $b_2' = 13$, $b_3' = 18$. В цьому випадку $b_1' = p_1 - b_1$, $b_{2,3}' = b_{2,3}$, тому в якості шифртексту доцільно вибрати значення *N'*=177768. Крім того, при k_i=m_i (k_1=m_1=-1; $k_{2,3}$=$m_{2,3}$=1) шифртекст *N'*=(-5475·17+2775·13+2701·18)mod 202575=194193; залишки b_i' та b_i будуть однаковими.

Якщо, аналогічно до попереднього, деякі значення k_i вибрати від'ємними (для прикладу, k_1=-19, k_2=-23, k_3=31), то згідно заданих в таблиці 3.12 вхідних параметрів отримуються такі результати: *N'*=(5475·17·(-19)+2775·13·(-23)+2701·18·31)mod202575=124458, $b_1' = 27$, $b_2' = 66$, $b_3' = 33$, q_1=2, q_2=-54, q_3=46, b_1=(27·2) mod 37=17; b_2=(-66·54) mod 73=13; b_3=(33·46) mod 75=18. Шифртекстом може виступати або параметр *N'*=124458, або конкатенація значень b_i': 276633.

3.4.1 Оцінка криптостійкості симетричного алгоритму шифрування в системі залишкових класів

Криптостійкість запропонованого симетричного методу шифрування на основі СЗК, описаного виразом (3.19), ґрунтується на пошуку всіх можливих варіантів параметрів k_i та модулів криптоперетворень p_i . Використання функції Ейлера $\phi(p_i)$ дозволяє обчислити кількість взаємно простих чисел із заданим p_i, причому максимальне значення буде у випадку $\phi(p_{imax}^{(l)})$, тобто $p_{imax}^{(l)}$ - максимальне просте число розрядності l [185]. Для фіксованого максимального p_1=$p_{imax}^{(l)}$ кількість варіантів вибору p_2 становитиме $\phi(p_1) = p_{imax}^{(l)_1}$, для p_3 – відповідно $\phi(p_2)$, …, для p_k – $\phi(p_{k-1})$. Отже, набори модулів криптоперетворення у запропонованому методі можна отримати такою кількістю способів: $\prod_{i=1}^{k-1}\phi(p_i)$. Найбільша криптостійкість буде у випадку, коли $\phi(p_i)$ приймають максимальні значення, тобто коли $p_i^{(l)}$ - найбільші прості числа певної розрядності.

В запропонованому підході для шифрування, крім модулів криптоперетворень, використовуються ще параметри k_i, вибір яких теж можна здійснити $\prod_{i=1}^{k-1}\phi(p_i)$ способами. Отже, загальна складність математичної атаки з врахуванням часової складності КТЗ буде обчислюватися згідно такого співвідношення: $O\left(k \cdot l^2 \cdot \left(\prod_{i=1}^{k-1}\phi(p_i)\right)^2\right)$.

Як видно з оцінки часової складності, збільшення криптостійкості можна досягнути завдяки збільшенню кількості модулів p_i і, відповідно, параметрів k_i (ключів), їх розрядності, а також вибором таких модулів, для яких значення $\phi(p_i)$ буде максимальним.

Якщо вважати, що всі модулі мають розрядність n, то загальна стійкість оцінюватиметься виразом $O(log_2(k-1) \cdot k \cdot l^6)$. Графік залежності стійкості від розрядності l та кількості модулів k представлено на рисунку 3.5. Видно, що із збільшенням цих параметрів криптостійкість алгоритму різко зростає. Завдяки параметрам k і l, метод шифрування може бути налаштований для

досягнення необхідного рівня криптостійкості. Це робить його універсальним для використання в різних сценаріях, від стандартних до високозахищених середовищ.

Крім того, розроблений метод добре підходить для інтегрованих систем, зокрема в Інтернеті речей (IoT), завдяки можливості адаптації обчислювальної складності. Оптимізація параметрів дозволяє досягти балансу між криптографічною безпекою і обмеженими ресурсами IoT-пристроїв.

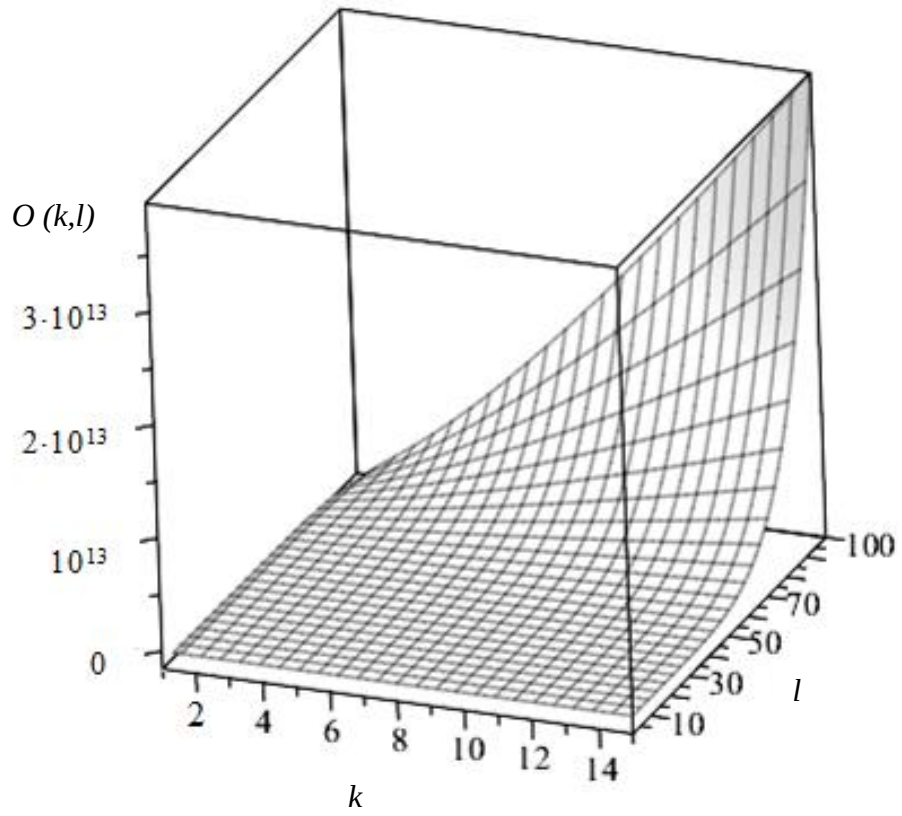


Рисунок 3.5 - Залежність криптостійкості алгоритму від розрядності модулів та їх кількості

Чисельні розрахунки показують, що для досягнення такої ж криптостійкості, як і симетричного криптоалгоритму AES [34, 284] з довжиною ключа 256 біт, розрядність модулів повинна становити приблизно 45 біт. Таким чином, запропонований метод демонструє високу стійкість, адаптивність і перспективність для застосування в сучасних і майбутніх системах захисту інформації, особливо в умовах обмежених ресурсів.

3.5 Асиметричні алгоритми шифрування у системі залишкових класів

Під час генерації ключів для асиметричного шифрування у СЗК [286] обидва абоненти повинні вибрати відомі тільки їм обом системи модулів p_i, які виступають таємними ключами, і відповідно обчислити параметри P_i та f_i. Звідси випливає симетричність запропонованої криптосистеми. Далі кожен абонент довільно вибирає цілі числа w_i, які виступають відкритими ключами і для яких виконуються умови $1 \le |w_i| \le p_i$ та НСД ($|w_i|$, p_i)=1. Якщо p_i є простим числом, то друга умова виконується завжди. Наявність відкритих ключів свідчить про асиметричність даної криптосистеми.

При шифруванні блок відкритого тексту $S<P$ спочатку записується в СЗК у вигляді залишків згідно виразу (3.1). Слід зазначити, що конкатенація залишків b_i відразу ж може виступати шифртекстом. Однак зловмисник, перехопивши декілька повідомлень, може зробити висновки про порядок параметрів p_i, що значно спростить криптоаналіз перехопленого тексту. Для усунення даного недоліку процес шифрування полягає в тому, що при відновленні десяткового числа із його залишків згідно формули $S = \left(\sum_{i=1}^{k} b_i P_i f_i\right) mod\, P$, де $P = \prod_{i=1}^{k} p_i > S$, $P_i = \frac{P}{p_i}$, f_i шукається з виразу $f_i P_i \, mod\, p_i = 1$, k – кількість модулів, множення відбувається не на коефіцієнти f_i, а на вибрані отримувачем відкриті ключі w_i, тобто [286]:

$$S' = \left(\sum_{i=1}^{k} b_i P_i w_i\right) mod\, P. \tag{3.26}$$

Знайдене число S' являється шифртекстом. В свою чергу, отримувач при розшифруванні обчислює величини:

$$r_i = \left(f_i (w_i^{-1} \, mod\, p_i)\right) mod\, p_i, \tag{3.27}$$

та визначає зашифровані залишки:

$$b_i^{'} = S^{'} mod\, p_i. \tag{3.28}$$

Для отримання справжніх залишків b_i виконуються такі операції:

$$b_i = \left(b_i^{'} r_i\right) mod\, p_i = \left(b_i^{'} f_i w_i^{-1}\right) mod\, p_i. \tag{3.29}$$

Тоді відновлення десяткового числа S, яке є відкритим текстом, відбувається згідно КТЗ. Крім того, можна використати формулу, яка з нього випливає:

$$S = \left(\sum_{i=1}^{k} P_i f_i \left(\left(b_i^{'} f_i w_i^{-1}\right) mod\, p_i\right)\right) mod\, P =$$

$$= \left(\sum_{i=1}^{k} P_i f_i \left(\left(b_i^{'} r_i\right) mod\, p_i\right)\right) mod\, P. \tag{3.30}$$

Коректність запропонованої криптосистеми, яка поєднує в собі симетричність та асиметричність, випливає із основних властивостей конгруенцій. Крім того, потрібно врахувати, що p_i є дільником числа P, а також рівність $f_i = P_i^{-1} mod\, p_i$. Звідси можна отримати:

$$b_i = \left(b_i^{'} r_i\right) mod\, p_i = \left((S' mod\, p_i) \cdot (f_i w_i^{-1})\, mod\, p_i\right) mod\, p_i =$$

$$= \left(\left(\left(\sum_{j=1}^{k} b_j w_j P_j\right) mod\, P\right) mod\, p_i \cdot (f_i w_i^{-1})\, mod\, p_i\right) mod\, p_i =$$

$$= \left((b_i w_i P_i)\, mod\, P \cdot (f_i w_i^{-1})\, mod\, p_i\right) mod\, p_i = (b_i f_i P_i)\, mod\, p_i = b_i. \tag{3.31}$$

Нехай, для прикладу, виберемо кількість модулів v=4, відкриті ключі w_1=31, w_2=41, w_3=43, w_4=53 та модулі p_1=43, p_2=59, p_3=71, p_4=79. Тоді відповідно P=14230033. В якості відкритого тексту виберемо слово LINE. Якщо встановити найпростіше правило числового кодування буквенних символів, коли кожній букві відповідає її номер в алфавіті (нумерація починається з нуля), то даному слову відповідає число S=11081304. Результат шифрування та попередні розрахунки для розшифрування при використанні звичайної цілочисельної СЗК із вказаними модулями наведені в таблиці 3.15.

Отже, в результаті шифрування згідно формули (3.26) отримується такий шифртекст: S'=(330931·32·31+241187·42·41+200423·50·43+ +180127·53·59)mod 14230033=1710119.

Він являє собою суперпозицію декількох параметрів. Тому не знаючи ключів (модулів), зловмисник, який перехопив це повідомлення, отримає дуже мало інформації про відкритий текст.

Після пошуку параметрів $b_i^{'}$, $w_i^{-1}\, mod\, p_i$ та r_i розшифрування відбувається згідно виразу (3.25):

S=(330931·29·((9·37)mod43)+241187·47·((4·40)mod59)+200423·7·((13·53) mod71)+180127·34·((6·22)mod79))mod14230033=1177944010 mod 4230033= =11081304.

Отже, числове значення відкритого тексту та результат розшифровування є однаковими.

Слід зазначити, що вибір деяких значень параметра w_i від'ємними дозволяє спростити проміжні обчислення за рахунок знакозмінності доданків у сумі (3.26) та, відповідно, зменшити розрядності операндів.

Інший підхід для запропонованого методу шифрування в СЗК полягає в тому, що блок відкритого тексту розбивається на підблоки, які менші від вибраних модулів. Вважається, що ці підблоки є залишками b_i за вибраними модулями. Після оголошення параметрів w_i відбувається використання КТЗ згідно з виразом (3.14).

Далі за формулами (3.27), (3.28) шукаються параметри r_i, b_i', та b_i. Шифртекстом може виступати або значення *S'*, або, коли потрібне швидке розшифрування, - параметри $b_i^{'}$. Відкритий текст утворює конкатенація значень b_i.

Для вибраного вище відкритого тексту LINE=(11081304), модулів p_1=43, p_2=59, p_3=71 p_4=79 та коефіцієнтів w_1=31, w_2=41, w_3=43, w_4=59, врахувавши дані таблиці 3.13, згідно (3.22) будемо мати: *S'*=(330931·11·31+241187·8·41+200423·13·43+180127·4·59)mod 14230033= =4982444.

Таблиця 3.15

Результат шифрування та попередні розрахунки для розшифрування при використанні звичайної цілочисельної СЗК

i	1	2	3	4
S	11081304			
p_i	43	59	71	79
P	14230033			
b_i	32	42	50	53
P_i	330931	241187	200423	180127
$P_i \bmod p_i$	3	54	61	7
f_i	29	47	7	34
w_i	31	41	43	59
S'	1710119			
$b_i^{'}$	9	4	13	6
$w_i^{-1} \, mod \, p_i$	25	36	38	75
r_i	37	40	53	22

Згідно з формулами (3.23), (3.24) отримуються такі результати: $b_1' = 34$, $b_2' = 12$, $b_3^{'} = 19$; $b_4^{'} = 72$; b_1=(34·29·25) mod 43=11; b_2=(12·47·36) mod 59=8; b_3=(19·7·38) mod 71=13, b_4=(72·34·75) mod 79=4. Конкатенація значень b_i дозволяє отримати відкритий текст LINE=(11081304). За угодою між

відправником та отримувачем шифртекстом може бути або число *S'*=4982444, або число, що являє собою конкатенацію значень b_i': 34121972.

3.5.1 Використання МДФ СЗК для асиметричного шифрування

Для спрощення розрахунків і, відповідно, зменшення часу при розшифруванні доцільно використовувати набори модулів, що утворюють МДФ СЗК [259], для якої виконується умова f_i=±1. У цьому випадку доданки у сумі (3.30) мають різні знаки, що зменшує проміжні результати обчислень. Для прикладу виберемо p_1=49, p_2=50, p_3=69, p_4=71. Параметри w_i та відкритий текст *S* залишаються такі ж самі, як і в попередньому прикладі. Результат шифрування та попередні розрахунки для розшифрування при використанні МДФ СЗК із вказаними модулями згідно формул (3.26)-(3.31) наведені в таблиці 3.16.

Таблиця 3.16

Результат шифрування та попередні розрахунки для розшифрування при використанні МДФ СЗК

i	1	2	3	4
S	11081304			
p_i	49	50	69	71
P	12002550			
b_i	3	4	42	50
P_i	244950	240051	173950	169050
$P_i \bmod p_i$	48mod 49= =-1mod 49	1	1	70mod 71= =-1mod 71
f_i	-1	1	1	-1
w_i	31	41	43	59
S'	10816314			
b_i'	5	14	12	32
$w_i^{-1} \, mod \, p_i$	19	11	61	65
r_i	-19	11	-8	6

Отже, в результаті шифрування згідно формули (3.26) отримується такий шифртекст:

S'=(244950·3·31+240051·4·41+173950·42·43+169050·50·59)mod 12002550 =874999914mod 12002550=10816314.

Згідно формули (3.30) можна отримати відкритий текст: *S*=(-244950·((-5·19)mod49)+ 240051·((14·11)mod50)+ 173950·((-12·8)mod69) - 169050·((32·6)mod71))mod 12002550 =-921246 mod 12002550=11081304.

Видно, що проміжні обчислення виконуються над меншими числами в порівнянні з попереднім відповідним прикладом, що дозволяє пришвидшити процес розшифрування повідомлення.

У випадку, коли підблоки відкритого тексту виступають залишками b_i за модулями p_i, які утворюють МДФ СЗК, то для вибраних вище параметрів *S* та w_i, врахувавши дані таблиці 3.16, згідно (3.26) отримується: *S'*=(244950·11·31+240051·8·41+ 173950·13·43+169050·4·59)mod 12002550= =299398528mod 12002550=11337328.

При розшифруванні за формулами (3.24), (3.25) отримується такі результати: $b_1' = 2$, $b_2' = 28$, $b_3' = 7$; $b_4' = 48$; b_1=(-2·19) mod 49=11; b_2=(28·11) mod50=8; b_3=(-7·8) mod 69=13, b_4=(48·6) mod 71=4. Після конкатенації значень b_i утворюється відкритий текст LINE=(11081304). Аналогічно до попереднього, за згодою абонентів шифртекстом може бути або число *S'*=11337328, або число, що являє собою конкатенацію значень b_i': 02280748.

3.5.2 Оцінка криптостійкості розробленого асиметричного алгоритму шифрування з використанням СЗК

Для оцінки криптостійкості розробленого асиметричного криптоалгоритму з використанням СЗК необхідно здійснити повний перебір усіх можливих варіантів взаємно простих модулів криптоперетворень p_i та знайти відповідні їм значення P_i та f_i або розв'язати задачу факторизації

(аналогічно криптосистемі RSA). Знаючи зазначені параметри і перехопивши у каналі зв'язку S' та w_i, зловмисник зможе знайти відкритий текст S.

Тому оцінка криптостійкості зводиться до визначення часових складностей пошуку p_i, P_i, b_i. Значення P_i знаходиться згідно формули $P_i = \frac{P}{p_i}$, $P = \prod_{i=1}^{k} p_i$. Оскільки для l – розрядних чисел необхідно виконати k операцій множення та ділення на p_i, то часова складність для визначення P_i оцінюється як $O1(k \cdot l^2(l+1)^2) \approx O1(kl^4)$.

Для кожного набору модулів часова складність знайденого на основі формули (3.26) добутку $b_i P_i w_i$ оцінюється як $O2(2l^2)$ (b_i, P_i, w_i - l – розрядні числа). Крім того, в (2) здійснюється сумування всіх наборів $b_i P_i w_i$ в залежності від кількості модулів, тому часова складність цієї операції оцінюється як $O3(kl)$.

Найбільш трудомісткою операцією при оцінці криптостійкості запропонованого асиметричного алгоритму шифрування у СЗК є визначення набору попарно взаємно простих модулів криптоперетворення p_i. Пошук p_i здійснюється повним перебором l – бітних чисел . Застосування функції Ейлера $\phi(p_i)$ дозволяє визначити кількість взаємно простих чисел із заданим p_i. Максимальне значення функції Ейлера досягається тоді, коли p_i буде найбільшим простим числом заданої розрядності n, тобто $\phi i_{i_{max}}$ [287]. Для l-бітного фіксованого максимального простого модуля $p_1 = p_{imax}^{(n)}$ кількість варіантів вибору модуля p_2 буде становити $\phi(p_1) = p_{imax}^{(n)_1}$, відповідно для p_3 буде $\phi(p_2)$, …, для p_k – $\phi(p_{k-1})$. Вважатимемо, що $p_1 > p_2 > \cdots > p_{k-1} > p_k$, причому тільки один із цих модулів може бути складеним. Тоді набори модулів криптоперетворень розробленого асиметричного алгоритму шифрування у СЗК можна отримати такою кількістю способів:

$$Z = \prod_{i=1}^{k-1} \phi(p_i). \qquad (3.32)$$

Слід відмітити, що криптостійкість розробленого алгоритму досягає свого максимуму, коли $\phi(p_i^{(l)})$ приймають найбільші значення, тобто $p_i^{(l)}$ - максимальні прості числа. Отже, для наборів з k модулів часова складність буде становити $O4(l^k)$. З врахуванням оцінок часових складностей базових операцій загальна складність математичної атаки становитиме $O(k^2 \cdot l^{k+7})$. Графік її залежності від розрядності модулів та їх кількості представлено на рисунку 3.6.

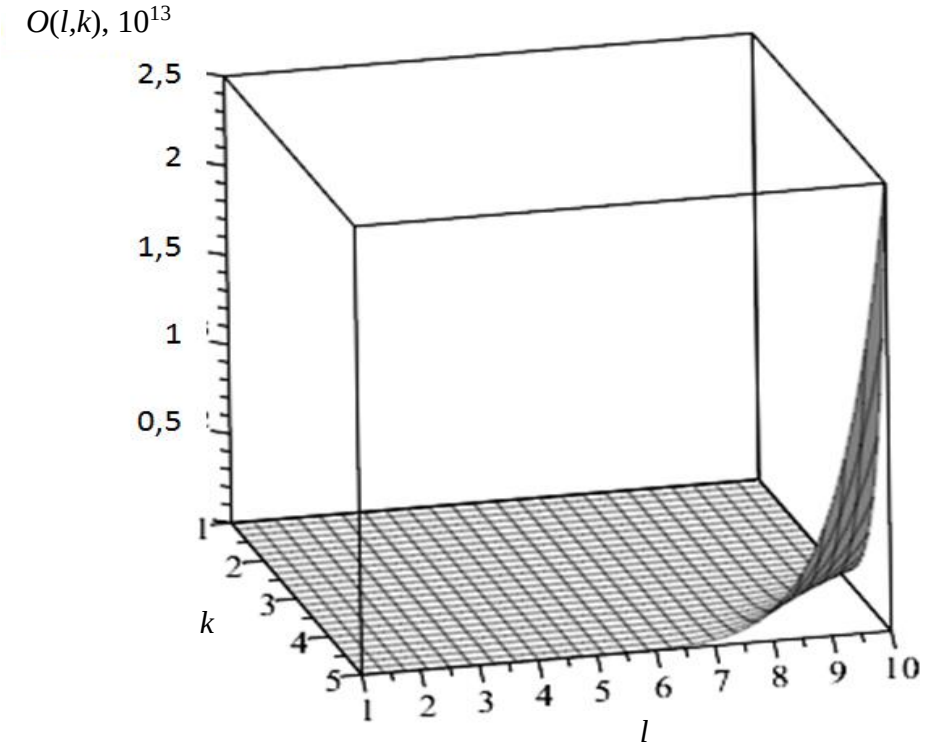


Рисунок 3.6 - Залежність криптостійкості запропонованого криптоалгоритму від розрядності та кількості модулів

Як видно з аналітичної оцінки часової складності та її графічної залежності, збільшення криптостійкості можна досягнути завдяки збільшенню кількості модулів p_i, їх розрядності, а також вибором таких модулів, для яких значення $\phi(p_i)$ буде максимальним. Причому підвищення стійкості в 2 рази можна досягнути при кількості модулів k=5 та їх розрядності l=74 біти.

Висновки до третього розділу

1. Розроблено три методи побудови системи модулів МДФ СЗК: факторизації, повного перебору та заміни. Представлено графічну залежність четвертого модуля від двох попередніх при одному відомому модулі. Розглянуто різні розрядності наборів модулів. Показано, що у наборах з чотирьох модулів однакової розрядності у модифікованій досконалій формі системи залишкових класів перший та четвертий модулі є від'ємними, другий та третій – додатніми. Це дозволяє більш раціонально використовувати регістри розрядної сітки.

2. Розглянуто теоретичні основи системи залишкових класів, її досконалої та модифікованої досконалої форм, визначено їх переваги та недоліки. Показано, що найбільш поширені на даний час модулі у вигляді степеня двійки, чисел Мерсена та чисел Ферма потребують виконання операції пошуку оберненого елемента та множення на нього, що ускладнює відновлення десяткового числа за його залишками при використанні китайської теореми про залишки. Модифікована досконала форма системи залишкових класів дозволяє спростити цю процедуру.

3. Розроблено симетричний криптоалгоритм у СЗК та МДФ СЗК в якому шифртекстом виступає набір залишків b_i по відповідних наборах модулів (ключів), а розшифрування або відновлення десяткового числа за його залишками відбувається згідно КТЗ.

4. Запропоновано метод симетричного шифрування на основі КТЗ, в якому блок відкритого тексту розбивається на підблоки, які менші відповідного модуля і виступають залишками від ділення деякого числа, яке є шифртекстом, на ці модулі, а відновлення відкритого тексту відбувається на основі пошуку залишків шифртексту по відповідним модулям.

5. Проведено оцінки стійкості запропонованих методів на основі теореми розподілу простих чисел та функції Ейлера та досліджено при якій розрядності та кількості модулів СЗК, розроблені симетричні системи захисту

забезпечують таку ж стійкість, як і ключ найбільшої довжини алгоритму AES. Встановлено, що із збільшенням кількості модулів їх розрядність зменшується.

6. Проведено оцінку часової складності математичної атаки на основі теореми розподілу простих чисел та функції Ейлера, та досліджено залежність складності від розрядності n та кількості модулів m. Аналіз проведених досліджень показав, що із збільшенням вказаних параметрів складність криптоаналізу зростає.

7. Розроблено високопродуктивні симетричні криптоалгоритми на основі СЗК та її МДФ, які, на відміну від класичних, дозволяють забезпечити необхідний рівень захисту даних за рахунок збільшення розмірності вхідних параметрів (розміру повідомлення, розрядності та кількості модулів). Особливості даного підходу полягають в тому, що при відновленні числа за його залишками з використанням КТЗ множення відбувається не на обернені елементи за модулем, а на довільно вибрані коефіцієнти (ключі) симетричного шифрування в СЗК, що дозволяє підвищити криптостійкість алгоритму.

8. Отримано аналітичні вирази оцінки стійкості, які вказують, що збільшення криптостійкості можна досягнути при збільшенні розрядності, кількості модулів та ключів, а також вибором таких модулів, для яких значення функції Ейлера буде максимальним. Представлено графічну залежність криптостійкості від розрядності та кількості модулів. Встановлено, що для досягнення такої ж криптостійкості, як і у симетричного криптоалгоритму AES-256, розрядність модулів повинна становити приблизно 45 біт.

9. Вперше запропоновані асиметричні криптоалгоритми на основі СЗК та її МДФ, які за рахунок збільшення розмірності вхідних параметрів (розміру повідомлення, розрядності та кількості модулів) забезпечують необхідний рівень захисту даних. Особливості даного підходу полягають в тому, що обрані системи модулів виступають таємними ключами, а при відновленні числа за його залишками з використанням КТЗ множення відбувається не на обернені елементи за модулем, а на довільно вибрані

коефіцієнти (відкриті ключі) асиметричного шифрування в СЗК, в результаті чого досягається підвищення криптостійкості алгоритму. Отримано аналітичні вирази, які вказують, що високої криптостійкості можна досягнути при збільшенні розрядності вхідних параметрів, кількості модулів (ключів), а також вибором таких модулів, для яких значення функції Ейлера буде максимальним. Представлено графічну залежність криптостійкості від розрядності та кількості модулів. Встановлено, що криптостійкість запропонованих алгоритмів, аналогічно криптосистемі RSA, ґрунтується на вирішенні задачі факторизації або повного перебору наборів модулів.

Основні результати третього розділу відображені у роботах [165, 259, 271, 275-278, 280, 281, 286].

РОЗДІЛ 4. ТЕОРЕТИЧНІ ОСНОВИ СИМЕТРИЧНИХ ПОЛІНОМІАЛЬНИХ КРИПТОАЛГОРИТМІВ НА ОСНОВІ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

4.1 Метод пошуку оберненого полінома за модулем в кільці Z[x] на основі методу невизначених коефіцієнтів

Однією з найбільш важливих і водночас складних операцій в теорії алгебри [288] та інших математичних застосуваннях [289] є пошук мультиплікативного оберненого полінома за модулем в кільці Z[x] [290, 291]. Поширеність зазначеної операції зумовлена її застосуванням у сучасній поліноміальній симетричній та асиметричній криптографії [292, 293] (зокрема, постквантовій [294]), паралельних та розподілених обчисленнях на основі системи залишкових класів [295, 296] в кільці Z[x] [296, 297], кодуванні даних на основі поліноміальної модулярної арифметики [298, 299], обробці сигналів і зображень [300, 301], вирішенні певних задач лінійного програмування [302] та інших додатках прикладної та дискретної математики [289, 303, 304].

Методи пошуку мультиплікативного оберненого полінома в кільці Z[x], як і в кільці цілих чисел, характеризуються значними обчислювальними складностями. Тому розробка нових та вдосконалення існуючих методів обчислення мультиплікативного оберненого полінома за модулем в кільці Z[x] є надзвичайно актуальною задачею [305, 306].

Нехай потрібно знайти обернений поліном за модулем $m(x) = r^{-1}(x)\, mod\, g\,(x)$ в кільці поліномів $Z(x)$, де $r(x)$ і $g(x)$ – взаємно прості поліноми ($\text{НСД}\big(r(x), g(x)\big) = w, w \in Z$), і $deg\, r\,(x) = n, deg\, g\,(x) = u$, для якого виконується рівність:

$$r(x) \cdot m(x) \equiv w\, mod\, g\,(x). \tag{4.1}$$

При цьому степінь полінома $deg\, m\,(x) = u - 1$, оскільки залишок по модулю $g(x)$ буде поліномом степеня $l - 1$. Нехай $f(x) = r(x) \cdot m(x)$ і $r(x) = A_n x^n + A_{n-1} x^{n-1} + \cdots + A_1 x + A_0$, $g(x) = B_u x^u + B_{u-1} x^{u-1} + \cdots + B_1 x + B_0$, тоді $m(x) = C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0$, де $A_i, B_j, C_k \in Z$, $i = 0 \ldots n, j = 0 \ldots u, k = 0 \ldots u - 1$. На основі методу невизначених коефіцієнтів рівність (4.1) для поліномів $r(x)$, $g(x)$ і $m(x)$ запишеться таким чином:

$$\left(\left(A_n x^n + A_{n-1} x^{n-1} + \ldots + A_1 x + A_0\right) \cdot \left(C_k x^k + C_{k-1} x^{k-1} + \ldots + C_1 x + C_0\right)\right) \bmod (B_u x^u + B_{u-1} x^{u-1} + \cdots + B_1 x^1 + B_0) = w, \quad (4.2)$$

або

$$\left((A_n x^n + A_{n-1} x^{n-1} + \cdots + A_1 x + A_0) \cdot (C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0)\right) mod(B_u x^u + + B_{u-1} x^{u-1} + \cdots + B_1 x^1 + B_0) - w = 0. \quad (4.3)$$

Спочатку потрібно знайти значення $f(x) = r(x) \cdot m(x)$:

$$f(x) = A_n x^n \cdot (C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0) + A_{n-1} x^{n-1} (C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0) + \cdots + A_1 x (C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0) + A_0 (C_k x^k + C_{k-1} x^{k-1} + \cdots + C_1 x + C_0) == A_n C_k x^{n+k} + (A_n C_{k-1} + A_{n-1} C_k) x^{n+k-1} + (A_n C_{k-2} + A_{n-1} C_{k-1} + A_{n-2} C_k) x^{n+k-2} + \cdots + (A_0 C_1 + A_1 C_0) x + A_0 C_0, \quad (4.4)$$

У виразі (4.4) $deg\, f\,(x) = n + k$. Введемо позначення: $F_{n+k} = A_n C_k$, $F_{n+k-1} = (A_n C_{k-1} + A_{n-1} C_k)$, $F_{n+k-2} = (A_n C_{k-2} + A_{n-1} C_{k-1} + A_{n-1} C_k)$, …, $F_0 = A_0 C_0$. Тоді рівність (4.4) перепишеться наступним чином:

$$f(x) = F_{n+k} x^{n+k} + F_{n+k-1} x^{n+k-1} + \cdots + F_1 x + F_0. \quad (4.5)$$

Постає задача визначення коефіцієнтів $C_k \in Z$, для яких виконується умова (4.3). Тому спочатку необхідно знайти залишок від ділення $f(x)$ на $g(x)$:

$$(F_{n+k}x^{n+k} + F_{n+k-1}x^{n+k-1} + \cdots + F_1x + F_0)\, mod(\, B_lx^l + B_{l-1}x^{l-1} + \cdots + B_1x^1 + B_0). \quad (4.6)$$

Розглянемо многочлен:

$$f(x) - \frac{F_{n+k}}{B_l}x^{n+k-l}g(x) = f_1(x),\, B_l \neq 0. \quad (4.7)$$

Враховуючи (4.5), вираз (4.7) запишеться таким чином:

$$(F_{n+k}x^{n+k} + F_{n+k-1}x^{n+k-1} + \cdots + F_1x + F_0) - \\ -\frac{F_{n+k}}{B_l}x^{n+k-l}(B_ux^u + B_{u-1}x^{u-1} + \cdots + B_1x^1 + B_0) = f_1(x), \quad (4.8)$$

причому $deg\, f\,(x) > deg\, f_1\,(x)$ і коефіцієнт при старшому степені визначається згідно співвідношення $f_1(x) = \left(\frac{F_{n+k-1}\cdot B_u - F_{n+k}\cdot B_{u-1}}{B_l^2}\right) = F_{1nk}$, $deg\, f_1\,(x) = n + k - 1 = n_1$.

Якщо $deg\, f_1\,(x) > deg\, g\,(x)$, тоді можна записати наступну рівність:

$$f_1(x) - \left(\frac{F_{n+k-1}\cdot B_u - F_{n+k}\cdot B_{u-1}}{B_u^2}\right)x^{n+k-l-1}g(x) = f_1(x) - F_{1nk}x^{n+k-l-1}g(x) = \\ f_2(x), \quad (4.9)$$

де $deg\, f_1\,(x) > deg\, f_2\,(x)$, причому коефіцієнт $f_2(x) = F_{2n}$ і $deg\, f_2\,(x) = n_2$. Неважко переконатися, що $deg\, f_2\,(x) > deg\, g\,(x)$, тобто $n_2 \geq u$. Тому можна продовжити дану процедуру і записати наступну рівність:

$$f_2(x) - \frac{F_{2nk}}{B_u} x^{n_2-u} g(x) = f_3(x), \tag{4.10}$$

де для степенів поліномів виконується умова $deg\, f_2\,(x) > deg\, f_3\,(x)$, коефіцієнт при максимальному степені полінома $f_3(x)$ буде F_{3nk} і $deg\, f_3\,(x) = n_3$.

Якщо $n_3 \geq l$, то аналогічно отримується рівність:

$$f_3(x) - \frac{F_{3nk}}{B_u} x^{n_3-u} g(x) = f_4(x) \tag{4.11}$$

,

де $deg\, f_2\,(x) > deg\, f_3\,(x)$. Відповідно до вищезазначених позначень, $f_3(x) = F_{3nk}$ і $deg\, f_3\,(x) = n_3$. Слід відмітити, що степені побудованих многочленів $f_1(x), f_2(x), f_3(x)$, … зменшуються $(n_1 > n_2 > n_3 > \cdots)$, тому після скінченої кількості кроків s отримається такий многочлен:

$$f_s(x) - \frac{F_{snk}}{B_u} x^{n_s-u} g(x) = f_{s+1}(x). \tag{4.12}$$

У ньому степені поліномів задовольняють нерівність $deg\, f_s\,(x) > deg\, f_{s+1}\,(x)$. Згідно вищезазначених співвідношень, значення шуканого залишку буде таким:

$$(F_{n+k}x^{n+k} + F_{n+k-1}x^{n+k-1} + \cdots + F_1 x + F_0)\, mod(\, B_u x^u + B_{u-1}x^{u-1} + \cdots + B_1 x^1 + B_0) = f_{r+1}(x).$$

Додавання всіх рівнянь (4.9)-(4.12) приводить до такого співвідношення:

$$f(x)-\frac{F_{n+k}}{B_l}x^{n+k-u}g(x)+f_1(x)-\frac{F_{1nk}}{B_u}x^{n_1-u}g(x)+f_2(x)-\frac{F_{2nk}}{B_u}x^{n_2-u}g(x)+$$

$$f_3(x)-\frac{F_{3nk}}{B_u}x^{n_3-u}g(x)+\ldots+f_s(x)-\frac{F_{snk}}{B_u}x^{n_s-u}g(x)=$$

$$(f_1(x)+f_2(x)+f_3(x)+\cdots+f_s(x))\Rightarrow \tag{4.13}$$

$$\Rightarrow f(x)-\left(\frac{F_{n+k}}{B_u}x^{n+k-u}+\frac{F_{1nk}}{B_u}x^{n_1-u}+\frac{F_{2nk}}{B_u}x^{n_2-u}+\frac{F_{3nk}}{B_u}x^{n_3-u}+\ldots+\frac{F_{snk}}{B_u}x^{n_s-u}\right)=f_{s+1}(x).$$

В результаті перетворень (4.13) отримується многочлен $f_{r+1}(x)=L_{r-1}x^{r-1}+L_{r-2}x^{r-2}+\cdots+L_1x+L_0$ порядку $deg\,f_{r+1}(x)=r-1$. З врахуванням умови (4.2) метод невизначених коефіцієнтів для обчислення величин $C_k\in Z$, де $k=1\ldots u-1$, приводить до системи з r рівнянь і r невідомих, які потрібно знайти:

$$L_{r-1}=0,\ L_{r-2}=0,\ldots,L_1=0,L_0=w. \tag{4.14}$$

З даних рівнянь обчислюється значення $C_k\in Z$. Схема пошуку оберненого поліному в кільці $Z(x)$ на основі методу невизначених коефіцієнтів представлено на рисунку 4.1.

4.1.1 Приклад застосування розробленого методу

Нехай задано многочлени $r(x)=x^2+3x+1$ і $g(x)=x^3+3x^2+2x+1$. Необхідно знайти $m(x)=r(x)^{-1}\,mod\,g(x)=(x^2+3x+1)^{-1}\,mod(x^3+3x^2+2x+1)$. Згідно представлених теоретичних положень многочлен $m(x)$ буде мати вигляд $m(x)=Ax^2+Bx+C$. Взявши до уваги (4.4), можна отримати добуток $r(x)\cdot m(x)=(x^2+3x+1)\cdot(Ax^2+Bx+C)=Ax^4+3Ax^3+Ax^2+Bx^3+3Bx^2+Bx+Cx^2+3Cx+C=Ax^4+(3A+B)x^3+(A+3B+C)x^2+(B+3C)x+C$. Далі, використовуючи співвідношення (4.9)-(4.12), отримується залишок від ділення $r(x)\cdot m(x)\,mod\,g(x)=(Ax^4+$

$(3A+B)x^3+(A+3B+C)x^2+(B+3C)x+C)\,mod(x^3+3x^2+2x+1)$: $Ax^4+(3A+B)x^3+(2A+3B)x^2+(A+2B)x+B=(Ax+B)\cdot(x^3+3x^2+2x+1)+\big((-A+C)x^2+(-A-B+3C)x+C-B\big)$.

Її розв'язок визначає шукані коефіцієнти: A=-1, B=-2, C=-1. Таким чином, обчислено значення оберненого полінома за модулем в кільці Z[x]. $m(x)=r(x)^{-1}\,mod\,g\,(x)=(x^2+3x+1)^{-1}\,mod(x^3+3x^2+2x+1)=-x^2-2x-1$.

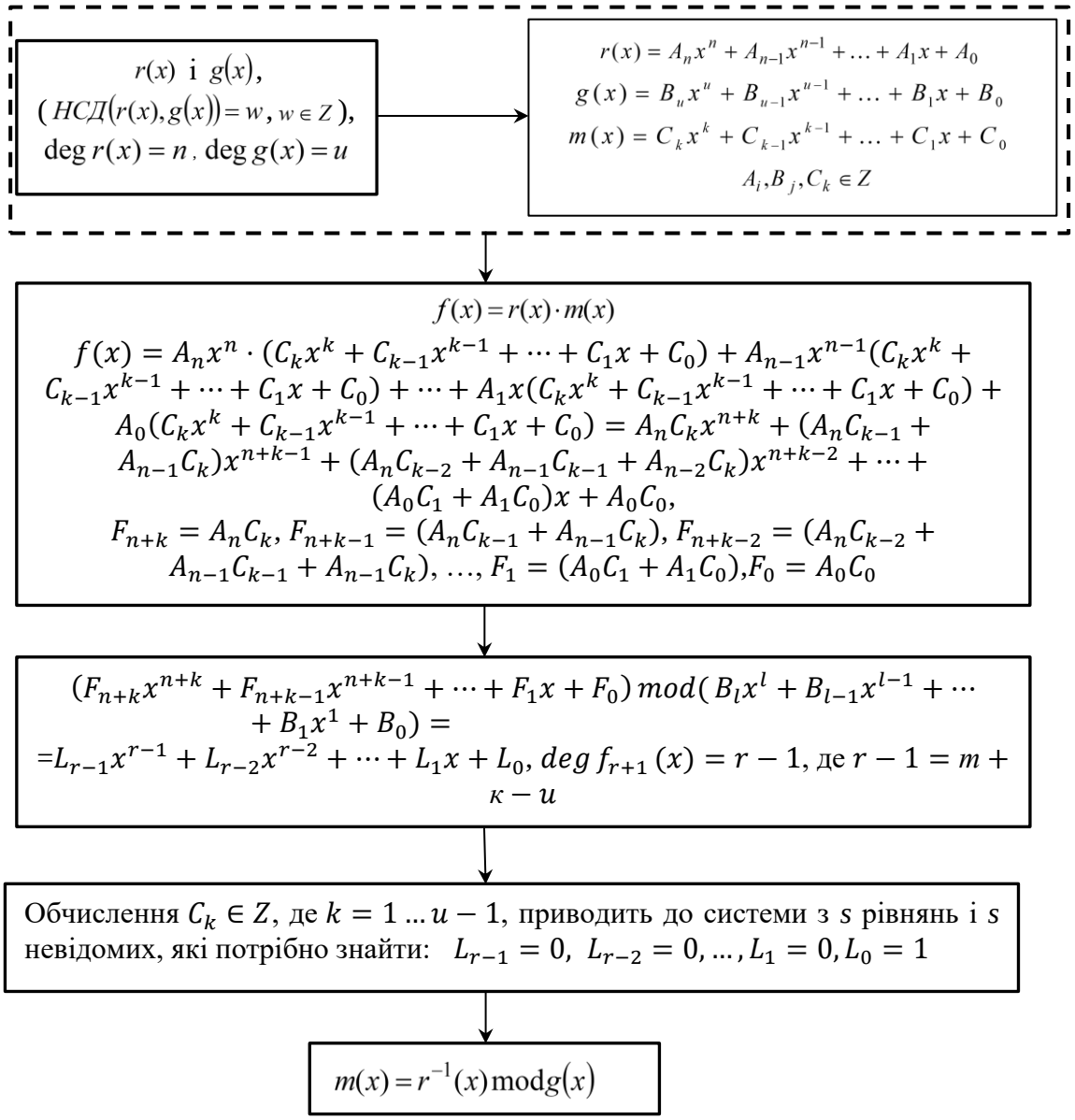


Рисунок 4.1 – Схема методу пошуку оберненого поліному в кільці поліномів

В результаті у залишку отримується поліном $f_3(x) = (-A + C)x^2 + (-A - B + 3C)x + C - B$, степінь якого менший від deg $g(x)$. Нехай для спрощення обчислень $f_3(x) = w = 1$, тобто $(-A + C)x^2 + (-A - B + 3C)x + C - B = 1$. Тоді умова (4.14) для пошуку невідомих коефіцієнтів A, B і C приводить до такої системи рівнянь:

$$\begin{cases} C - A = 0 \\ 3C - A - B = 0 \\ C - B = 1 \end{cases}$$

4.1.2 Оцінка часової складності запропонованого алгоритму обчислення оберненого полінома в кільці Z[x]

При побудові аналітичних виразів оцінки часової складності обчислення оберненого елемента у кільці поліномів згідно класичного та запропонованого методів необхідно визначити складності найбільш трудомістких операцій, а саме:

1. Добуток двох поліномів.
2. Пошук залишку двох поліномів.

На першому етапі методу, згідно (4.4), найбільш обчислювально складною є операція множення двох поліномів $(A_n x^n + A_{n-1}x^{n-1} + \cdots + A_1 x + A_0) \cdot (C_k x^k + C_{k-1}x^{k-1} + \cdots + C_1 x + C_0)$. Її часова складність для поліномів степеня n досліджувалась у роботі [171] і складає $O(n\,log_2\,n)$ бітових операцій, де логарифм береться при основі 2. З врахуванням складності пошуку залишків [171] загальна оцінка складе $O(2n\,log_2\,n)$ бітових операцій.

Відомий метод пошуку оберненого полінома в кільці поліномів базується на використанні алгоритму Евкліда та його наслідку. У [273] відмічено, що часова складність пошуку НСД($p(x)$,$q(x)$) над полем $Z[X]$ за алгоритмом Евкліда має верхню межу $O(n\,log_2^2\,n)$, де $n = max\{deg(p), deg(q)\}$. Крім того, найкраща відома асимптотична оцінка

зворотнього агоритму Евкліда рівна $O(n\,log_2 n\,log_2 log_2 n) \approx O(n\,log_2 n)$ [273].

З врахуванням часових складностей базових операцій загальна часова складність класичного методу пошуку оберненого елемента в кільці поліномів складатиме $O\left(nlog_2 n \cdot (1 + log_2 n)\right)$. Оскільки в запропонованому методі відсутній пошук НСД поліномів, то, відповідно, часова складність зменшиться: $O1(2n\,log_2 n)$. На рисунку 4.2 представлені графіки, які характеризують залежності часових складностей запропонованого та класичного підходів для пошуку оберненого елемента в кільці поліномів від їх степенів.

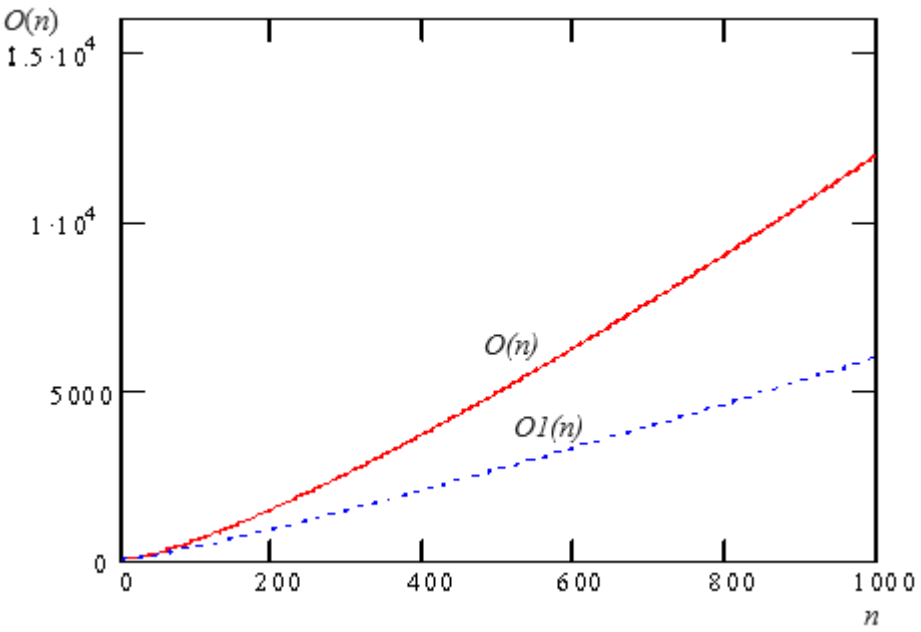


Рисунок 4.2 - Графіки залежності часових складностей запропонованого та класичного підходів для пошуку оберненого елемента в кільці поліномів від їх степенів

В результаті чисельного експерименту встановлено, що складності істотно зростають із збільшенням степенів поліномів.

Ефективність розробленого методу у порівнянні з класичним визначається як відношення відповідних складностей:

$$E(n) = \frac{O(n)}{O1(n)} = \frac{n \log n \cdot (1 + \log n)}{2n \log n} = \frac{1 + \log n}{2}. \quad (4.15)$$

З виразу (4.15) випливає, що знайдена ефективність розробленого підходу логарифмічно зростає із збільшенням степенів поліномів і при n=256 в 4,5 разів переважає відомий метод.

4.2 Метод відновлення полінома на основі додавання добутку модулів поліномів

Теоретичною основою при відновленні полінома за його залишками у відповідному кільці є алгебра і теорія чисел [307], зокрема китайська теорема про залишки (КТЗ) в поліноміальній формі [308]. Будь-який поліном $N(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ можна представити у вигляді залишків $b_i(x) = c_r x^r + c_{r-1} x^{r-1} + \cdots c_1 x + c_0$ від ділення на незвідні многочлени $p_i(x) = s_z x^z + s_{z-1} x^{z-1} + \cdots + s_1 x + s_0$, які називаються поліноміальними модулями:

$$b_i(x) = N(x) \, mod \, p_i \, (x). \quad (4.16)$$

де $deg \, b_i = deg \, N - deg \, p_i$.

При цьому необхідною і достатньою умовою є нерівність $N(x) < P(x) = \prod_{i=1}^{k} p_i \, (x)$, де k - кількість незвідних поліномів. Тоді вихідний поліном можна однозначно відновити на основі КТЗ:

$$N(x) = \left(\sum_{i=1}^{k} m_i(x) P_i(x) b_i \, (x) \right) mod \;\; P(x), \quad (4.17)$$

де $P_i(x) = \frac{P(x)}{p_i(x)}$, $m_i(x) = P_i^{-1}(x) \, mod \, p_i \, (x)$.

Ще одним методом відновлення полінома у відповідному кільці за його залишками є алгоритм Гарнера [198], основою якого є співвідношення:

$$N(x) = n_0(x) + n_1(x)p_1(x) + n_2(x)p_1(x)p_2(x) + \ldots + n_{n-1}(x)p_1(x)p_2(x)\ldots p_{i-1}(x), \quad (4.18)$$

де $0 \le n_i(x) < p_{i+1}(x)$, i=0, 1, …, k-1,

$$n_i(x) = \frac{b_{i+1}(x) - (n_0(x) + n_1(x)p_1(x) + \ldots + n_{i-1}(x)p_1(x)p_2(x)\ldots p_{i-1}(x))}{p_1(x)p_2(x)\ldots p_i(x)} \, mod \, p_{i+1}(x). \quad (4.19)$$

В цьому випадку поліноми $n_i(x)$ обчислюються послідовно один за одним на основі рекурентної формули (4.19). Крім того, і алгоритм Гарнера, і КТЗ придатні для аналогічних операцій у цілочисельній арифметиці.

Основними недоліками розглянутих вище методів відновлення полінома за його залишками є їх строго послідовна структура, що приводить до унеможливлення процесу розпаралелення обчислень, виконання операцій над поліномами вищих порядків (зокрема, обчислення залишку за модулем $P(x)$) необхідності пошуку мультиплікативного оберненого елемента за модулем у кільці поліномів.

Для пошуку останнього найбільш поширеними є такі методи [273]: перебір всіх можливих варіантів, за допомогою розширеного алгоритму Евкліда, на основі функції Ейлера. Дані підходи характеризуються значною обчислювальною складністю.

Тому метою є розробка методів відновлення полінома за його залишками у кільці поліномів на основі додавання добутку та додавання залишків модулів з можливістю розпаралелення обчислень та уникнення процедури пошуку мультиплікативного оберненого полінома. При цьому результати проміжних обчислень не будуть виходити за межі встановленого

діапазону, що усуває необхідність виконання операції знаходження залишку за порівняно великим модулем $P(x)$.

Розглянемо систему конгруенцій, яка будується на основі співвідношення (4.1):

$$\begin{cases} b_1(x) \equiv N(x)\, mod\, p_1\,(x) \\ b_2(x) \equiv N(x)\, mod\, p_2\,(x) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ b_k(x) \equiv N(x)\, mod\, p_k\,(x). \end{cases} \tag{4.20}$$

Будь-яку конгруенцію за модулем $p_1(x)$ з залишком $b_1(x)$ (наприклад, $a(x)\, mod\, p_1\,(x) \equiv b_1(x)$) можна представити у вигляді $a(x) = \gamma(x)p_1(x) + b_1(x)$, де $\gamma(x)$ - поліном, який вказує, скільки разів потрібно додавати модуль $p_1(x)$ до залишку $b_1(x) = N_1(x)$, щоб виконувалося співвідношення $N_2(x)\, mod\, p_2\,(x) \equiv b_2(x)$. Тут $N_2(x) = N_1(x) + \gamma_1(x)p_1(x)$. Далі необхідно додавати добуток $p_1(x)p_2(x)$ до тих пір, поки не буде виконуватись конгруенція $N_3(x)\, mod\, p_3\,(x) \equiv b_3(x)$, де $N_3(x) = N_2(x) + \gamma_2(x)p_1(x)p_2(x)$. Дана процедура продовжується до тих пір, поки не буде виконуватись останнє рівняння (4.20). Аналітично це записується таким чином:

$$N_1(x) = b_1(x);$$

$$N_2(x) = N_1(x) + \gamma_1(x)p_1(x) = b_1(x) + \gamma_1(x)p_1(x);\ N_2(x)\, mod\, p_2\,(x) \equiv b_2(x);$$

$$N_3(x) = N_2(x) + \gamma_2(x)p_1(x)p_2(x) = b_1(x) + \gamma_1(x)p_1(x) + \gamma_2(x)p_1(x)p_2(x);$$

$$N_3(x)\, mod\, p_3\,(x) \equiv b_3(x); \tag{4.21}$$

$$N_i(x) = N_{i-1}(x) + \gamma_{i-1}(x)p_1(x)p_2(x) \ldots p_{i-1}(x);\ N_i(x)\, mod\, p_i\,(x) \equiv b_i(x);$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$N_k(x) = N_{k-1}(x) + \gamma_{k-1}(x)p_1(x)p_2(x) \ldots p_{k-1}(x);\ N_k(x)\, mod\, p_k\,(x) \equiv r_k(x).$$

Пошук $\gamma_i(x)$ здійснюється методом невизначених коефіцієнтів, причому на кожному кроці алгоритму степінь поліному $\gamma_i(x) = A_i x^i +$

$A_{i-1}x^{i-1}+\ldots+A_1x+A_0$ буде на 1 менший степеня $p_{i+1}(x)$: $deg\,\gamma_i\,(x) = deg\,p_{i+1}\,(x)-1$.

Розглянемо приклад. Нехай задана система порівнянь:

$$\begin{cases} N(x)\,mod(x^2+x+1) \equiv x+3 \\ N(x)\,mod(x^2+x+2) \equiv 2x+5 \\ N(x)\,mod(\,x^3+2x+1) \equiv x^2+4x+1 \end{cases} \quad (4.22)$$

Не зменшуючи загальності задачі можна вважати, що для поліномів $p_1(x), p_2(x), \ldots, p_k(x)$ їхні степені задовольняють нерівностям $deg\,p_1 \geq deg\,p_2 \geq deg\,p_3 \geq \cdots \geq deg\,p_k$. Ці умови дозволяють робити більші кроки при виконанні ітерацій.

Знайдемо значення $a(x)$ з співвідношення $a(x)\,mod\,p_1\,(x) \equiv b_1(x)$. Його можна представити у вигляді $(\gamma(x)(x^2+x+1)+x+3)\,mod(x^2+x+2) = 2x+5$. Тут $\gamma(x)$ - поліном, який вказує, скільки разів потрібно додавати модуль $p_1(x) = x^2+x+1$ до залишку $x+3 = N_1(x)$, щоб виконувалося співвідношення $N_2(x)\,mod(x^2+x+2) \equiv 2x+5$, де $N_2(x) = x+3+\gamma_1(x)(x^2+x+1)$. Оскільки $p_1(x) = x^2+x+1$ - поліном 2 степеня, то шуканий параметр $\gamma_1(x)$ доцільно представити у вигляді полінома першого степеня: $\gamma_1(x) = A_1x+A_0$. Отримається добуток $\gamma_1(x)(x^2+x+1) = (A_1x+A_0)(x^2+x+1) = A_1x^3+(A_1+A_0)x^2+(A_1+A_0)x+A_0$. Для пошуку невідомих коефіцієнтів A_i розглянемо конгруенцію $(A_1x^3+(A_1+A_0)x^2+(A_1+A_0)x+A_0)\,mod(x^2+x+2) == \big((x^2+x+2)(A_1x+A_0)+(-A_1x-A_0)\big)\,mod(x^2+x+2) = x+2$.

Отже, з врахуванням останнього виразу коефіцієнти A_1 та A_0 приймають значення -1, -2 і відповідно $\gamma(x) = -x-2$, $N_2(x) = -x^3-3x^2-3x-2$.

Далі необхідно додавати добуток $p_1(x)p_2(x) = (x^2+x+1)(x^2+x+2) = (x^4+2x^3+4x^2+3x+2)$, поки не буде виконуватись конгруенція

$N_3(x)\,mod\,p_3\,(x) = b_3(x)$, де $x^2 + 4x + 1 = \big((-x^3 - 3x^2 - 3x - 2) + \gamma_2(x)(x^4 + 2x^3 + 4x^2 + 3x + 2)\big)\,mod(x^3 + 2x + 1)$. Спочатку обчислюється значення $(-x^3 - 3x^2 - 3x - 2)\,mod(x^3 + 2x + 1) = -3x^2 - x - 1$, яке розміщується у лівій частині попереднього рівняння: $4x^2 + 5x + 2 = \gamma_2(x)(x^4 + 2x^3 + 4x^2 + 3x + 2)\,mod(x^3 + 2x + 1)$. Зменшивши степінь модуля на 1, параметр $\gamma_2(x)$ потрібно шукати у вигляді $\gamma_2(x) = A_2x^2 + A_1x + A_0$, тобто $4x^2 + 5x + 2 = (A_2x^2 + A_1x + A_0)(x^4 + 2x^3 + 4x^2 + 3x + 2)\,mod(x^3 + 2x + 1)$. Оскільки $(x^4 + 2x^3 + 4x^2 + 3x + 2)\,mod(x^3 + 2x + 1) = 2x^2 - 2x$, то можна отримати$4x^2 + 5x + 2 = (A_2x^2 + A_1x + A_0)(2x^2 - 2x)\,mod(x^3 + 2x + 1)$. Далі обчислюється значення $(A_2x^2 + A_1x + A_0)(2x^2 - 2x) = 2A_2x^4 + 2A_1x^3 + (2A_0 - 2A_2)x^2 - 2A_1x - 2A_0$ і шукається залишок $(2A_2x^4 + 2A_1x^3 + (2A_0 - 2A_2)x^2 - 2A_1x - 2A)\,mod(x^3 + 2x + 1) = (2A_0 - 6A_2)x^2 +$

$+(-6A_1 - 2A_1)x - 2A_0 - 2A_1$. Пошук невідомих коефіцієнтів зводиться до розв'язку системи рівнянь:

$$\begin{cases} 2A_0 - 6A_2 = 4 \\ -6A_1 - 2A_2 = 5, \\ -2A_0 - 2A_1 = 2 \end{cases} \text{або} \begin{cases} A_0 - 3A_2 = 2 \\ -6A_1 - 2A_2 = 5. \\ -A_0 - A_1 = 1 \end{cases}$$

Її розв'язком будуть значення $A_2 = -\frac{13}{16}, A_1 = -\frac{9}{16}, A_0 = -\frac{7}{16}$. Отже, підставивши A_2, A_1, A_0, отримається відновлений поліном по залишках згідно запропонованого алгоритму:

$$N(x) = (-x^3 - 3x^2 - 3x - 2) + (4x^2 + 5x + 2)(x^4 + 2x^3 + 4x^2 + 3x + 2)$$
$$= 4x^6 + 13x^5 + 28x^4 + 35x^3 + 28x^2 + 13x + 2.$$

Отже, розв'язком системи (4.22) є поліном, який отримано без використання громіздких операцій та необхідності контролю переповнення розрядної сітки при виконанні проміжних обчислень.

Слід відмітити, що даний метод подібний до алгоритму Гарнера, однак у ньому уникається пошук оберненого елемента за модулем в кільці поліномів для отримання відповідних коефіцієнтів.

4.2.1 Метод відновлення полінома на основі додавання залишку від добутку модулів

Для спрощення обчислень, які використовуються у запропонованому методі, можна додавати не добуток поліномів-модулів, а залишок цього добутку від ділення на відповідний поліном. Математичний запис даного методу виглядає таким чином:

$$N_1(x) = b_1(x);\ p_{11}(x) = p_1(x)\ mod\ p_2\ (x);$$
$$(N_1(x) + \gamma_1(x)p_{11}(x))\ mod\ p_2\ (x) = b_2(x);\ N_2(x) = N_1(x) +$$
$$\gamma_1(x)p_1(x);\ p_{12}(x) = p_1(x)p_2(x)\ mod\ p_3\ (x);$$
$$(N_2(x) + \gamma_2(x)p_{12}(x))\ mod\ p_3\ (x) = b_3(x),\ N_3(x) = N_2(x) +$$
$$\gamma_2(x)p_1(x)p_2(x);\ p_{13}(x) = p_1(x)p_2(x)p_3(x)\ mod\ p_4\ (x);$$
$$(N_{i-1}(x) + \gamma_{i-1}(x)p_{1i-1}(x))\ mod\ p_i\ (x) = b_i(x);\ N_i(x) = N_{i-1}(x) +$$
$$\gamma_{i-1}(x)p_1(x)p_2(x)p_3(x)\ \dots p_{i-1}(x);$$
$$p_{1i}(x) = p_1(x)p_2(x)\ \dots p_i(x)\ mod\ p_{i+1}\ (x); \qquad (4.23)$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$(N_{k-1}(x) + \gamma_{k-1}(x)p_{1k-1}(x))\ mod\ p_k\ (x) = r_k(x);\ N(x) = N_k(x) =$$
$$N_{k-1}(x) + \gamma_{k-1}(x)p_1(x)p_2(x)p_3(x)\ \dots p_{k-1}(x).$$

Розглянемо приклад відновлення полінома за його залишками за допомогою додавання залишку від добутку модулів на основі системи (4.22).

Оскільки $(x^2 + x + 1)\ mod(x^2 + x + 2) \equiv -1$, то з першого порівняння (4.22) отримуємо конгруенцію $\big(x + 3 - \gamma_1(x)\big)\ mod(x^2 + x + 2) \equiv 2x + 5$, в якій необхідно визначити поліном першого степеня $\gamma_1(x) = A_1x + A_0$ методом невизначених коефіцієнтів. Комбінуючи дві останні рівності, можна отримати, що $A_1 = -1, A_0 = -2$, відповідно $\gamma_1(x) = -x - 2$. На наступному етапі необхідно знайти залишок від добутку

$p_1(x)p_2(x)\ mod\ p_3\ (x) = \big((x^2 + x + 1)(x^2 + x + 2)\big)\ mod(x^3 + 2x + 1) =$

$(x^4 + 2x^3 + 4x^2 + 3x + 2)\ mod(x^3 + 2x + 1) = 2x^2 - 2x$

і тоді $N_2(x) = -x^3 - 3x^2 - 3x - 2.$

Звідси

$$-(x^3 + 3x^2 + 3x + 2) + \gamma_2(x)(2x^2 - 2)\, mod(x^3 + 2x + 1) = x^2 + 4x + 1$$

або

$$\gamma_2(x)(2x^2 - 2)\, mod(x^3 + 2x + 1) = 4x^2 + 5x + 2.$$

В результаті отримується поліном, аналогічний до попереднього прикладу.

Отже, розроблені методи пошуку оберненого елемента в кільці поліномів дозволяють уникати виконання складних операцій, зокрема, ділення з остачею і пошуку оберненого елемента, та проводити обчислення над поліномами меншого порядку в порівнянні з класичною КТЗ та алгоритмом Гарнера.

4.2.2 Дослідження часової складності запропонованих методів

Для розрахунку часової залежності розроблених методів необхідно визначити найбільш трудомісткі базові арифметичні операції. Запропонований алгоритм містить множення, додавання та пошук залишків в кільці поліномів. У роботі [273] доведено, що задача множення $p(x) \cdot q(x)$ вимагає $O(4n \log n)$ (логарифм береться при основі 2) бітових операцій, де $n = max\{deg(p(x)), deg(q(x))\}$ – найбільший степінь поліному. З врахуванням складності пошуку залишків [307] загальна оцінка складе - $O(5n \log n)$ бітових операцій**.** Слід відмітити, що розроблені алгоритми вимагають $\frac{(k^2+k)}{2}$ операцій множення, де k - кількість модулів, та знаходження залишку на кожному кроці. Тому загальна складність асимптотично наближається до $O_1\big((k^2 + k)n \log_2 n\big)$.

У класичному алгоритмі Гарнера необхідно k – разів здійснювати пошук мультиплікативного оберненого елемента в кільці многочленів. У [292] зазначено, що часова складність згаданої операції в стандартному базисі

$GF(q^n)$ над $GF(q)$ рівна $O\left(n\,log_2{}^2\,n\,log_2 log_2\,n\right)$. Крім того, у класичному алгоритмі Гарнера присутні ті ж операції, що і у розроблених – додавання, множення та пошук залишків. З врахуванням цього його часова складність становить $O_2\left((k^2+k)(n\,log_2\,n)+kn\,log_2{}^2\,n\,log_2 log_2\,n\right)$.

Тому запропонований алгоритм відновлення числа за його залишками дозволяє зменшити часову складність з $O_2\left((k^2+k)(n\,log_2\,n)+kn\,log_2{}^2\,n\,loglog\,n\right)$ до $O_1\left((k^2+k)n\,log_2\,n\right)$. На рисунку 4.3 представлені графіки, які характеризують залежності часових складностей запропонованого та класичного підходів відновлення полінома по його залишках в кільці поліномів при $k=10$ і n=1,…,1000.

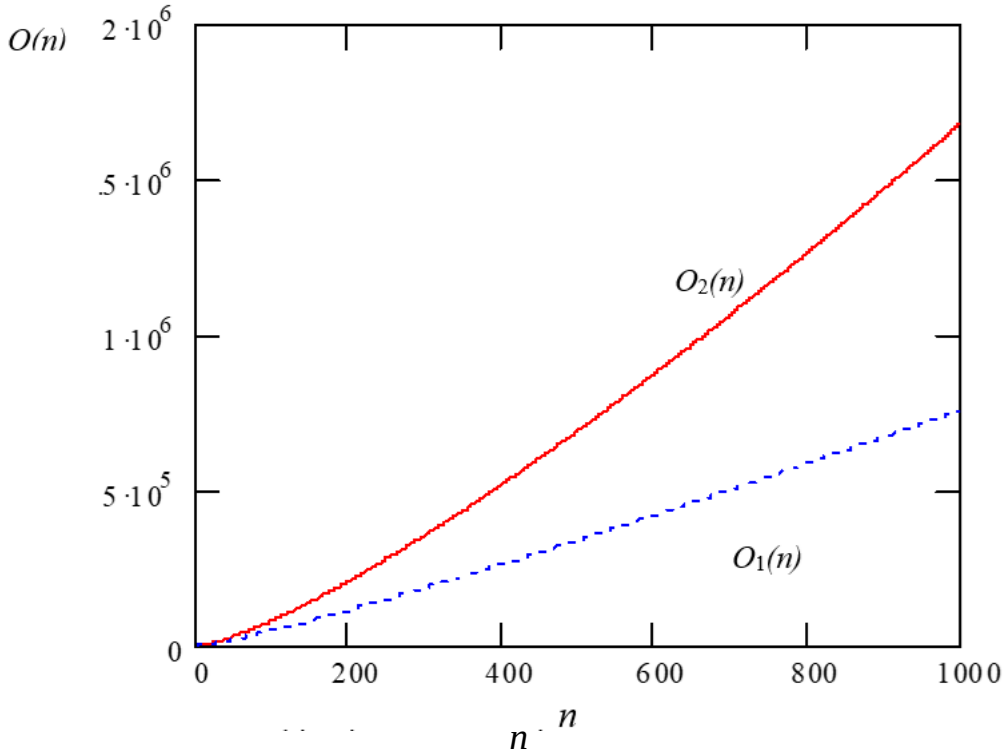


Рисунок 4.3 – Графічні залежності часових складностей розробленого методу $O_1(n)$ і алгоритму Гарнера $O_2(n)$

Важливо відмітити, що предствлені аналітичні вирази часових складностей вказують на їх квадратичну залежність відносно кількості модулів і лінійно логарифмічну по розрядності.

З рисунку 4.3 видно, що використання розробленого методу відновлення полінома по його залишках у відповідному кільці, який грунтується на основі

додавання добутку модулів-поліномів, дозволяє зменшити часову складність. Результати чисельного експерименту вказують на зростання обох часових складностей при збільшені розмірності вхідних параметрів.

Ефективність розроблених методів визначатиметься співвідношенням часових складностей і в загальному випадку представлятиметься таким чином:

$$E(n,k) = \frac{O_2(k,n)}{O_1(k,n)} = \frac{\left((k^2+k)(n\log_2 n) + kn\,\log_2{}^2\, n\,\log_2\log_2 n\right)}{\left((k^2+k)n\log_2 n\right)} \approx$$

$$\approx 1 + \frac{\log_2 n\,\log_2\log_2 n}{k+1} \qquad (4.24)$$

На рисунку 4.4 представлена графічна залежність ефективності запропонованого методу у порівнянні з алгоритмом Гарнера. Вхідні параметри вибиралися у таких діапазонах: $1 \le k \le 20$, а $1 \le n \le 1000$.

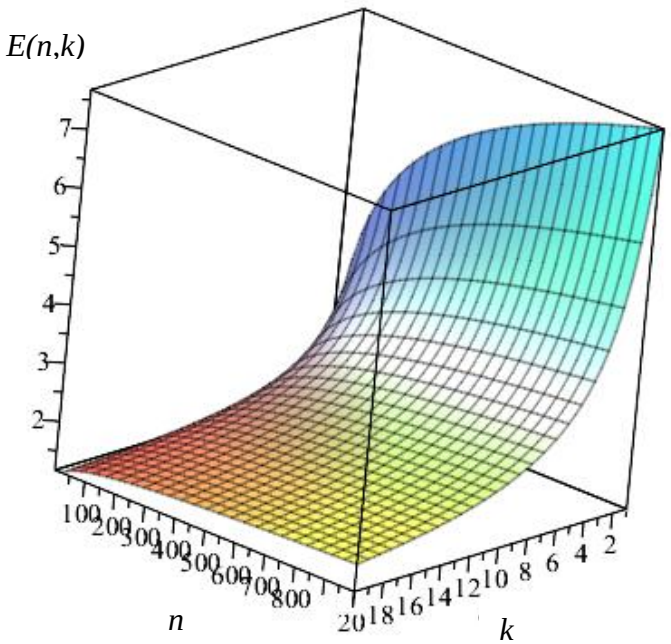


Рисунок 4.4 – Графічна залежність ефективності розробленого методу у порівнянні з алгоритмом Гарнера від кількості модулів k і степенів полінома

Слід відмітити, що ефективність запропонованого методу логарифмічно зростає із збільшенням степеня полінома, і пропорційно зменшується при збільшені кількості модулів. Наприклад для значень $k = 10 \quad n = 256$ ефективність зростає у 641 разів.

4.3 Криптосистема Рабіна в поліноміальній системі числення

Згідно класичної криптосистеми Рабіна (в якості таємного ключа вибираються два великі прості числа p і q, а відкритим ключем є їх добуток $n = p \cdot q$), у поліноміальній системі таємним ключем будуть два незвідні поліноми [309]:

$$p(x) = p_a x^a + p_{a-1} x^{a-1} + \ldots + p_1 x + p_0,$$
$$q(x) = q_b x^b + q_{b-1} x^{b-1} + \ldots + q_1 x + q_0, \quad (4.25)$$

де a і b – степені поліномів відповідно $p(x)$ та $q(x)$. Тоді відкритим ключем буде добуток цих поліномів:

$$n(x) = p(x) \cdot q(x) = p_a q_b x^{a+b} + (p_{a-1} q_b + p_a q_{b-1}) x^{a+b-1} + \cdots + (p_1 q_0 + p_0 q_1) x + p_0 q_0. \quad (4.26)$$

Відкритий текст m також потрібно представити у вигляді полінома:

$$m(x) = m_c x^c + m_{c-1} x^{c-1} + \ldots + m_1 x + m_0, \quad (4.27)$$

де c – степінь полінома $m(x)$.

Блок відкритого тексту $m(x)$ повинен бути максимальним, але не перевищувати $n(x)$. Шифрування відбувається за допомогою піднесення $m(x)$ до квадрату та пошуку його залишку за модулем, який є відкритим ключем:

$$d(x) = m(x)^2 \, mod \, n\,(x) =$$
$$= (m_c x^c + m_{c-1} x^{c-1} + \cdots + m_1 x + m_0)^2 \, mod (p_a q_b x^{a+b} + (p_{a-1} q_b + p_a q_{b-1}) x^{a+b-1} + \cdots + (p_1 q_0 + p_0 q_1) x + p_0 q_0) = d_e x^e + d_{e-1} x^{e-1} + \ldots + d_1 x + d_0. \quad (4.28)$$

Для розшифрування необхідно знайти залишки від ділення шифртексту на значення таємного ключа:

$$y(x) = d(x)\,mod\,p\,(x) = (d_e x^e + d_{e-1}x^{e-1} + \ldots + d_1 x + d_0)\,mod(\,p_a x^a + \\ + p_{a-1}x^{a-1} + \ldots + p_1 x + p_0) = y_f x^f + y_{f-1}x^{f-1} + \ldots + y_1 x + y_0,$$
$$z(x) = d(x)\,mod\,q\,(x) = (d_e x^e + d_{e-1}x^{e-1} + \ldots + d_1 x + d_0)\,mod(\,q_b x^b + \\ + q_{b-1}x^{b-1} + \ldots + q_1 x + q) = z_g x^g + z_{g-1}x^{g-1} + \ldots + z_1 x + z_0. \quad (4.29)$$

Після цього від знайдених поліномів $y(x)$ та $z(x)$ шукаються квадратні корені за відповідними модулями $p(x)$ та $q(x)$, які є таємним ключем:

$$\eta(x) = \sqrt{y(x)}\,mod\,p\,(x) = \sqrt{y_f x^f + y_{f-1}x^{f-1} + \ldots + y_1 x + x_0}\,mod(p_a x^a + \\ + p_{a-1}x^{a-1} + \ldots + p_1 x + p_0) = \eta_r x^r + \eta_{r-1}x^{r-1} + \ldots + \eta_1 x + \eta_0,$$
$$\lambda(x) = \sqrt{z(x)}\,mod\,q\,(x) = \sqrt{z_g x^g + z_{g-1}x^{g-1} + \ldots + z_1 x + z_0}\,mod(q_b x^b + \\ + q_{b-1}x^{b-1} + \ldots + q_1 x + q_0) = \lambda_s x^s + \lambda_{s-1}x^{s-1} + \ldots + \lambda_1 x + \lambda_0. \quad (4.30)$$

Основною модифікацією криптографічного алгоритму Рабіна в кільці поліномів є запропонований підхід обчислення квадратичного лишку на основі додавання модуля, яке виконується до тих пір, поки результат не буде повним квадратом. На відмінну від існуючих [207], цей метод дає змогу зменшити часову складність, і, відповідно, збільшити розмірності вхідних параметрів алгоритму. Причому максимальна кількість кроків при реалізації даного підходу буде $\zeta = \sqrt{max(\,deg(\,y(x), p(x))}$. Математично це можна записати так:

$$(y_f x^f + y_{f-1}x^{f-1} + \ldots + y_1 x + y_0) + i \cdot (p_a x^a + p_{a-1}x^{a-1} + \ldots + p_1 x + p_0) = \\ = (\eta_r x^r + \eta_{r-1}x^{r-1} + \ldots + \eta_1 x + \eta_0)^2 = \eta(x)^2, \quad (4.31)$$

$$(z_g x^g + z_{g-1}x^{g-1} + \ldots + z_1 x + z_0) + j \cdot (q_b x^b + q_{b-1}x^{b-1} + \ldots + q_1 x + q_0) =$$
$$= (\lambda_s x^s + \lambda_{s-1}x^{s-1} + \ldots + \lambda_1 x + \lambda_0)^2 = \lambda(x)^2,$$

де $i, j = 0, 1, 2, \ldots$.

З отриманих двох пар $(\eta(x);\ -\eta(x) = p(x) - \eta(x))$ та $(\lambda(x);\ -\lambda(x) = q(x) - \lambda(x))$ формується чотири можливих пари систем з двох конгруенцій:

$$\begin{cases} m_1(x)\ mod\ p\ (x) = \eta(x); \\ m_1(x)\ mod\ q\ (x) = \lambda(x); \end{cases} \quad \begin{cases} m_2(x)\ mod\ p\ (x) = p(x) - \eta(x); \\ m_2(x)\ mod\ q\ (x) = \lambda(x); \end{cases} \tag{4.32}$$

$$\begin{cases} m_3(x)\ mod\ p\ (x) = \eta(x); \\ m_3(x)\ mod\ q\ (x) = q(x) - \lambda(x); \end{cases} \quad \begin{cases} m_4(x)\ mod\ p\ (x) = p(x) - \eta(x); \\ m_4(x)\ mod\ q\ (x) = q(x) - \lambda(x); \end{cases}$$

Розв’язок однієї із систем (4.32) і буде шуканим відкритим текстом. На рисунку 4.5 предсталена схема шифрування/розшифрування криптосистеми Рабіна в поліноміальній системі числення.

Системи (4.32) розв’язуються на основі КТЗ для поліномів з використанням алгоритму Евкліда та його наслідку:

$$p(x) = q(x)\mu_1(x) + \sigma_1(x);$$
$$q(x) = \sigma_1(x)\mu_2(x) + \sigma_2(x);$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots..$$
$$\sigma_{k-2}(x) = \sigma_{k-1}(x)\mu_k(x) + \sigma_k(x);$$
$$\sigma_{k-1}(x) = \sigma_k(x)\mu_{k+1}(x) + 1(x).$$

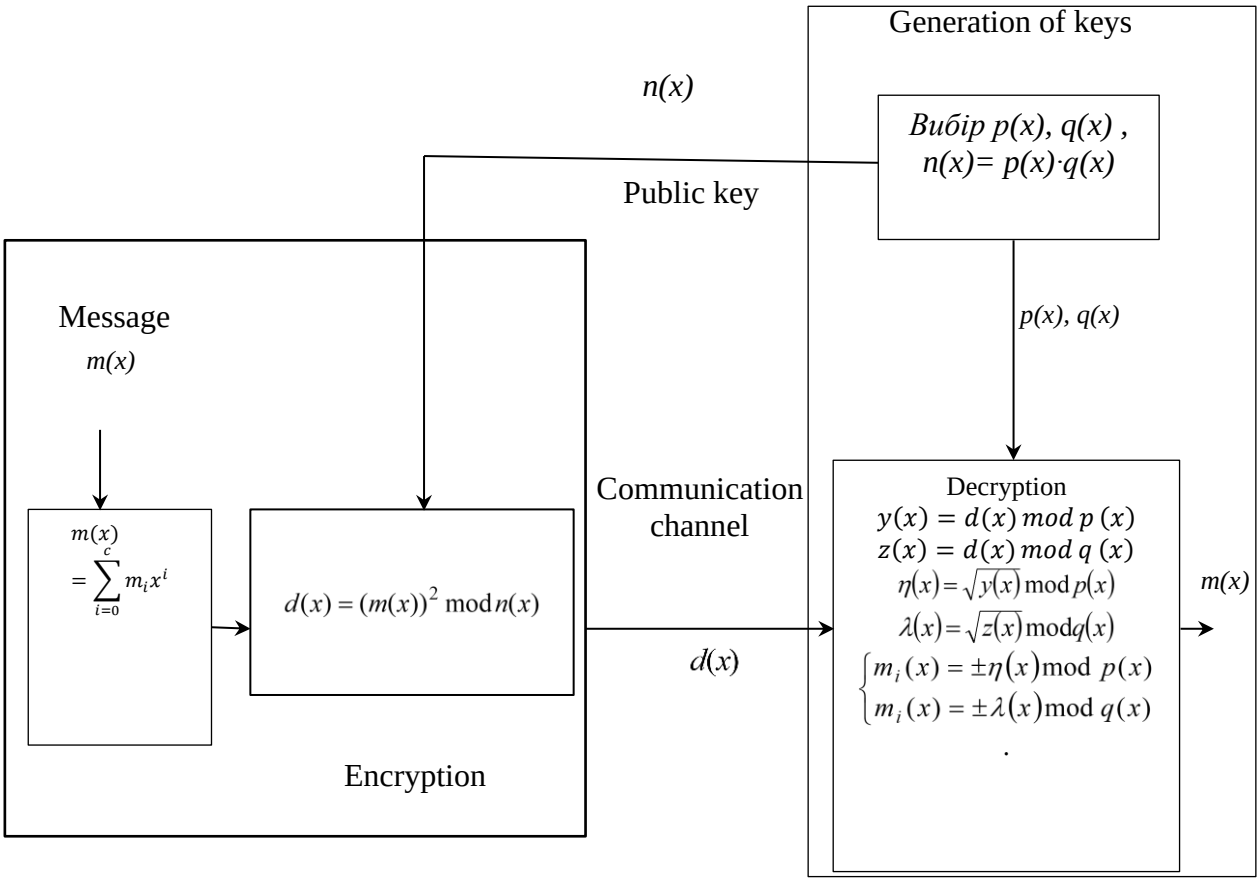


Рисунок 4.5 – Схема шифрування/розшифрування криптосистеми Рабіна в поліноміальній системі числення

Тоді $\sigma_k(x)$ є найбільшим спільним дільником поліномів $p(x)$ та $q(x)$. Якщо $\sigma_k(x) = const = \sigma_k$, то поліноми є взаємно простими. Суть наслідку алгоритму Евкліда полягає в його проходженні в зворотному порядку: $\sigma_k(x) = \sigma_{k-2}(x) - \sigma_{k-1}(x)\mu_k(x) = \sigma_{k-2}(x) - \mu_k(x) \cdot (\sigma_{k-3}(x) - \sigma_{k-2}(x) \cdot \mu_{k-1}(x)) =$ …Даний процес продовжується до тих пір, поки після ділення на $\sigma_k(x)$ не отримається вираз $p(x) \cdot \theta(x) + q(x) \cdot \tau(x) = 1$.

Далі обчислюються чотири значення $m_i(x)$ з рівнянь (8):

1) $m_1(x) = (p(x) \cdot \theta(x) \cdot \lambda(x) + q(x) \cdot \tau(x) \cdot \eta(x)) \, mod \, n\,(x)$;

2) $m_2(x) = (p(x) \cdot \theta(x) \cdot \lambda(x) + q(x) \cdot \tau(x) \cdot (p(x) - \eta(x))) \, mod \, n\,(x)$;

3) $m_3(x) = (p(x) \cdot \theta(x) \cdot (q(x) - \lambda(x)) + q(x) \cdot \tau(x) \cdot \eta(x)) \, mod \, n\,(x)$;

4) $m_4(x) = (p(x) \cdot \theta(x) \cdot (q(x) - \lambda(x)) + q(x) \cdot \tau(x) \cdot (p(x) - \eta(x)))\, mod\, n\,(x)$.

Одне із знайдених значень $m_i(x)$ відповідатиме відкритому тексту.

Приклад застосування даного методу. Нехай $p(x) = x^2 + 3x + 1$, $q(x) = x^2 + x + 3$. Тоді $n(x) = p(x) \cdot q(x) = (x^2 + 3x + 1) \cdot (x^2 + x + 3) = x^4 + 4x^3 + 7x^2 + 10x + 3$. Відкритий текст представимо у вигляді полінома $m(x) = x^3 + 3x^2 + 2x + 4$. Процес шифрування матиме такий вигляд: $d(x) = m(x)^2\, mod\, n\,(x) = (x^3 + 3x^2 + 2x + 4)^2\, mod(\, x^4 + 4x^3 + 7x^2 + 10x + 3) = 4x^3 + 19x^2 + +30x + 22$ - шифртекст.

Для розшифрування спочатку потрібно знайти залишки від ділення шифртексту на значення таємного ключа:

$y(x) = d(x)\, mod\, p\,(x) = (4x^3 + 19x^2 + 30x + 22)\, mod(x^2 + 3x + 1) = 5x + 15$;

$z(x) = d(x)\, mod\, q\,(x) = (4x^3 + 19x^2 + 30x + 22)\, mod(x^2 + x + 3) = 3x - 23$.

Для пошуку квадратного кореня $\eta(x) = \sqrt{y(x)}\, mod\, p\,(x) = \sqrt{5x + 15}\, mod(x^2 + 3x + 1)$ до значення $y(x) = 5x + 15$ потрібно додати модуль $p(x) = x^2 + 3x + 1$ один раз, щоб в результаті вийшов повний квадрат: $\eta^2(x) = 5x + 15 + x^2 + 3x + 1 = x^2 + 8x + 16 = (x + 4)^2$. Отже, $\eta(x) = x + 4$ і $p(x) - \eta(x) = x^2 + 2x - 3$.

Аналогічно шукається квадратний корінь $\lambda(x) = \sqrt{z(x)}\, mod\, q\,(x) = \sqrt{3x - 23}\, mod(x^2 + x + 3)$. Видно, що коефіцієнт біля x^2 має бути квадратом деякого числа. Однак значення $x^2 + x + 3 + 3x - 23 = x^2 + 4x - 20$ та $4x^2 + 4x + 12 + 3x - 23 = 9x^2 + 7x - 11$ не є повним квадратом. Такій вимозі задовольняє вираз $9x^2 + 7x - 11 + 27 + 3x - 23 = 9x^2 + 12x + 4 = (3x + 2)^2 = (\lambda(x))^2$. Звідси $\lambda(x) = 3x + 2$ і $q(x) - \lambda(x) = x^2 - 2x + 1$.

Далі формується чотири систем конгруенцій для визначення відкритого тексту:

$\begin{cases} m_1(x)\, mod(x^2+3x+1) = x+4; \\ m_1(x)\, mod(x^2+x+3) = 3x+2; \end{cases}$ $\begin{cases} m_2(x)\, mod(x^2+3x+1) = x^2+2x-3; \\ m_2(x)\, mod(x^2+x+3) = 3x+2; \end{cases}$

$\begin{cases} m_3(x) \bmod\left(x^2+3x+1\right) = x+4; \\ m_3(x) \bmod\left(x^2+x+3\right) = x^2-2x+1; \end{cases}$ $\begin{cases} m_4(x)\, mod(x^2+3x+1) = x^2+2x-3; \\ m_4(x)\, mod(x^2+x+3) = x^2-2x+1; \end{cases}$

Ці системи розв'язуються за допомогою КТЗ з використанням алгоритму Евкліда та його наслідку:

$$x^2+3x+1 = (x^2+x+3)+(2x-2);$$

$$x^2+x+3 = (2x-2)\cdot(0{,}5x+1)+5;$$

$$5 = (x^2+x+3)-(0{,}5x+1)\cdot(2x-2)$$

$$= (x^2+x+3)-(0{,}5x+1)\left((x^2+3x+1)-(x^2+x+3)\right) =$$

$$-(0{,}5x+1)(x^2+3x+1)+(0{,}5x+2)(x^2+x+3).$$

Звідси:

$$-(0{,}1x+0{,}2)(x^2+3x+1)+(0{,}1x+0{,}4)(x^2+x+3) = 1.$$

Далі розглядається чотири випадки (в двох останніх випадках для спрощення можна скористатися співвідношеннями $m_3(x) = n(x) - m_1(x)$, $m_4(x) = n(x) - m_2(x)$:

1) $m_1(x) = \left(-(0{,}1x+0{,}2)(x^2+3x+1)\right.$

$$\left.+(0{,}1x+0{,}4)(x^2+x+3)\right)\, mod(x^4+4x^3+7x^2+10x+3)$$

$$= x^2+4x+5;$$

2) $m_2(x) = \left(-(0{,}1x+0{,}2)(x^2+3x+1)(x^2-2x+1)+(0{,}1x+0{,}4)(x^2+\right.$

$\left.x+3)(x+4)\right) mod(x^4++4x^3+7x^2+10x+3) = x^3+3x^2+2x+4;$

3)$m_3(x) = (x^4+4x^3+7x^2+10x+3)-(x^2+4x+5) = x^4+4x^3+6x^2+$

$6x-2;$

4) $m_4(x) = (x^4 + 4x^3 + 7x^2 + 10x + 3) - (x^3 + 3x^2 + 2x + 4) = x^4 + 3x^3 + 4x^2 + 8x - 1$.

Звідси видно, що другий випадок відповідає відкритому тексту.

4.3.1 Оцінка обчислювальної складності запропонованого алгоритму шифрування на основі модульної арифметики многочленів

Для проведення розрахунку часової складності алгоритму шифрування Рабіна на основі поліноміальної арифметики необхідно виділити найбільш трудомісткі операції, до яких відносяться:

– піднесення полінома до квадрату;

– пошук залишку полінома по модулю іншого полінома;

– пошук квадратичного лишку у кільці поліномів;

– алгоритм Евкліда для двох поліномів;

– пошук мультиплікативного оберненого елемента в кільці поліномів;

– КТЗ для поліномів.

При побудові аналітичних виразів оцінки часових складностей криптосистеми Рабіна у кільці поліномів згідно класичного та запропонованого методів необхідно визначити складності зазначених базових операцій.

На етапі шифрування найбільш обчислювально складною є операція піднесення до квадрату за модулем в кільці поліномів, тобто $d(x) = m(x)^2 \, mod \, n\,(x)$. У роботі [310] проведені дослідження часової складності добутку поліномів степеня n для фіксованого простого p у полі поліномів $F_p[X]$. Доведено, що дана задача вимагає $O(4n \log n)$ бітових операцій, де логарифм береться при основі 2. З врахуванням складності пошуку залишків [307] загальна оцінка складе $O(5n \log n)$ бітових операцій**.**

У процесі розшифрування необхідно знайти залишки від ділення шифртексту на значення таємного ключа $y(x)$, $z(x)$ з часовою складністю $O(n \log n)$ [273]. Після цього від знайдених поліномів $y(x)$ та $z(x)$ шукаються

квадратні корені за відповідними модулями $p(x)$ та $q(x)$, у результаті чого отримуються $\lambda(x)$ і $\eta(x)$. Як зазначено у [211], для вирішення даної задачі необхідно $O(n^2 k \log n)$ бітових операцій, де k – розмір у бітах вхідних коефіцієнтів поліномів. У запропонованому методі обчислення квадратичного лишку в кільці поліномів здійснюється на основі додавання модуля. Верхня асимптотична оцінка складності цього методу становитиме $O\left(\sqrt{\zeta} \cdot n\right) = O\left(n^{\frac{3}{2}}\right)$ бітових операцій.

На заключному етапі формуються чотири можливих пари систем з двох конгруенцій, вирішення яких на основі КТЗ для поліномів з використанням алгоритму Евкліда та його наслідку дозволяє отримати відкритий текст. Часова складність пошуку НСД$(p(x), q(x))$ над полем $F_p[X]$ за алгоритмом Евкліда має верхню межу $O\left(n \log_2{}^2 n\right)$, де $n = max\{deg(p), deg(q)\}$ [273]. Крім того, у [311] зазначено, що найкраща відома асимптотична оцінка пошуку мультиплікативного оберненого елемента в кільці многочленів рівна $O(n \log_2 n \log_2 \log_2 n) \approx O(n \log_2 n)$. В [273] відмічено, що існує алгоритм відновлення поліномів по залишках згідно КТЗ з часовою складністю $O(\psi n \log \psi n)$, де ψ - кількість модулів. Зокрема, для випадку двох поліномів $O(2n \log_2 2n)$.

Враховуючи складності базових операцій, загальна часова складність класичного криптографічного алгоритму Рабіна в кільці поліномів складатиме $O(n \log_2 n \cdot (nk + \log_2 n + 7) + 2n \log_2 2n)$. Оскільки в запропонованому методі обчислення квадратичного лишку здійснюється на основі операції додавання модуля, то, відповідно, зменшиться і його часова складність: $O1\left(5n \log_2 n + n^{\frac{3}{2}} + n \log_2 n + n \log_2{}^2 n + n + 2n \log_2 2n\right)$.

На рисунку 4.6 в логарифмічній шкалі представлені графіки, які характеризують залежності часових складностей запропонованого та класичного підходів для реалізації криптографічного алгоритму Рабіна в кільці поліномів від їх степенів при $k = 10$. В результаті чисельного

експерименту можна відмітити, що функції складностей істотно зростають із збільшенням степенів поліномів.

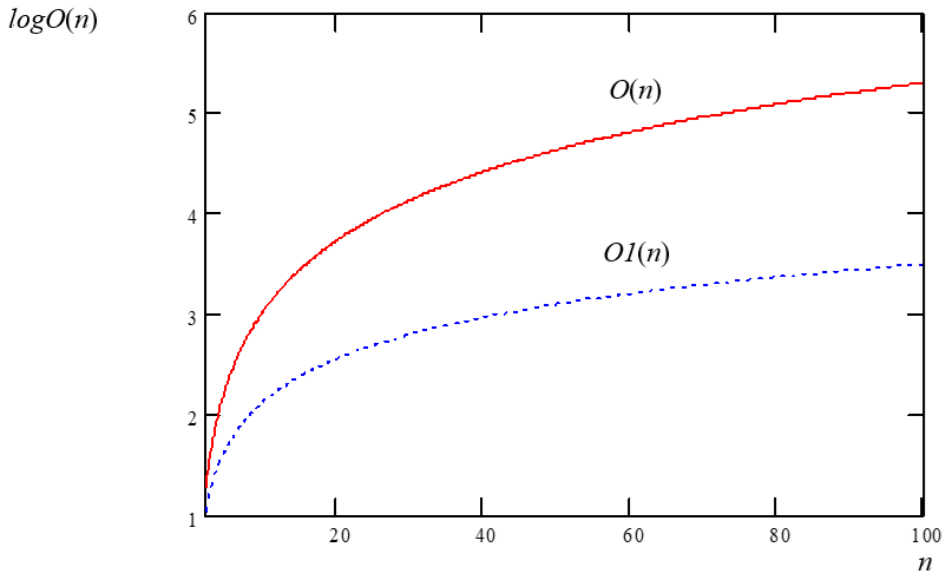


Рисунок 4.6 - Графічна залежність часових складностей запропонованого та класичного підходів для реалізації криптографічного алгоритму Рабіна в кільці поліномів

Проведені дослідження свідчать про те, що реалізація запропонованого алгоритмічного забезпечення криптоалгоритму Рабіна на основі операції додавання дозволяє зменшити часову складність базових операцій у порівнянні з класичним. Крім того, даний підхід повинен спростити апаратну реалізацію та структурну складність при схемотехнічному проектуванні криптосистеми Рабіна у кільці поліномів за рахунок використання однотипних суматорів замість перемножувачів та мультиплексорів.

Ефективність криптоалгоритму Рабіна на основі операції додавання визначатиметься співвідношенням часових складностей і в загальному випадку представлятиметься таким чином:

$$E(n,k) = \frac{O(k,n)}{O_1(n)} = \frac{(n\log_2 n\cdot(nk+\log_2 n+7)+2n\log_2 2n)}{\left(5n\log_2 n+n^{\frac{3}{2}}+n\log_2 n+n\log_2{}^2 n+n+2n\log_2 2n\right)}. \qquad (4.33')$$

Результати чисельного експерименту вказують, що ефективність запропонованого методу зростає із збільшенням степеня полінома, і

пропорційно зменшується при збільшені кількості модулів. Наприклад для значень $k = 3$ $n = 512$ ефективність зростає у 79 разів.

4.4 Криптографічні поліноміальні симетричні методи шифрування в системі залишкових класів

Довільний поліном $N(x)$ в ПСЗК представляється у вигляді залишків $b_i(x)$ від ділення $N(x)$ на кожен із системи попарно взаємно простих модулів-поліномів $p_i(x)$:

$$b_i(x) = N(x) \mod p_i(x). \tag{4.34}$$

Відновлення поліному $N(x)$ відбувається, як правило, на основі китайської теореми про залишки (КТЗ) [312] в кільці поліномів $Z[x]$:

$$N(x) = \left(\sum_{i=1}^{s} b_i(x) M_i(x) m_i(x)\right) mod\ P\ (x), \tag{4.35}$$

де $P(x) = \prod_{i=1}^{s} p_i(x)$, $M_i(x) = \frac{P(x)}{p_i(x)}$, $m_i(x)$ шукається з виразу $m_i(x) = M_i^{-1}(x)\, mod\, p_i\ (x)$, s – кількість модулів. При цьому для степенів поліномів повинна виконуватися нерівність $\deg N(x) < \deg P(x)$.

Суть одного з методів поліноміального симетричного шифрування в ПСЗК [313] полягає в тому, що при відновленні полінома за його залишками у сумі (4.35) множення відбувається не на параметри $m_i(x) = M_i^{-1}(x)\, mod\, p_i\ (x)$, а на довільно вибрані поліноми $k_i(x)$, взаємно прості з $p_i(x)$.

Отже, для генерування ключів обидва абоненти повинні вибрати відомі тільки їм обом системи модулів $p_i(x)$ та відповідні поліноми $k_i(x)$, для яких виконуються такі умови: $1 < deg\, k_i\ (x) < deg\, p_i\ (x)$ та НСД $(k_i(x), p_i(x)) = 1$. Якщо $p_i(x)$ є незвідним поліномом, то друга умова виконується завжди. Відповідно, і відправнику, і отримувачу відомі параметри $M_i(x)$ та $m_i(x)$.

Для шифрування буквенну інформацію необхідно записати у числовій формі. Найпоширенішим класичним методом є заміна букви на її номер в алфавіті, причому нумерація починається з 0. Після цього її необхідно представити у вигляді поліному з коефіцієнтами, які відображають буквенну інформацію, тобто відкритий текст $N(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 x^o$, де a_i - послідовність цифрового представлення букв, $i = \overline{0 \ldots n}$, ($n$+1) – довжина повідомлення. Далі блок відкритого тексту $N(x)$ записується в ПСЗК згідно виразу (4.34). Шифрування відбувається при відновленні числа в позиційну систему числення згідно такого виразу:

$$N'(x) = \left(\sum_{i=1}^{s} b_i(x) M_i(x) k_i(x)\right) mod\, P(x). \quad (4.36)$$

Знайдений поліном є шифртекстом, який передається по відкритому каналі зв'язку від одного абонента до іншого.

При розшифруванні спочатку обчислюються такі значення:

$$q_i(x) = \left(m_i(x)\left(k_i^{-1}(x) \bmod p_i(x)\right)\right) mod\, p_i(x);\ b_i'(x) = N'(x) \bmod p_i(x) \quad (4.37)$$

Для отримання істинних залишків $b_i(x)$ необхідно виконати перетворення згідно співвідношення:

$$b_i(x) = \left(b_i'(x) q_i(x)\right) mod\, p_i(x) = \left(b_i'(x) m_i(x) k_i^{-1}(x)\right) mod\, p_i(x). \quad (4.38)$$

Відповідно, відновлення полінома $N(x)$, який є відкритим текстом, здійснюється згідно формули (4.35) або можна використати вираз, який з неї випливає:

$$N(x) = \left(\sum_{i=1}^{s} M_i(x) m_i(x) \left(\left(b_i'(x) m_i(x) k_i^{-1}(x)\right) mod\, p_i(x)\right)\right) mod\, P(x) =$$

$$= \left(\sum_{i=1}^{s} M_i(x) m_i(x) \left(\left(b_i'(x) q_i(x)\right) mod\, p_i(x)\right)\right) mod\, P(x). \quad (4.39)$$

На рисунку 4.7 представлена схема запропонованого поліноміального методу шифрування на основі КТЗ.

Коректність запропонованої криптосистеми встановлюється з властивостей конгруенцій, врахувавши подільність $P(x)$ на $p_i(x)$ та рівність $m_i(x) = M_i^{-1}(x)\, mod\, p_i\,(x)$.

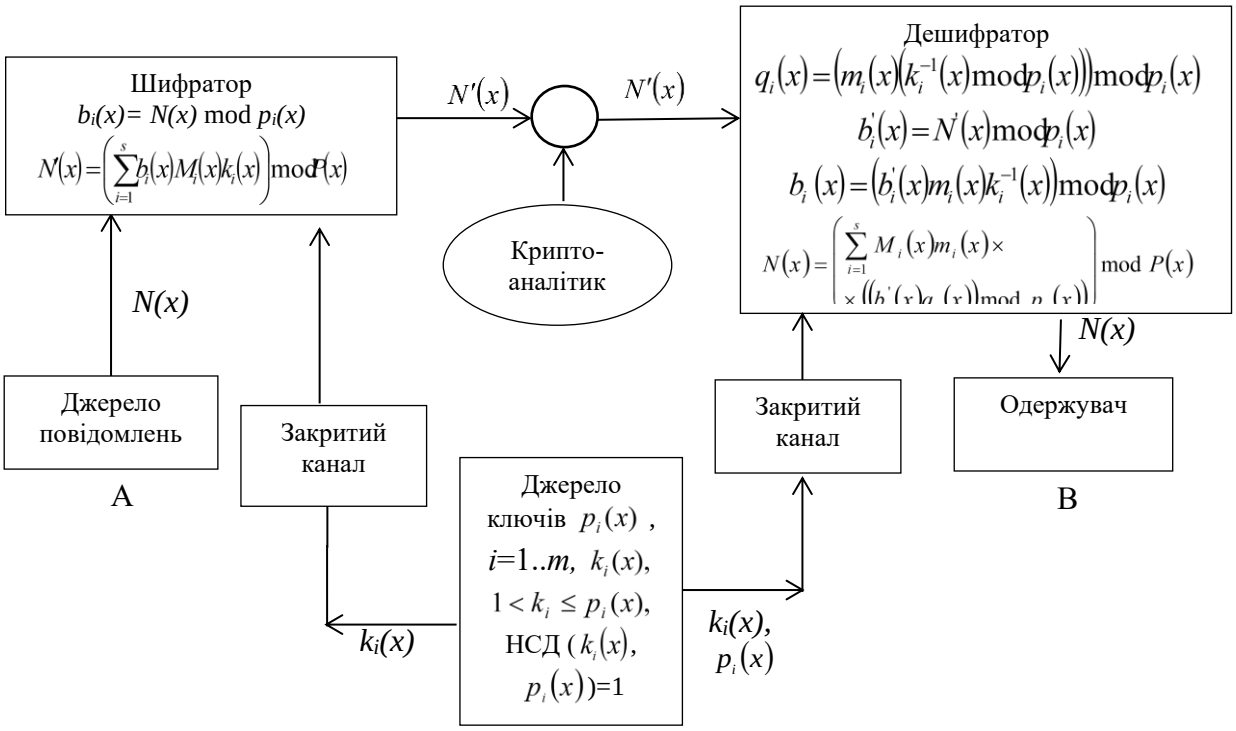


Рисунок 4.7 - Схема запропонованого поліноміального симетричного шифрування в ПСЗК

Тоді отримуємо:

$$b_i(x) = \left(b_i'(x) q_i(x)\right) mod\, p_i\,(x) =$$

$$\left(\left(N'^{(x)\, mod\, p_i(x)}\right)\left(m_i(x) k_i^{-1}(x)\right) mod\, p_i\,(x)\right) mod\, p_i\,(x) =$$

$$\left(\begin{array}{c}\left(\left(\sum_{j=1}^{s} b_j(x) k_j(x) M_j(x)\right) mod\, P\,(x)\right) mod\, p_i\,(x) \\ \left(m_i(x) k_i^{-1}(x)\right) mod\, p_i\,(x)\end{array}\right) mod\, p_i\,(x) =$$

$$\left(\left(b_i(x) k_i(x) M_i(x)\right) mod\, P\,(x) \cdot \left(m_i(x) k_i^{-1}(x)\right) mod\, p_i\,(x)\right) mod\, p_i\,(x) =$$

$$= \left(b_i(x) m_i(x) M_i(x)\right) mod\, p_i\,(x) = b_i(x). \qquad (4.40)$$

На рисунку 4.8 представлено блок-схему симетричного поліноміального алгоритму шифрування у ПСЗК.

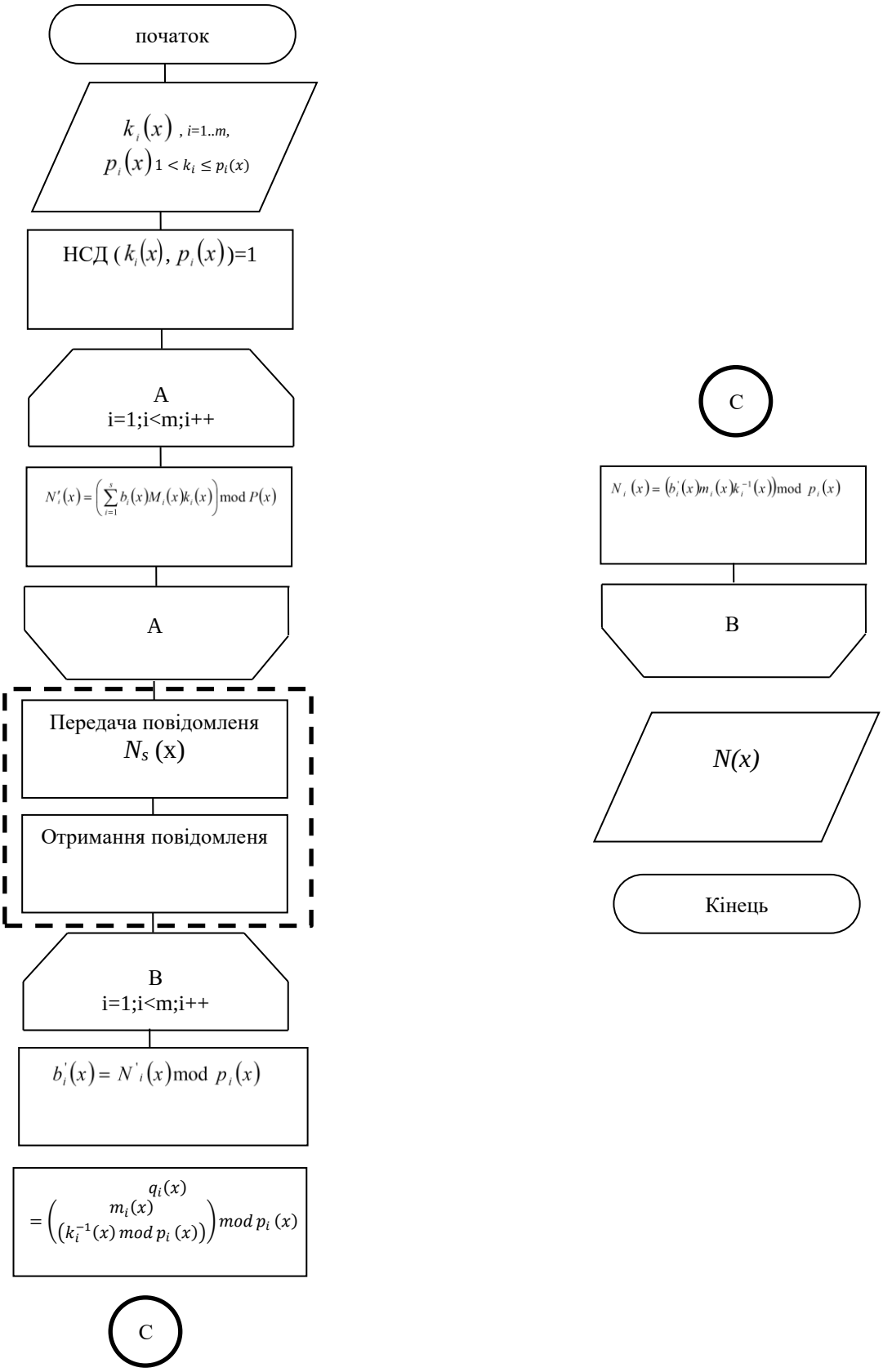


Рисунок 4.8 – Блок-схема симетричного поліноміального алгоритму шифрування у ПСЗК

4.4.1 Приклад симетричного поліноміального шифрування у ПСЗК

Нехай потрібно зашифрувати відкритий текст (polynomial symmetric methods of encryption in the system of residual classes (PSMESRC)) PSMESRC =(15181205181703), який відповідає поліному $N(x) = 15x^6 + 18x^5 + 12x^4 + 5x^3 + 18x^2 + 17x + 3$. Згідно розробленої поліноміальної симетричної криптосистеми для трьох модулів (s=3) $p_1(x)=x^2+x+1$, $p_2(x)= x^3+x+1$, $p_3(x)= x^2+1$ та вибраних коефіцієнтів $k_1(x)=x^2+2x+3$, $k_2(x)= x^3+x^2+1$, $k_3(x)= x^2+3x+2$ розраховуються всі параметри: $P(x) = p_1(x) \cdot p_2(x) \cdot p_3(x) = x^7 + x^6 + 3x^5 + 3x^4 + 4x^3 + 3x^2 + 2x + 1$,

$M_1(x) = \frac{P(x)}{p_1(x)} = \frac{x^7+x^6+3x^5+3x^4+4x^3+3x^2+2x+1}{x^2+x+1} = x^5 + 2x^3 + x^2 + x + 1,$

$M_2(x) = \frac{P(x)}{p_2(x)} = \frac{x^7+x^6+3x^5+3x^4+4x^3+3x^2+2x+1}{x^3+x+1} = x^4 + x^3 + 2x^2 + x + 1,$

$M_3(x) = \frac{P(x)}{p_3(x)} = \frac{x^7+x^6+3x^5+3x^4+4x^3+3x^2+2x+1}{x^2+1} = x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1,$

Пошук $m_i(x) = M_i^{-1}(x)\, mod\, p_i\,(x)$ здійснюється на основі методу невизначених коефіцієнтів. Спочатку шукається $m_1(x) = M_1^{-1}(x)\, mod\, p_1\,(x) = (x^5 + 2x^3 + x^2 + x + 1)^{-1}\, mod(x^2 + x + 1) =$ $= (x + 2)^{-1}\, mod(x^2 + x + 1)$. Для цього записується співвідношення $(x + 2)(Ax + B)\, mod(x^2 + x + 1) = 1$ і після перетворення отримується $(Ax^2 + (2A + B)x + 2B)\, mod(x^2 + x + 1) = (A + B)x + 2B - A = 1$. З останньої рівності випливає, що $2B - A = 1, B + A = 0$, звідки $A = \frac{1}{3}, B = \frac{2}{3}$.

Отже, шуканий обернений поліном приймає вигляд $(x + 2)^{-1}\, mod(x^2 + x + 1) = \frac{1}{3}x + \frac{2}{3}$.

Аналогічним чином здійснюється пошук $m_2(x) = M_2^{-1}(x)\, mod\, p_2\,(x) = (x^4 + x^3 + 2x^2 + x + 1)^{-1}\, mod(x^3 + x + 1) =$ $(x^2 - x)^{-1}\, mod(x^3 + x + 1)$. Для цього записується $(x^2 - x)(Ax^2 + Bx + C)\, mod(x^3 + x + 1) = 1$, де $Ax^2 + Bx + C$ - обернений поліном за модулем.

Потрібно знайти коефіцієнти A, B, C , які задовольняють останню рівність. Після перетворення $(2Ax^4 + 2Bx^3 + (A + 2C)x^2 + Bx +$

$C)\,mod(x^3+x+1) =$ $= (\mathrm{C}-A-B)x^2+(-C-B)x+A-B=1$ отримується $2\mathrm{C}-A$ =0, $C-B-A=0$, $-C-B=0$, $A-B=1$. Звідси $A=\frac{2}{3}$, $B=-\frac{1}{3}$, $C=\frac{1}{3}$. Отже, шуканий обернений поліном набуває такого вигляду: $(x^2-x)^{-1}\,mod(x^3+x+1)=\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}$. Таким же чином обчислюється значення $m_3(x)=M_3^{-1}(x)\,mod\,p_3\,(x)=(x^5+x^4+2x^3+2x^2+2x+1)^{-1}\,mod(x^2+1)=$ $=(x)^{-1}\,mod(x^2+1)$. Застосувавши метод невизначених коефіцієнтів, можна виконати такі перетворення:$(x)(Ax+B)\,mod(x^2+1)=(Ax^2+Bx)\,mod(x^2+1)\Rightarrow Bx-A=1$. Остання рівність приводить до системи рівнянь, яка дає змогу обчислити значення коефіцієнтів $A=-1, B=0$ і тим самим знайти обернений поліном за модулем $m_3(x)=-x$. Тоді:

$b_1(x)=N(x)\,mod\,p_1\,(x)=(15x^6+18x^5+12x^4+5x^3+18x^2+17x+3)\,mod(x^2+x+1)=$ $=-7x-13$, $b_2(x)=N(x)\,mod\,p_2\,(x)=(15x^6+18x^5+12x^4+5x^3+18x^2+17x+3)\,mod(x^3+x+1)=3x^2+48x+31$,

$b_3(x)=N(x)\,mod\,p_3\,(x)=(15x^6+18x^5+12x^4+5x^3+18x^2+17x+3)\,mod(x^2+1)==30x-18$.

Отже, згідно виразу (4.36), шифртекстом є таке значення:

$$N'(x)=\left(\sum_{i=1}^{s}b_i(x)M_i(x)k_i(x)\right)mod\,P\,(x)$$

$$=((x^5+2x^3+x^2+x+1)\cdot(-7x-13)(x^2+2x+3)+$$

$$+(x^4+x^3+2x^2+x+1)(3x^2+48x+31)(x^3+x^2+1)+$$

$$+(x^5+x^4+2x^3+2x^2+2x+1)(30x-18)(x^2+3x+2))\,mod(\,x^7+x^6+3x^5+3x^4+4x^3+3x^2+2x+1)=$$

$$=\text{-}64x^6\text{-}250x^5\text{-}360x^4\text{-}545x^3\text{-}492x^2\text{-}403x\text{-}172$$

Параметри $k_i^{-1}(x)\,mod\,p_i\,(x)$ обчислюються на основі методу невизначених коефіцієнтів. Для пошуку оберненого полінома $k_1^{-1}(x)\,mod\,p_1\,(x)=(x^2+2x+3)^{-1}\,mod(x^2+x+1)=(x+2)^{-1}\,mod(x^2+x+1)$ записується $(x+2)(Ax+B)\,mod(x^2+x+1)=1\Rightarrow$ $\Rightarrow(Ax^2+(2A+B)x+2B)\,mod(x^2+x+1)=(A+B)x+2B-A=1$, де $(Ax+B)$ -

шукане значення. Для знаходження коефіцієнтів A, B обчислюється залишок і прирівнюються відповідні значення: $(A+B)x+2B-A=1$, звідки $A=-\frac{1}{3},\ B=\frac{1}{3}$. Отже, $k_1^{-1}(x)\ mod\ p_1\ (x)=\frac{1}{3}(-x+1)$.

Аналогічно здійснюється пошук $k_2^{-1}(x)\text{mod}\ p_2(x)=\left(x^3+x^2+1\right)^{-1}\ \text{mod}\left(x^3+x+1\right)=\left(x^2-x\right)^{-1}\ \text{mod}\left(x^3+x+1\right)$ $(x^2-x)(Ax^2+Bx+C)\ mod(x^3+x+1)=1$, де Ax^2+Bx+C - обернений поліном за модулем. Після перетворення $(2Ax^4+2Bx^3+(A+2C)x^2+Bx+C)\ mod(x^3+x+1)=(\text{C}-A-B)x^2+(-C-B)x+\ +A-B=1$ отримується система з трьох рівнянь з трьома невідомими $C-B-A=0, -C-B=0, A-B=1$. Звідси $A=\frac{2}{3},\ B=-\frac{1}{3},\ C=\frac{1}{3}$. Отже, шуканий обернений поліном в Z[x] прийме вигляд: $(x^2-x)^{-1}\ mod(x^3+x+1)=\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}$, $k_3^{-1}(x)\ mod\ p_3\ (x)=-\frac{3}{10}x+\frac{1}{10}$.

На наступному етапі обчислюються такі величини:

$b_1'(x)=N'(x)\ mod\ p_1\ (x)=$ (-64x^6-250x^5-360x^4-545x^3 −

-492x^2-403x-172) $mod(x^2+x+1)=-21x-39$;

$b_1'(x)=N'(x)\ mod\ p_1\ (x)=$ (-64x^6-250x^5-360x^4-545x^3 −

-492x^2-403x-172) $mod(x^3+x+1)=54x^2+124x+59$;

$b_1'(x)=N'(x)\ mod\ p_1\ (x)=$ (-64x^6-250x^5-360x^4-545x^3 −

-492x^2-403x-172) $mod(x^2+1)=-108x+24$.

Крім того, для розшифрування необхідно знайти ще такі параметри: $q_1(x)=\left(m_1(x)\left(k_1^{-1}(x)\ mod\ p_1\ (x)\right)\right)mod\ p_1\ (x)=\left(\left(\frac{1}{3}x+\frac{2}{3}\right)\left(-\frac{1}{3}x+\frac{1}{3}\right)\right)mod(x^2+x+1)=\frac{1}{3}$,

$q_2(x)=\left(m_2(x)\left(k_2^{-1}(x)\ mod\ p_2\ (x)\right)\right)mod\ p_2\ (x)=\left(\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}\right)\times\left(\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}\right)mod(x^3+x+1)=\left(-\frac{10}{27}x^4+\frac{7}{27}x^3-\frac{20}{27}x^2+\frac{8}{27}x-\frac{7}{27}\right)mod(x^3+x+1)=\ =\frac{1}{9}x^2-\frac{2}{9}x+\frac{5}{9}$.

$$q_3(x) = \left(m_3(x)\left(k_3^{-1}(x)\, mod\, p_3\,(x)\right)\right) mod\, p_3\,(x) = \left((-x)\left(-\frac{3}{10}x + \frac{1}{10}\right)\right) mod(x^3 + x^2 + x + 3) = = \left(\frac{3}{10}x^2 - \frac{1}{10}x\right) mod(x^2 + 1) = -\frac{1}{10}x - \frac{3}{10}.$$

Тоді за формулою (4.39) відновлюється вхідне повідомлення, яке є відкритим текстом:

$$N(x) = \left(\sum_{i=1}^{s} M_i(x)m_i\,(x)\left(\left(b_i^{'}(x)m_i(x)k_i^{-1}(x)\right) mod\, p_i\,(x)\right)\right) mod\, P\,(x) =$$

$$\left(\sum_{i=1}^{s} M_i(x)m_i(x)\left(\left(b_i^{'}(x)q_i(x)\right) mod\, p_i\,(x)\right)\right) mod\, P\,(x) =$$

$$= (\left((x^5 + 2x^3 + x^2 + x\ \ + 1)\left(\frac{1}{3}x + \frac{2}{3}\right)\left(\left(\frac{1}{3}\right)(-21x - 39)\right) mod(x^2 + x + 1)\right) + \left((x^4 + x^3 + 2x^2 + x + 1)\left(\frac{2}{3}x^2 - \frac{1}{3}x + \frac{1}{3}\right)\right)$$

$$\times \left(\left(\frac{1}{9}x^2 - \frac{2}{9}x + \frac{5}{9}\right)(54x^2 + 124x + 59)\right) mod(x^3 + x + 1)$$

$$+(x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)(-x) \times$$

$$\times \left((-108x + 24)\left(-\frac{1}{10}x - \frac{3}{10}\right)\right) mod(x^2 + 1))\, mod\, x^7 + x^6 + 3x^5 + 3x^4 +$$

$$4x^3 + 3x^2 + 2x + 1 = 15x^6 + 18x^5 + 12x^4 + 5x^3 + 18x^2 + 17x + 3$$

Пришвидшення процесу шифрування та розшифрування можна досягнути у випадку, коли абоненти виберуть параметри $k_i(x) = 1$. Однак це призведе до зменшення стійкості криптосистеми. Процес шифрування відбувається за спрощеною формулою:

$$N^{'}(x) = (\sum_{i=1}^{s} b_i(x)M_i(x))\, mod\, P\,(x). \qquad (4.41)$$

Слід відмітити, що в процесі розшифрування усувається операція пошуку оберненого полінома за модулем та множення на нього, оскільки $q_i(x) = m_i(x)$. Відновлення відкритого тексту відбувається на основі таких співвідношень:

$$b_i(x) = \left(b_i^{'}(x)m_i(x)\right) mod\, p_i\,(x);$$

$$N(x) = \left(\sum_{i=1}^{s} M_i(x)m_i\,(x)\left(\left(b_i^{'}(x)m_i(x)\right) mod\, p_i\,(x)\right)\right) mod\, P\,(x). \qquad (4.42)$$

Для таких самих вхідних параметрів, що і у попередньому прикладі, згідно формул (4.41) і (4.42), отримується такий шифртекст:

$$N^{'}(x) = \left(\sum_{i=1}^{s} b_i(x)M_i(x)\right) mod\, P\,(x) = ((x^5 + 2x^3 + x^2 + x + 1) \cdot (-7x - 13)$$

$$+(x^4 + x^3 + 2x^2 + x + 1)(3x^2 + 48x + 31) + (x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)$$

$$\cdot (30x - 18))\, mod(x^7 + x^6 + 3x^5 + 3x^4 + 4x^3 + 3x^2 + 2x + 1) =$$

$$= 26x^6 + 50x^5 + 113x^4 + 121x^3 + 117x^2 + 53x.$$

Для розшифрування необхідно обчислити додаткові параметри згідно формули (4.37):

$$b_1^{'}(x) = N^{'}(x)\, mod\, p_1\,(x) = = (26x^6 + 50x^5 + 113x^4 + 121x^3 + 117x^2 +$$

$$53x)\, mod(x^2 + x + 1) = -x - 20;$$

$$b_2^{'}(x) = N^{'}(x)\, mod\, p_2\,(x)(26x^6 + 50x^5 + 113x^4 + 121x^3 + 117x^2 +$$

$$53x)\, mod(x^3 + x + 1) = -20x^2 - 79x - 45;$$

$$b_3^{'}(x) = N^{'}(x)\, mod\, p_3\,(x) = (26x^6 + 50x^5 + 113x^4 + 121x^3 + 117x^2 +$$

$$53x)\, mod(x^2 + 1) = -18x - 30.$$

Процес розшифрування здійснюється згідно формули (4.42):

$$N(x)=\left(\sum_{i=1}^{s}M_i(x)m_i(x)\left(\left(b_i^{'}(x)m_i(x)\right)\bmod p_i(x)\right)\right)\bmod P(x)=$$

$$=\left(\begin{array}{l}\left(\begin{array}{l}\left(x^5+2x^3+x^2+x+1\right)\left(\frac{1}{3}x+\frac{2}{3}\right)\times\\ \times\left(\left(\frac{1}{3}x+\frac{2}{3}\right)(-x-20)\right)\bmod\left(x^2+x+1\right)\end{array}\right)+\\ +\left(\begin{array}{l}\left(x^4+x^3+2x^2+x+1\right)\left(\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}\right)\times\\ \times\left(\left(\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}\right)\left(-20x^2-79x-45\right)\right)\bmod\left(x^3+x+1\right)\end{array}\right)+\\ +\left(\begin{array}{l}\left(x^5+x^4+2x^3+2x^2+2x+1\right)(-x)\times\\ \times\left((-18x-30)(-x)\right)\bmod\left(x^2+1\right)\end{array}\right)\end{array}\right)\bmod\left(\begin{array}{l}x^7+x^6+3x^5+3x^4+\\ +4x^3+3x^2+2x+1\end{array}\right)=$$

$$=15x^6+18x^5+12x^4+5x^3+18x^2+17x+3.$$

Таке спрощення призводить до зменшення обчислювальної складності за рахунок уникнення операції пошуку параметрів $q_i(x)$ та $k_i(x)^{-1}\,mod\,p_i$.

4.4.2 Поліноміальний симетричний метод шифрування на основі КТЗ

Інший поліноміальний метод симетричного шифрування на основі КТЗ полягає в тому, що відкритий текст $N(x)$ розбивається на блоки – поліноми $N_i(x)$ меншого порядку, ніж вибрані поліноміальні модулі [314-316]. Ці блоки виступатимуть залишками $b_i(x)$ по вибраних модулях, причому, якщо $deg\,p_i\,(x)=n$, то $deg\,N_i\,(x)\le n-1$, тобто $N_i(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_0x^o$. Після вибору параметрів $k_i(x)$ шифрування відбувається згідно з виразом (4.36). Шифртекстом буде значення $N^{'}(x)$.

Розшифрування відбувається за формулами (4.37), (4.38), згідно яких шукаються параметри $q_i(x)$, $b_i(x)=N_i(x)\,mod\,p_i\,(x)=N_i(x)$, та $b'_i(x)$. Конкатенація коефіцієнтів a_{n-1} поліномів $N_i(x)$ утворює відкритий текст. Слід зазначити, що у випадку, коли потрібне швидке розшифрування, шифртекстом можуть виступати також параметри $b^{'}_i(x)$.

На рисунку 4.9 представлено схему поліноміального симетричного методу шифрування на основі КТЗ.

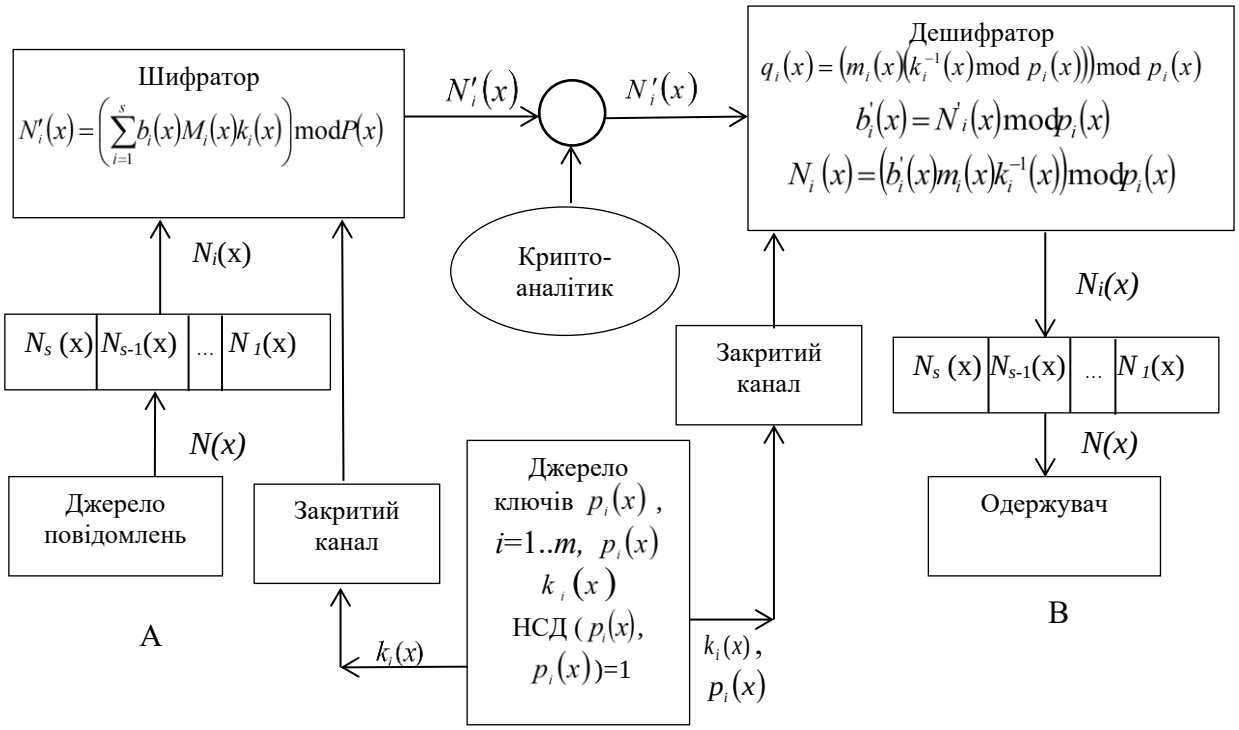


Рисунок 4.9 - Поліноміальний симетричний метод шифрування на основі КТЗ

Для шифрування даним методом вибраний вище відкритий текст (polynomial symmetric methods of encryption in the system of residual number classes (PSMESRNC)) PSMESRNC =(15181205181713 03) розбивається на чотири блоки по два символи: PS=1518, ME=1205, SR =1817, NC=1303. Далі формуються відкриті тексти у вигляді поліномів: $N_1(x) = 15x + 18$, $N_2(x) = 12x + 5$, $N_3(x) = 18x + 17$ і $N_4(x) = 13x + 3$, які є залишками по обраним модулях $p_1(x)=x^2+x+1$, $p_2(x)= x^3+x+1$, $p_3(x)= x^2+1$, та вибираються відповідні коефіцієнти $k_1(x)=x^2+2x+3$, $k_2(x)= x^3+x^2+1$, $k_3(x)= x^2+3x+2$. На основі формули (4.41) обчислюється шифртекст для першого блоку:

$$N_1(x)' = \left(\sum_{i=1}^{s} b_i(x)M_i(x)k_i(x)\right) mod\ P(x) = ((15x + 18)(x^5 + 2x^3 + x^2 + x + 1)(x^2 + 2x + 3) + (x^4 + x^3 + 2x^2 + x + 1)(15x + 18)(x^3 + x^2 + 1) + (x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)(15x + 18)(x^2 + 3x + 2)) mod \begin{pmatrix} x^7 + x^6 + 3x^5 + 3x^4 + \\ +4x^3 + 3x^2 + 2x + 1 \end{pmatrix} = 105x^6 + 33x^5 + 135x^4 + 81x^3 + 78x^2 + 21x - 21$$

При розшифруванні за формулами (4.37), (4.38) отримуються такі результати: $b_1^{1\prime}(x) = 45x + 54$, $b_2^{1\prime}(x) = 15x^2 + 48x + 36$, $b_3^{1\prime}(x) = -27x - 69$. Відновлення повідомлення відбувається на основі співвідношення (4.39):

$$b_i^1(x) = \left(b_i^{1\prime}(x)q_i(x)\right) mod\, p_i\,(x) = \left(b_i^{1\prime}(x)m_i(x)k_i^{-1}(x)\right) mod\, p_i\,(x),\ b_1^1(x) =$$

$$(45x + 54)\left(\frac{1}{3}x + \frac{2}{3}\right)\left(-\frac{1}{3}x + \frac{1}{3}\right) mod(x^2 + x + 1) = (45x + 54) \times$$

$$\times \left(-\frac{1}{9}x^2 - \frac{1}{9}x + \frac{2}{9}\right) mod(x^2 + x + 1) = (-5x^3 - 11x^2 + 4x + 12)\, mod(x^2 +$$

$$x + 1) = 15x + 18$$

Для другого блоку вхідного повідомлення $N_2(x) = 12x + 5$ отримується таке значення шифртескту:

$$N_2(x)' = \left(\sum_{i=1}^{s} b_i(x)M_i(x)k_i\,(x)\right) mod\, P\,(x) = ((12x + 5)(x^5 + 2x^3 + x^2 +$$

$$x + 1)(x^2 + 2x + 3) + \ (x^4 + x^3 + 2x^2 + x + 1)(12x + 5)(x^3 + x^2 + 1) +$$

$$+(x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)(12x + 5)(x^2 + 3x + 2))\, mod(x^7 + x^6 +$$

$$3x^5 + 3x^4 + 4x^3 + 3x^2 + 2x + 1) = \ 37x^6 - 30x^5 + 14x^4 - 48x^3 - 41x^2 -$$

$$49x - 45$$

Згідно формули (4.37) отримуються залишки: $b_1^{2\prime}(x) = 36x + 15$ $b_2^{2\prime}(x) = 12x^2 + 29x + 10$, $b_3^{2\prime}(x) = -31x - 27$. Відновлення другого блоку вхідного повідомлення відбувається згідно співвідношення (4.39):

$$b_1^2(x) = (36x + 15)\left(\frac{1}{3}x + \frac{2}{3}\right)\left(-\frac{1}{3}x + \frac{1}{3}\right) mod(x^2 + x + 1) = (36x + 15) \times$$

$$\times \left(-\frac{1}{9}x^2 - \frac{1}{9}x + \frac{2}{9}\right) mod(x^2 + x + 1) = \left(-4x^3 - \frac{17}{3}x^2 + \frac{19}{3}x + \frac{10}{3}\right) mod(x^2 +$$

$$x + 1) = 12x + 5.$$

Шифртекст для третього блоку вхідного повідомлення $N_3(x) = 18x + 17$ буде мати такий вигляд:

$$N_3(x)' = \left(\sum_{i=1}^{s} b_i(x)M_i(x)k_i\,(x)\right) mod\, P\,(x) = ((18x + 17)(x^5 + 2x^3 + x^2 +$$

$$x + 1)(x^2 + 2x + 3) + (x^4 + x^3 + 2x^2 + x + 1)(18x + 17)(x^3 + x^2 + 1) +$$

$$(x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1) \times (18x + 17)(x^2 + 3x + 2))\, mod(x^7 + x^6 +$$

$$3x^5 + 3x^4 + 4x^3 + +3x^2 + 2x + 1) =$$

$$= 103x^6 + 12x^5 + 116x^4 + 42x^3 + 43x^2 - 7x - 39$$

Тоді, згідно формули (4.37), отримуються залишки: $b_1^{3\prime}(x) = 54x + 51$, $b_2^{3\prime}(x) = 18x^2 + 53x + 34$, $b_3^{3'}(x) = -37x - 69$. Відновлення відбувається з застосуванням співвідношення (4.38):

$$b_i^3(x) = \left(b_i^{3'}(x)q_i(x)\right) mod\, p_i\,(x) = \left(b_i^{3'}(x)m_i(x)k_i^{-1}(x)\right) mod\, p_i\,(x).$$

Звідси

$$b_1^3(x) = (54x + 51)\left(\frac{1}{3}x + \frac{2}{3}\right)\left(-\frac{1}{3}x + \frac{1}{3}\right) mod(x^2 + x + 1) = (54x + 51) \times$$

$$\times \left(-\tfrac{1}{9}x^2 - \tfrac{1}{9}x + \tfrac{2}{9}\right) mod(x^2 + x + 1) = \left(-6x^3 - \tfrac{35}{3}x^2 + \tfrac{19}{3}x + \tfrac{34}{3}\right) mod(x^2 +$$

$$x + 1) = 18x + 17.$$

Відповідно, зашифрованим повідомленням четвертого блоку $N_4(x) = 13x + 3$ згідно (4.36) буде такий поліном:

$$N_4(x)' = \left(\sum_{i=1}^{s} b_i(x)M_i(x)k_i\,(x)\right) mod\, P\,(x) = ((13x + 3)(x^5 + 2x^3 + x^2 + x +$$

$$1)(x^2 + 2x + 3) + \ (x^4 + x^3 + 2x^2 + x + 1)(13x + 3)(x^3 + x^2 + 1) + (x^5 + x^4 +$$

$$2x^3 + 2x^2 + 2x + 1) \times$$

$$\times (13x + 3)(x^2 + 3x + 2))\, mod(x^7 + x^6 + 3x^5 + 3x^4 + 4x^3 + 3x^2 + 2x + 1)$$

$$= 28x^6 - 47x^5 - 9x^4 - 81x^3 - 71x^2 - 70x - 56$$

Тоді, згідно (4.37), отримуються такі залишки: $b_1^{4\prime}(x) = 39x + 9$ $b_2^{4\prime}(x) = 13x^2 + 29x + 6$, $b_3^{4'}(x) = -36x - 22$. Відновлення четвертого блоку відбувається на основі виразу (4.38):

$$b_i^4(x) = \left(b_i^{4\prime}(x)q_i(x)\right) mod\, p_i\,(x) = \left(b_i^{4\prime}(x)m_i(x)k_i^{-1}(x)\right) mod\, p_i\,(x),$$

$$b_1^4(x) = (39x + 9)\left(\tfrac{1}{3}x + \tfrac{2}{3}\right)\left(-\tfrac{1}{3}x + \tfrac{1}{3}\right) mod(x^2 + x + 1) = (39x + 9)$$

$$\left(-\tfrac{1}{9}x^2 - \tfrac{1}{9}x + \tfrac{2}{9}\right) mod(x^2 + x + 1) = \left(-\tfrac{13}{3}x^3 - \tfrac{16}{3}x^2 + \tfrac{23}{3}x + 2\right) mod(x^2 +$$

$$x + 1) = 13x + 3$$

Конкатенація коефіцієнтів залишків $b_i(x)$ відповідає вхідному тексту PSMESRNC =(15181205181713O3). Згідно домовленостей між абонентами, шифртекстом може виступати або параметр $N_i(x)'$, або залишки $b_i^{j'}(x)$, де j – номер блоку повідомлення. Блок-схема розробленого поліноміального симетричного методу шифрування на основі КТЗ представлено на рисунку 4.10.

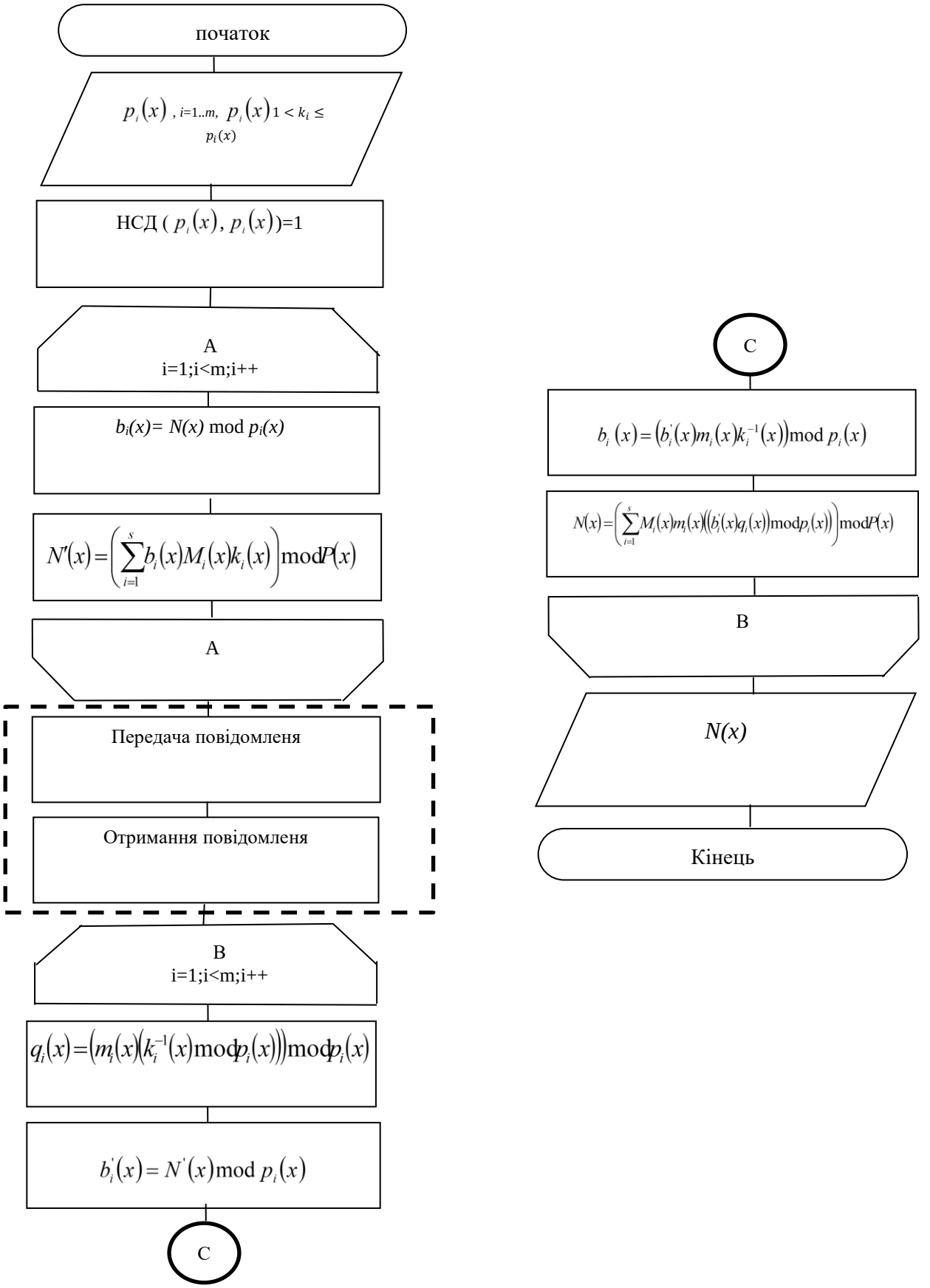


Рисунок 4.10 – Блок-схема симетричного поліноміального алгоритму шифрування на основі КТЗ

4.4.3 Оцінка криптостійкості поліноміального симетричного алгоритму шифрування в системі залишкових класів

Запропонований поліноміальний симетричний метод шифрування на основі ПСЗК є криптостійким через складність пошуку всіх можливих варіантів параметрів та модулів криптоперетворень. Для його криптоаналізу необхідно здійснити повний перебір всіх взаємнопростих поліномів в кільці $Z[x]$ над простим полем Галуа $GF(p)$, де p - просте число. Найбільша складність буде у випадку, якщо поліном $f(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 x^0$ є незвідним. Кількість $S_p(n)$ незвідних поліномів $f(x)$ степеня n можна обчислити за такою формулою:

$$S_p(n) = \frac{1}{n}\sum_{n|d} \mu(d) p^{n/d}, \tag{4.43}$$

де $\mu(d)$ - функція Мебіуса [317, 318]. Вона дорівнює 1, якщо d – дільник степеня n з парною кількістю простих множників; -1, якщо d - дільник степеня n з непарною кількістю простих множників; 0, якщо d містить квадрат простого множника. Відповідно, кількість модулів l не може перевищувати $S_p(n)$.

В таблиці 4.1 наведено значення кількості незвідних поліномів для степенів n=4, 8,16, 32, 64, 96, 128 і параметра p=3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101.

Отже, щоб знайти кількість незвідних поліномів $f(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 x^0$ над $GF(p)$, потрібно визначити всі дільники степеня n, для кожного з них обчислити функцію Мебіуса, підставити їх у формулу (4.43) та просумувати.

Наприклад, для незвідних поліномів над $GF(p)$ степеня n=3 з дільниками 1 (значення функції Мебіуса дорівнює 1) та 3 (значення функції Мебіуса дорівнює -1) їх кількість можна знайти за формулою:

$$S_p(n) = \frac{1}{3}\sum_{n|d} \mu(d) p^{n/d} = \frac{1}{3}\left(\mu(1)p^{3/1} + \mu(3)p^{3/3}\right) = \frac{(p^3-p)}{3}.$$

Таблиця 4.1

Функція Мебіуса для перших 100 натуральних чисел [319]

d	1	2	3	4	5	6	7	8	9	10
$\mu(d)$	1	-1	-1	0	-1	1	-1	0	0	1
d	11	12	13	14	15	16	17	18	19	20
$\mu(d)$	-1	0	-1	1	1	0	-1	0	-1	0
d	21	22	23	24	25	26	27	28	29	30
$\mu(d)$	1	1	-1	0	0	1	0	0	-1	-1
d	31	32	33	34	35	36	37	38	39	40
$\mu(d)$	-1	0	-1	1	1	0	-1	1	1	0
d	41	42	43	44	45	46	47	48	49	50
$\mu(d)$	-1	-1	-1	0	0	1	-1	0	0	0
d	51	52	53	54	55	56	57	58	59	60
$\mu(d)$	1	0	-1	0	1	0	1	1	-1	0
d	61	62	63	64	65	66	67	68	69	70
$\mu(d)$	-1	1	0	0	1	-1	-1	0	1	-1
d	71	72	73	74	75	76	77	78	79	80
$\mu(d)$	-1	0	-1	1	0	0	1	-1	-1	0
d	81	82	83	84	85	86	87	88	89	90
$\mu(d)$	0	1	-1	0	1	1	1	0	-1	0
d	91	92	93	94	95	96	97	98	99	100
$\mu(d)$	1	0	1	1	1	0	-1	0	0	0
D	1	2	3	4	5	6	7	8	9	10
$\mu(d)$	1	-1	-1	0	-1	1	-1	0	0	1
D	11	12	13	14	15	16	17	18	19	20
$\mu(d)$	-1	0	-1	1	1	0	-1	0	-1	0
D	21	22	23	24	25	26	27	28	29	30
$\mu(d)$	1	1	-1	0	0	1	0	0	-1	-1
D	31	32	33	34	35	36	37	38	39	40
$\mu(d)$	-1	0	-1	1	1	0	-1	1	1	0
D	41	42	43	44	45	46	47	48	49	50
$\mu(d)$	-1	-1	-1	0	0	1	-1	0	0	0
D	51	52	53	54	55	56	57	58	59	60
$\mu(d)$	1	0	-1	0	1	0	1	1	-1	0
D	61	62	63	64	65	66	67	68	69	70
$\mu(d)$	-1	1	0	0	1	-1	-1	0	1	-1
D	71	72	73	74	75	76	77	78	79	80
$\mu(d)$	-1	0	-1	1	0	0	1	-1	-1	0
D	81	82	83	84	85	86	87	88	89	90
$\mu(d)$	0	1	-1	0	1	1	1	0	-1	0
D	91	92	93	94	95	96	97	98	99	100
$\mu(d)$	1	0	1	1	1	0	-1	0	0	0

Для полінома степеня 32 (дільниками є числа 1, 2, 4, 8, 16, 32) над *GF*(2) згідно формули (10) кількість незвідних поліномів буде така:

$$S_2(32) = \frac{1}{32}\sum_{32|d}\mu(d)2^{32/d} = \frac{1}{32}\begin{pmatrix}\mu(1)2^{32/1} + \mu(2)2^{32/2} + \mu(4)2^{32/4} + \\ +\mu(8)2^{32/8} + \mu(16)2^{32/16} + \mu(32)2^{32/32}\end{pmatrix}$$

$$= \frac{1}{32}(2^{32} - 2^{16}) = 2^{27} - 2^{11} = 134215680.$$

Нижче наведено приклад розрахунку кількості незвідних поліномів для *n* = 32 і *GF*(p), де *p*=3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43:

$$S_3(32) = \frac{1}{32}\sum_{32|d}\mu(d)3^{\frac{32}{d}} = \begin{pmatrix}\mu(1)3^{\frac{32}{1}} + \mu(2)3^{\frac{32}{2}} + \mu(4)3^{\frac{32}{4}} + \\ +\mu(8)3^{\frac{32}{8}} + \mu(16)3^{\frac{32}{16}} + \mu(32)3^{\frac{32}{32}}\end{pmatrix} =$$

$$= \frac{1}{32}(3^{32} - 3^{16}) = 57906879556410$$

$$S_5(32) = \frac{1}{32}\sum_{32|d}\mu(d)5^{\frac{32}{d}} = \begin{pmatrix}\mu(1)5^{\frac{32}{1}} + \mu(2)5^{\frac{32}{2}} + \mu(4)5^{\frac{32}{4}} + \\ +\mu(8)5^{\frac{32}{8}} + \mu(16)5^{\frac{32}{16}} + \mu(32)5^{\frac{32}{32}}\end{pmatrix} =$$

$$= \frac{1}{32}(5^{32} - 5^{16}) = 727595761413574*10^{6}$$

$$S_7(32) = \frac{1}{32}\sum_{32|d}\mu(d)7^{\frac{32}{d}} = \begin{pmatrix}\mu(1)7^{\frac{32}{1}} + \mu(2)7^{\frac{32}{2}} + \mu(4)7^{\frac{32}{4}} + \\ +\mu(8)7^{\frac{32}{8}} + \mu(16)7^{\frac{32}{16}} + \mu(32)7^{\frac{32}{32}}\end{pmatrix} =$$

$$= \frac{1}{32}(7^{32} - 7^{16}) = 34513364820121*10^{11}$$

$$S_{11}(32) = \frac{1}{32}\sum_{32|d}\mu(d)11^{\frac{32}{d}} = \begin{pmatrix}\mu(1)11^{\frac{32}{1}} + \mu(2)11^{\frac{32}{2}} + \mu(4)11^{\frac{32}{4}} + \\ +\mu(8)11^{\frac{32}{8}} + \mu(16)11^{\frac{32}{16}} + \mu(32)11^{\frac{32}{32}}\end{pmatrix} =$$

$$= \frac{1}{32}(11^{32} - 11^{16}) = 65980552329226*10^{17}$$

$$S_{13}(32) = \frac{1}{32}\sum_{32|d}\mu(d)13^{\frac{32}{d}} = \begin{pmatrix}\mu(1)13^{\frac{32}{1}} + \mu(2)13^{\frac{32}{2}} + \mu(4)13^{\frac{32}{4}} + \\ +\mu(8)13^{\frac{32}{8}} + \mu(16)13^{\frac{32}{16}} + \mu(32)13^{\frac{32}{32}}\end{pmatrix} =$$

$$= \frac{1}{32}(13^{32} - 13^{16}) = 138368519930263*10^{20}$$

$$S_{17}(32) = \frac{1}{32}\sum_{32|d} \mu(d)17^{\frac{32}{d}} = \begin{pmatrix} \mu(1)17^{\frac{32}{1}} + \mu(2)17^{\frac{32}{2}} + \mu(4)17^{\frac{32}{4}} + \\ +\mu(8)17^{\frac{32}{8}} + \mu(16)17^{\frac{32}{16}} + \mu(32)17^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(17^{32} - 17^{16}) = 7399723733626 46*10^{23}$$

$$S_{19}(32) = \frac{1}{32}\sum_{32|d} \mu(d)19^{\frac{32}{d}} = \begin{pmatrix} \mu(1)19^{\frac{32}{1}} + \mu(2)19^{\frac{32}{2}} + \mu(4)19^{\frac{32}{4}} + \\ +\mu(8)19^{\frac{32}{8}} + \mu(16)19^{\frac{32}{16}} + \mu(32)19^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(19^{32} - 19^{16}) = 259995153315273*10^{25}$$

$$S_{23}(32) = \frac{1}{32}\sum_{32|d} \mu(d)23^{\frac{32}{d}} = \begin{pmatrix} \mu(1)23^{\frac{32}{1}} + \mu(2)23^{\frac{32}{2}} + \mu(4)23^{\frac{32}{4}} + \\ +\mu(8)23^{\frac{32}{8}} + \mu(16)23^{\frac{32}{16}} + \mu(32)23^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(23^{32} - 23^{16}) = 117527845345372*10^{28}$$

$$S_{29}(32) = \frac{1}{32}\sum_{32|d} \mu(d)29^{\frac{32}{d}} = \begin{pmatrix} \mu(1)29^{\frac{32}{1}} + \mu(2)29^{\frac{32}{2}} + \mu(4)29^{\frac{32}{4}} + \\ +\mu(8)29^{\frac{32}{8}} + \mu(16)29^{\frac{32}{16}} + \mu(32)29^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(29^{32} - 29^{16}) = 195697804967027*10^{31}$$

$$S_{31}(32) = \frac{1}{32}\sum_{32|d} \mu(d)31^{\frac{32}{d}} = \begin{pmatrix} \mu(1)31^{\frac{32}{1}} + \mu(2)31^{\frac{32}{2}} + \mu(4)31^{\frac{32}{4}} + \\ +\mu(8)31^{\frac{32}{8}} + \mu(16)31^{\frac{32}{16}} + \mu(32)31^{\frac{32}{32}} \end{pmatrix}$$

$$= \frac{1}{32}(31^{32} - 31^{16}) = 165357624391381*10^{32}$$

$$S_{37}(32) = \frac{1}{32}\sum_{32|d} \mu(d)37^{\frac{32}{d}} = \begin{pmatrix} \mu(1)37^{\frac{32}{1}} + \mu(2)37^{\frac{32}{2}} + \mu(4)37^{\frac{32}{4}} + \\ +\mu(8)37^{\frac{32}{8}} + \mu(16)37^{\frac{32}{16}} + \mu(32)37^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(37^{32} - 37^{16}) == 475669375729533*10^{34}$$

$$S_{43}(32) = \frac{1}{32}\sum_{32|d} \mu(d)43^{\frac{32}{d}} = \begin{pmatrix} \mu(1)43^{\frac{32}{1}} + \mu(2)43^{\frac{32}{2}} + \mu(4)43^{\frac{32}{4}} + \\ +\mu(8)43^{\frac{32}{8}} + \mu(16)43^{\frac{32}{16}} + \mu(32)43^{\frac{32}{32}} \end{pmatrix} =$$

$$= \frac{1}{32}(43^{32} - 43^{16}) = 583231000811536*10^{36}$$

В таблиці 4.2 наведено значення кількості незвідних поліномів для степенів *n*=4, 8,16, 32, 64, 96, 128 і параметра *p*=3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43.

Таблиця 4.2

Кількість незвідних поліномів для різних степенів *n* і значень параметра *p*.

p \ *n*	4	8	16	32	64	96	128
2	3	30	4080	134215680	2,8823E+17	8,25293E+26	2,65846E+36
3	18	810	2690010	5,791E+13	5,36513E+28	6,6278E+43	9,21108E+58
5	150	48750	9,54E+09	7,276E+20	8,47033E+42	1,31477E+65	2,29589E+87
7	588	720300	2,08E+12	3,451E+25	1,90588E+52	1,40327E+79	1,1624E+106
11	3630	26793030	2,87E+15	6,598E+31	6,96549E+64	9,80452E+97	1,5526E+131
13	7098	1,02E+08	4,16E+16	1,384E+34	3,06334E+69	9,0425E+104	3,0029E+140
17	20808	8,72E+08	3,04E+18	7,4E+37	8,76095E+76	1,383E+116	2,4561E+155
19	32490	2,12E+09	1,8E+19	2,6E+39	1,08156E+80	5,9989E+120	3,7433E+161
23	69828	9,79E+09	3,83E+20	1,175E+42	2,21005E+85	5,5412E+128	1,563E+172
29	176610	6,25E+10	1,56E+22	1,957E+45	6,12762E+91	2,5582E+138	1,2015E+185
31	230640	1,07E+11	4,55E+22	1,654E+46	4,3749E+93	1,5433E+141	6,1247E+188
37	468198	4,39E+11	7,71E+23	4,757E+48	3,62018E+98	3,6736E+148	4,1938E+198
41	706020	9,98E+11	3,98E+24	1,27E+50	2,5822E+101	6,9981E+152	2,1337E+204
43	854238	1,46E+12	8,54E+24	5,832E+50	5,4425E+102	6,7717E+154	9,4788E+206
47	1219368	2,98E+12	3,54E+25	1,005E+52	1,6147E+105	3,4604E+158	8,3429E+211
53	1971918	7,78E+12	2,42E+26	4,695E+53	3,5276E+108	3,5336E+163	3,982E+218
59	3028470	1,84E+13	1,35E+27	1,452E+55	3,3756E+111	1,046E+168	3,6463E+224
61	3460530	2,4E+13	2,3E+27	4,221E+55	2,8506E+112	2,5668E+169	2,6002E+226
67	5036658	5,08E+13	1,03E+28	8,497E+56	1,1551E+115	2,0937E+173	4,2694E+231
71	6351660	8,07E+13	2,61E+28	5,434E+57	4,7245E+116	5,4768E+175	7,1426E+234
73	7098228	1,01E+14	4,06E+28	1,322E+58	2,7957E+117	7,8836E+176	2,501E+236
79	9735960	1,9E+14	1,44E+29	1,655E+59	4,3848E+119	1,5486E+180	6,1526E+240
83	11862858	2,82E+14	3,17E+29	8,042E+59	1,0347E+121	1,7751E+182	3,426E+243
89	15683580	4,92E+14	9,69E+29	7,505E+60	9,0112E+122	1,4427E+185	2,5984E+247
97	22129968	9,8E+14	3,84E+30	1,179E+62	2,2244E+125	5,5952E+188	1,5833E+252
101	26012550	1,35E+15	7,33E+30	4,297E+62	2,9538E+126	2,7076E+190	2,7921E+254

Чисельний експеримент показує, що із розширенням поля Галуа та зростанням степеня кількість незвідних поліномів збільшується. При вказаних параметрах p і n вона буде лежати в таких діапазонах: $30 \le S_p(8) \le 1{,}46 \cdot 10^{12}$, $4080 \le S_p(16) \le 8{,}54 \cdot 10^{24}$, $1{,}34 \cdot 10^8 \le S_p(32) \le 5{,}832 \cdot 10^{50}$, $2{,}8823 \cdot 10^{17} \le S_p(64) \le 5{,}4425 \cdot 10^{104}$, $8{,}25293 \cdot 10^{26} \le S_p(96) \le 6{,}7717 \cdot 10^{154}$ і $2{,}65846 \cdot 10^{36} \le S_p(n) \le 9{,}4788 \cdot 10^{206}$. Крім того, задану кількість незвідних поліномів можна отримати як зміною степеня n, так і розширенням поля $GF(p)$. Наприклад, $S_{43}(8) \approx S_7(16)$, $S_{43}(16) \approx S_7(32)$, $S_{43}(32) \approx S_7(64)$, $S_{43}(64) \approx S_{13}(96)$, $S_{43}(96) \approx S_{17}(128)$.

На рисунку 4.11 представлена графічна залежність кількості незвідних поліномів від цих параметрів в логарифмічній шкалі з основою 10.

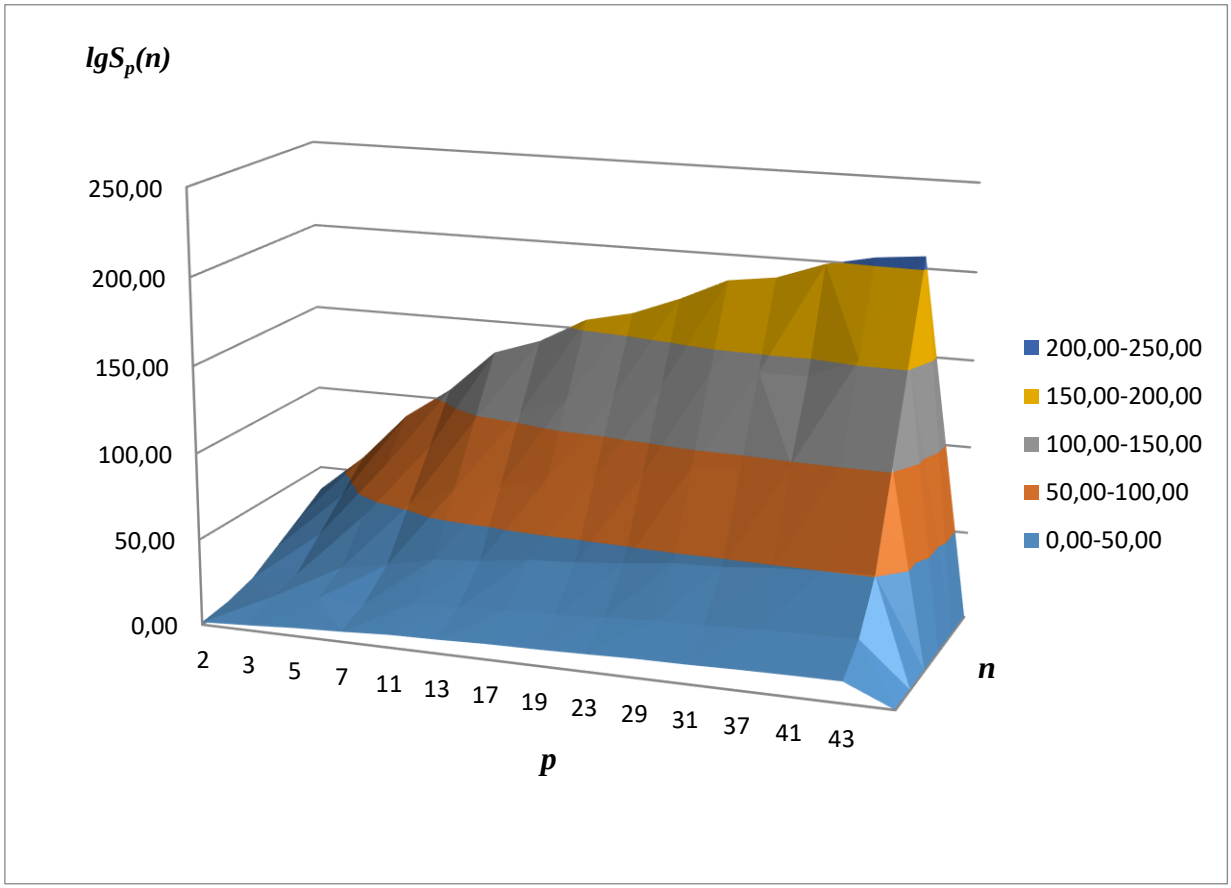


Рисунок 4.11 – Кількість взаємнопростих модулів $S_p(n)$ в залежності від поля Галуа p та степеня n.

В загальному випадку стійкість запропонованої криптосистеми з l модулів буде визначатися як сумарний час повного перебору всіх незвідних поліномів та складності виконання з кожним обчислень згідно такої формули [321]:

$$O(n,l) = C_{S_p(n)}^{l} \cdot n^2 \, log_2 \, l = \frac{\left(S_p(n)\right)! \cdot n^2 \cdot log_2 \, l}{\left(S_p(n)-l\right)! \cdot l!}. \quad (4.44)$$

Наприклад, стійкість криптосистеми для $l = 2\ldots7$ та n = 32 у полі Галуа GF(3) визначається так:

$$O(32,2) = C_{57906879556410}^{2} \cdot 32^2 \, log_2 \, 2 = \frac{(57906879556410)!}{(57906879556408)! \cdot 2!} 32^2 \, log_2 \, 2 \approx$$

$$\approx 5{,}879 \cdot 10^{29}$$

$$O(32,3) = C_{57906879556410}^{3} \cdot 32^2 \, log_2 \, 3 = \frac{(57906879556410)!}{(57906879556407)! \cdot 3!} 32^2 \, log_2 \, 3 \approx$$

$$\approx 1{,}674 \cdot 10^{43}$$

$$O(32,4) = C_{57906879556410}^{4} \cdot 32^2 \, log_2 \, 4 = \frac{(57906879556410)!}{(57906879556406)! \cdot 4!} 32^2 \, log_2 \, 4 \approx$$

$$\approx 9{,}595 \cdot 10^{56}$$

$$O(32,5) = C_{57906879556410}^{5} \cdot 32^2 \, log_2 \, 5 = \frac{(57906879556410)!}{(57906879556405)! \cdot 5!} 32^2 \, log_2 \, 5 \approx$$

$$\approx 1{,}29 \cdot 10^{70}$$

$$O(32,6) = C_{57906879556410}^{6} \cdot 32^2 \, log_2 \, 6 = \frac{(57906879556410)!}{(57906879556404)! \cdot 6!} 32^2 \, log_2 \, 6 \approx$$

$$\approx 1{,}386 \cdot 10^{83}$$

$$O(32,7) = C_{57906879556410}^{7} \cdot 32^2 \, log_2 \, 7 = \frac{(57906879556410)!}{(57906879556403)! \cdot 7!} 32^2 \, log_2 \, 7 \approx$$

$$\approx 1{,}245 \cdot 10^{96}$$

З представлених розрахунків видно, що при додаванні одного модуля стійкість зростає приблизно на 13 порядків.

На рисунку 4.12 в логарифмічній шкалі з основою 10 представлені графіки залежностей стійкості $O(n,l)$ запропонованого симетричного поліноміального алгоритму шифрування у ПСЗК від кількості модулів l для степенів полінома n=4, 8 і параметра p=2, p=3, p=5, p=7.

З рисунка видно, що усі графіки носять однаковий дзвоноподібний характер. Стійкість істотно зростає при збільшенні степеня та розмірності поля Галуа p і досягає свого максимуму при $l = \frac{S_p(n)}{2}$. Це означає, що криптоаналіз запропонованого алгоритму вимагає комбінаторної складності, яка приводить до NP-повної задачі.

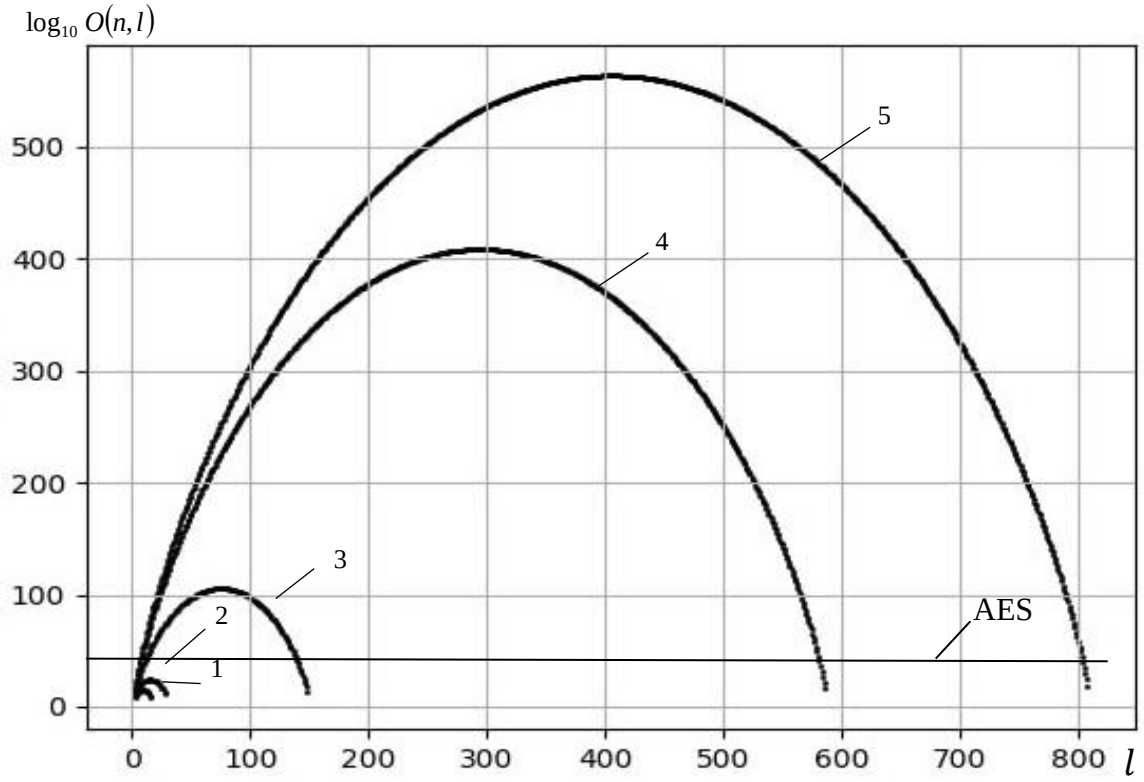


Рисунок 4.12 – Графіки залежностей стійкості $O(n,l)$ в логарифмічній шкалі від кількості модулів lі степенів полінома n (лінія 1 - p=2, n=4, лінія 2 – p=2, n=8, лінія 3 – p=5, n=4, лінія 4 – p=7, n=4, лінія 5 – p=3, n=8)

4.5 Теоретичні основи двоключового поліноміального симетричного шифрування в ПСЗК

У ПСЗК довільний поліном $L(x)$ представляється залишками $l_i(x)$ від ділення $L(x)$ на кожен із системи попарно взаємно простих модулів-поліномів $m_i(x)$:

$$l_i(x) = L(x) \bmod m_i(x). \tag{4.45}$$

Для відновлення поліному $L(x)$, як правило, використовується китайська теорема про залишки (КТЗ) [312] в кільці поліномів $Z_p[x]$ (p -просте число, яке визначає розмірність поля Галуа):

$$L(x) = \left(\sum_{i=1}^{w} l_i(x) B_i(x) b_i(x)\right) mod\ M\ (x), \tag{4.46}$$

де $M(x) = \prod_{i=1}^{w} m_i(x)$, $B_i(x) = \frac{M(x)}{m_i(x)}$, $b_i(x)$ шукається з виразу $b_i(x) = B_i^{-1}(x)\ mod\ m_i\ (x)$, w – кількість модулів. При цьому для степенів поліномів повинна виконуватися нерівність $\deg L(x) < \deg M(x)$.

Запропонований двоключовий поліноміальний симетричний метод шифрування в ПСЗК полягає в тому, що при відновленні полінома за його залишками у сумі (4.46) множення відбувається не на параметри $b_i(x) = B_i^{-1}(x)\ mod\ m_i\ (x)$, а на довільно вибрані поліноми-ключ $s_i(x)$, взаємно прості з $m_i(x)$.

На початковому етапі шифрування буквенну інформацію потрібно перекодувати у числову форму. Найчастіше використовуються ASCII-коди або спосіб, коли кожна літера замінюється на її відповідний номер в алфавіті. Нумерацію літер доцільно починати з 1. Тоді будь-яке повідомлення M запишеться як вектор рядок $M = a_1 a_2 \dots a_n$, де a_i - послідовність цифрового представлення букв, $i = \overline{1 \dots n}$, n – кількість літер або довжина повідомлення.

Далі вибирається довільний поліном $V(x) = \alpha_r x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_0$ порядку r в кільці $Z_p[x]$ такий, що в певній точці $x=c$, $c \in Z$, значення $V(c) = M$. Отже, параметр x, крім полінома $s_i(x)$, виступає в якості ще одного таємного ключа. Слід зауважити, що значення α_0 зручно шукати з виразу $\alpha_0 = M - \alpha_r x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_1 x$. Для генерації поліноміальних ключів абоненти вибирають відомі тільки їм обом системи модулів $m_i(x)$ та відповідні їм поліноми $s_i(x)$, для яких виконуються такі умови: $1 < deg\, s_i(x) < deg\, m_i(x)$ та НСД($s_i(x), m_i(x)$)=1. Друга умова виконується завжди, коли $m_i(x)$ є незвідним поліномом. Відповідно, і відправнику, і отримувачу відомі параметри $B_i(x)$, $b_i(x)$ і $x=c$.

Далі блок відкритого тексту $V(x)$ записується в ПСЗК згідно виразу (4.45). Шифрування відбувається при відновленні числа в позиційну систему числення згідно такого виразу:

$$V'(x) = \left(\sum_{i=1}^{s} l_i(x) B_i(x) s_i(x)\right) mod\, M(x). \qquad (4.47)$$

Знайдений поліном є шифртекстом, який передається по відкритому каналі зв'язку від одного абонента до іншого.

При розшифруванні спочатку обчислюються такі параметри:

$$q_i(x) = \left(b_i(x)\left(s_i^{-1}(x)\, mod\, m_i(x)\right)\right) mod\, m_i(x);$$
$$l_i'(x) = V'(x)\, mod\, m_i(x) \qquad (4.48)$$

Для отримання істинних залишків $l_i(x)$ необхідно виконати перетворення згідно співвідношення:

$$l_i(x) = \left(l_i'(x) q_i(x)\right) mod\, m_i(x) = \left(l_i'(x) b_i(x) s_i^{-1}(x)\right) mod\, m_i(x). \qquad (4.49)$$

Відповідно, відновлення полінома $V(x)$, який є відкритим текстом, здійснюється згідно формули (4.46). Крім цього, можна використати вираз, який з неї випливає:

На рисунку 4.13 представлена схема запропонованого поліноміального методу шифрування на основі КТЗ.

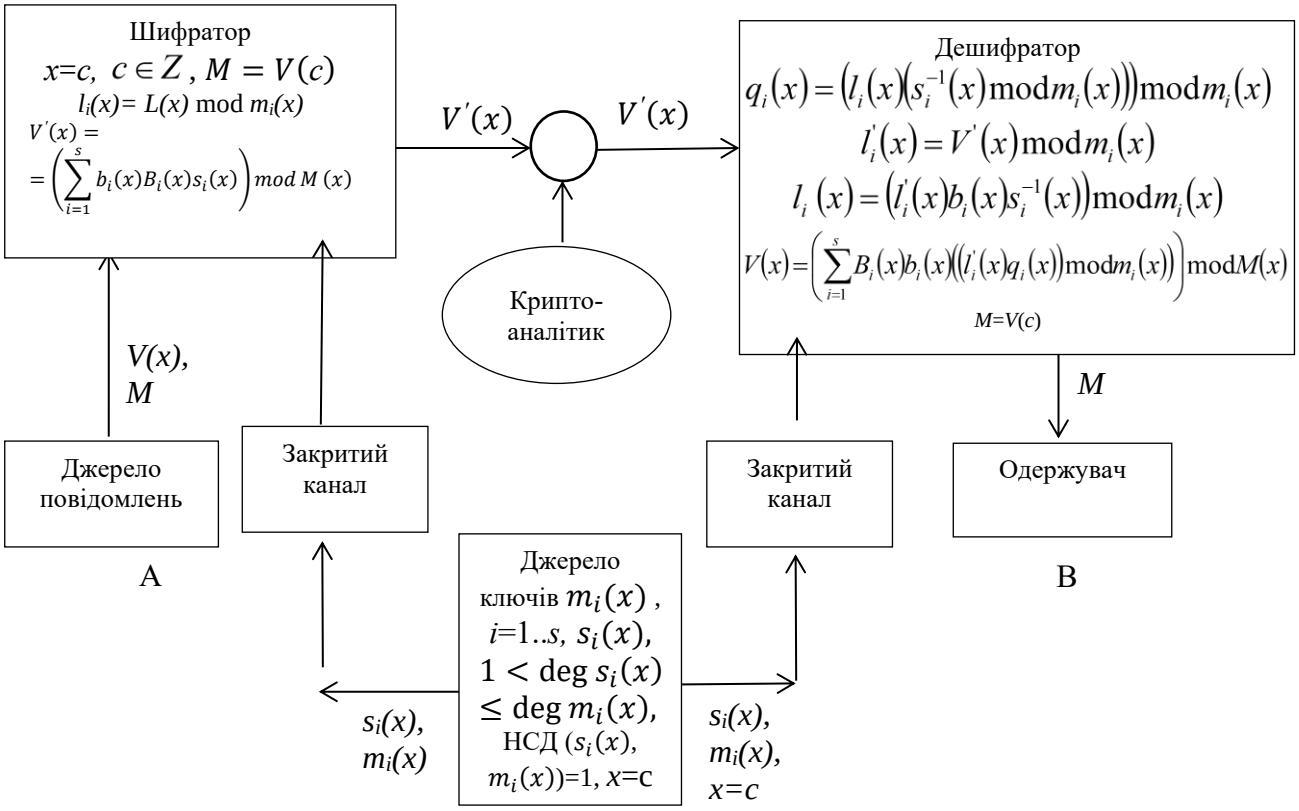


Рисунок 4.13 - Схема запропонованого двоключового поліноміального симетричного шифрування в ПСЗК

$$V(x) = \left(\sum_{i=1}^{s} B_i(x)b_i\,(x)\left(\left(l_i'(x)b_i(x)s_i^{-1}(x)\right) mod\ m_i\,(x)\right)\right) mod\ M\,(x) =$$

$$= \left(\sum_{i=1}^{s} B_i(x)b_i(x)\left(\left(l_i'(x)q_i(x)\right) mod\ m_i\,(x)\right)\right) mod\ M\,(x). \qquad (4.50)$$

Для отримання вхідного повідомлення потрібно знайти значення $V(\mathrm{c})=M$.

Коректність запропонованої криптосистеми встановлюється формальним доведенням, яке грунтується на властивостях конгруенцій, подільності $P(x)$ на $p_i(x)$ та рівності $b_i(x) = B_i^{-1}(x)\, mod\, m_i\,(x)$. Тоді отримуємо:

$$l_i(x) = \left(l_i'(x)q_i(x)\right) mod\, m_i\,(x)$$

$$= \left(\left(V'(x)\, mod\, m_i\,(x)\right) \cdot \left(b_i(x)s_i^{-1}(x)\right) mod\, m_i\,(x)\right) mod\, m_i\,(x) =$$

$$= \left(\left(\left(\sum_{j=1}^{s} l_j(x)s_j(x)B_j(x)\right) mod\, M\,(x)\right) mod\, m_i\,(x)\right.$$

$$\left. \cdot \left(b_i(x)s_i^{-1}(x)\right) mod\, m_i\,(x)\right) mod\, m_i\,(x) =$$

$$= \left(\left(l_i(x)s_i(x)B_i(x)\right) mod\, M\,(x) \cdot \left(b_i(x)s_i^{-1}(x)\right) mod\, m_i\,(x)\right) mod\, m_i\,(x) =$$

$$= \left(l_i(x)b_i(x)B_i(x)\right) mod\, m_i\,(x) = b_i(x). \qquad (4.51)$$

4.5.1 Приклад симетричного поліноміального шифрування у ПСЗК

Нехай потрібно зашифрувати відкритий текст «encryption polynomial (EP)», який у цифровій формі згідно ASCI-кодування матиме такий вигляд: EP=(101112). Для x=4 вибиремо, наприклад, вихідний поліном $L(x) = 19x^6 + 22x^5 - 9x^4 + 37x^3 + 33x^2 + 23x + \alpha_0$. Тоді $\alpha_0 = 101112 - L(4) = 76$. Отже, остаточний вигляд вхідного поліному буде такий: $L(x) = 19x^6 + 22x^5 - 9x^4 + 37x^3 + 33x^2 + 23x + 76$. Далі згідно запропонованої двоключової поліноміальної симетричної криптосистеми для трьох модулів (w=3) потрібно вибрати поліноми-модулі $m_i(x)$ та поліноми-ключі $s_i(x)$, взаємно прості з $m_i(x)$. Нехай, для прикладу, $m_1(x)=x^2+x+1$, $m_2(x)= x^3+x+1$,

$m_3(x)= x^2+3$ та $s_1(x)=x^2+x+3$, $s_2(x)= x^3+x^2+1$, $s_3(x)= x^2+2x+3$. Після цього обчислюються всі необхідні параметри для шифрування:

$$M(x) = m_1(x) \cdot m_2(x) \cdot m_3(x) = x^7 + x^6 + 5x^5 + 5x^4 + 8x^3 + 7x^2 + 6x + 3,$$

$$B_1(x) = \frac{M(x)}{m_1(x)} = \frac{x^7+x^6+5x^5+5x^4+8x^3+7x^2+6x+3}{x^2+x+1} = x^5+4x^3 + x^2+3x+3,$$

$$B_2(x) = \frac{M(x)}{m_2(x)} = \frac{x^7+x^6+5x^5+5x^4+8x^3+7x^2+6x+3}{x^3+x+1} = x^4 + x^3 + 4x^2 + 3x + 3,$$

$$B_3(x) = \frac{M(x)}{m_3(x)} = \frac{x^7+x^6+5x^5+5x^4+8x^3+7x^2+6x+3}{x^2+3} = x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1,$$

На наступному етапі здійснюється пошук значень $b_i(x) = B_i^{-1}(x)\, mod\, m_i\,(x)$ з використанням методу невизначених коефіцієнтів. Шукається $b_1(x) = B_1^{-1}(x)\, mod\, m_1\,(x) = (x^5+4x^3 + x^2+3x+3)^{-1}\, mod(x^2 + x + 1) =$ $= (x + 5)^{-1}\, mod(x^2 + x + 1)$. Далі записується співвідношення $(x + 5)(Ax + B)\, mod(x^2 + x + 1) = 1$ і після перетворення отримується $(Ax^2 + (5A + B)x + 5B)\, mod(x^2 + x + 1) = (4A + B)x + 5B - A = 1$. З останньої рівності випливає, що $5B - A = 1{,}4A + B = 0$, звідки $A = -\frac{1}{21}, B = \frac{4}{21}$. Отже, шуканий обернений поліном набере вигляду $b_1(x) = (x + 5)^{-1}\, mod(x^2 + x + 1) = -\frac{1}{21}x + \frac{4}{21}$.

Аналогічним чином здійснюється пошук $b_2(x) = B_2^{-1}(x)\, mod\, m_2\,(x) = (x^4 + x^3 + 4x^2 + 3x + 3)^{-1}\, mod(x^3 + x + 1)$ $= (3x^2 + x + 2)^{-1}\, mod(x^3 + x + 1)$. Для цього записується $(3x^2 + x + 2)(Ax^2 + Bx + C)\, mod(x^3 + x + 1) = 1$, де $Ax^2 + Bx + C$ - обернений поліном за модулем. Потрібно знайти коефіцієнти A, B, C, які задовольняють останню рівність. Після перетворення $(3Ax^4 + (A + 3B)x^3 + (2A + B + 3C)x^2 + (2B + C)x)\, mod(x^3 + x + 1) =$ $(-A + B + 3C)x^2 + +(-4A - B + C)x - A - 3 + 2C = 1$ отримується: $-A + B + 3C = 0$, $-4A - B + C = 0$, $-A - 3 + 2C = 1$. Звідси $A = \frac{4}{39}$, $B =$

$-\frac{11}{39}$, $C = \frac{5}{39}$. Отже, шуканий обернений поліном набуває такого вигляду: $b_2(x) = (3x^2 + x + 2)^{-1}\, mod(x^3 + x + 1) = \frac{4}{39}x^2 - \frac{11}{39}x + \frac{5}{39}$.

Таким же чином обчислюється значення $b_3(x) = B_3^{-1}(x)\, mod\, m_3\,(x) = (x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)^{-1}\, mod(x^2 + 3) =$ $= (5x + 4)^{-1}\, mod(x^2 + 3)$. Застосувавши метод невизначених коефіцієнтів, можна отримати: $(5x + 4)(Ax + B)\, mod(x^2 + 3) = (5Ax^2 + (4A + 5B)x + 4B)\, mod(x^2 + 3) \Rightarrow$ $\Rightarrow (4A + 5B)x + 4B - 15A = 1$. Остання рівність приводить до системи рівнянь $4A + 5B = 0$ і $4B - 15A = 1$, яка дає змогу обчислити значення коефіцієнтів $A = -\frac{5}{91}, B = \frac{4}{91}$ і тим самим знайти обернений поліном за модулем $b_3(x) = -\frac{5}{91}x + \frac{4}{91}$.

Тоді:

$l_1(x) = L(x)\, mod\, m_1\,(x) = (19x^6 + 22x^5 - 9x^4 + 37x^3 + 33x^2 + 23x + 76)\, mod(x^2 + x + 1) = -41x + 77,$

$l_2(x) = L(x)\, mod\, m_2\,(x) = (19x^6 + 22x^5 - 9x^4 + 37x^3 + 33x^2 + 23x + 76)\, mod(x^3 + x + 1) = 39x^2 + 55x + 80,$

$l_3(x) = L(x)\, mod\, m_3\,(x) = (19x^6 + 22x^5 - 9x^4 + 37x^3 + 33x^2 + 23x + 76)\, mod(x^2 + 3) = 110x - 617.$

Отже, згідно виразу (4.41), шифртекстом є такий поліном:

$$V'(x) = \left(\sum_{i=1}^{s} l_i(x)B_i(x)s_i\,(x)\right) mod\, M\,(x) = ((x^5+4x^3 + x^2+3x+3) \cdot (-41x + 77)(x^2 + x + 3) +$$
$$+(x^4 + x^3 + 4x^2 + 3x + 3)(39x^2 + 55x + 80)(x^3 + x^2 + 1) +$$
$$+(x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)(110x - 617)(x^2 + 3x + 2))\, mod \begin{pmatrix} x^7 + x^6 + 8x^5 + 6x^4 + \\ +22x^3 + 12x^2 + 21x + 9 \end{pmatrix} =$$
$$= -1330x^6 - 1889x^5 - 3663x^4 - 3852x^3 - 4391x^2 - 2791x - 246$$

При розшифруванні для обчислення параметрів $s_i^{-1}(x)\, mod\, m_i\,(x)$ знову доцільно використати метод невизначених коефіцієнтів. Для першого полінома-ключа $s_1^{-1}(x)\, mod\, m_1\,(x) = (x^2 + 2x + 3)^{-1}\, mod(x^2 + x + 1) =$

$(x+2)^{-1}\,mod(x^2+x+1)$. Далі записується $(x+2)(Ax+B)\,mod(x^2+x+1)=1\Rightarrow(Ax^2+(2A+B)x+2B)\,mod(x^2+x+1)==(A+B)x+2B-A=1$, звідки $A=-\frac{1}{3},\ B=\frac{1}{3}$. Отже, $s_1^{-1}(x)\,mod\,m_1\,(x)=\frac{1}{3}(1-x)$.

Аналогічно $s_2^{-1}(x)\,mod\,m_2\,(x)=(x^3+x^2+1)^{-1}\,mod(x^3+x+1)==(x^2-x)^{-1}\,mod(x^3+x+1)$, $(x^2-x)(Ax^2+Bx+C)\,mod(x^3+x+1)=1$, де Ax^2+Bx+C - обернений поліном за модулем. Після перетворення $(Ax^4+(B-A)x^3+(C-B)x^2-Cx)\,mod(x^3+x+1)=(\mathrm{C}-A-B)x^2+(-C-B)x++A-B=1$ отримується система з трьох рівнянь з трьома невідомими $C-B-A=0, -C-B=0, A-B=1$. Звідси $A=\frac{2}{3},\ B=-\frac{1}{3},\ C=\frac{1}{3}$. Отже, шуканий обернений поліном в Z[x] прийме вигляд: $s_2^{-1}(x)\,mod\,m_2\,(x)=(x^2-x)^{-1}\,mod(x^3+x+1)=\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}$.

Аналогічно знаходиться значення $s_3^{-1}(x)\,mod\,m_3\,(x)=(x^2+3x+2)^{-1}\,mod(x^2+3)=(3x-1)^{-1}\,mod(x^2+3)=-\frac{3}{28}x-\frac{1}{28}$.

Після того, як знайдені ключі для розшифрування, необхідно знайти ще параметри $q_i(x)$:

$$q_1(x)=\left(b_1(x)\left(s_1^{-1}(x)\,mod\,p_1\,(x)\right)\right)mod\,p_1\,(x)=\left(\left(-\frac{1}{21}x+\frac{4}{21}\right)\left(\frac{1}{3}(1-x)\right)\right)mod(x^2+x+1)=\left(\frac{1}{63}x^2-\frac{5}{63}x+\frac{4}{63}\right)mod(x^2+x+1)=-\frac{2}{21}x+\frac{1}{21},$$

$$q_2(x)=\left(b_2(x)\left(s_2^{-1}(x)\bmod p_2(x)\right)\right)\bmod p_2(x)=\left(\begin{array}{l}\left(\frac{4}{39}x^2-\frac{11}{39}x+\frac{5}{39}\right)\times\\ \times\left(\frac{2}{3}x^2-\frac{1}{3}x+\frac{1}{3}\right)\end{array}\right)\bmod\left(x^3+x+1\right)=$$

$$=\left(\frac{8}{117}x^4-\frac{2}{9}x^3-\frac{25}{117}x^2-\frac{16}{117}x+\frac{5}{117}\right)\bmod\left(x^3+x+1\right)=\frac{17}{117}x^2+\frac{2}{117}x+\frac{31}{117},$$

$$q_3(x)=\left(b_3(x)\left(s_3^{-1}(x)\,mod\,p_3\,(x)\right)\right)mod\,p_3\,(x)=\left(\left(-\frac{5}{91}x+\frac{4}{91}\right)\left(-\frac{3}{28}x-\frac{1}{28}\right)\right)mod(x^3+3)=\left(\frac{15}{2548}x^2-\frac{1}{364}x-\frac{1}{637}\right)mod(x^2+3)=-\frac{1}{364}x-\frac{1}{52}.$$

На наступному етапі обчислюються «неправильні» залишки:

$$l_1'(x) = V'(x)\,mod\,m_1\,(x) = \begin{pmatrix}\text{-}1330x^6\text{-}1889x^5\text{-}3663x^4 - \\ \text{-}3852x^3\text{-}4391x^2\text{-}2791x\text{-}246\end{pmatrix} mod(x^2 + x + 1)$$

$$= -174x + 852;$$

$$l_2'(x) = V'(x)\,mod\,m_2\,(x) = \begin{pmatrix}\text{-}1330x^6\text{-}1889x^5\text{-}3663x^4 - \\ \text{-}3852x^3\text{-}4391x^2\text{-}2791x\text{-}246\end{pmatrix} mod(x^3 + x + 1)$$

$$= -169x^2 + 175x + 387;$$

$$l_3'(x) = V'(x)\,mod\,m_3\,(x) = \begin{pmatrix}\text{-}1330x^6\text{-}1889x^5\text{-}3663x^4 - \\ \text{-}3852x^3\text{-}4391x^2\text{-}2791x\text{-}246\end{pmatrix} mod(x^2 + 3) =$$

$$= -8236x + 15870.$$

Далі за формулою (4.50) відновлюється вхідний поліном:

$$L(x) = \left(\sum_{i=1}^{s} B_i(x)b_i\,(x)\left(\left(l_i'(x)b_i(x)s_i^{-1}(x)\right) mod\,m_i\,(x)\right)\right) mod\,M\,(x) =$$

$$= \left(\sum_{i=1}^{s} B_i(x)b_i(x)\left(\left(l_i'(x)q_i(x)\right) mod\,m_i\,(x)\right)\right) mod\,M\,(x) =$$

$$= \left(\begin{array}{l}\left(\begin{array}{l}(x^5{+}4x^3 + x^2{+}3x{+}3)\left(-\frac{1}{21}x + \frac{4}{21}\right)\times \\ \times\left(\left(-\frac{2}{21}x + \frac{1}{21}\right)(-174x + 852)\right) mod(x^2 + x + 1)\end{array}\right) + \\ + \left(\begin{array}{l}(x^4 + x^3 + 4x^2 + 3x + 3)\left(\frac{4}{39}x^2 - \frac{11}{39}x + \frac{5}{39}\right)\times \\ \times\left(\left(\frac{17}{117}x^2 + \frac{2}{117}x + \frac{31}{117}\right)(-169x^2 + 175x + 387)\right) mod(x^3 + x + 1)\end{array}\right) + \\ + \left(\begin{array}{l}(x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)\left(-\frac{5}{91}x + \frac{4}{91}\right)\times \\ \times\left((-8236x + 15870)\left(-\frac{1}{364}x - \frac{1}{52}\right)\right) mod(x^2 + 1)\end{array}\right)\end{array}\right)$$

$$mod(x^7 + x^6 + 5x^5 + 5x^4 + 8x^3 + 7x^2 + 6x + 3) = 19x^6 + 22x^5 - 9x^4 +$$

$$+37x^3 + 33x^2 + 23x + 76$$

Для знаходження відкритого повідомлення потрібно знайти значення цього полінома при *x*=4:

$$L(4) = 19(4)^6 + 22(4)^5 - 9(4)^4 + 37(4)^3 + 33(4)^2 + 23 \cdot 4 + 76 = 101112,$$

що відповідає повідомленню encryption polynomial (EP) у цифровій формі згідно ASCI-кодів.

Пришвидшення процесу шифрування та розшифрування можна досягнути у випадку, коли абоненти виберуть параметри $s_i(x) = 1$. Однак це призведе до зменшення стійкості криптосистеми. Процес шифрування відбувається за спрощеною формулою:

$$V'(x) = (\sum_{i=1}^{s} b_i(x)M_i(x)) \, mod \, P\,(x). \qquad (4.52)$$

Слід відмітити, що в процесі розшифрування усувається операція пошуку оберненого полінома за модулем $s_i^{-1}(x) \, mod \, m_i\,(x)$ та множення на нього, оскільки $q_i(x) = b_i(x)$. Відновлення відкритого тексту відбувається на основі таких співвідношень:

$$l_i(x) = \left(l_i'(x)b_i(x)\right) mod \, m_i\,(x);$$

$$L(x) = \left(\sum_{i=1}^{s} B_i(x)b_i\,(x)\left(\left(l_i'(x)b_i(x)\right) mod \, m_i\,(x)\right)\right) mod \, M\,(x). \qquad (4.53)$$

Для таких самих вхідних параметрів, що і у попередньому прикладі, згідно формул (4.52) і (4.53), отримується такий шифртекст:

$$V'(x) = \left(\sum_{i=1}^{s} l_i(x)B_i(x)\right) mod\, M\,(x)$$

$$= ((x^5{+}4x^3 + x^2{+}3x +) \cdot (-41x + 77)$$

$$+ (x^4 + x^3 + 4x^2 + 3x + 3)(39x^2 + 55x + 80)$$

$$+ (x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1)(110x$$

$$- 617))\, mod \begin{pmatrix} x^7 + x^6 + 5x^5 + 5x^4 + 8x^3 + \\ +7x^2 + 6x + 3 \end{pmatrix}$$

$$= \ 108\mathrm{x}^6 - 336x^5 - 270x^4 - 330x^3 - 458x^2 - 611x - 146.$$

Для розшифрування необхідно обчислити додаткові параметри згідно формули (4.48):

$$l_1'(x) = V'(x)\, mod\, m_1\,(x) =$$

$$= \begin{pmatrix} 108\mathrm{x}^6 - 336x^5 - 270x^4 - 330x^3 - \\ -458x^2 - 611x - 146 \end{pmatrix} mod(x^2 + x + 1) = -87x + 426$$

$$l_2'(x) = V'(x)\, mod\, m_2\,(x) =$$

$$= \begin{pmatrix} 108\mathrm{x}^6 - 336x^5 - 270x^4 - 330x^3 - \\ -458x^2 - 611x - 146. \end{pmatrix} mod(x^3 + x + 1) =$$

$$= 256x^2 - 131x - 44$$

$$l_3'(x) = V'(x)\, mod\, m_3\,(x) = \begin{pmatrix} 108\mathrm{x}^6 - 336x^5 - 270x^4 - 330x^3 - \\ -458x^2 - 611x - 146. \end{pmatrix} mod(x^2 +$$

$$1) = -2545x - 4118.$$

Процес розшифрування здійснюється згідно формули (4.53):

$$L(x) = \left(\sum_{i=1}^{s} B_i(x)b_i\,(x)\left(\left(l_i'(x)b_i(x)\right) mod\, m_i\,(x)\right)\right) mod\, M\,(x) =$$

$$= \left(\begin{array}{l} \left(\begin{array}{l} (x^5+4x^3+x^2+3x+3)\left(-\frac{1}{21}x+\frac{4}{21}\right)\left(\begin{array}{l} (-87x+426)\times \\ \times\left(-\frac{1}{21}x+\frac{4}{21}\right) \end{array} \right) mod(x^2+x+1) \end{array} \right)+ \\ +\left(\begin{array}{l} (x^4+x^3+4x^2+3x+3)\left(\frac{4}{39}x^2-\frac{11}{39}x+\frac{5}{39}\right)\times \\ \times\left(\left(\frac{4}{39}x^2-\frac{11}{39}x+\frac{5}{39}\right)\times(256x^2-131x-44)\right) mod(x^3+x+1) \end{array} \right)+ \\ +\left(\begin{array}{l} (x^5+x^4+2x^3+2x^2+2x+1)\left(-\frac{5}{91}x+\frac{4}{91}\right)\times \\ \times\left((-2545x-4118)\left(-\frac{5}{91}x+\frac{4}{91}\right)\right) mod(x^2+3) \end{array} \right) \end{array} \right)$$

$mod(x^7+x^6+5x^5+5x^4+8x^3+7x^2+6x+3) = 19x^6+22x^5-9x^4+37x^3+33x^2+23x+76$

Таке спрощення призводить до зменшення обчислювальної складності за рахунок уникнення операції пошуку параметрів $q_i(x)$ та $s_i(x)^{-1} \, mod \, m_i$.

4.5.2 Оцінка криптостійкості поліноміального двоключового симетричного алгоритму шифрування в системі залишкових класів

Пропонований симетричний метод шифрування, який базується на поліноміальній арифметиці ПСЗК, відрізняється високим рівнем криптостійкості. Вона забезпечується завдяки складності задачі ідентифікації всіх можливих варіантів параметрів та модулів криптографічних перетворень. Ключовим аспектом розробленого підходу є необхідність здійснення повного перебору всіх взаємнопростих поліномів у кільці многочленів Z[x] в полі Галуа *GF*(*p*). Криптоаналіз стає особливо складним у випадку застосування незвідних поліномів.

Загальна кількість можливих модулів *l* обмежена величиною $S_p(n)$, яка визначається через функцію Мебіуса згідно формули (4.43).

Отже, щоб знайти кількість незвідних поліномів над *GF*(*p*), потрібно визначити всі дільники степеня *n*, для кожного з них обчислити функцію Мебіуса, підставити їх у формулу (4.43) та просумувати.

В результаті проведених досліджень (таблиця 4.2) встановлено, що розширення поля Галуа і збільшення степенів поліномів призводить до зростання кількості незвідних поліномів.

Наприклад, $3 \le S_p(4) \le 2{,}6 \cdot 10^7$ $30 \le S_p(8) \le 1{,}35 \cdot 10^{15}$, $4080 \le S_p(16) \le 7{,}33 \cdot 10^{30}$, $1{,}34 \cdot 10^8 \le S_p(32) \le 4{,}297 \cdot 10^{62}$, $2{,}8823 \cdot 10^{17} \le S_p(64) \le 2{,}9538 \cdot 10^{126}$, $8{,}25293 \cdot 10^{26} \le S_p(96) \le 2{,}707 \cdot 10^{190}$ і $2{,}65846 \cdot 10^{36} \le S_p(n) \le 2{,}7921 \cdot 10^{254}$.

Крім того, можна отримати задану кількість незвідних поліномів, змінюючи степінь n або розширюючи поле $GF(p)$, наприклад $S_{101}(8) \approx S_{11}(16)$, $S_{47}(16) \approx S_7(32)$, $S_{43}(32) \approx S_7(64)$, $S_{47}(64) \approx S_{13}(96)$, $S_{67}(96) \approx S_{17}(128)$.

На рисунку 4.12 представлена графічна залежність кількості незвідних поліномів від цих параметрів в логарифмічній шкалі з основою 10.

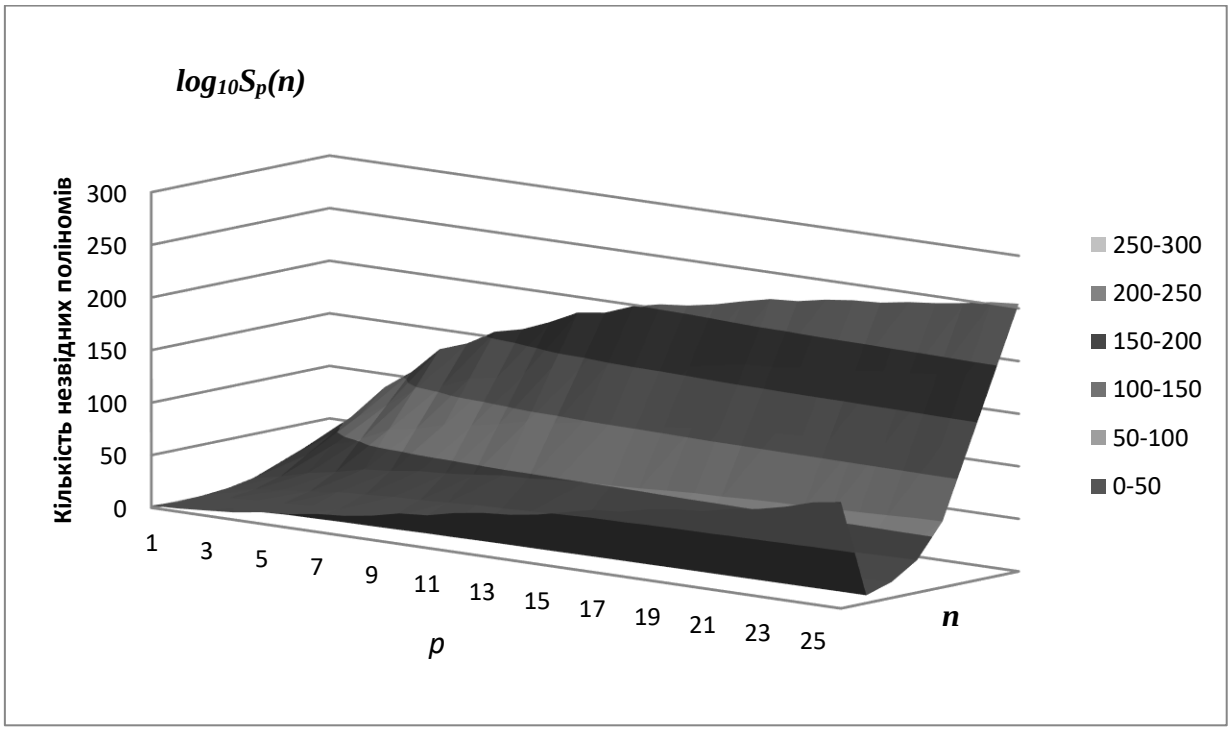


Рисунок 4.12 – Кількість взаємнопростих модулів $S_p(n)$ в залежності від поля Галуа p та степеня n.

Стійкість запропонованої системи ґрунтується на вирішенні двох важливих задач: повний перебір всіх незвідних поліномів і пошук коренів поліномів.

Важливо відзначити, що обчислення точних коренів для поліномів, вище четвертого ступеня, в загальному випадку є досить складним завданням, і часто використовуються наближені методи, часова складність яких залежить від порядку поліномів n. Зокрема, відомі такі основні методи [320]:

1. Метод Бернуллі або метод Ньютона (Ньютона-Рафсона) використовується для ітеративного знаходження коренів. Часова складність кожної ітерації зазвичай є $O(n)$, оскільки потрібно обчислити значення полінома та його похідної. Загальна складність залежить від кількості ітерацій, яка необхідна для досягнення заданої точності.

2. Метод Безу або синтетичного ділення використовується для розкладання полінома на множники. Часова складність залежить від степеня полінома та складності коефіцієнтів. Загалом, ці методи мають складність$O(n^2)$.

3. Метод Дюранда-Кернера або Лагерра – ітеративні методи, схожі на метод Ньютона. Їх складність оцінюється як $O(n^2)$ і також визначається в основному кількістю ітерацій, необхідних для досягнення бажаної точності, та складністю обчислень на кожній ітерації.

В загальному випадку стійкість запропонованої криптосистеми з l модулів буде визначатися як сумарний час повного перебору всіх незвідних поліномів та визначення коренів рівняння з врахуванням складності виконання кожної операції згідно формули:

$$O(n,l) = C^{l}_{S_p(n)} \cdot n^4 \, log \, l = \frac{\left(S_p(n)\right)! \cdot n^4 \cdot log \, l}{(S_p(n)-l)! \cdot l!}. \quad (4.55)$$

Наприклад, стійкість криптосистеми для $l = 2\ldots8$ та $n = 64$ у полі Галуа $GF(5)$ визначається так: $O(64,2) = C^{2}_{5365130969207050000000000000000} \cdot 32^4 \, log \, 2 =$

$\frac{(536513096920705000000000000)!}{(536513096920704999999999998)!\cdot 2!} \times 32^4\, log\, 2 \approx 7{,}269 \cdot 10^{63}$, $O(64{,}3) \approx 2{,}06{\cdot}10^{92}$, $O(64{,}4) \approx 3{,}49{\cdot}10^{120}$, $O(64{,}5) \approx 4{,}34{\cdot}10^{148}$, $O(64{,}6) \approx 4{,}71{\cdot}10^{176}$, $O(64{,}7) \approx 2{,}73{\cdot}10^{204}$, $O(64{,}8) \approx 1{,}703 \cdot 10^{232}$.

З проведених чисельних результатів видно, що при додаванні одного модуля стійкість зростає приблизно на 30 порядків.

На рисунку 4.13, в логарифмічній шкалі з основою 10, представлені криві, які відображають залежності стійкості запропонованого симетричного поліноміального алгоритму шифрування у скінченних полях від кількості модулів для різних степенів полінома (n=4, 8) та параметрів (p=2, p=3, p=5, p=7).

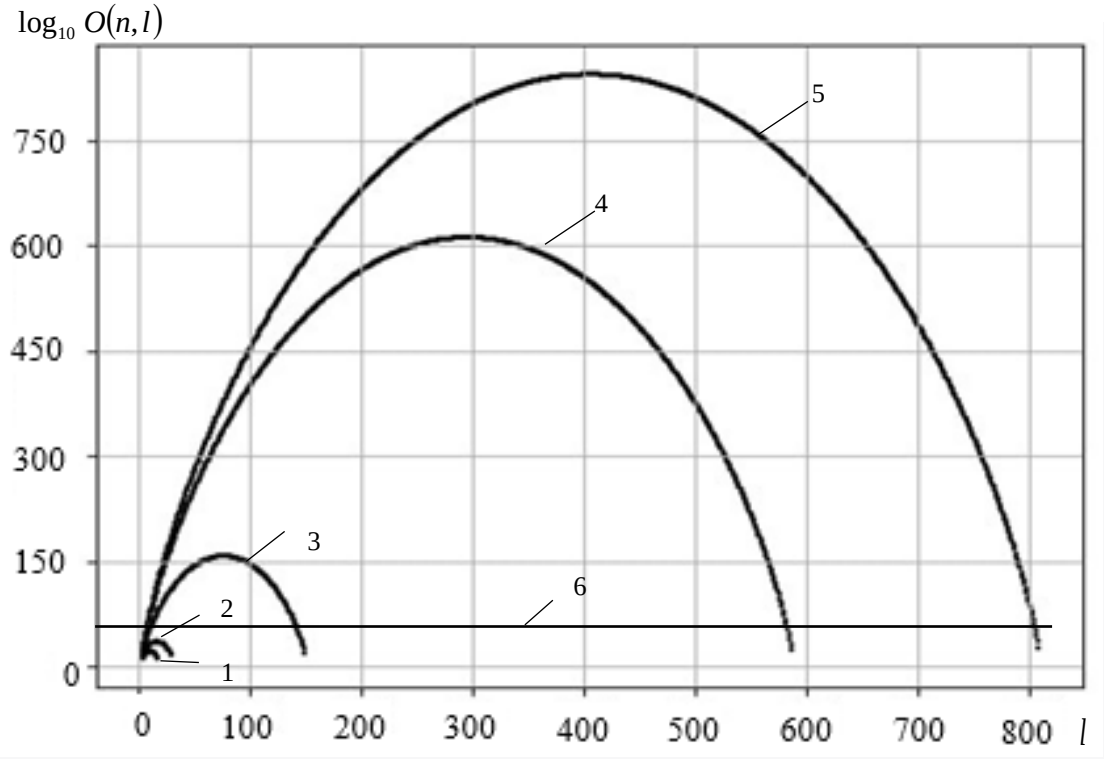


Рисунок 4.13 – Графіки залежностей стійкості $O(n,l)$ від кількості модулів lі степенів полінома n (лінія 1 - p=2, n=4, лінія 2 – p=2, n=8, лінія 3 – p=5, n=4, лінія 4 – p=7, n=4, лінія 5 – p=3, n=8, лінія 6 – стійкість алгоритму шифрування AES [218])

З аналізу рисунка випливає, що стійкість запропонованого симетричного поліноміального алгоритму шифрування значно збільшується при зростанні як

степеня полінома n, так і розмірності поля Галуа p. При певних значеннях n і p, тобто при $l = \frac{S_p(n)}{2}$ спостерігається максимальна стійкість.

Це свідчить про те, що атаки на цей алгоритм вимагають значної обчислювальної складності і часто можуть бути формалізовані як NP-повні задачі.

Такий дзвоноподібний характер кривих стійкості може бути пов'язаний з властивостями математичних конструкцій, що лежать в основі алгоритму, і відображає взаємозв'язок між параметрами системи та її стійкістю до атак.

Висновки до четвертого розділу

1. Вперше запропоновано алгоритм пошуку оберненого полінома в кільці Z[x] на основі методу невизначених коефіцієнтів. Представлено математичний опис розробленого підходу та наведено приклад застосування. Побудовано аналітичні вирази часових складностей в залежності від порядку поліномів для запропонованого методу та відомого, який грунтується на алгоритмі Евкліда та його наслідку. В результаті проведених досліджень встановлено ефективність алгоритму на основі методу невизначених коефіцієнтів, в якому відсутній пошук НСД поліномів. Однією з основних переваг є зменшення часової складності з $O(n\log n\cdot(1+\log n))$ до $O1(2n\log n)$ в порівнянні з відомим. Представлено графічні залежності часових складностей. Встановлено, що ефективність запропонованих методів логарифмічно зростає із збільшенням степенів поліномів.

2. Вперше запропоновані методи відновлення полінома в кільці Z[x], які за рахунок операцій додавання добутку модулів та добутку залишків модулів дають змогу розпаралелити обчислення та уникнути процедури пошуку мультиплікативного оберненого полінома, що в свою чергу, призводить до збільшення ефективності. В результаті чого досягається ефект коли результати

проміжних обчислень виходять за межі встановленого діапазону, що усуває необхідність виконання операції знаходження залишку за поліномом порівняно великого порядку. Отримано аналітичні вирази часових складностей в залежності від порядку поліномів та кількості модулів-поліномів запропонованого підходу та алгоритму Гарнера, які вказують, що використання розробленого методу дає змогу зменшити часову складність. Представлено графічну залежність часової складності і ефективності. Встановлено, що ефективність запропонованих методів логарифмічно зростає із збільшенням степенів поліномів, і пропорційно зменшується при збільшені кількості модулів

3. Розроблено симетричні криптографічні алгоритми, які грунтуються на поліноміальній системі залишкових класів. Розроблено математичне забезпечення та схеми запропонованого поліноміального симетричного шифрування в ПСЗК. З метою оцінки його стійкості було проведено дослідження та побудовано аналітичні вирази, які вказують, що криптоаналіз запропонованого алгоритму вимагає комбінаторної складності, що приводить до NP-повної задачі. Встановлено, що стійкість істотно зростає при збільшенні степеня та розмірності поля Галуа p і досягає свого максимуму у випадку, коли кількість модулів рівна половині можливої кількості незвідних поліномів із заданими степенями поліномів та порядками поля Галуа. Це означає, що пошук ефективного алгоритму для розв'язання даної задачі вимагає значних обчислювальних ресурсів і часу.

4. Вперше запропоновано та розроблено симетричний криптографічний алгоритм, що ґрунтується на системі поліноміальних залишкових класів. Результати підтверджують, що стійкість цього алгоритму суттєво залежить від степеня полінома та розмірності поля Галуа, а також від оптимальної кількості модулів. Проведене дослідження показує, що криптоаналіз даного алгоритму вимагає значних обчислювальних ресурсів і може бути віднесений до класу NP-повних задач. Такий підхід використання системи поліноміальних залишкових класів для розробки криптографічних алгоритмів пропонує новий

шлях для забезпечення захисту конфіденційної інформації у системах з обмеженими обчислювальними ресурсами.

Основні результати четвертого розділу відображені у роботах [295, 304, 306, 309, 313].

РОЗДІЛ 5. ТЕОРЕТИЧНІ ОСНОВИ ІЄРАРХІЧНИХ СИМЕТРИЧНИХ ПОЛІНОМІАЛЬНИХ КРИПТОАЛГОРИТМІВ НА ОСНОВІ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

5.1 Симетричний криптоалгоритм на основі ієрархічної системи залишкових класів

Довільне десяткове число N в СЗК можна задати сукупністю залишків b_i від ділення цього числа на вибрані модулі p_i, які мають бути попарно взаємно прості:

$$b_i = N \bmod p_i. \tag{5.1}$$

Установити взаємно–однозначну відповідність між цілим числом N в десятковій системі з діапазону $[0, P)$, де $P = \prod_{i=1}^{l} p_i$, $N < P$, і його залишками дозволяє СЗК:

$$N = \left(\sum_{i=1}^{l} b_i M_i m_i\right) mod\ P, \tag{5.2}$$

де $M_i = \frac{P}{p_i}$, m_i знаходиться згідно з виразом $m_i = M_i^{-1} mod\, p_i$, l — кількість модулів.

Інший спосіб полягає у послідовному додаванні добутку попередньо розглянутих модулів. Наприклад, до залишку b_1 модуль p_1 додають до тих пір, поки не буде виконуватись конгруенція $N_1 \bmod p_2 = b_2$ ($N_1 = b_1 + \gamma_1 p_1$, γ_1— кількість додавань модуля p_1). Далі до N_1 додають вже добуток модулів $p_1 p_2$, аж поки не буде виконуватись рівність $N_2 \bmod p_3 = b_3$ ($N_2 = N_1 + \gamma_2 p_1 p_2$, γ_2 — кількість додавань $p_1 p_2$). На i-му кроці (i=1 ,…, l–1) має виконуватися конгруенція $N_i \bmod p_{i+1} = b_{i+1}$ ($N_i = N_{i-1} + \gamma_i p_1 p_2 p_3 \ldots p_i$). На рисунку 1 представлена схема відновлення десяткового представлення числа за його залишками на основі додавання добутку модулів.

Важливо відмітити, що при використанні даного підходу результати проміжних обчислень не будуть виходити за межі встановленого діапазону P, що унеможливлює переповнення розрядної сітки і усуває необхідність виконання операції знаходження залишку за модулем P. Такий спосіб подібний до алгоритму Гарнера, згідно якого для обчислення коефіцієнтів γ_i потрібно використовувати методи пошуку оберненого елемента за відповідним модулем: $\gamma_i = (b_{i+1} - N_i) \cdot ((p_1 p_2 \dots p_i) \, mod \; p_{i+1})^{-1} \, mod \; p_{i+1}$.

$$N = \underbrace{\underbrace{\underbrace{\underbrace{b_1 + \gamma_1 p_1}_{\substack{mod\, p_2 \\ \| \\ b_2}} + \gamma_2 p_1 p_2}_{\substack{mod\, p_3 \\ \| \\ b_3}} + \gamma_3 p_1 p_2 p_3}_{\substack{mod\, p_4 \\ \| \\ b_4}} + \dots + \gamma_{k-1} p_1 p_2 p_3 \dots p_{l-1}}_{\substack{mod\, p_l \\ \| \\ b_l}},$$

У той же час γ_i можна обчислити за формулою $\gamma_i = (b_{i+1} - N_i) \cdot ((p_1 p_2 \dots p_i) \, mod \; p_{i+1})^{-1} \, mod \; p_{i+1}$.

5.1.1 Правила кодування тексту

Для шифрування текстової інформації її спочатку потрібно перевести в числову форму. Зазвичай при цьому використовують відомі таблиці відповідностей між літерами та числами (наприклад, ASCII-коди, порядкові номери літери у відповідному алфавіті). Однак при використанні десяткової системи числення виникає незручність, коли номер букви починається з нуля (наприклад, а = 00, b = 01 тощо). Тому для перекодування букв українського та англійського алфавітів, цифр, а також деяких спеціальних символів, використовується принцип, коли нумерація символів представлена трьохзначним числом, починаючи зі 100. Це дозволить ефективно застосовувати запропонований симетричний метод

шифрування/розшифрування у ступінчатій СЗК. У таблиці 5.1 наведено символьно-числову відповідність для кодування 167 символів. Очевидно, що таблицю 1 можна розширити до 900 символів.

Таблиця 5.1

Кодування літер українського та англійського алфавітів, символів та цифр

а	б	в	г	д	е	є	ж	з	и	і	ї	й	к	л	м
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115
н	о	п	р	с	т	у	Ф	х	ц	ч	ш	щ	ь	ю	я
116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131
	\n	.	,	:	!	?	%	#	@	№	;	^	*	(	)
132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147
-	+	=	_	<	>	{	}	[	]	|	/	`	~	\	"
148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163
\\	$	&	a	b	c	d	E	f'	g	h	i	j	k	l	m
164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179
n	o	p	q	r	s	t	u	v	w	x	y	z	A	Б	B
180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	A	B	C
212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243
T	U	V	W	X	Y	Z	1	2	3	4	5	6	7	8	9
244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259
0	—	−	«	»	ґ	Я									
260	261	262	263	264	265	266									

5.1.2 Теоретичні основи симетричного шифрування у ІСЗК

Збільшення об’ємів опрацювання, передачі та зберігання інформації неминуче призводить до збільшення кількості модулів та їх величини. Це, в свою чергу, зумовлює ускладнення апаратного забезпечення та збільшення часу виконання операцій. Зменшити величину модулів і, відповідно, числових операндів дозволяє ІСЗК [321, 322].

Нехай система модулів першого рівня p_1, p_2, …, p_l, які розміщені в порядку зростання, забезпечує можливість блоків відкритого тексту N в діапазоні $[0, P)$, де $P = \prod_{i=1}^{l} p_i$. За формулою (5.1) обчислюються залишки b_i. Далі для кожного модуля p_i першого рівня вибирається нова система модулів

другого рівня: q_{i1}, q_{i2}, …, q_{il}, вважаючи для спрощення, що кількість модулів у кожній системі на будь-якому рівні однакова і дорівнює l. Відповідно, обчислюються залишки $b_{ij}=b_i \bmod q_{ij}$.

Далі, в свою чергу, залишки другого рівня з дотриманням відповідних вимог записуються аналогічно в системі модулів третього рівня. Така процедура триває до останнього, k-го рівня.

Отже, кожному залишку r-ого рівня відповідає l систем з l^{r-1} залишками. Таким чином, на $r + 1$ рівень передається l^r залишків ($l >1$, $r = 1, 2,\ldots, k – 1$) і шифртекст складається із l^k чисел, які є залишками останнього рівня ІСЗК. Якщо кількість рівнів дорівнює 1, то має місце звичайне симетричне шифрування у СЗК [323]. Набори модулів відомі для відправника і отримувача.

Як правило, кількість рівнів, визначається умовами конкретної задачі. Такий процес переходу до менших модулів помітно спрощує реалізацію елементарного арифметичного пристрою і скорочує час виконання арифметичних операцій. На рисунку 5.1 наведено схему опрацювання залишків на відповідних рівнях.

Слід зазначити, що для задач криптографії кількості модулів на різних рівнях доцільно вибирати різними. Це значно підвищить складність криптоаналізу такої системи шифрування, хоча і ускладнить апаратну реалізацію.

Якщо відкрите повідомлення, переведене у числову форму, не задовольняє умову $N < P$, то його необхідно поділити на блоки, які відповідають вказаній нерівності.

Розшифрування в ІСЗК відбувається в зворотному порядку. Весь шифртекст розбивається на блоки, кількість яких становить l^k. Далі, застосовуючи один із методів відновлення числа із його залишків, наприклад, КТЗ (5.2), отримуються залишки вищого рівня. На кожному з рівнів в процесі розшифрування кількість залишків зменшується в l разів.

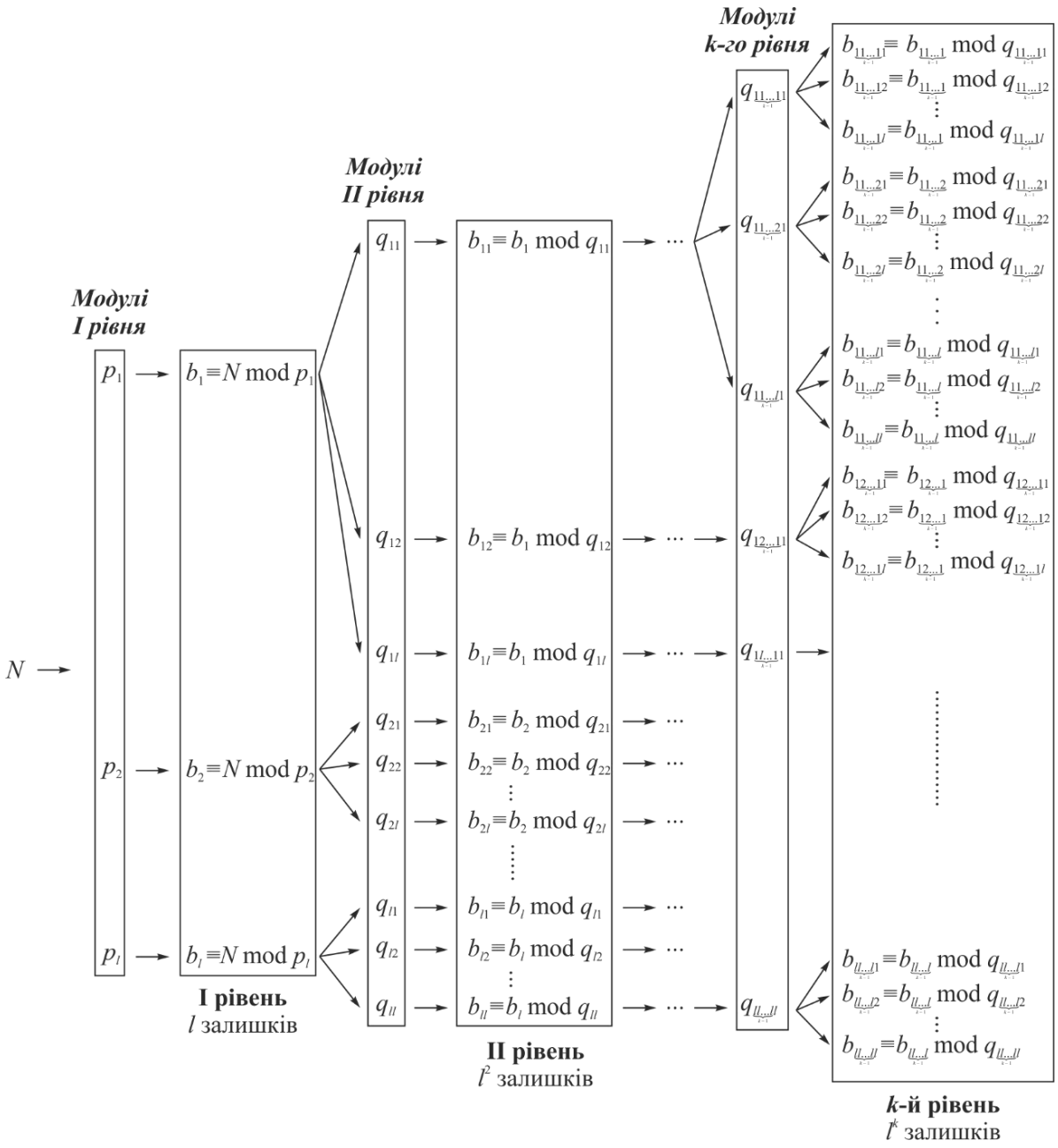


Рисунок 5.1 - Схема передачі залишків по рівнях

На останньому рівні розшифрування отримується числове значення вхідного повідомлення. На рисунку 5.2 представлена загальна схема симетричного шифрування на основі ІСЗК.

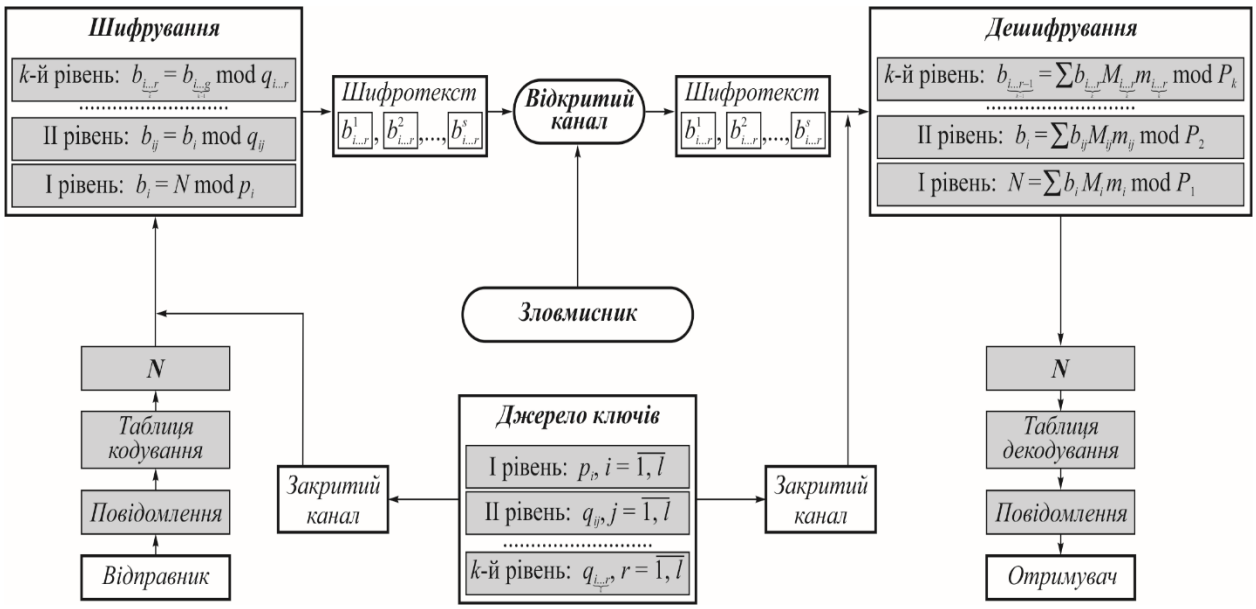


Рисунок 5.2 – Схема симетричного шифрування та розшифрування на основі ІСЗК

5.1.3 Приклад симетричного методу шифрування у ІСЗК

Для прикладу візьмемо дворівневу ІСЗК з наборами модулів І рівня p_1=135789, p_2=195437, p_3=3456785 та ІІ рівня q_{11}=187, q_{12}=355, q_{13}=489, q_{21}=731, q_{22}=574, q_{23}=199, q_{31}=389, q_{32}=591, q_{33}=292. Нехай відкритий текст «I love Ukraine», який розбиваємо по 5 символів, враховуючи пробіли: N_1= «I lov», N_2= «e Ukr», N_3= «aine». За таблицею кодування (табл. 1), одержуємо три 15-и цифрових числа: N_1 = 233132178181188, N_2 = 171132245177184, N_3 = 167175180171. Результати ієрархічного шифрування показані в таблицях 5.2-5.4 для відповідних N_1, N_2, N_3.

Таблиця 5.2

Шифрування N_1

Блок шифрування	*І рівень*		*ІІ рівень*	
N_1 = 233132178181188	p_1=135789	b_1= 22125	q_{11}=187	b_{11}=59
			q_{12}=355	b_{12}=115
			q_{13}=489	b_{13}=120

Продовження таблиці 5.2

N_1 = 233132178181188	p_2=195437	b_2= 34683	q_{21}=731	b_{21}=326
			q_{22}=574	b_{22}=243
			q_{23}=199	b_{23}=57
	p_3=3456785	b_2= 2235408	q_{31}=389	b_{31}=214
			q_{32}=591	b_{32}=246
			q_{33}=292	b_{33}=148

Для блоку «S_1=I lov» відповідає шифротекст (59, 115, 120, 326, 243, 57, 214, 246, 148).

Таблиця 5.3

Шифрування N_2

Блок шифрування	*I рівень*		*II рівень*	
N_2 = 171132245177184	p_1=135789	b_1= 68004	q_{11}=187	b_{11}=123
			q_{12}=355	b_{12}=199
			q_{13}=489	b_{13}=33
	p_2=195437	b_2=5648	q_{21}=731	b_{21}=531
			q_{22}=574	b_{22}=482
			q_{23}=199	b_{23}=76
	p_3=3456785	b_2= 548389	q_{31}=389	b_{31}=288
			q_{32}=591	b_{32}=532
			q_{33}=292	b_{33}=13

Набір залишків шифротексту (123, 199, 33, 531, 482, 76, 288, 532, 13) другого блоку S_2= «e Ukr».

Таблиця 5.4

Шифрування N_3

Блок шифрування	I рівень		II рівень	
N_3 = 167175180171	p_1=135789	b_1= 46500	q_{11}=187	b_{11}=124
			q_{12}=355	b_{12}=350
			q_{13}=489	b_{13}=45
	p_2=195437	b_2=129304	q_{21}=731	b_{21}=648
			q_{22}=574	b_{22}=154
			q_{23}=199	b_{23}=153
	p_3=3456785	b_2=1600786	q_{31}=389	b_{31}=51
			q_{32}=591	b_{32}=358
			q_{33}=292	b_{33}=42

Отже, набір залишків шифротексту, що відповідає третьому блоку N_3= «aine», дорівнює (124, 350, 45, 648, 154, 153, 51, 358, 42). В результаті проведених обчислень на основі розробленого симетричного методу з використанням ІСЗК отримується шифротекст (59, 115, 120, 326, 243, 57, 214, 246, 148, 123, 199, 33, 531, 482, 76, 288, 532, 13, 124, 350, 45, 648, 154, 153, 51, 358, 42).

Процес розшифрування шифротексту (59, 115, 120, 326, 243, 57, 214, 246, 148, 123, 199, 33, 531, 482, 76, 288, 532, 13, 124, 350, 45, 648, 154, 153, 51, 358, 42) відбувається у зворотньому порядку, тобто з другого рівня за модулями q_{11}=187, rq_{12}=355, q_{13}=489, q_{21}=731, q_{22}=574, q_{23}=199, q_{31}=389, q_{32}=591, q_{33}=292. Після відновлення залишків першого рівня відбувається розшифруванням за модулями: p_1=135789, p_2=195437, p_3=3456785. Оскільки системи модулів на першому рівні складаються із 3-х модулів, то відповідно на другому буде 9, то шифротекст розбиваємо його на блоки по 9, кожний з яких у свою чергу ділимо на набір по 3 залишки.

Таким чином перший блок залишків (59, 115, 120, 326, 243, 57, 214, 246, 148) розбивається на систему по три залишки та застосовується китайська

теорема про залишки (5.2). Аналогічно робимо і з наступними двома наборами по дев’ять чисел. Результати розшифрування кожного з блоків представлені в таблицях 6, 7 та 8 відповідно.

Таблиця 5.6

Розшифрування першого блоку (59, 115, 120, 326, 243, 57, 214, 246, 148)

		I			II			III		
II рівень	b_{ij}	b_{11}=59	b_{12}=115	b_{13}=120	b_{21}=326	b_{22}=243	b_{23}=57	b_{31}=214	b_{32}=246	b_{33}=148
	q_{ij}	q_{11}=187	q_{12}=355	q_{13}=489	q_{21}=731	q_{22}=574	q_{23}=199	q_{31}=389	q_{32}=591	q_{33}=292
	M_{ij}	173 595	91 443	66 385	114 226	145 469	419 594	172 572	113 588	229 899
	M_{ij}mod q_{ij}	59	208	370	190	247	102	245	116	95
	$m_{ij} = M_{ij}^{-1} \, mod \; P$	168	227	226	227	165	80	208	107	83
	$b_i = \left(\sum_{j=1}^{l} b_{ij} M_{ij} m_{ij} \right)$	22125			34683			2235408		
I рівень	p_i	p_1=135789			p_2=195437			p_3=3456785		
	M_i	675 583 690 045			469 393 378 365			26 538 194 793		
	$M_i$mod p_i	10 951			22 934			456 348		
	$m_i = M_i^{-1} \, mod \; P$	31954			120284			515812		
	$N_1 = \left(\sum_{i=1}^{l} b_i M_i m_i \right) m$	233 132 178 181 188								
	S_1 (з табл. 1)	**I lov**								

Таблиця 5.7

Розшифрування другого блоку (123, 199, 33, 531, 482, 76, 288, 532, 13)

		I			II			III		
II рівень	b_{ij}	b_{11}=123	b_{12}=199	b_{13}=33	b_{21}=531	b_{22}=482	b_{23}=76	b_{31}=288	b_{32}=532	b_{33}=13
	q_{ij}	q_{11}=187	q_{12}=355	q_{13}=489	q_{21}=731	q_{22}=574	q_{23}=199	q_{31}=389	q_{32}=591	q_{33}=292
	M_{ij}	173595	91443	66385	114 226	145 469	419 594	172 572	113 588	229 899
	M_{ij}mod q_{ij}	59	208	370	190	247	102	245	116	95
	$m_{ij} = M_{ij}^{-1} \, mod \; P$	168	227	226	227	165	80	208	107	83

Продовження таблиці 5.7

	$b_i = \left(\sum_{j=1}^{l} b_{ij} M_{ij} m_{ij}\right) mod$	68004	5648	548389
І рівень	p_i	p_1=135789	p_2=195437	p_3=3456785
	M_i	675 583 690 045	469 393 378 365	26 538 194 793
	$M_i$mod p_i	10 951	22 934	456 348
	$m_i = M_i^{-1} mod\ P$	31954	120284	515812
	$N_2 = \left(\sum_{i=1}^{l} b_i M_i m_i\right) mod\ P$	171 132 245 177 184		
	S_2 (з табл. 1)	**e Ukr**		

Таблиця 5.8

Розшифрування третього блоку (124, 350, 45, 648, 154, 153, 51, 358, 42)

		I			II			III		
ІІ рівень	b_{ij}	b_{11}=124	b_{12}=350	b_{13}=45	b_{21}=648	b_{22}=154	b_{23}=153	b_{31}=51	b_{32}=358	b_{33}=42
	q_{ij}	q_{11}=187	q_{12}=355	q_{13}=489	q_{21}=731	q_{22}=574	q_{23}=199	q_{31}=389	q_{32}=591	q_{33}=292
	M_{ij}	173595	91443	66385	114 226	145 469	419 594	172 572	113 588	229 899
	M_{ij}mod q_{ij}	59	208	370	190	247	102	245	116	95
	$m_{ij} = M_{ij}^{-1} mod\ P$	168	227	226	227	165	80	208	107	83
	$b_i = \left(\sum_{j=1}^{l} b_{ij} M_{ij} m_{ij}\right) mo$	46500			129304			1600786		
І рівень	p_i	p_1=135789			p_2=195437			p_3=3456785		
	M_i	675 583 690 045			469 393 378 365			26 538 194 793		
	$M_i$mod p_i	10 951			22 934			456 348		
	$m_i = M_i^{-1} mod\ P$	31954			120284			515812		
	$N_3 = \left(\sum_{i=1}^{l} b_i M_i m_i\right) mod$	167 175 180 171								
	S_3 (з табл. 1)	**aine**								

Отже, після поєднання розшифрованих текстів усіх блоків, отримується вхідне повідомлення «I love Ukraine».

5.2 Симетричний криптоалгоритм в поліноміальній ієрархічній системі залишкових класів

Будь-який поліном $N(x)$ над полем $Z_p[x]$ у ПСЗК можна задати сукупністю залишків $b_i(x)$ від його ділення на кожен із системи модулів $p_i(x)$ (i=1, 2, …, l, l – кількість модулів), причому усі поліноми $p_i(x)$ є попарно взаємно прості:

$$b_i(x) = N(x) \bmod p_i(x). \tag{5.8}$$

Це означає, що між поліномом $N(x)$ над полем $Z_p[x]$ з проміжку $[0, P(x))$, де $P(x) = \prod_{i=1}^{l} p_i\,(x)$— причому степінь ($\deg N(x) < \deg P(x)$), в поліноміальній системі числення і його залишками $b_i(x)$ існує взаємно–однозначна відповідність. Одним із найпоширеніших способів відновлення полінома $N(x)$ із його залишків є використання КТЗ:

$$N(x) = \left(\sum_{i=1}^{l} b_i(x)\, M_i(x) m_i(x)\right) mod\, P\,(x) \tag{5.9}$$

де $M_i(x) = \frac{P(x)}{p_i(x)}$, $m_i(x)$ знаходиться згідно з виразом $m_i(x) = M_i(x)^{-1}\, mod\, p_i\,(x)$.

Слід зазначити, що у [324] представлено два інших підходи щодо відновлення поліномів, які характеризуються меншою часовою складністю. Основна ідея, яких полягає в тому, що будь-яку конгруенцію за модулем $p_1(x)$ з залишком $b_1(x)$ (наприклад, $a(x)\, mod\, p_1\,(x) \equiv b_1(x)$) можна представити у вигляді $a(x) = \gamma(x)p_1(x) + b_1(x)$, де $\gamma(x)$ - поліном, який вказує, скільки разів потрібно додавати модуль-поліном $p_1(x)$ до залишку $b_1(x) = N_1(x)$, щоб виконувалося співвідношення $N_2(x)\, mod\, p_2\,(x) \equiv b_2(x)$. Тут $N_2(x) = N_1(x) + \gamma_1(x)p_1(x)$. Далі необхідно додавати добуток $p_1(x)p_2(x)$ до тих пір,

поки не буде виконуватись конгруенція $N_3(x)\,mod\,p_3\,(x) \equiv b_3(x)$, де $N_3(x) = N_2(x) + \gamma_2(x)p_1(x)p_2(x)$. Дана процедура продовжується до тих пір, поки не буде виконуватись останнє рівняння (5.11). Аналітично це записується таким чином:

$$
\begin{aligned}
&N_1(x) = b_1(x);\\
&N_2(x) = N_1(x) + \gamma_1(x)p_1(x) = b_1(x) + \gamma_1(x)p_1(x)\,;\ N_2(x)\,mod\,p_2\,(x) \equiv b_2(x);\\
&N_3(x) = N_2(x) + \gamma_2(x)p_1(x)p_2(x) = b_1(x) + \gamma_1(x)p_1(x) + \gamma_2(x)p_1(x)p_2(x);\ N_3(x)\,mod\,p_3\,(x) \equiv b_3(x);\\
&N_i(x) = N_{i-1}(x) + \gamma_{i-1}(x)p_1(x)p_2(x)\ldots p_{i-1}(x);\ N_i(x)\,mod\,p_i\,(x) \equiv b_i(x);\\
&\qquad\qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (5.11)\\
&N_k(x) = N(x) = N_{k-1}(x) + \gamma_{k-1}(x)p_1(x)p_2(x)\ldots p_{k-1}(x);\\
&N_k(x)\,mod\,p_k\,(x) \equiv r_k(x).
\end{aligned}
$$

Пошук $\gamma_i(x)$ здійснюється методом невизначених коефіцієнтів, причому на кожному кроці алгоритму степінь поліному $\gamma_i(x) = A_i x^i + A_{i-1}x^{i-1} + \ldots + A_1 x + A_0$ буде на 1 менший степеня $p_{i+1}(x)$: $deg\,\gamma_i\,(x) = deg\,p_{i+1}\,(x) - 1$. На рисунку 5.3 представлено схему відновлення полінома по його залишках з використанням методу, який ґрунтується на додаванні добутку поліномів-модулів:

Цей підхід дозволяє забезпечити обмеження результатів проміжних обчислень в межах встановленого діапазону $P(x)$. Що, в свою чергу, запобігає виникненню переповнення діапазону обчислень і усуває потребу у виконанні операції знаходження залишку за модулем $P(x)$.

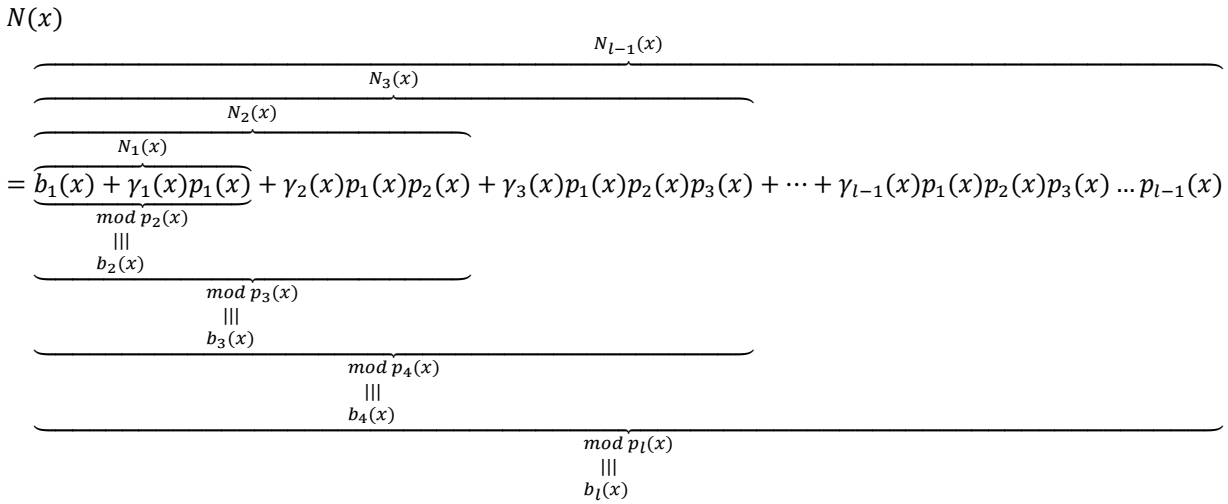


Рисунок 5.3 – Схема відновлення полінома по його залишках

Методика схожа на алгоритм Гарнера, де для обчислення поліномів $\gamma_i(x)$ використовується метод невизначених коефіцієнтів: $\gamma_i(x) = (b_{i+1}(x) - N_i(x))((p_1(x)p_2(x)p_3(x) \dots p_i(x))^{-1} \, mod \, p_{i+1}(x)) \, mod \, p_{i+1}(x)$.

Для спрощення обчислень, які використовуються у запропонованому методі, можна додавати не добуток поліномів-модулів, а залишок цього добутку від ділення на відповідний поліном. Математичний запис даного методу виглядає таким чином:

$$N_1(x) = b_1(x);\ p_{11}(x) = p_1(x) \, mod \, p_2(x);$$
$$(N_1(x) + \gamma_1(x)p_{11}(x)) \, mod \, p_2(x) = b_2(x);\ N_2(x) = N_1(x) + \gamma_1(x)p_1(x);$$
$$p_{12}(x) = p_1(x)p_2(x) \, mod \, p_3(x);$$
$$(N_2(x) + \gamma_2(x)p_{12}(x)) \, mod \, p_3(x) = b_3(x),\ N_3(x) = N_2(x) +$$
$$\gamma_2(x)p_1(x)p_2(x);\ p_{13}(x) = p_1(x)p_2(x)p_3(x) \, mod \, p_4(x);$$
$$(N_{i-1}(x) + \gamma_{i-1}(x)p_{1i-1}(x)) \, mod \, p_i(x) = b_i(x);\ N_i(x) = N_{i-1}(x) +$$
$$\gamma_{i-1}(x)p_1(x)p_2(x)p_3(x) \dots p_{i-1}(x);$$
$$p_{1i}(x) = p_1(x)p_2(x) \dots p_i(x) \, mod \, p_{i+1}(x); \qquad (5.12)$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$(N_{k-1}(x) + \gamma_{k-1}(x)p_{1k-1}(x)) \, mod \, p_k(x) = r_k(x);\ N(x) = N_k(x) = N_{k-1}(x) +$$
$$\gamma_{k-1}(x)p_1(x)p_2(x)p_3(x) \dots p_{k-1}(x).$$

Даний підхід відображає стратегію оптимізації арифметичних обчислень для систем з степенями поліномів, добуток яких менший за діапазон обчислень. Його основна мета - запобігти переповненню діапазону та уникнути необхідності виконувати операції за модулем $P(x)$, що сприяє підвищенню точності та ефективності обчислень.

Цей метод розглядається як аналог алгоритму Гарнера, в якому використовуються методи знаходження оберненого поліному за модулем для визначення коефіцієнтів:

$$\gamma_i(x) = \big(b_{i+1}(x) - N_i(x)\big)\Big(\big(p_1(x)p_2(x)\dots p_{i-1}(x)p_i(x)\big)\, mod\, p_{i+1}(x)\Big)^{-1} mod\, p_{i+1}(x)$$

Впровадження даного підходу передбачає обмеження проміжних обчислень до встановленого діапазону $P(x)$, що забезпечує надійну роботу, запобігає можливим втратам точності та дозволяє підтримувати оптимальну продуктивність операцій у системах з обмеженим числовим представленням.

5.2.1 Теоретичні основи ієрархічної поліноміальної системи залишкових класів

Ієрархічна поліноміальна система залишкових класів (ІПСЗК) ґрунтується на математичних принципах поліномів над скінченими полями [325, 326]. Зі зростанням обсягів опрацювання, передачі та зберігання інформації необхідно збільшувати кількість поліномів та їх порядку. Це призводить до ускладнення апаратного забезпечення та збільшення часу виконання операцій. У зв'язку з цим виникає необхідність оптимізації, а одним з підходів до цього є ІПСЗК, яка дає змогу зменшити порядок поліномів. Для спрощення вважаємо, що кількість поліномів у кожній системі на будь-якому рівні однакова і дорівнює l.

Нехай головна система $p_1(x)$, $p_2(x)$, …, $p_l(x)$ забезпечує можливість виконання операцій в діапазоні $[0, P_1(x))$, де $P_1(x) = p_1(x)p_2(x)\dots p_l(x) = \prod_{i=1}^{l} p_i(x)$. Максимальний діапазон обчислень, який можна отримати в цій

системі при перемноженні, становить $(p_l(x)\text{-}1)^2$. Тоді всі залишки головної системи можна представити в новій системі з основами $q_{11}(x)$, $q_{12}(x)$, …, $q_{1l}(x)$, $q_{21}(x)$, $q_{22}(x)$, …, $q_{2l}(x)$, …, $q_{l1}(x)$, $q_{l2}(x)$, …, $q_{ll}(x)$ з відповідними діапазонами. Далі, в свою чергу, залишки другого рівня з дотриманням відповідних вимог записуються в системі модулів третього рівня. Така процедура триває до останнього (k-го) рівня. Кількість рівнів, як правило, визначається та залежить від умов забезпечення необхідного рівня захисту для конкретної задачі. Такий процес переходу до поліномів (модулів) меншого порядку помітно спрощує реалізацію. Крім того, ІПСЗК можуть мати великий ключовий простір, що робить їх стійкими до атак з використанням методів перебору та дозволяють паралельну обробку на багатоядерних або розподілених системах, що підвищує швидкість виконання операцій.

5.2.2 Теоретичні основи симетричного шифрування у ієрархічній поліноміальній системі залишкових класів

У симетричному шифруванні з використанням ІПСЗК на кожному із k рівнів ($k \geq 1$), кількість яких обумовлюється обома абонентами, знаходять залишки по відповідній системі модулів, які передають на наступний рівень. Кожному залишку r-ого рівня відповідає l систем з l^{r-1} залишками. Таким чином на $r + 1$ рівень передається l^r залишків ($l > 1$, $r = 1, 2, \ldots, k - 1$). Отже, шифротекст складається із l^k поліномів, які є залишками останнього рівня ІПСЗК (рис.5.4). Набори модулів відомі для відправника і отримувача.

Для шифрування буквенна інформація записується у числовій формі з використанням ASCII-кодів. Після цього її необхідно представити у вигляді поліному з коефіцієнтами, які відображають буквенну інформацію, тобто відкритий текст $N(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 x^o$, де a_i - послідовність цифрового представлення букв, $i = \overline{0 \ldots n}$, ($n$+1) – довжина повідомлення. Далі блок відкритого тексту $N(x)$ записується в ПСЗК згідно виразу (5.9).

Таким чином, шифротекстом у розробленому поліноміальному ієрархічному методі є набір залишків, отриманих за формулою (5.8),

останнього рівня кожного блоку (рис. 5.4). Відновлення вхідного повідомлення (поліному) відбувається у зворотньому порядку починаючи з *k*-го рівня до першого на основі використанням виразів (5.11).

В результаті проведених обчислень *k*-го рівня отримуються значення, які є залишками (поліномами) *k*-1 рівня. На кожному з рівнів в процесі розшифрування кількість залишків поліномів зменшується в *l* разів.

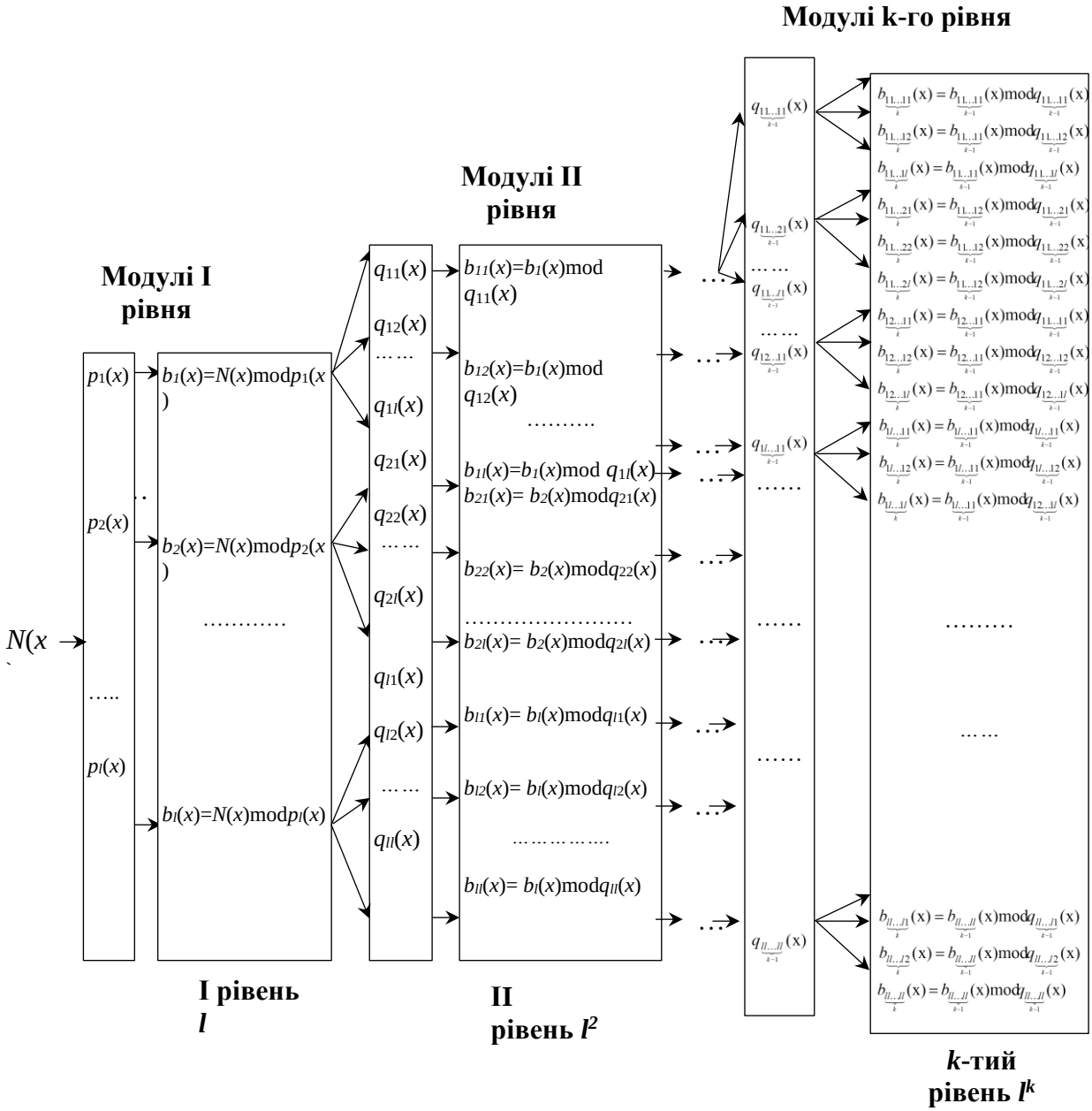


Рисунок 5.4 – Схема передачі поліномів-залишків по рівнях

На останньому рівні розшифрування отримуємо початковий вхідний поліном, який відповідає вхідному повідомленню.

На рисунку 5.5 представлена загальна схема симетричного шифрування на основі ІПСЗК.

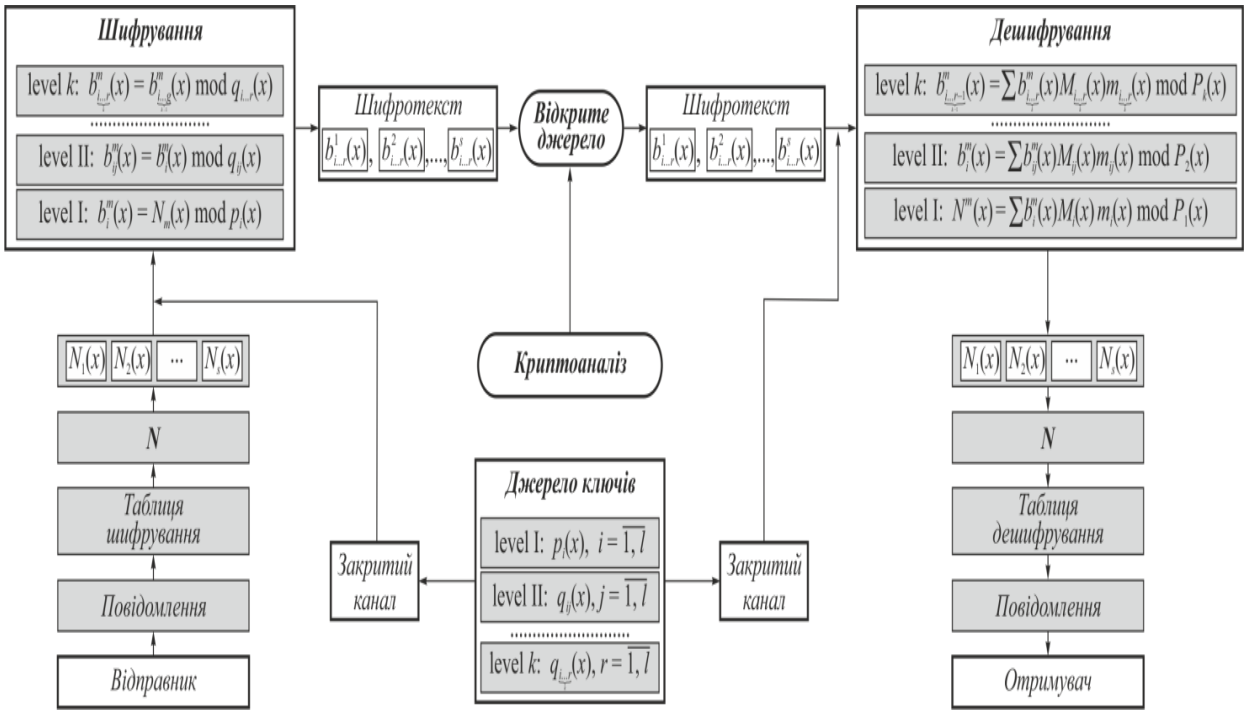


Рисунок 5.5 – Схема симетричного шифрування та розшифрування на основі ІПСЗК

5.2.3 Приклад симетричного шифрування та розшифрування на основі ІПСЗК

Для прикладу розглянемо дворівневу ІПСЗК з наборами модулів І рівня $p_1(x) = x^4 + x^3 + 3x^2 + x + 1$, $p_2(x) = x^4 + x^3 + x^2 + 2x + 1$, $p_3(x) = x^4 + 2x^3 + x^2 + 3x + 1$. Для спрощення обчислень модулі ІІ рівня вибиралися однакові для кожного з $p_i(x)$: $q_{11}(x) = q_{21}(x) = q_{31}(x) = x^3 + x^2 + 3x + 1$, $q_{12}(x) = q_{22}(x) = q_{32}(x) = x^3 + 3x^2 + x + 1$, $q_{13}(x) = q_{23}(x) = q_{33}(x) = x^3 + 2x^2 + x + 1$. Нехай відкритий текст «Cryptography» перетворюється у поліноном, в якому коефіцієнтами є ASCII – коди, відповідні буквам відкритого повідомлення. В результаті отримується відкритий текст $N(x) = 67x^{11} + 114x^{10} + 121x^9 + 112x^8 + 116x^7 + 111x^6 + 103x^5 + 114x^4 + 97x^3 + 112x^2 + 104x + 121$.

Результати його шифрування за допомогою ІПСЗК показані в таблиці 5.9.

Таблиця 5.9 - Шифрування $S(x)$

<table>
<tr><td colspan="5">$N(x) = 67x^{11} + 114x^{10} + 121x^9 + 112x^8 + 116x^7 + 111x^6 + 103x^5 + 114x^4 + 97x^3 + 112x^2 + 104x + 121$</td></tr>
<tr><td colspan="3">i</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td colspan="3">Модулі першого рівня $p_i(x)$</td><td>$x^4 + x^3 + 3x^2 + x + 1$</td><td>$x^4 + x^3 + x^2 + 2x + 1$</td><td>$x^4 + 2x^3 + x^2 + 3x + 1$</td></tr>
<tr><td colspan="3">Залишки першого рівня $b_i(x)$</td><td>$712x^3 – 2319x^2 – 389x – 975$</td><td>$–171x^3 – 152x^2 + 146x + 203$</td><td>$9417x^3 – 827x^2 + 11671x + 4695$</td></tr>
<tr><td colspan="3">Модулі другого рівня $q_{ij}(x)$, j=1, 2, 3</td><td colspan="3">$q_{i1}(x) = x^3 + x^2 + 3x + 1$; $q_{i2}(x) = x^3 + 3x^2 + x + 1$; $q_{i3}(x) = x^3 + 2x^2 + x + 1$.</td></tr>
<tr><td rowspan="3">Залишки другого рівня $b_{ij}(x)$</td><td rowspan="3">j</td><td>1</td><td>$–3031x^2 – 1747x – 1687$</td><td>$19x^2 + 659x + 374$</td><td>$–10244x^2 – 16580x – 4722$</td></tr>
<tr><td>2</td><td>$–4455x^2 – 323x – 1687$</td><td>$361x^2 + 317x + 374$</td><td>$–29078x^2 + 2254x – 4722$</td></tr>
<tr><td>3</td><td>$–3743x^2 – 323x – 1687$</td><td>$190x^2 + 317x + 374$</td><td>$–19661x^2 + 2254x – 4722$</td></tr>
</table>

Отже, відкритому повідомленню «$N(x)$= Cryptography» відповідає шифротекст ($-591x^2 - 5270x - 1687$, $-4943x^2 - 918x - 1687$, $-2767x^2 - 918x - 1687$, $-103x^2 + 476x + 252$, $361x^2 + 12x + 252$, $129x^2 + 12x + 252$, $9337x^2 + 15811x + 4733$, $26615x^2 - 1467x + 4733$, $17976x^2 - 1467x + 4733$).

Процес розшифрування відбувається у зворотньому порядку, тобто з другого рівня на основі КТЗ за модулями $q_{ij}(x)$, i, j =1, 2, 3. Після відновлення залишків першого рівня відбувається розшифрування за модулями p_i, i=1, 2, 3.

Процес і результати розшифрування представлені в таблиці 5.10.

Таблиця 5.10

Проміжні обчислення та результат розшифрування повідомлення

i			1	2	3
Залишки другого рівня $b_{ij}(x)$	j	1	$-3031x^2 - 1747x - 1687$	$19x^2 + 659x + 374$	$-10244x^2 - 16580x - 4722$
		2	$-4455x^2 - 323x - 1687$	$361x^2 + 317x + 374$	$-29078x^2 + 2254x - 4722$
		3	$-3743x^2 - 323x - 1687$	$190x^2 + 317x + 374$	$-19661x^2 + 2254x - 4722$
Модулі другого рівня $q_{ij}(x)$, j=1, 2, 3			$q_{i1}(x) = x^3 + x^2 + 3x + 1;\ q_{i2}(x) = x^3 + 3x^2 + x + 1;$ $q_{i3}(x) = x^3 + 2x^2 + x + 1.$		
Базисні числа $M_{ij} = \frac{\prod_j q_{ij}(x)}{q_{ij}(x)}$, i, j=1, 2, 3			$M_{i1}(x)=x^6 + 5x^5 + 8x^4 + 7x^3 + 6x^2 + 2x + 1;$ $M_{i2}(x)=x^6 + 3x^5 + 6x^4 + 9x^3 + 6x^2 + 4x + 1;$ $M_{i3}(x)=x^6 + 4x^5 + 7x^4 + 12x^3 + 7x^2 + 4x + 1.$		
Базисні числа $m_{ij}(x) = M_{ij}^{-1}(x) \bmod p_{ij}$, j=1, 2, 3			$m_{i1}(x) = 0{,}45175438596491x^2 + 0{,}27192982456140x + 1{,}23245614035088;$ $m_{i2}(x) = -1{,}08333333333333x^2 - 3{,}33333333333333x - 0{,}91666666666666;$ $m_{i3}(x) = 0{,}63157894736842x^2 + 1{,}52631578947368x + 0{,}68421052631579.$		
Залишки першого рівня $b_i(x) = \left(\sum_{j=1}^{l} b_{ij}(x) M_{ij(x)} m_{ij(x)}\right) mod\ P_{ij}(x)$			$712x^3 - 2319x^2 - 38+9x - 975$	$-171x^3 - 152x^2 + 146x + 203$	$9417x^3 - 827x^2 + 16+71x + 4695$
Модулі першого рівня $p_i(x)$			$x^4 + x^3 + 3x^2 + x + 1$	$x^4 + x^3 + x^2 + 2x + 1$	$x^4 + 2x^3 + x^2 + 3x + 1$
Базисні числа $M_i(x) = \frac{P(x)}{p_i(x)}$			$x^8 + 3x^7 + 4x^6 + 8x^5 + 10x^4 + 8x^3 + 8x^2 + 5x + 1$	$x^8 + 3x^7 + 6x^6 + 11x^5 + 10x^4 + 13x^3 + 7x^2 + 4x + 1$	$x^8 + 2x^7 + 5x^6 + 7x^5 + 8x^4 + 9x^3 + 6x^2 + 3x + 1$
Базисні числа $m_i(x) = M_i^{-1}(x) \bmod p_i$			$-0{,}32090933121861x^3 - 0{,}47105471847740x^2 - 0{,}95083267248216x - 0{,}57467618292361$	$-0{,}75641025641025x^3 - 0{,}38461538461539x^2 - 0{,}44871794871795x - 0{,}98717948717949$	$1{,}07731958762887x^3 + 1{,}93298969072165x^2 + 0{,}53608247422680x + 2{,}56185567010309$
Відновлення вхідного поліному $N(x) = \left(\frac{\sum_{i=1}^{l} b_i(x)\, M_i(x)}{m_i(x)}\right) mod\ P(x)$			$67x^{11} + 114x^{10} + 121x^9 + 112x^8 + 116x^7 + 111x^6 + 103x^5 + 114x^4 + 97x^3 + 112x^2 + 104x + 121$		
$N(x)$			Cryptography		

Отже, після розшифрування отримується вхідний поліном $N(x) = 67x^{11} + 114x^{10} + 121x^9 + 112x^8 + 116x^7 + 111x^6 + 103x^5 + 114x^4 + 97x^3 + 112x^2 + 104x + 121$, який відповідає повідомленню «Cryptography».

5.3 Оцінка криптостійкості симетричних криптоалгоритмів в ієрархічній системі залишкових класів

5.3.1 Оцінка криптостійкості симетричного криптоалгоритму в цілочисельній ієрархічній системі залишкових класів

Як зазначено у статті [313], складність криптоаналізу симетричного методу шифрування у цілочисельній СЗК, при умові, що модулі криптоперетворення є простими числами, становить $O\left(\left(\frac{2^{n+1}}{n}\right)^l n^2\right)$, де n – розрядність модулів. При досліджені криптостійкості розробленого алгоритму з використанням ІСЗК необхідно враховувати кількість рівнів k та зміну розрядності модулів на кожному з них. Тому загальна часова складність криптоаналізу по всіх рівнях обчислюється згідно такої формули:

$$O(n,k,l) = \prod_k \left(\frac{2^{n_k+1}}{n_k}\right)^{l_k} {n_k}^2 \tag{5.3}$$

де $n_k = n - k + 1$ – розрядність на k-ому рівні, $l_k = l^k$ – кількість модулів на k-ому рівні.

З врахуванням значень параметрів n_k і l_k формула (5.3) набуде такого вигляду:

$$O(n,k,l) = \prod_k \left(\frac{2^{n-k+1}}{n-k+1}\right)^{l^k} (n-k+1)^2 \tag{5.4}$$

Для прикладу криптостійкість запропонованого алгоритму для k рівневої ІСЗК при кількості модулів на першому рівні l=3 описується виразом:

$$O(n,k) = \prod_k \left(\frac{2^{n-k+1}}{n-k+1}\right)^{3^k} (n-k+1)^2 \quad (5.5)$$

На рисунку 5.6 представлений графік, який відображає логарифмічну залежність складності криптоаналізу від розрядності *n*, кількості модулів *l* та рівнів *k*. Явно спостерігається, що зі зростанням цих параметрів складність криптоаналізу збільшується.

Результати експериментальних досліджень вказують, що стійкість запропонованої симетричної криптосистеми, яка базується на використанні ієрархічної системи залишкових класів зростає із збільшення кількості рівнів та розрядності вхідних параметрів (модулів).

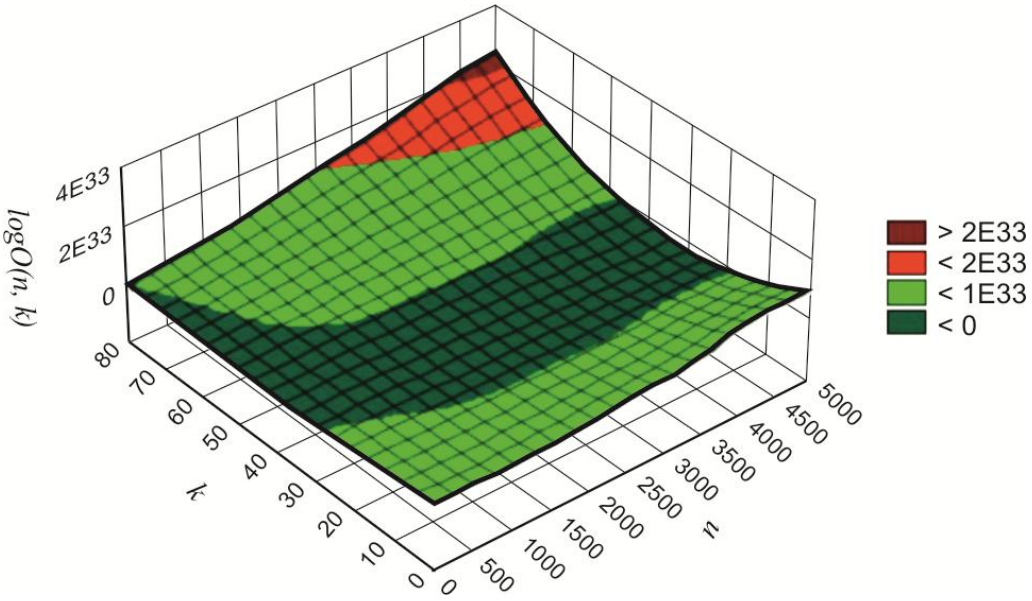


Рисунок 5.6 - Залежність логарифму складності криптоаналізу від розрядності *n,* кількості модулів *l* та рівнів *k*

З огляду літературних джерел відомо [327, 328], що криптоаналіз сучасного симетричного шифру AES-256 становить 2^{255} бітових операцій. З рівності $\prod_k \left(\frac{2^{n-k+1}}{n-k+1}\right)^{l^k} (n-k+1)^2 = 2^{255}$ можна знайти розрядності, кількості рівнів та кількості модулів ІСЗК, які забезпечують таку ж стійкість, як і алгоритм AES-256.

Після логарифмування за основою 2 та виконання елементарних арифметичних перетворень, можна одержати:

$$\sum_k (l^k(n-k+1) + (2-l^k)log_2(n-k+1)) = 255. \quad (5.6)$$

Розв'язок рівняння (5.6) відносно n для різних значень l та k, одержимо можливі значення n (таблиця 5.11).

Таблиця 5.11. Криптостійкість розробленого алгоритму в ІСЗК при різних розрядностях модулів, кількостях модулів та рівнів

n=8					
k	l=2	l=3	l=4	l=5	l=6
1	16	21	26	31	36
2	38,38529	64,34852	98,69703	141,4308	192,5499
3	70,87552	161,7245	**322,4294**	**573,4804**	**935,368**
4	118,3685	**383,2921**	**1012,66**	**2251,919**	**4410,793**
5	186,3685	**873,2921**	**3064,66**	**8505,919**	**19966,79**
6	**280,1008**	**1908,024**	**8863,823**	**30619,05**	**85989,95**
n=12					
k	l=2	l=3	l=4	l=5	l=6
1	24	32,41504	40,83007	49,24511	57,66015
2	61,08114	107,199	168,398	244,6782	**336,0395**
3	121,1496	**294,1508**	**602,4385**	**1086,081**	**1785,147**
4	220,7706	**772,7267**	**2101,278**	**4736,218**	**9347,264**
5	**386,7706**	**1993,727**	**7227,278**	**20367,22**	**48233,26**

Таблиця 5.11 демонструє, що запропонований симетричний криптоалгоритм на основі ІСЗК забезпечує рівень криптографічної стійкості, порівнянний з алгоритмом AES-256, при певних значеннях параметрів n, l, k.

Зокрема, для 8-бітних модулів при l=2 потрібно 6 рівнів, щоб перевищити стійкість AES-256. Збільшення кількості модулів приводить до зменшення рівнів шифрування (якщо l=3, то k=4; якщо l=4-6, то k=3).

Для 12-бітних модулів при l=2 стійкість запропонованого криптоалгоритму перевищує стійкість AES-256 на п'ятому рівні. При

збільшенні кількості модулів (l=3-5) умова перевищення стійкості AES-256 виконується на третьому рівні, а при l=6 – на другому. Встановлено, що при n=8, l=5, k=4 і при n=12, l=4, k=4 спостерігається підвищення криптостійкості до криптоаналітичних атак приблизно в 9 разів.

Таким чином, варіація розрядності модулів, їх кількості та рівнів ієрархії дозволяє досягнути відповідної криптографічної стійкості запропонованого алгоритму залежно від конкретної задачі.

5.3.2 Оцінка криптостійкості на основі функції Ейлера

Більш точну оцінку криптографічної стійкості запропонованого методу шифрування на основі ІСЗК можна досягнути з використанням функції Ейлера $\varphi(p_i)$. Крім того максимальне значення $\varphi_{\max}(p_i)$ отримується тоді, коли $p_{i\max}^{(n)}$ - максимальне просте число із заданої розрядності n, оскільки $\varphi_{\max}(p_{i\max}^{(n)}) = p_{i\max}^{(n)} - 1$. Наприклад, для 16 - розрядних чисел $p_{i\max}^{(16)} = 509$, $\varphi_{\max}(p_{i\max}^{(8)}) = 508$.

Відповідно мінімальне значення функції Ейлера $\varphi_{\min}(p_i)$ буде в тому випадку, коли p_i розкладається в добуток послідовних простих чисел одиничного степеня: $p_i = \prod_{m=1}^{l} p_m$, де l – кількість множників, p_l =2, 3, 5, … - послідовні прості числа. Для 8 - розрядних чисел $p_i^{(8)}$ мінімальне значення $\varphi_{\min}(p_i^{(8)})$ буде тоді, коли $p_i^{(16)} = 5 \cdot 7 \cdot 11 = 385$, і відповідно $\varphi_{\min}(p_i^{(16)}) = \varphi(5) \cdot \varphi(7) \cdot \varphi(11) = 4 \cdot 6 \cdot 10 = 240$.

У випадку максимально фіксованого простого числа p_1 кількість можливих варіантів вибору p_2 обмежено значенням функції Ейлера $\phi(p_1)$, для p_3 – відповідно $\phi(p_2)$, …, $p_l - \phi(p_{l-1})$. Припускаючи, що лише один з модулів

може бути складеним і $p_1 > p_2 > \cdots > p_{l-1} > p_l$, загальна кількість всіх можливих наборів модулів на першому рівні ІСЗК може бути оцінена за допомогою такої формули:

$$K = \prod_{i=1}^{l-1} \phi(p_i). \quad (5.6)$$

З врахування того факту, що на кожному наступному рівні квадратично зростає кількість модулів, а їх розрядність квадратично зменшується, то загальна кількість наборів модулів для *k* рівнів ІСЗК можна оцінити за такою формулою:

$$K = \prod_k \prod_{i=1}^{l^k-1} \phi\left(p_i^k\right). \quad (5.7)$$

Даний метод дозволяє провести оцінку криптостійкості алгоритму шифрування в залежності від значення параметра *k,* яке може бути максимальним, мінімальним або приймати проміжні значення. Варто зауважити, що при максимальному значенні *k* забезпечується найбільша криптостійкість алгоритму шифрування, а при мінімальному - найменша.

З врахуванням того, що в оцінці стійкості значення функції Ейлера на першому рівні є *n* - розрядними числами, а зі збільшенням рівнів зростає кількість модулів, наприклад на *k* – рівні буде l^k, і відповідно зменшуються їх розрядності до *n-k*+1, складність пошуку кількості варіантів ключів обчислюється згідно співвідношення $O(\prod_k (l^k - 1) \cdot (n - k + 1)^2)$. Тому загальна оцінка стійкості становитиме:

$$O(\prod_k (l^k - 1) \cdot (n - k + 1)^4). \quad (5.8)$$

В таблиці 5.6 представлено результати чисельного експерименту для параметрів $l = 3,4,5$ *k*=1,2 3 4 5 та розрядностей n =64, 128, 512, 1024, 2048

бітів.

Таблиця 5.12

Криптостійкість на кожному з рівнів симетричної системи шифрування в ієрархічній системі залишкових класів

O(n), l=3					
k / n	1	2	3	4	5
8	8192	19208	33696	50000	61952
16	131072	405000	998816	2284880	5018112
32	2097152	7388168	21060000	56582480	1.49E+08
64	33554432	1.26E+08	3.84E+08	1.11E+09	3.14E+09
128	5.37E+08	2.08E+09	6.55E+09	1.95E+10	5.72E+10
256	8.59E+09	3.38E+10	1.08E+11	3.28E+11	9.76E+11
512	1.37E+11	5.45E+11	1.76E+12	5.37E+12	1.61E+13
1028	2.2E+12	8.76E+12	2.84E+13	8.69E+13	2.62E+14
2048	3.52E+13	1.4E+14	4.56E+14	1.4E+15	4.22E+15
4096	5.63E+14	2.25E+15	7.3E+15	2.25E+16	6.79E+16
O(n), l=4					
k / n	1	2	3	4	5
8	12288	36015	81648	159375	261888
16	196608	759375	2420208	7283055	21212928
32	3145728	13852815	51030000	1.8E+08	6.29E+08
64	50331648	2.36E+08	9.31E+08	3.53E+09	1.33E+10
128	8.05E+08	3.9E+09	1.59E+10	6.23E+10	2.42E+11
256	1.29E+10	6.34E+10	2.62E+11	1.04E+12	4.13E+12
512	2.06E+11	1.02E+12	4.26E+12	1.71E+13	6.81E+13
1028	3.3E+12	1.64E+13	6.87E+13	2.77E+14	1.11E+15
2048	5.28E+13	2.63E+14	1.1E+15	4.46E+15	1.79E+16
4096	8.44E+14	4.22E+15	1.77E+16	7.16E+16	2.87E+17
O(n), l=5					
k / n	1	2	3	4	5
8	16384	57624	160704	390000	799744
16	262144	1215000	4763584	17822064	64779264
32	4194304	22164504	1E+08	4.41E+08	1.92E+09
64	67108864	3.78E+08	1.83E+09	8.64E+09	4.05E+10
128	1.07E+09	6.24E+09	3.13E+10	1.52E+11	7.39E+11
256	1.72E+10	1.01E+11	5.16E+11	2.56E+12	1.26E+13
512	2.75E+11	1.64E+12	8.39E+12	4.19E+13	2.08E+14
1028	4.4E+12	2.63E+13	1.35E+14	6.78E+14	3.38E+15
2048	7.04E+13	4.21E+14	2.17E+15	1.09E+16	5.45E+16
4096	1.13E+15	6.75E+15	3.48E+16	1.75E+17	8.76E+17

Для визначення загальної стійкості на всіх 5 рівнях для відповідних розрядностей вхідних параметрів і різної кількості модулів наобхідно скористатися формулою (5.8), результати приведено в таблиці 5.13.

Таблиця 5.13

Криптостійкість симетричної системи шифрування в ієрархічній системі залишкових класів

l / n	3	4	5
8	1.64239E+22	1.50815E+24	4.73223E+25
16	6.07931E+29	5.58244E+31	1.75164E+33
32	2.74634E+36	2.52188E+38	7.91306E+39
64	5.6438E+42	5.18253E+44	1.62615E+46
128	8.18206E+48	7.51333E+50	2.3575E+52
256	1.00586E+55	9.23652E+56	2.8982E+58
512	1.14122E+61	1.04795E+63	3.2882E+64
1028	1.24454E+67	1.14282E+69	3.5859E+70
2048	1.33079E+73	1.22202E+75	3.83441E+76
4096	1.40914E+79	1.29397E+81	4.06018E+82

Результати проведених досліджень свідчать про підвищення криптографічної стійкості із зростанням вхідних параметрів криптосистеми. Зокрема, встановлено, що додавання одного модуля призводить до збільшення стійкості приблизно в 92 рази, а додавання двох модулів забезпечує зростання стійкості у 2881 разів.

5.4 Оцінка стійкості симетричного криптоалгоритму в поліноміальній ієрархічній системі залишкових класів

Для отримання оцінки криптостійкості симетричного криптоалгоритму в ІПСЗК на основі закону розподілу незвідних поліномів, першочергово потрібно визначити їх кількість на кожному рівні з використанням формули (4.43). На першому рівні кількість буде визначатися згідно співвідношення:

$$S_p(n) = \frac{1}{n}\sum_{n|d} \mu\,(d) p^{n/d}$$

Оскільки кількість поліномів модулів на кожному з рівнів зростає, для спрощення обчислень припускаємо, що їх кількість збільшується пропорційно, тобто, наприклад при трьох модулях на першому рівні кількість на другому рівні буде 3^2, відповідно на *k*-тому рівні буде 3^k . Для розрахунку криптографічної стійкості запропонованого алгоритму, слід врахувати те, що на кожному наступному рівні порядок полінома зменшується мінімум на 1. Тому отримання чисельних експериментів стійкості ієрархічного симетричного поліноміального алгоритму шифрування для k=4 при вхідних поліномах (ключах) n=32 необхідно знайти значення кількості незвідних поліномів $S_p(32)$, $S_p(31)$, $S_p(30)$, $S_p(29)$ (таблиця 5.6).

Таблиця 5.6

Кількість незвідних поліномів для різних степенів n=32, 31, 30, 29 і значень параметра p=2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,101.

n / *p*	32	31	30	29
2	134215680	69273666	35790265.5	18512790
3	5.791E+13	1.99249E+13	6.863E+12	2.37E+12
5	7.276E+20	1.50213E+20	3.1044E+19	6.42E+18
7	3.451E+25	5.08953E+24	7.5131E+23	1.11E+23
11	6.598E+31	6.19172E+30	5.8165E+29	5.47E+28
13	1.384E+34	1.09871E+33	8.7333E+31	6.95E+30
17	7.4E+37	4.49319E+36	2.7312E+35	1.66E+34
19	2.6E+39	1.41254E+38	7.6822E+36	4.18E+35
23	1.175E+42	5.27474E+40	2.3698E+39	1.07E+38
29	1.957E+45	6.96588E+43	2.4821E+42	8.85E+40
31	1.654E+46	5.50619E+44	1.8354E+43	6.12E+41
37	4.757E+48	1.32706E+47	3.7062E+45	1.04E+44
41	1.27E+50	3.19844E+48	8.0611E+46	2.03E+45
43	5.832E+50	1.4001E+49	3.3646E+47	8.09E+45
47	1.005E+52	2.20634E+50	4.8508E+48	1.07E+47
53	4.695E+53	9.14514E+51	1.783E+50	3.48E+48
59	1.452E+55	2.54127E+53	4.4508E+51	7.8E+49
61	4.221E+55	7.14271E+53	1.21E+52	2.05E+50

67	8.497E+56	1.30905E+55	2.0189E+53	3.12E+51
71	5.434E+57	7.90036E+55	1.1498E+54	1.68E+52
73	1.322E+58	1.86916E+56	2.6458E+54	3.75E+52
79	1.655E+59	2.16311E+57	2.8294E+55	3.7E+53
83	8.042E+59	1.00014E+58	1.2451E+56	1.55E+54
89	7.505E+60	8.70419E+58	1.0106E+57	1.17E+55
97	1.179E+62	1.25476E+60	1.3367E+58	1.43E+56
101	4.297E+62	4.39138E+60	4.4928E+58	4.6E+56

На рисунку 5.7 представлена залежність кількості незвідних поліномів в логарифмічній шкалі з основою 10 для різних степенів n=32, 31, 30, 29 і значень параметра p=2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,101.

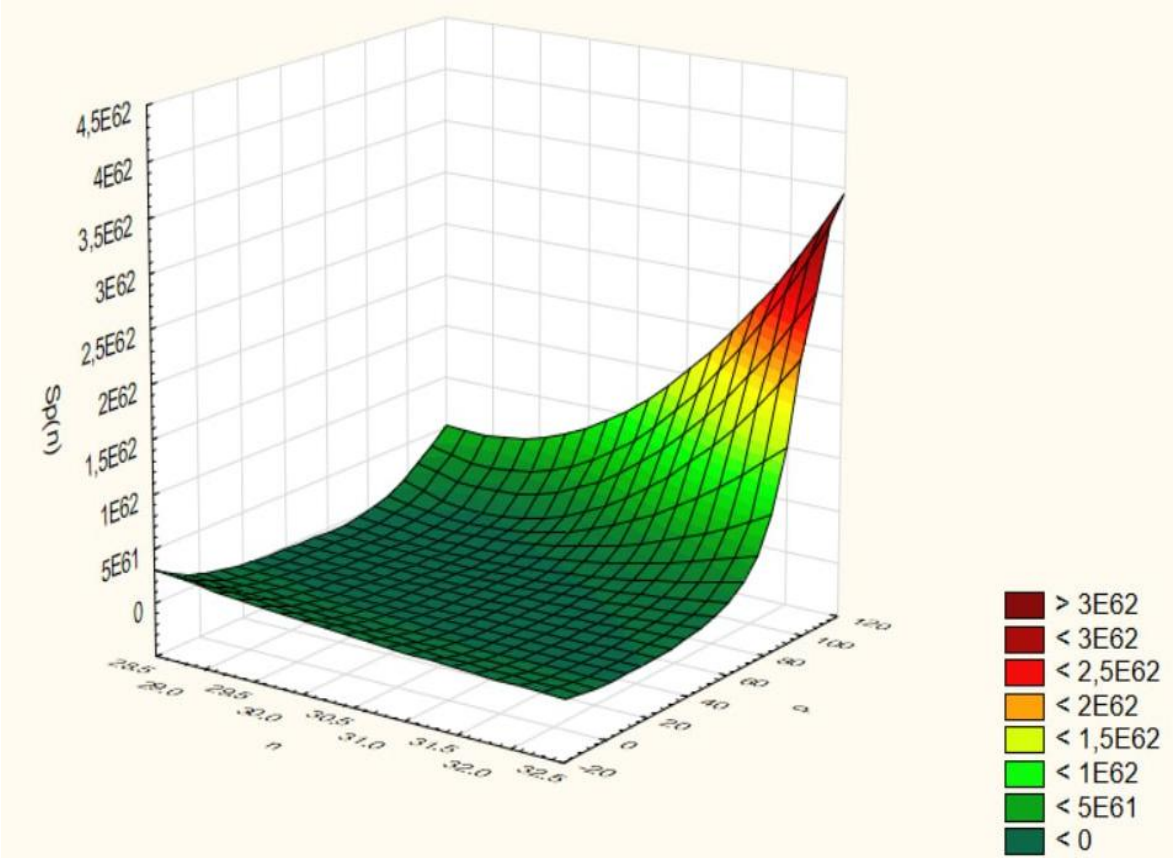


Рисунок 5.7 – Кількість незвідних поліномів в залежності від порядку полінома n і параметру p.

В загальному випадку стійкість запропонованої криптосистеми з l модулів та k рівнів буде визначатися як добуток часу повного перебору всіх

незвідних поліномів та складності виконання з кожним обчислень згідно такої формули:

$$O(n,k,l) = \prod_{k=1}^{n-k+1} C_{S_p(n-k+1)}^{l^k} \cdot (n-k+1)^2 log_2 l^k =$$

$$= \prod_{k=1}^{n-k+1} \frac{\left(S_p(n-k+1)\right)! \cdot (n-k+1)^2 \cdot (k)\, log_2 l}{\left(S_p(n-k+1)-l^k\right)! \cdot l^k!}. \qquad (4.44)$$

$$\prod_{k=1}^{n-k+1} \frac{S_p(n-k+1) \cdot S_p\big((n-k+1)-1\big) \cdot S_p\big((n-k+1)-2\big) \cdot \ldots S_p((n-k+1)-l^k+1)}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot l^k}$$

$$\cdot (n-k+1)^2 \cdot k\, log_2 l$$

На рисунку 5.8 представлено залежність стійкості до криптоаналітичних атак від кількості ієрархічних рівнів k та ступеня полінома n при l=3.

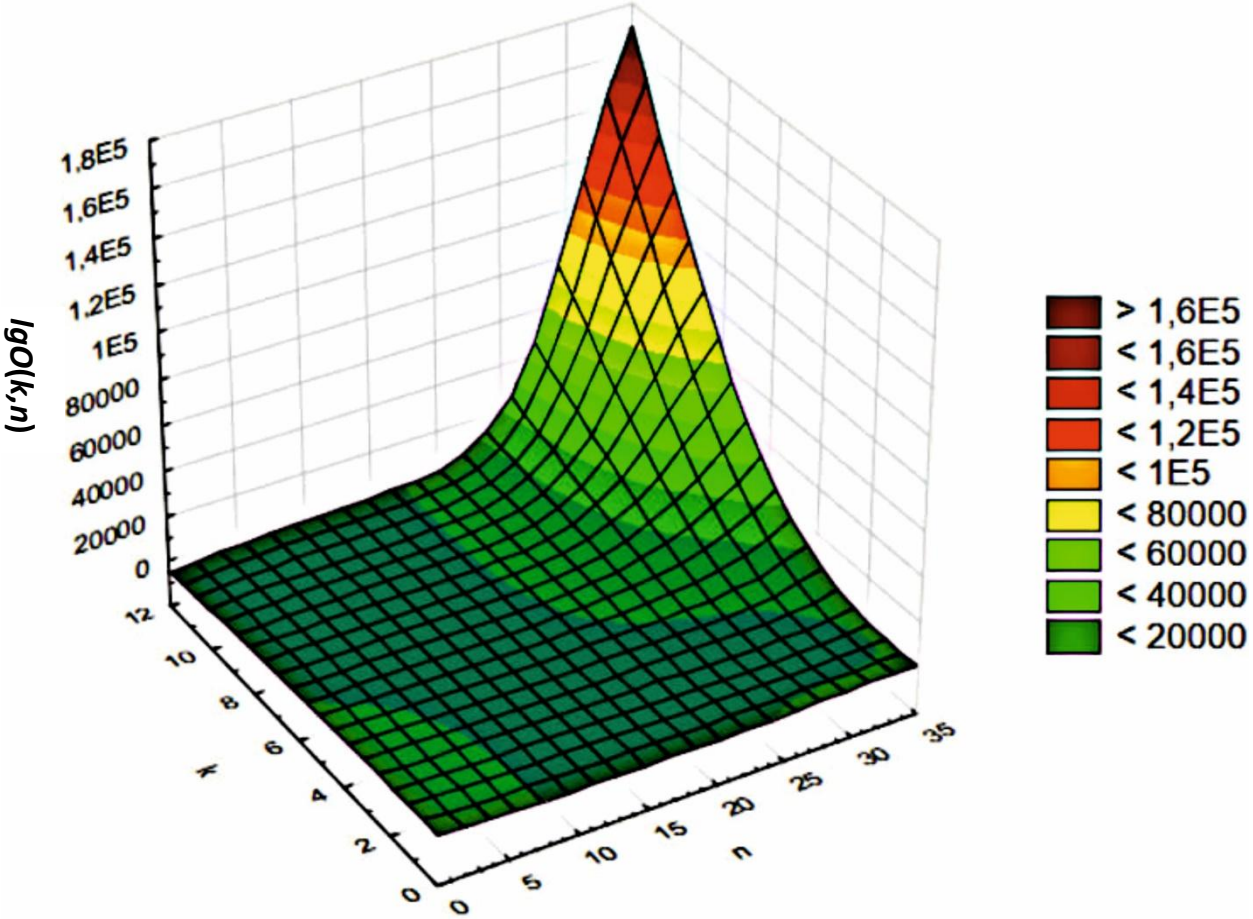


Рисунок 5.8 – Стійкість криптосистеми в ІПСЗК в залежності від порядку полінома n і кількості модулів l

На основі аналізу поверхні можна зробити висновок, що стійкість криптосистеми в ІПСЗК значною мірою залежить від вибору параметрів *n* і *l*. Збільшення порядку полінома та кількості модулів сприяє зростанню криптографічної безпеки, і для забезпечення максимальної захищеності системи слід обирати великі значення обох параметрів.

Наприклад, стійкість криптосистеми для $l = 2,3$, *k*=3 та $n = 32$ у полі Галуа GF(2) визначається як сумарне значення на 3 рівнях.

На першому рівні отримуються наступні значення:

$$O(32,2) = C^2_{134215680} \cdot 32^2\, log\, 2 = \frac{(134215680)!}{(134215678)! \cdot 2!} 32^2\, log\, 2 \approx 1{,}46183 \cdot 10^{19}$$

$$O(32,3) = C^3_{134215680} \cdot 32^2\, log\, 2 = \frac{(134215680)!}{(134215677)! \cdot 3!} 32^2\, log\, 2 \approx 6{,}54 \cdot 10^{26}$$

На другому рівні:

$$O(31,2) = C^4_{69273666} \cdot 31^2\, 2log\, 4 = \frac{(69273666)!}{(69273662)! \cdot 4!} 31^2\, 2log\, 4$$

$$\approx 1{,}82499 \cdot 10^{32}$$

$$O(31,3) = C^9_{69273666} \cdot 31^2\, 2log\, 9 = \frac{(69273666)!}{(69273658)! \cdot 9!} 31^2 2\, log\, 9$$

$$\approx 1{,}92553 \cdot 10^{67}$$

На третьому рівні відповідні складності будуть:

$$O(30,2) = C^8_{18512790} \cdot 30^2\, 3log\, 8 = \frac{(18512790)!}{(18512782)! \cdot 8!} 30^2 3\, log\, 8$$

$$\approx 9{,}43674 \cdot 10^{55}$$

$$O(30,3) = C^{27}_{18512790} \cdot 30^2 3\, log\, 27 = \frac{(18512790)!}{(18512763)! \cdot 27!} 30^2\, 3log\, 27$$

$$\approx 4{,}2201 \cdot 10^{170}$$

Загальна стійкість запропонованої поліноміальної ієрархічної симетричної системи шифрування буде добуток відповідних складностей на трьох рівнях: $O(32,2) \approx 9{,}43674 \cdot 10^{107}$, $O(32,3) \approx 1{,}251393 \cdot 10^{264}$. Слід

відзначити, що збільшення кількості модулів на однин при вхідних поліномах степеня n = 32 призводить до суттєвого зростання стійкості запропонованого симетричного ієрархічного поліноміального криптографічного алгоритму в $4{,}97066 \cdot 10^{156}$ разів і на трьох рівнях при вхідних параметрах n = 32, p=2, $l = 3$ спостерігається підвищення криптостійкості у 3,34 рази в порівнянні з AES-256.

5.5 Методологія криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої та поліноміальної систем залишкових класів

Створення методології криптографічного захисту інформації на основі розроблених методів дозволить побудувати єдину стратегію використання цілочисельного, модифікованого досконалого та поліноміального СЗК, а також векторно-модульних методів модулярного множення та експоненціювання для зменшення часової складності, підвищення стійкості алгоритмів, спеціалізованого програмного забезпечення в симетричних та асиметричних криптосистемах, ефективно будувати швидкодіючі криптографічні системи захисту інформації.

Представлена в [329] методологія криптографічного захисту інформаційних потоків на основі використання симетричних та асиметричних криптосистем в цілочисельній, модифікованій досконалій та поліноміальній СЗК (її структурно-аналітичне відображення представлено на рис. 5.8) складається з восьми етапів: 1) процес формування множини блоків відкритого тексту (N, $N(x)$) для цілочисельних і поліноміальних криптографічних систем; 2) Встановлення вимог щодо основних параметрів цілочисельних та поліноміальних симетричних та асиметричних криптографічних систем та захищеності інформації; 3) вибір запропонованих цілочисельних та поліноміальних криптосистем; 4) створення множини базових операцій; 5) формування набору методів виконання операцій; 6) вибір

цілочисельної та поліноміальної форм СЗК; 7) програмна реалізація цілочисельних та поліноміальних симетричних та асиметричних криптосистем. Детальний опис запропонованих етапів та взаємозвязок між ними представлено нижче [329].

Етап 1. Процес формування множини блоків відкритого тексту (N, N(x)) для цілочисельних і поліноміальних криптографічних систем в СЗК. На початковому етапі користувач повинен подати у цифровому вигляді набори блоків відкритого тексту для цілочисельного шифрування $N = \left\{\bigcup_{i=1}^{M_1} N_i\right\} = \left\{N_1, N_2, \ldots, N_{M_1}\right\}$, і у вигляді поліномів для поліноміального $N(x) = \left\{\bigcup_{i=1}^{M_2} N_i\,(\text{x})\right\} = \left\{N_1(\text{x}), N_2(\text{x}), \ldots, N_{M_1}(\text{x})\right\}$ (M_1, M_2 – кількість блоків відкритого тексту). У основних асиметричних криптографічних системах, таких як RSA, Рабіна та Ель-Гамаль, значення відкритого тексту не повинні перевищувати відповідний параметр відкритого ключа.

Величина M_1 визначається шляхом ділення числового значення відкритого тексту на цей параметр. Для цілочисельних та поліноміальних криптографічних систем в СЗК блоки повідомлень повинні задовольняти нерівності $N<P$ і $\deg N(x)<\deg P(x)$ відповідно, де $P = \prod_{i=1}^{s} p_i$, $P(x) = \prod_{i=1}^{l} p_i(x)$, p_i, $p_i(x)$ - системи попарно взаємно простих модулів та модулів-поліномів, *s, l* – кількість модулів.

При передачі блоків у зашифрованому вигляді по відкритих каналах зв'язку виникають можливі загрози. В результаті формується визначена користувачем множина загроз $МЗ = \left\{\bigcup_{i=1}^{z_1} МЗ_i\right\} = \left\{МЗ_1, МЗ_2, \ldots, МЗ_{z_2}\right\}$, де z_1 – їх кількість.

Етап 2. Вибір модулів та ключів для цілочисельного та поліноміального шифрування в СЗК. На даному етапі відбувається вибір відкритих та таємних ключів для цілочисельних та поліноміальних: симетричних та асиметричних криптоалгоритмів у СЗК $p_i, НСД(p_i, p_{i-1}) = 1, P = \prod_{i=1}^{k} p_i$, k – кількість модулів (ключів) для цілочисельної, $p_i(\text{x}), НСД\left(p_i(\text{x}), p_{i-1}(\text{x})\right) = 1, P(x) = \prod_{i=1}^{k} p_i\,(\text{x})$), двоключового поліноміального симетричного шифрування в СЗК

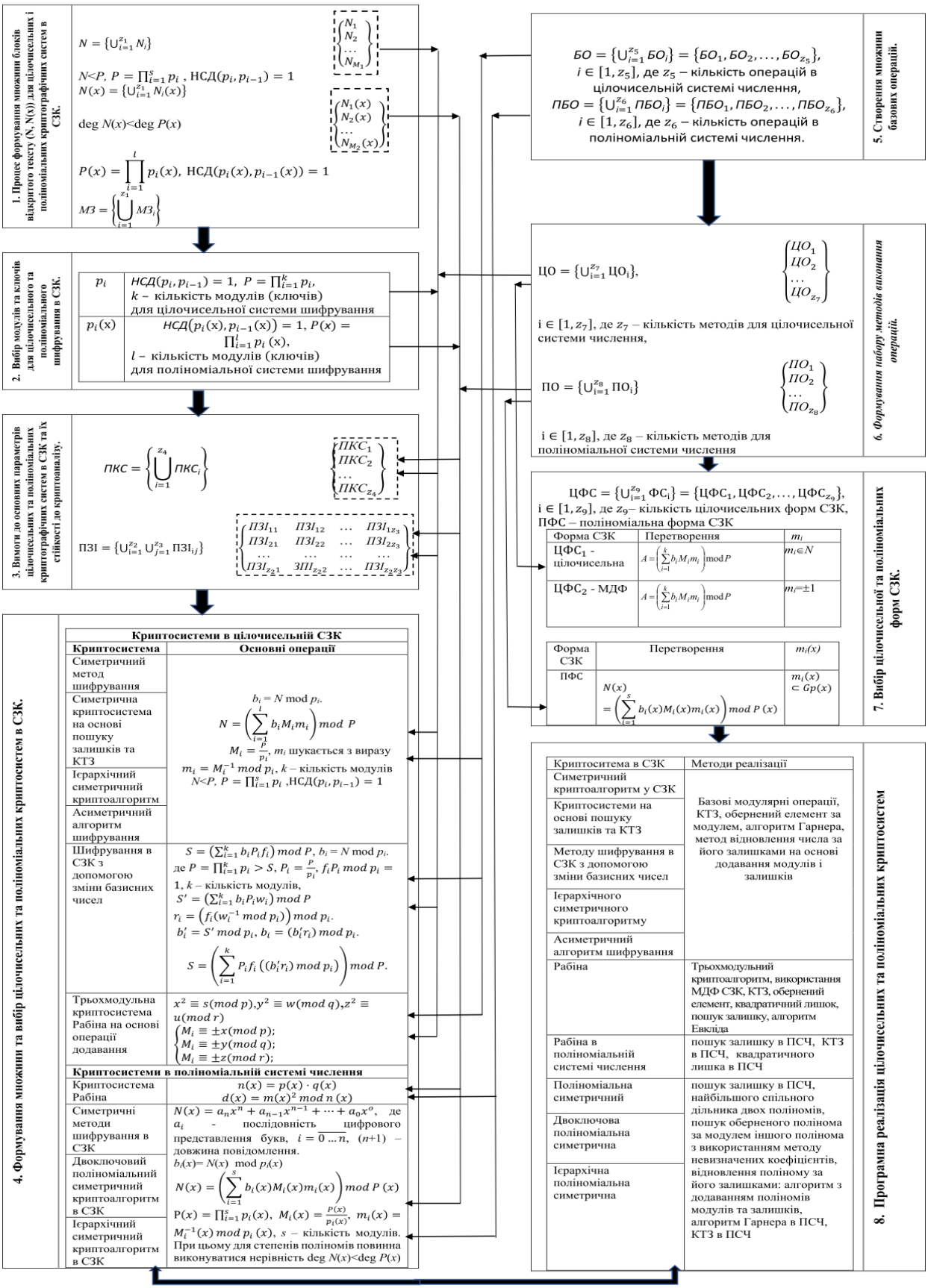


Рисунок 5.8 Схемо-аналітичне представлення методології криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої та поліноміальної систем залишкових класів

(при генерації поліноміальних ключів абоненти вибирають відомі тільки їм обом системи модулів $m_i(x)$ та відповідні їм поліноми $s_i(x)$, для яких виконуються такі умови: $1<deg\, s_i\,(x)<deg\, m_i\,(x)$ та НСД($s_i(x), m_i(x)$)=1), ієрархічних цілочисельних в СЗК, поліноміальних алгоритмів шифрування в ІПСЗК (основна система модулів (ключів) першого рівня $p_1(x)$, $p_2(x)$, …, $p_l(x)$ забезпечує виконання операцій у діапазоні [0, $P_1(x)$)), де $P_1(x) = p_1(x)p_2(x)\dots p_l(x) = \prod_{i=1}^{l} p_i\,(x)$. Причому, максимально можливий діапазон обчислень дорівнює $(p_l(x)\text{-}1)^2$. На наступному ієрархічному рівні головна система модулів представляється в новій системі з відповідними основами та діапазонами $q_{11}(x)$, $q_{12}(x)$, …, $q_{1l}(x)$, $q_{21}(x)$, $q_{22}(x)$, …, $q_{2l}(x)$, …, $q_{l1}(x)$, $q_{l2}(x)$, …, $q_{ll}(x)$. Ця процедура триває до останнього (k-го) рівня).

Етап 3. Встановлення вимог щодо основних параметрів цілочисельних та поліноміальних криптографічних систем в СЗК та їх стійкості до криптоаналізу. На основі сформованих множин загроз та блоків відкритого тексту відбувається формування вимог до кількісних показників стійкості криптографічного алгоритму $ПЗІ = \left\{\bigcup_{i=1}^{z_2}\bigcup_{j=1}^{z_3} ПЗІ_{ij}\right\}$, які записуються у вигляді матриці, та основних параметрів криптографічних систем $ПКС = \left\{\bigcup_{i=1}^{z_4} ПКС_i\right\} = \{ПКС_1, ПКС_2, \dots, ПКС_{z_4}\}$, $i \in [1, z_4]$, де z_4 – кількість вимог.

Для кожного фрагмента відкритого тексту $N = \left\{\bigcup_{i=1}^{M_1} N_i\right\} = \{N_1, N_2, \dots, N_{M_1}\}$, $N(x) = \left\{\bigcup_{i=1}^{M_2} N_i\,(\text{x})\right\} = \{N_1(\text{x}), N_2(\text{x}), \dots, N_{M_2}(\text{x})\}$ та кожної ймовірної загрози $МЗ_i, i \in [1, z_2]$ встановлюється набір показників захищеності $ПЗІ = \left\{\bigcup_{i=1}^{z_2}\bigcup_{j=1}^{z_3} ПЗІ_{ij}\right\}$, де z_2 –позначає кількість показників для певної загрози та множина вимог до параметрів криптосистеми $ПКС_i, i \in [1, z_3]$, де z_3 – кількість вимог.

Етап 4. Формування множини та вибір цілочисельних та поліноміальних криптосистем в СЗК. На четвертому етапі відбувається вибір розроблених цілочисельних та поліноміальних симетричних та асиметричних криптосистем в залежності від поставлених завдань щодо рівня захисту:

симетричний метод шифрування у СЗК, симетрична криптосистема на основі пошуку залишків та КТЗ, методу шифрування в СЗК з допомогою зміни базисних чисел, ієрархічний симетричний криптоалгоритм на основі СЗК, асиметричний алгоритм шифрування у СЗК, трьохмодульної криптосистеми Рабіна на основі операції додавання, асиметричного алгоритму шифрування Ель-Гамаля з використанням системи залишкових класів та векторно-модульного алгоритму модулярного експоненціювання. Аналогічно до цілочисельних застосувань, поліноми можна успішно використовувати в якості відкритого і зашифрованого тексту, відкритого і таємного ключів при використанні в криптографічних алгоритмах. Серед розроблених поліноміальних криптоалгоритмів на даному етапі можна вибрати: криптосистему Рабіна в поліноміальній системі числення, криптографічні поліноміальні симетричні методи шифрування в поліноміальній системі залишкових класів, двоключовий поліноміальний симетричний криптоалгоритм в СЗК, ієрархічний поліноміальний симетричний криптоалгоритм в поліноміальній СЗК.

Для генерування ключів цілочисельних симетричних та асиметричних криптосистем, процесу шифрування та розшифровування використовуються такі операції: пошук залишку, найбільшого спільного дільника, використання розширеного алгоритму Евкліда, оберненого елемента за модулем, відновлення десяткового числа за його залишками (алгоритм з додаванням модулів та залишків, алгоритм Гарнера, китайська теорема про залишки), квадратичний лишок, модулярне множення та модулярне експоненціювання. Для забезпечення формування симетричних та асиметричних криптосистем поліноміальній системі числення (ПСЧ) використовуються операції: пошук залишку в ПСЧ, найбільшого спільного дільника двох поліномів, пошук оберненого полінома за модулем іншого полінома на основі методу невизначених коефіцієнтів, відновлення поліному за його залишками (алгоритм з додаванням поліномів модулів та залишків, алгоритм Гарнера та

КТЗ в ПСЧ), пошук кореня квадратного полінома в ПСЧ (алгоритм додавання модуля до тих пір, поки не буде повний квадрат).

Етап 5. Створення множини базових операцій. Для забезпечення процесу шифрування та розшифрування в цілочисельній та поліноміальній системах числення на п'ятому етапі формується відповідна множина базових операцій, які використовуються у симетричих та асиметричних криптоалгоритмах $БО = \{\bigcup_{i=1}^{z_5} БО_i\} = \{БО_1, БО_2, \dots, БО_{z_5}\}$, $i \in [1, z_5]$, де z_5 – кількість операцій в цілочисельній системі числення, $ПБО = \{\bigcup_{i=1}^{z_6} ПБО_i\} = \{ПБО_1, ПБО_2, \dots, ПБО_{z_6}\}$, $i \in [1, z_6]$, де z_6 – кількість операцій в поліноміальній системі числення.

Етап 6. Формування набору методів виконання операцій. На шостому етапі здійснюється формування набору методів для реалізації операцій в цілочисельній та поліноміальній системі числення, визначених на попередньому етапі: $ЦО = \{\bigcup_{i=1}^{z_7} ЦО_i\} = \{ЦО_1, ЦО_2, \dots, ЦО_{z_7}\}$, $i \in [1, z_7]$, де z_7 – кількість методів для цілочисельної системи числення, $ПО = \{\bigcup_{i=1}^{z_8} ПО_i\} = \{ПО_1, ПО_2, \dots, ПО_{z_8}\}$, $i \in [1, z_8]$, де z_8 – кількість методів для поліноміальної системи числення. Зокрема, для операцій в цілочисельній системі числення: визначення залишку, модульного множення та піднесення до степеня можуть бути застосовані векторно-модульний підхід, тоді як інші операції реалізуються через додавання чи множення модулів.

Для поліноміальних операцій: пошук залишку в ПСЧ, найбільшого спільного дільника двох поліномів (здійснюються на основі операції ділення поліномів), пошук оберненого полінома за модулем іншого полінома (з використанням методу невизначених коефіцієнтів), відновлення поліному за його залишками (алгоритм з додаванням поліномів модулів та залишків, алгоритм Гарнера в ПСЧ, китайська теорема про залишки в ПСЧ), пошук кореня квадратного полінома в ПСЧ (на основі операції дадавання модуля).

Етап 7. Вибір цілочисельної та поліноміальних форм СЗК. Для оптимізації обчислень за рахунок розпаралелення і зменшення розмірності

операндів на четвертому етапі більшість операцій можна виконувати в СЗК, множина форм якої формується на сьомому етапі: $ЦФС = \left\{\bigcup_{i=1}^{z_9} ФС_i\right\} = \left\{ЦФС_1, ЦФС_2, \ldots, ЦФС_{z_9}\right\}$, $i \in [1, z_9]$, де z_9 – кількість цілочисельних форм СЗК, *ПФС* – поліноміальна форма СЗК . Окрім традиційної цілочисельної форми, особливий інтерес для використання в асиметричних криптографічних системах становлять ДФ та МДФ СЗК, які є фундаментом для створення нових стандартів безпеки, які відповідають сучасним вимогам інформаційної безпеки та захисту даних.

Етап 8. Програмна реалізація цілочисельних та поліноміальних криптосистем. Восьмий етап передбачає програмну реалізацію запропонованих цілочисельних та поліноміальних симетричних та асиметричних криптосистем з використанням базових модулярних операцій (етап 4) на основі вибору методу їх виконання (етап 6) або з використанням відповідних форм СЗК (етап 7): методу шифрування у СЗК, на основі пошуку залишків та КТЗ, методу шифрування в СЗК з допомогою зміни базисних чисел, ієрархічного симетричного криптоалгоритму на основі СЗК, асиметричний алгоритм шифрування у СЗК, трьохмодульної криптосистеми Рабіна, асиметричного алгоритму шифрування Ель-Гамаля з використанням СЗК та векторно-модульного алгоритму модулярного експоненціювання, та для реалізації розроблених поліноміальних криптоалгоритмів: криптосистеми Рабіна в поліноміальній системі числення, криптографічного поліноміального симетричного методу шифрування в СЗК, двоключового поліноміального симетричного криптоалгоритму в СЗК, симетричного криптоалгоритму в ІПСЗК.

Ця методологія забезпечує комплексний підхід до розробки, реалізації та оптимізації запропонованих симетричних та асиметричних цілочисельних, поліноміальних криптосистем на основі використання СЗК, векторно-модульних методів пошуку залишків, модулярного множення та

експоненціювання, методу невизначених коефіцієнтів для пошуку оберненого поліному в поліноміальній системі числення, методів відновлення поліному за його залишками на основі додавання добутку модулів, що дає змогу досягти високого рівня захисту інформації при мінімальних витратах на обчислення.

Висновки до розділу 5

1. Вперше запропоновано симетричний криптоалгоритм, який базується на ІСЗК, особливістю якого є його ступінчаста структура, яка дозволяє послідовно зменшувати розрядність модулів на кожному рівні та забезпечувати необхідний рівень захищеності в залежності від кількості рівнів, модулів та їх розрядностей.

2. Проведені експериментальні дослідження симетричного криптоалгоритму в ІСЗК продемонстрували, що розроблений алгоритм має високу стійкість до атак завдяки використанню великих простих чисел у системі модулів та багаторівневій структурі шифрування. Встановлено, що криптографічна стійкість збільшується зі зростанням кількості модулів, їх розрядності та рівнів ієрархії.

3. Проведено порівняльний аналіз стійкості запропонованого алгоритму та алгоритму AES-256. Вказано, при яких значеннях вхідних параметрів (розрядності модулів, кількості модулів та рівнів ієрархії) алгоритм на основі СЗК демонструє стійкість, порівнянну з AES-256, забезпечуючи при цьому більшу гнучкість налаштувань та ефективність обчислень. Запропонована методика дозволяє змінювати розрядність модулів, кількість рівнів ієрархії та інші параметри для досягнення необхідного рівня захисту.

4. Вперше запропоновано симетричний криптоалгоритм, заснований на поліноміальній ієрархічній системі залишкових класів, що використовує поліном із коефіцієнтами, які відповідають кодованим символам, як відкритий текст. У якості таємних ключів застосовано систему незвідних поліномів, а

шифрування реалізується шляхом обчислення залишків вхідного полінома за відповідними модулями.

5. Розроблений криптоалгоритм в ІПСЗК відзначається ступінчастою структурою, яка забезпечує поступове зменшення порядку поліномів-модулів на кожному рівні, що, у свою чергу, дозволяє знизити обчислювальні витрати, дає можливість підвищити ефективність криптографічних операцій і, в свою чергу, адаптивно налаштовувати параметри алгоритму залежно від забезпечення необхідного рівня захисту.

6. Вперше розроблено методологію криптографічного захисту інформації в системі залишкових класів, яка забезпечує комплексний підхід до розробки, реалізації та оптимізації запропонованих симетричних та асиметричних цілочисельних, криптосистем на основі використання ПСЗК, векторно-модульних методів пошуку залишків, модулярного множення та експоненціювання, методу невизначених коефіцієнтів для пошуку оберненого поліному в поліноміальній системі числення, методів відновлення поліному за його залишками на основі додавання добутку модулів, що дає змогу досягти високого рівня захисту інформації при мінімальних витратах на обчислення.

Основні результати п’ятого розділу відображені у роботах [313, 321-324, 329].

РОЗДІЛ 6. ПРОГРАМНА РЕАЛІЗАЦІЯ КРИПТОАЛГОРИТМІВ В СИСТЕМІ ЗАЛИШКОВИХ КЛАСІВ

6.1 Програмна реалізація симетричних та асиметричних криптоалгоритмів в СЗК

У сучасних інформаційно-комунікаційних системах важливе місце займає реалізація криптографічних алгоритмів, які забезпечують захист даних від несанкціонованого доступу. Одним із підходів до оптимізації криптоалгоритмів є використання СЗК для підвищення ефективність обчислень та забезпечення високої стійкості. У рамках даного дослідження було розроблено програмну систему, яка реалізує криптоалгоритми в СЗК і може бути інтегрована в прикладні інформаційно-комунікаційні підсистеми.

На рисунку 6.1. представлено діаграму компонентів програмної системи реалізації криптоалгоритмів в системі залишкових класів із врахування її практичної імплементації в прикладні інформаційно-комунікаційні підсистеми.

Діаграма компонентів програмної системи реалізації криптоалгоритмів у СЗК відображає багаторівневу архітектуру, у якій прикладні інформаційні системи взаємодіють із криптографічними модулями через інформаційно-комунікаційну інфраструктуру. У верхній частині показано прикладні підсистеми система управління, підсистема аналізу та спеціалізована підсистема захисту в енергетичних системах. Вони здійснюють виклики криптографічних функцій і передають між собою дані, використовуючи модуль комунікації, який забезпечує обмін повідомленнями та слугує посередником між прикладним рівнем і криптографічною платформою.

Нижче розміщено основні криптографічні компоненти, реалізовані на основі СЗК. До них належать метод відновлення полінома та метод пошуку оберненого полінома, які забезпечують математичну підтримку поліноміальних ієрархічних обчислень. Поруч розташовані криптографічні поліноміальні симетричні методи, симетричні ієрархічні криптоалгоритми на

основі СЗК, а також ієрархічний поліноміальний симетричний криптоалгоритм ці компоненти відповідають за побудову симетричних схем шифрування, що використовують переваги розпаралелювання та модульного поділу даних у СЗК. Окремим блоком представлено асиметричні методи в СЗК, що забезпечують виконання криптографічних операцій, які ґрунтуються на модулярному експоненціюванні та векторно-модульних перетвореннях.

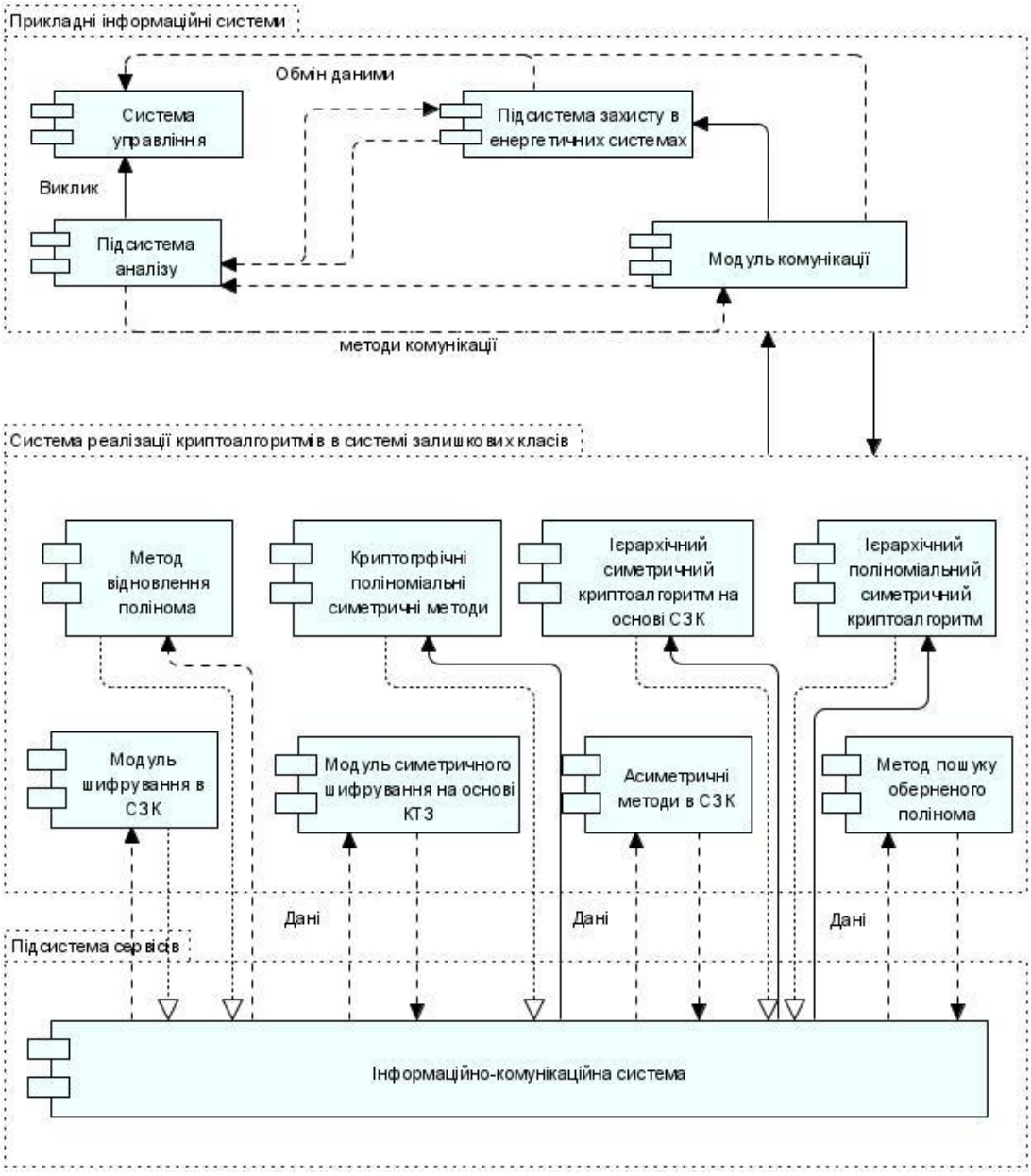


Рисунок 6.1 – Діаграма компонентів програмної системи реалізації криптоалгоритмів в системі залишкових класів

Також присутній модуль симетричного шифрування на основі КТЗ, що дозволяє реалізувати високошвидкісні криптографічні процедури завдяки незалежності підканалів СЗК.

Усі ці алгоритмічні методи об’єднані з модульними блоками шифрування, які забезпечують безпосереднє виконання операцій над даними. Потоки інформації від криптомодулів передаються в інформаційно-комунікаційну систему, що розташована внизу діаграми та слугує центральною інфраструктурною підсистемою. Вона забезпечує транспортування даних, їх маршрутизацію між модулями та повернення результатів назад до прикладних систем.

Таким чином, діаграма демонструє цілісну архітектуру, у якій прикладні інформаційні сервіси взаємодіють із багатокомпонентною криптографічною платформою через стандартизовані методи комунікації. Це дозволяє масштабувати систему, розширювати набір криптографічних алгоритмів і забезпечувати захищену обробку даних у різних доменах застосування.

Програма реалізована за допомогою мови програмування Python, з його переваг слід відзначити наступні: зрозумілий синтаксис, підтримка більшості сучасних парадигм програмування, кросплатформеність, підвищена продуктивність, інтерпретована мова, динамічна типизація, розрядність чисел обмежуються тільки розміром оперативної пам'яті.

Програма використовує додаткові модулі (рис. 6.2), а саме:

1. pycrypto, для порівняння швидкодії з AES;
2. tkinter, для побудови GUI інтерфейсу;
3. xlsxwriter, для збереження результатів порівняння швидкодії в таблиці Microsoft Office Excel.

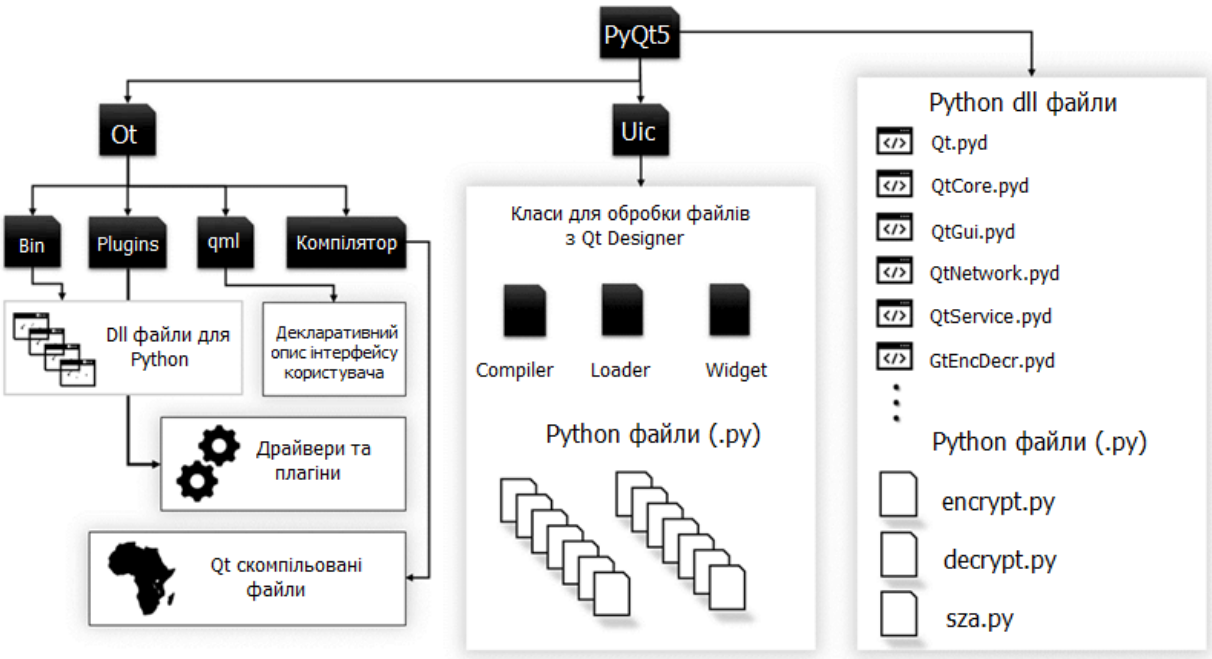


Рисунок 6.2 – Структура засобів програмної реалізації з використанням PyQt

У процесі реалізації програмної системи були створені два основних інтерфейси для взаємодії з користувачем: графічний (GUI) та командний (CLI). Такий підхід забезпечує гнучкість у використанні програмного забезпечення, дозволяючи користувачам обирати найбільш зручний спосіб взаємодії залежно від конкретних завдань та умов експлуатації.

З точки зору криптографічних алгоритмів, програмна система підтримує як симетричні, так і асиметричні методи шифрування. Оскільки їх реалізація має багато спільних аспектів, для пояснення принципів роботи системи використовується узагальнений опис, який охоплює обидва підходи. Це дозволяє більш ефективно продемонструвати загальну архітектуру системи та ключові етапи обробки даних.

Структурна організація програмного коду представлена у вигляді об'єктно-орієнтованої моделі, де основні компоненти реалізовані у формі класів. Взаємозв’язки між цими класами, їхні функціональні можливості та принципи взаємодії між собою зображені на діаграмі основних реалізованих

класів (рис. 6.3). Така структурна схема наочно демонструє логіку побудови програмної системи та сприяє подальшому вдосконаленню її функціональності.

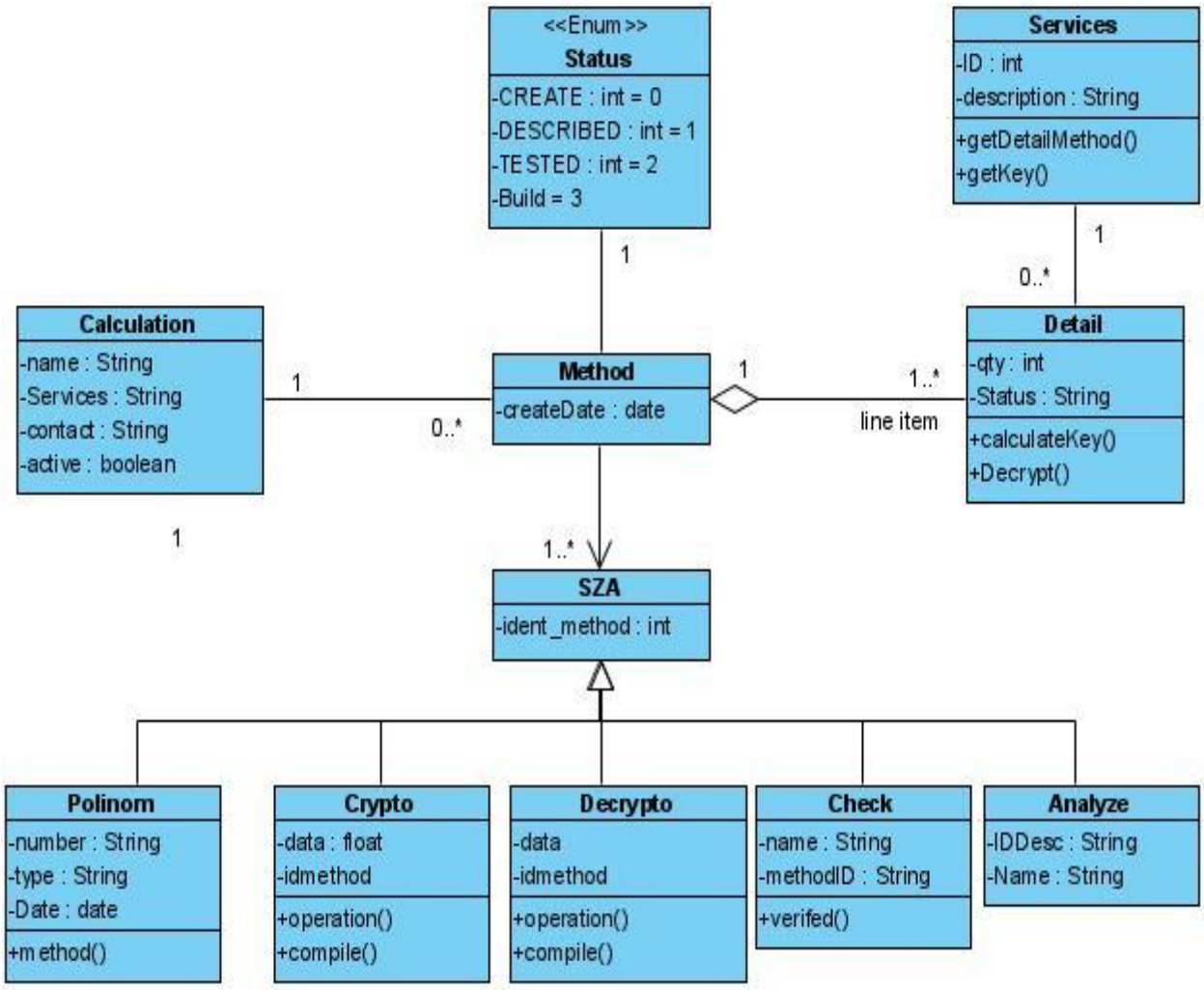


Рисунок 6.3 – Діаграма класів системи

Діаграма комунікації системи демонструє взаємодію основних компонентів під час виконання криптографічних операцій (рисунок 6.4). Вона складається з кількох рівнів: системного, представлення, управління та моделі.

Системний рівень представлений зовнішньою інформаційною системою, яка ініціює запити на виконання криптографічних операцій. Взаємодія розпочинається з ініціалізації методу, що відбувається через інтерфейс користувача (System UI), який приймає запит та передає його до

керуючого модуля (Control). Контролер, у свою чергу, викликає відповідні методи криптографічного алгоритму, що відповідає за виконання операцій шифрування або розшифрування.

Для успішного виконання операцій відбувається збір та підготовка необхідних даних. Інтерфейс користувача передає попередню інформацію до контролера, де вона перевіряється на коректність та узгоджується з вимогами системи. Контролер здійснює обробку вхідних даних та передає їх до моделі, яка включає кілька модулів: профіль користувача, документацію, вимоги, базу даних, математичний опис алгоритмів та фінальне криптографічне рішення. На цьому етапі інформація перевіряється, компілюється та формується у вигляді відповідного математичного представлення.

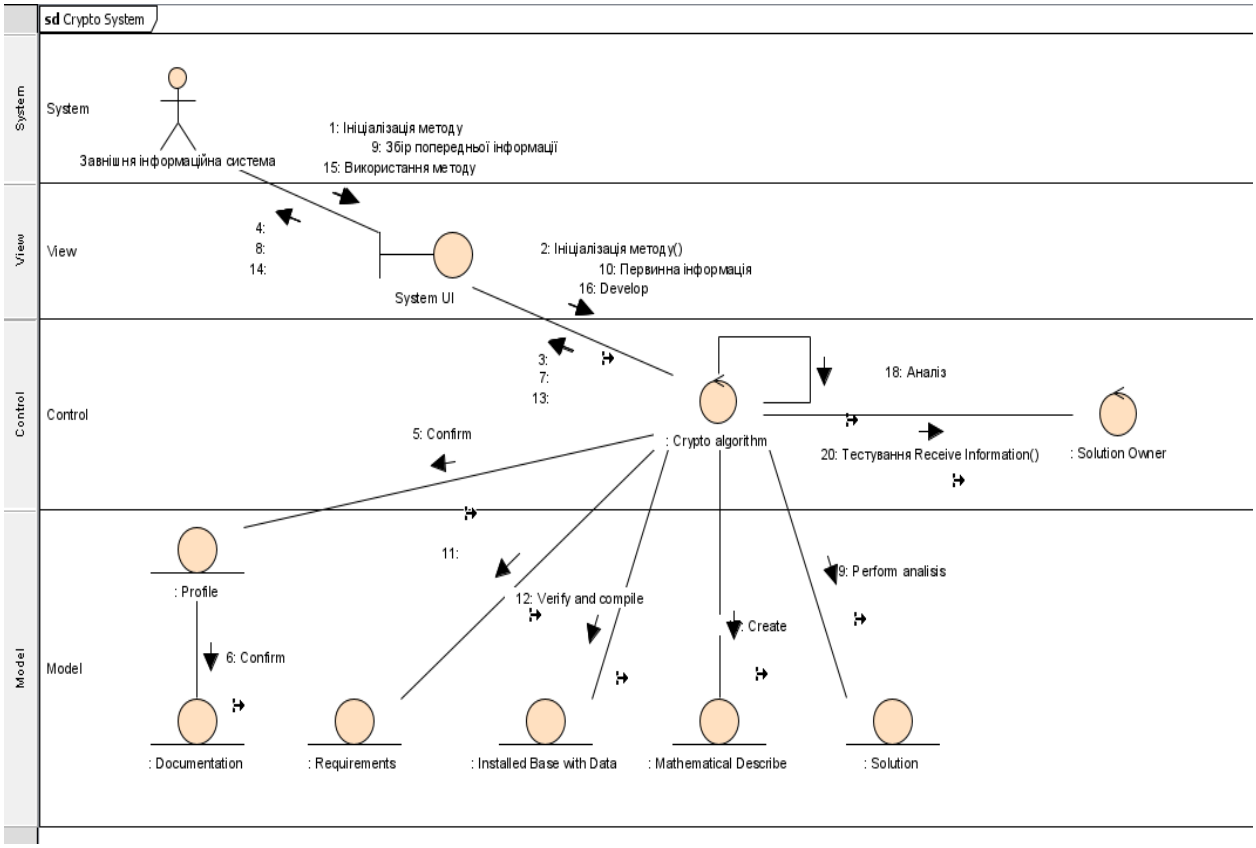


Рисунок 6.4 – Діаграма комунікації системи

Після цього здійснюється безпосередня криптографічна обробка, де контролер координує взаємодію між математичним модулем та модулем

рішення. Відбувається розрахунок криптографічного алгоритму, створення математичної моделі та її верифікація. Система аналізує отримані результати, проводить їх тестування та перевірку правильності виконання.

На завершальному етапі остаточне криптографічне рішення передається зовнішній інформаційній системі або користувачеві через інтерфейс для подальшого використання. Завдяки чіткій структурі взаємодії між компонентами, система забезпечує модульність, ефективну обробку запитів та високу продуктивність криптографічних алгоритмів.

Етапи роботи функції шифрування/розшифрування [295]:

1) Знаходження P, за формулою:

$$P = \prod_{i=1}^{v} p_i > S;$$

2.1) Повернення результату, за формулою (шифрування):

$$S' = (\sum_{i=1}^{v} b_i P_i w_i)$$

2.2) Повернення результату, за формулою (розшифрування):

$$S = \left(\sum_{i=1}^{v} P_i f_i ((b_i' f_i w_i^{-1}) mod\ p_i) \right) mod\ P$$

Для роботи з асиметричним та симетричним криптоалгоритмами в СЗК є два модуля resolution.py й resolution2.py, здійснюється їх запуск відповідно. Крім того, кожен з них містить функції шифрування (encrypt) й розшифруванням (decrypt). Процес роботи зазначених модулів описано нижче:

1) encrypt – функція приймає в якості параметрів два списки (масив) цілих чисел w (в resolution2.py k) й p, повідомлення в цифровій формі для шифрування num. Слід зазначити, що довільні елементи списків w_i та p_i повинні бути взаємно прості і задовольняти умовам:

$$1 \leq |w_i| \leq p_i \text{ та НСД}(|w_i|, p_i) = 1$$

Після виконання шифрування функція поверне шифротекст S' в вигляді цілого числа.

2) decrypt – функція, аналогічна по параметрам до encrypt, але параметр *num* має бути шифртекстом, а функція, відповідно, повертає розшифрований текст *S*.

Нижче наведено програмний код на мові Python реалізації зазначених модулів:

```
def encrypt(w: list, p: list, num: int) -> int:
    prod, result = 1, 0
    for i in range(len(p)): prod *= p[i]
    for i in range(len(p)):
        pp = prod // p[i]
        result += pp * (num % p[i]) * w[i]
    return result % prod
def decrypt(w: list, p: list, num: int) -> int:
    prod, result = 1, 0
    for i in range(len(p)): prod *= p[i]
    for i in range(len(p)):
        pp = prod // p[i]
        f = pow(pp, -1, p[i]) # функція pow(a, -1, b), знаходить обернене число, за модулем (a^-1 ≡ x (mod m))
        r = (f * pow(w[i], -1, p[i])) % p[i]
        result += pp * f * (((num % p[i]) * r) % p[i])
    return result % prod
```

На рисунках 6.5 та 6.6 наведені блок-схеми алгоритмів функцій encrypt та decrypt.

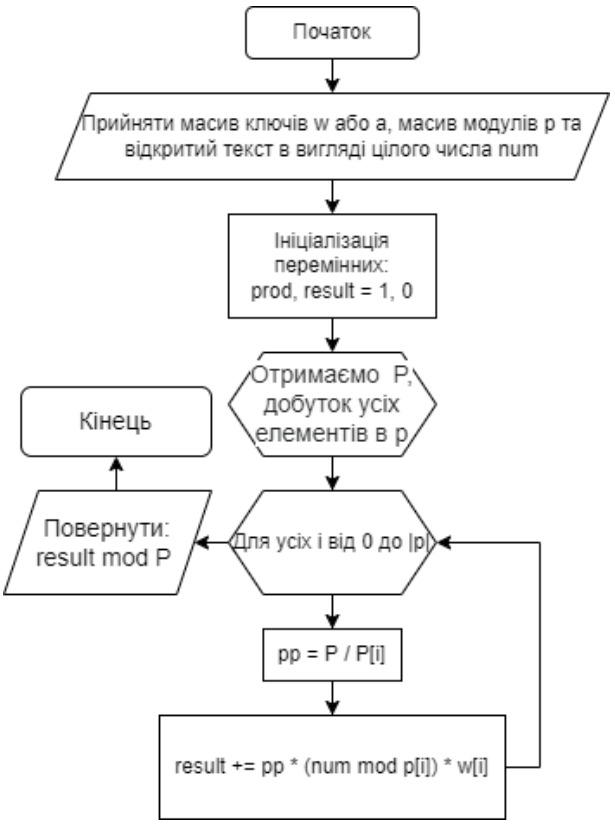


Рисунок 6.5 – Блок-схема encrypt

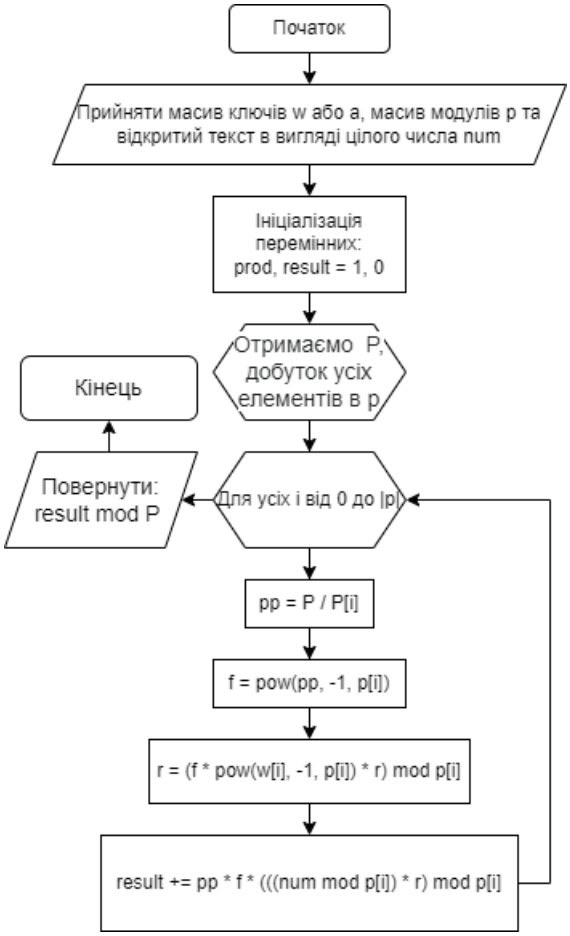


Рисунок 6.6 – Блок-схема decrypt

Для полегшення роботи з програмним засобом розроблено інтерфейс GUI, який є простий у використанні, але менш функціональний у порівнянні з CLI аналогом.

Основне вікно GUI предствлено на рисунку 6.7.

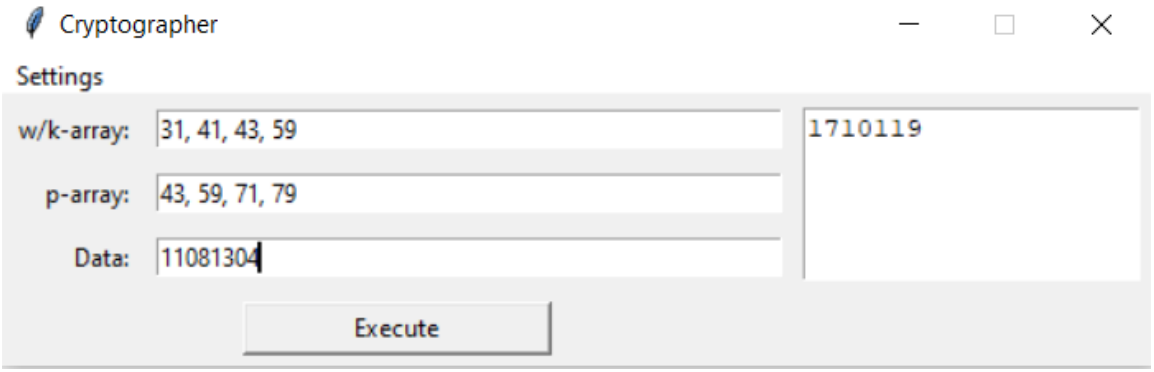


Рисунок 6.7 – Інтерфейс GUI

Меню Settings містить налаштування, в якому можна вибрати тип методу шифрування, й переключатися між шифруванням й розшифруванням, що зображено на рисунку 6.8.

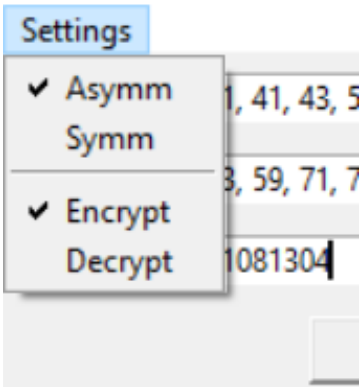


Рисунок 6.8 – Вибір методу шифрування та розшифрування

У полі w/a-array містяться відкриті ключі, при вводі яких всі елементи розділяються комою (,), аналогічно у p-array вводяться модулі. Повідомлення у цифровій формі, яке необхідно шифрувати/дешифрувати, подається у поле Data. Після внесення всіх необхідних даних запускає процес виконання функції за допомогою кнопки Execute. Слід відмітити, що в процесі реалізації

модуля шифрування та розшифрування повідомлення в Data можна вводити і текстове повідомлення, яке закодується звичайним алфавітним кодуванням в число, після чого шифрується за допомогою відповідної функції encrypt, приклад цього процесу зображений на рисунках 6.9 та 6.10.

Рисунок 6.9 – Вхідні дані у цифровій формі

Рисунок 6.10 – Вхідні дані у вигляді текстового повідомлення

При переході в режим розшифрування ця функція буде також доступною, але рекомендується вводити дані у вигляді цілого числа, яке було виведено в процесі шифруванні.

Розхроблено CLI інтерфейс, який реалізований з допомогою файлу, який також підтримує передавання аргументів для шифрування/розшифрування через консоль.

Приклад цього можна побачити на рисунку 6.11.

```
PS C:\Users\user\Python_code\encryption> python resolution.py -w 31 41 43 59 -p 43 59 71 79 -e LINE
1710119
```

Рисунок 6.11 – Шифрування інформаційних потоків

Параметри: -w і –p мають аналогічне значення як і в GUI інтерфейсі, але якщо використовувати файл resolution2.py, обов’язково параметр –w необхідно замінити на –k. Параметр –e вказує, що буде проводитися шифрування, й він приймає вхідні дані. Також можна бачити приклад розшифрування використовуючи такий метод на рисунку 6.12.

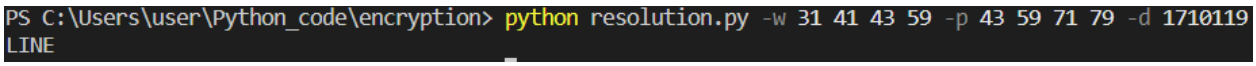


Рисунок 6.12 – Розшифрування

Для розшифрування використовується параметр –d й передане йому значення у вигляді шифротексту. В команді ключове слово python не обов'язкове, якщо використовується скомпільований файл resolution.exe, а не файл скріпт resulution.py (рисунок 6.13).

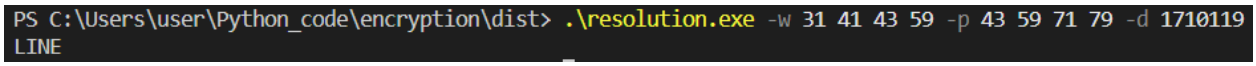


Рисунок 6.13 – Використання .exe файлу

При запуску CLI інтерфейсу, появляються запити на виконання певної дії, з переліку необхідно обрати: 2 – Encrypt, 1 – Decrypt, 0 – Generate, -1 – Check time (рис. 6.14), в залежності яку операцію потрібно виконати.

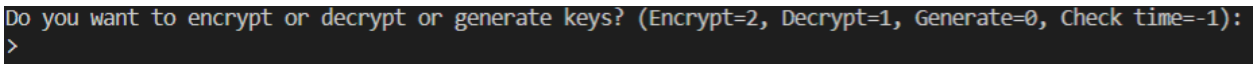


Рисунок 6.14 – Запит на виконання певних операцій

Якщо вибрати у вікні цього інтерфейсу 2, то відбувається процес шифрування з допомогою функції Encrypt. Для цього необхідно ввести всі параметри шифрування, тобто масив ключів –*w* через кому та масив модулів – *p* і відповідне повідомлення для шифрування. Після цього здійснюється виконання процесу шифрування, в результаті чого отримується Giphertext.

На рисунку 6.15 зображено стандартний приклад шифрування повідомлення LINE при w = 31, 41, 43, 59 та p = 43, 59, 71, 79, результатом виконання при цьому буде Giphertext: 1710119.

```
Do you want to encrypt or decrypt or generate keys? (Encrypt=2, Decrypt=1, Generate=0, Check time=-1):
> 2
Enter a w-array (Example: 1, 2, 3, ... n):
> 31, 41, 43, 59
Enter a p-array (Example: 1, 2, 3, ... n):
> 43, 59, 71, 79
Enter a opentext for encryption:
> LINE
Ciphertext: 1710119
```

Рисунок 6.15 – Виконання операції шифрування

Також як і GUI, CLI інтерфейс при вводі відкритого тексту, перевіряє чи текст є числовим, чи алфавітною стрічкою. У першому випадку дані передаються в функцію, без кодування алфавітним способом. Приклад виконання процесу шифрування при вхідних даних 11081304 представлено на рисунку 6.16, що відповідає повідомленню LINE.

```
Enter a opentext for encryption:
> 11081304
Ciphertext: 1710119
```

Рисунок 6.16 – Вхідні дані 11081304

В процесі розшифрування використовується функція Decrypt, її виконання ідентичне до Encrypt (рис. 6.17).

```
Do you want to encrypt or decrypt or generate keys? (Encrypt=2, Decrypt=1, Generate=0, Check time=-1):
> 1
Enter a w-array (Example: 1, 2, 3, ... n):
> 31, 41, 43, 59
Enter a p-array (Example: 1, 2, 3, ... n):
> 43, 59, 71, 79
Enter a ciphertext for decryption:
> 1710119
Do you want to encode the result? (y/n)
> y
Opentext: LINE
```

Рисунок 6.17 – Процес розшифрування

Крім того при реалізації отримується результат розшифрування у вигляді набору числових значень, які необхідно згідно кодової таблиці

перевести у текстове повідомлення, тобто у відкритий текст. В процесі вибору *n* отримується шифртекст (рис. 6.18).

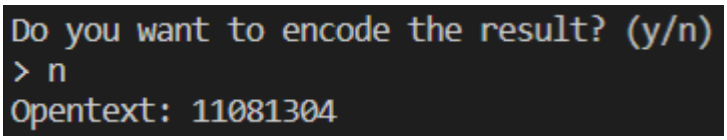


Рисунок 6.18 – Приклад зашифрованого відкритого повідомлення

Реалізація симетричного шифрування у СЗК resolution2.py або resolution2.exe здійснюється за допомогою функцій Ecnrypt і Decrypt в яких масив даних *w* змінюється на *k*. При вводі тексту слід уникати вводу зайвих символів, наприклад табуляцій чи пробілів в кінці вводі чи на початку, адже це може спричинити помилки при роботі програми.

Генерування ключів реалізовується в CLI, результат роботи представлено на рисунку 6.19.

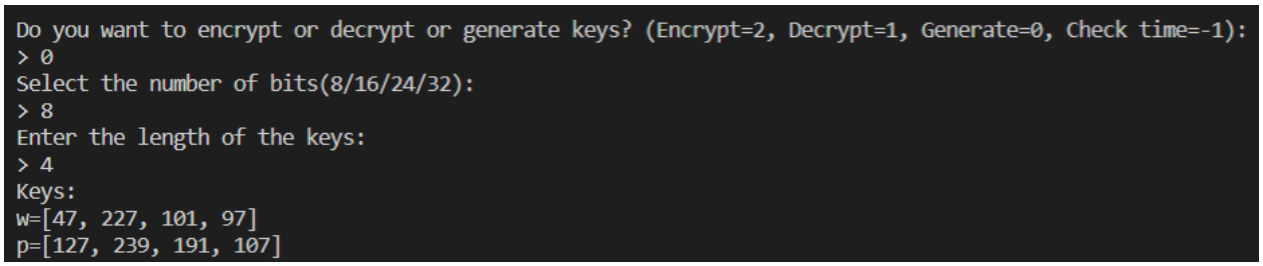


Рисунок 6.19 – Генерування ключів

Спочатку вибирається максимальна розрядність ключа і вводиться їх кількість, в даному прикладі, задається тільки 4 різні розрядності при виборі інших появиться помилка. Після введення програма виведе результат в вигляді двох списків *w* й *p*, у випадку resolution2.py - *k* і *p*.

Функція яка відповідає за процес генерування параметрів знаходиться в файлі generation.py й має назву generate_keys() з двома параметрами – кількість та розрядність ключів. Для перевірки на простоту використовується додаткова функція is_prime(), блок–схема якої наведена на рисунку 6.20.

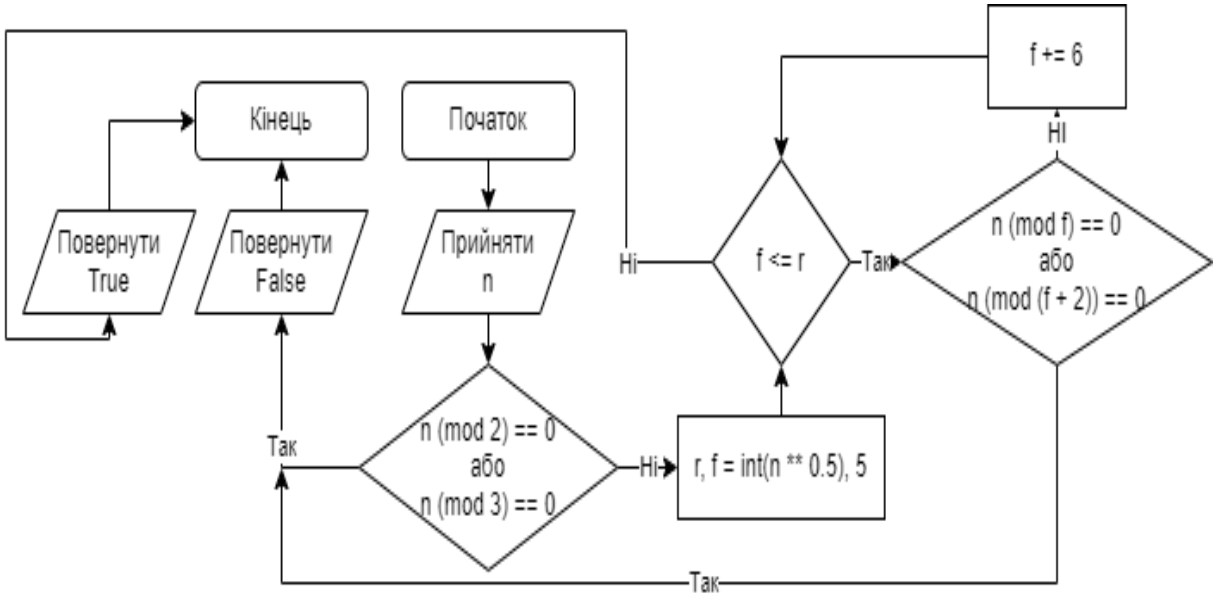


Рисунок 6.20 – Блок–схема is_prime()

Код на мові Python який реалізує функцію is_prime() представлено нижче.

```
def is_prime(n):
    if n % 2 == 0 or n % 3 == 0: return False
    r, f = int(n ** 0.5), 5
    while f <= r:
        if n % f == 0: return False
        if n % (f + 2) == 0: return False
        f += 6
    return True
```

Даний алгоритм перевірки також має назву “Тест на простоту використовуючи 6k+-1 оптимізацію”. Він відмічається високою швидкодією при будь–яких значеннях вхідних параметрів.

Для генерування ключів використовується функція generate_keys, її код представлено нижче:

```
def generate_keys(length, bits):
    if bits not in (8, 16, 24, 32):
        raise Exception("Wrong the number of the bits")
    max_bit, min_bit = 2 ** bits, 2 ** (bits - 8)
    a, b = [], []
    for i in range(length):
        x = randint(min_bit, max_bit)
        while not is_prime(x):
            x += 1
        y = randint(x, max_bit + 1)
        while not is_prime(y):
            y += 1
        a.append(x), b.append(y)
    return a, b
```

Блок-схема функції generate_keys, зображена на рисунку 6.21.

Функція generate_keys генерує ключі і перевіряється умова взаємнопростоти:

$$1 \leq |a_i| \leq b_i \text{ та НСД}(|a_i|, b_i) = 1.$$

Для порівняння швидкодії запропонованих криптоалгоритмів і відомих використовується функція Check time, при виборі якої вказується процес шифрування/розшифрування і виводиться результат обчислень (рис. 6.22).

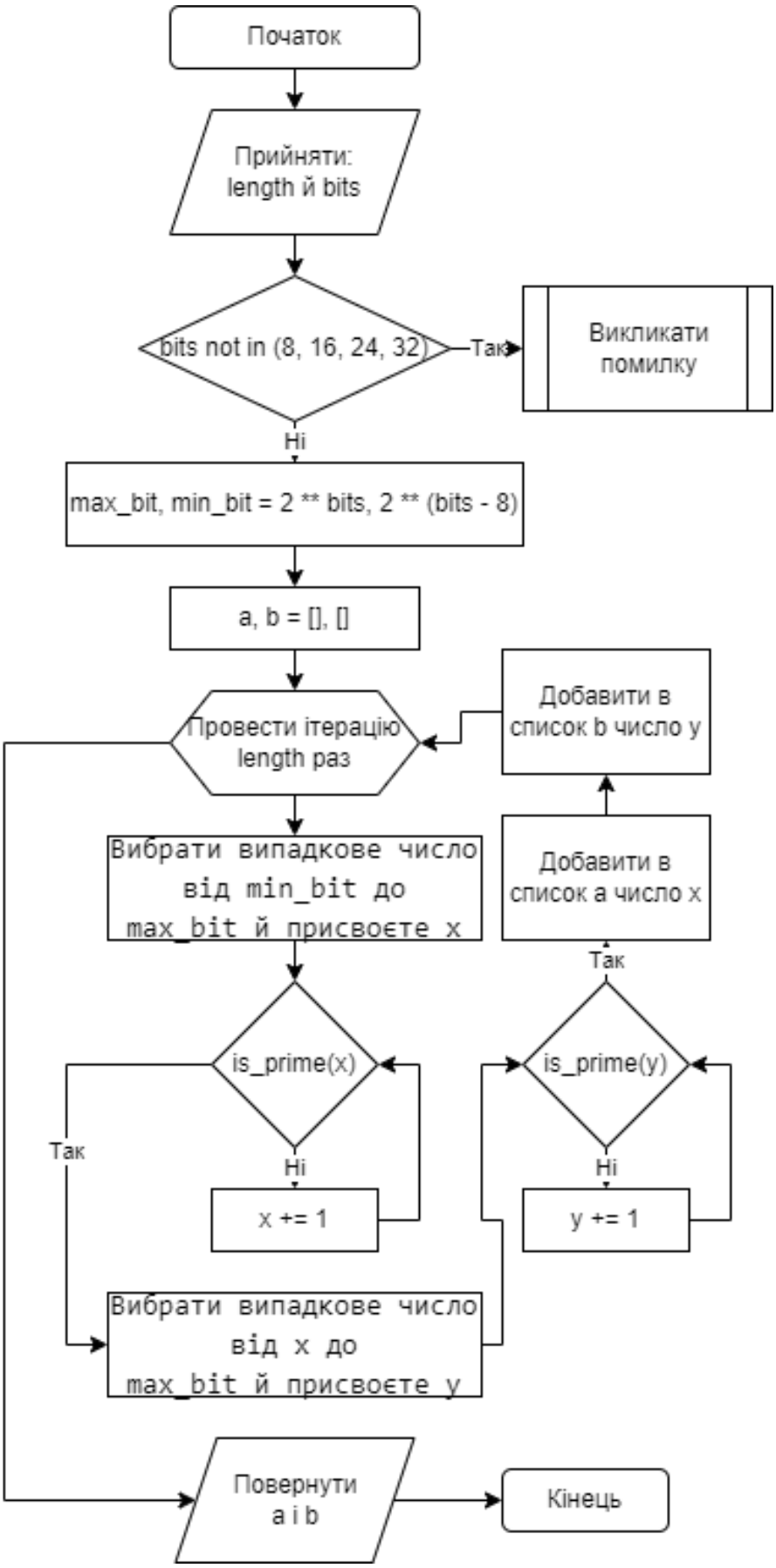


Рисунок 6.21 – Блок–схема функції generate_keys

Рисунок 6.22 – Результ виконання функції Check time

У випадку з файлом resolution.py або .exe, порівняння відбувається з алгоритмом RSA, який реалізовувався у модулі rsa.py. Слід зазначити, що для об’єктивності визначення часових характеристик відбувається *s* (вводиться користувачем) замірів і знаходиться середнє значення, для різної розрядності та процесів шифрування, або розшифрування. Швидкодія визначалася за допомогою вбудованого модуля timeit від Python із використанням функції repeat згідно виразу:

round(sum(timeit.repeat(stmt=test, number=number, repeat=5)) / 5, 5)

Масив всіх тестів та їх вхідних даних представлено нижче. Функції encrypt та decrypt використовуються в процесі асиметричного шифрування/розшифрування у СЗК, а encrypt2 та decrypt2 – згідно алгоритму RSA.

```
tests = [
    lambda: encrypt(w=[193, 7, 47, 113], p=[17, 3, 2, 227], num=11081304),
    lambda: encrypt2((2825, 21823), "LINE"),
```

lambda: encrypt(w=[19441, 14759, 46919, 51347], p=[3727, 10037, 1667, 61379], num=11081304),

lambda: encrypt2((896718305, 1538332949), "LINE"),

lambda: encrypt(w=[3341227, 9026887, 12226673, 1616621], p=[10216849, 10173959, 116101, 8420521], num=11081304),

lambda: encrypt2((7023302398829, 13501436698307), "LINE"),

lambda: encrypt(w=[2600395621, 2451048343, 3853117571, 1126795183], p=[869012423, 1425080537, 4197720061, 3478747583], num=11081304),

lambda: encrypt2((10493372830128350491, 15043166710414400477), "LINE"),

lambda: decrypt(w=[193, 7, 47, 113], p=[17, 3, 2, 227], num=7224),

lambda: decrypt2((14921, 21823), [21629, 16226, 6007, 11692]),

lambda: decrypt(w=[19441, 14759, 46919, 51347], p=[3727, 10037, 1667, 61379], num=3793995912705605),

lambda: decrypt2((1491839117, 1538332949), [514238408, 915228368, 400087027, 1283296618]),

lambda: decrypt(w=[3341227, 9026887, 12226673, 1616621], p=[10216849, 10173959, 116101, 8420521], num=90845309435412153753239000),

lambda: decrypt2((19382402650085, 13501436698307), [10947356661727, 2262821532403, 12148830514716, 2602448734982]),

lambda: decrypt(w=[2600395621, 2451048343, 3853117571, 1126795183], p=[869012423, 1425080537, 4197720061, 3478747583], num=36340083833939294476842988492107217 95),

lambda: decrypt2((19130392151640161311, 15043166710414400477), [8433464188612863724, 5731303454996900902, 11436009858380315906, 35916939674150021 48]),

]

В кінці, за допомогою бібліотеки *xlsxwriter*, відповідні показники (алгоритм, розрядність, кількість повторень, час виконання) інвертуються у

таблицю Excel. Для прикладу вибиралися вхідні параметри *n*=8, 16, 24, 32 біти при кількості повторень 10000 разів.

Результати роботи програми представлено на рисунку 6.23.

	A	B	C	D
1	**Алгоритм й тип**	**Розрядність**	**Повторення**	**Час виконання**
2	Encrypt	8	10000	0.07422
3	RSA Encrypt	8	10000	0.09263
4	Decrypt	8	10000	0.07941
5	RSA Decrypt	8	10000	0.28165
6	Encrypt	16	10000	0.08216
7	RSA Encrypt	16	10000	0.50518
8	Decrypt	16	10000	0.08513
9	RSA Decrypt	16	10000	0.69843
10	Encrypt	24	10000	0.18185
11	RSA Encrypt	24	10000	0.13799
12	Decrypt	24	10000	0.24573
13	RSA Decrypt	24	10000	0.38735
14	Encrypt	32	10000	0.26354
15	RSA Encrypt	32	10000	0.50862
16	Decrypt	32	10000	0.29852
17	RSA Decrypt	32	10000	0.91182

Рисунок 6.23 – Результат швидкодії запропонованого симетричного методу в СЗК і алгоритму RSA

В результаті проведених досліджень встановлено, що в залежності від розрядності вхідних параметрів та вибору процесу шифрування/розшифрування швидкодія запронованого методу є більшою в порівнянні з криптоалгоритмом RSA і переважає приблизно від 1,2 до 4 разів.

За необхідності систему можна розширювати шляхом додавання нових тестів. Для цього необхідно внести відповідні зміни до масиву *test*, де слід вказати тип операції та обраний криптографічний алгоритм. Обробка алгоритму здійснюється на основі значень, що зберігаються у масиві *alg_typ*, а визначення розрядності даних відбувається через параметри, зазначені у

масиві *bit_length*. Такий підхід забезпечує гнучкість у конфігурації тестування та адаптивність системи до різних криптографічних методів.

Для реалізації симетричного алгоритму шифрування у СЗК було розроблено програмний засіб, який виконується за допомогою файлу *resolution2.py* у форматі *.exe*. Порівняння часових характеристик та показників продуктивності здійснювалося з алгоритмом AES. Аналогічно до файлу *resolution.py*, тестування здійснюється за допомогою функції *timeit.repeat*, починаючи з розрядності 16 біт.

Функції AES, необхідні для проведення тестування, були взяті зі сторонньої бібліотеки *PyCrypto*, що забезпечує ефективну реалізацію криптографічних алгоритмів. У файлі *resolution2.py* представлено набір тестів, що використовуються для аналізу продуктивності, а також відкриті ключі, які застосовуються в алгоритмі AES.

keys_for_aes = [b'Sixteen byte key', b'Twenty-four byte key and', b'thirty two key jdad kdaso daskod']
tests = [
lambda: encrypt(k=[19441, 14759, 46919, 51347], p=[3727, 10037, 1667, 61379], num=11081304),
lambda: instance.encrypt(b"LINE"),
lambda: encrypt(k=[3341227, 9026887, 12226673, 1616621], p=[10216849, 10173959, 116101, 8420521], num=11081304),
lambda: instance.encrypt(b"LINE"),
lambda: encrypt(k=[2600395621, 2451048343, 3853117571, 1126795183], p=[869012423, 1425080537, 4197720061, 3478747583], num=11081304),
lambda: instance.encrypt(b"LINE"),
lambda: decrypt(k=[19441, 14759, 46919, 51347], p=[3727, 10037, 1667, 61379], num=3793995912705605),
lambda: instance.decrypt(b'@\xa76\xc9'),

lambda: decrypt(k=[3341227, 9026887, 12226673, 1616621], p=[10216849, 10173959, 116101, 8420521], num=90845309435412153753239000),

lambda: instance.decrypt(b'/\xee\xca\xde'),

lambda: decrypt(k=[2600395621, 2451048343, 3853117571, 1126795183], p=[869012423, 1425080537, 4197720061, 3478747583], num=36340083833939294476842988492107217 95),

lambda: instance.decrypt(b'\xc4\xb8m\xb6'),]

На рисунку 6.24 зображено результат швидкодії запропонованого методу шифрування і алгоритму AES. Для прикладу вибиралися вхідні параметри *n*= 16, 24, 32 біти при кількості повторень 10000 разів.

```
Do you want to encrypt or decrypt or generate keys? (Encrypt=2, Decrypt=1, Generate=0, Check time=-1):
> -1
16 Encrypt: 0.11321
16 AES Encrypt: 0.63409
16 Decrypt: 0.14642
16 AES Decrypt: 0.64672
24 Encrypt: 0.14612
24 AES Encrypt: 0.62031
24 Decrypt: 0.29
24 AES Decrypt: 0.67502
32 Encrypt: 0.39709
32 AES Encrypt: 0.67398
32 Decrypt: 0.43323
32 AES Decrypt: 0.67142
```

Рисунок 6.24 – Результат швидкодії запропонованого методу шифрування і алгоритму AES

У даному файлі результати обчислень зберігаються у таблиці під назвою *result_symmetrical.xlsx*, яка містить згенеровані дані тестування. Вміст цієї таблиці представлено на рисунку 6.25, що дозволяє наочно проаналізувати отримані результати та порівняти часові характеристики реалізованих алгоритмів.

Алгоритм й тип	Розрядність	Повторення	Час виконання
Encrypt	16	10000	0.11321
AES Encrypt	16	10000	0.63409
Decrypt	16	10000	0.14642
AES Decrypt	16	10000	0.64672
Encrypt	24	10000	0.14612
AES Encrypt	24	10000	0.62031
Decrypt	24	10000	0.29
AES Decrypt	24	10000	0.67502
Encrypt	32	10000	0.39709
AES Encrypt	32	10000	0.67398
Decrypt	32	10000	0.43323
AES Decrypt	32	10000	0.67142

Рисунок 6.25 – Результати моделювання швидкодії від розрядності і кількості повторень

На основі даних, отриманих із таблиць (рисунки 6.18 і 6.20), було побудовано діаграми, що відображають часові характеристики процесів шифрування та розшифрування для кожного алгоритму. Візуалізація результатів дозволяє оцінити ефективність реалізованих методів та порівняти продуктивність алгоритмів у різних умовах.

На рисунках 6.26 та 6.27 представлено результати проведених експериментальних досліджень швидкодії симетричних криптоалгоритмів у СЗК та алгоритму AES. Аналіз отриманих даних показує, що запропонований метод шифрування в СЗК демонструє значну перевагу в швидкодії, приблизно в два рази швидше порівняно з AES.

Цей результат обґрунтовує доцільність використання криптографічних алгоритмів у СЗК, особливо для застосувань, де критичним фактором є швидкість обробки даних. Використання запропонованого підходу може сприяти підвищенню продуктивності інформаційно-комунікаційних систем, що застосовують криптографічні механізми для захисту даних.

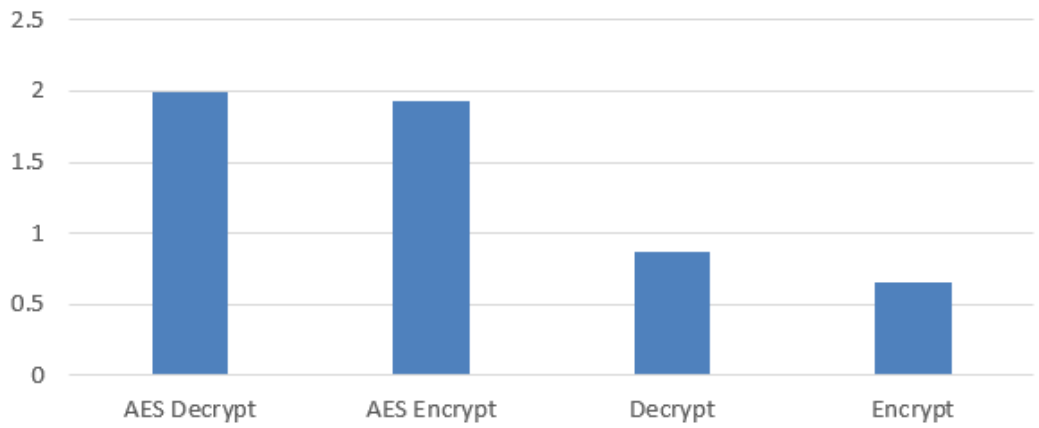


Рисунок 6.26 – Результат проведених досліджень швидкодії симетричних криптоалгоритмів у СЗК і AES

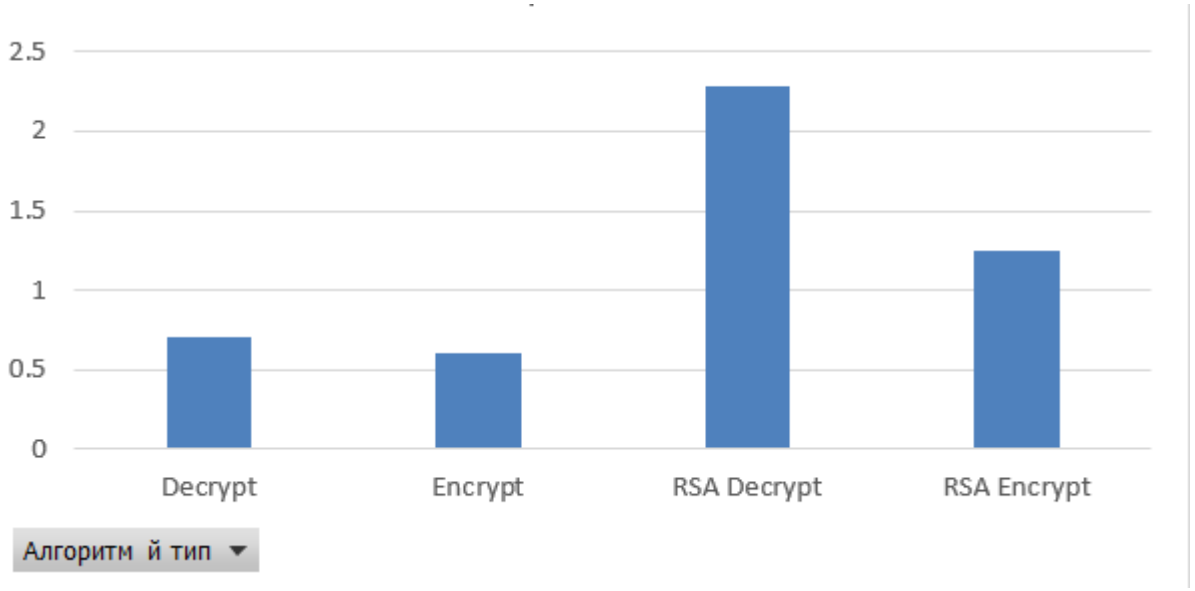


Рисунок 6.27 – Результат проведених досліджень швидкодії асиметричних криптоалгоритмів з використанням СЗК і RSA

Слід відмітити, що процес розшифрування з використанням СЗК приблизно в три рази швидший в порівнянні з криптоалгоритмом RSA.

Описаний процес кодування та декодування символів базується на стандартному алфавітному кодуванні, де кожен символ в алфавіті відповідає певному ASCII-коду в відкритому тексті і виконується наступним чином:

Кодування:

1. Ітерується кожен символ вхідного рядка.

2. Для кожного символу обчислюється його ASCII-код, з якого віднімається значення 65 для отримання індексу символу в алфавітній таблиці.

3. Якщо результат має лише одну цифру, до нього додається ведучий нуль для формування двоцифрового коду.

4. Усі отримані двоцифрові значення об’єднуються в один рядок, який потім перетворюється на ціле число.

5. Повертається отримане число.

Декодування:

1. Вхідне число конвертується у рядок.

2. Якщо кількість цифр в рядку непарна, перед рядком додається символ "0".

3. Кожні два символи інтерпретуються як окреме число.

4. Для кожного числа до отриманого результату додається значення 65, після чого за допомогою функції *chr()* відновлюється відповідний символ.

5. Усі символи об’єднуються в одну строку, яка і є результатом декодування.

Цей алгоритм дозволяє ефективно перетворювати текст у число та назад, що може бути корисним у криптографічних та інформаційних системах.

Нижче представлено програмний модуль, який реалізує ці функції на мові Python:

```
def transform(num: int) -> str:
    num = str(num)
    return num if len(num) == 2 else '0' + num

def alphabet_encode_decode(text: str, mode=True):
    if mode:
        result = int(''.join([transform(ord(x) - 65) for x in text.upper() if x.isalpha()]))
    else:
```

text = str(text)

text = '0' + text if len(text) % 2 != 0 else text

*result = ''.join([chr(int(text[i * 2:i * 2 + 2]) + 65) for i in range(len(text) // 2)])*

return result

Крім того визначимо деякі особливості інтерфейсу CLI:

– при закінченні виконання операції шифрування/розшифрування відбувається повернення в меню вибору дії, де можна вибрати, яку наступну дію здійснювати;

– при вводі неправильних даних програма виведе, що саме спричинило помилку, після чого, буде можливість знову спробувати ввести потрібні данні, приклад наведено на рисунку 6.28.

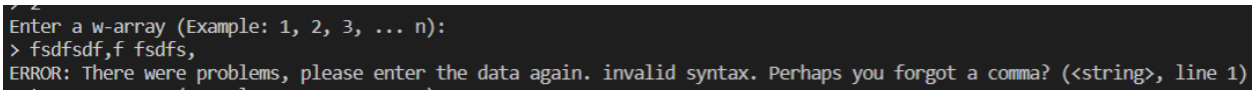


Рисунок 6.28 – Приклад помилки

– завершення роботи програми можна здійснити при виході з консолі, або використовуючи комбінацію клавіш Ctrl + Z.

– всі файли програми обов’язково повинні находитися в одному каталозі, щоб уникнути будь які помилки з пошуком модуля.

– щоб скомпілювати певний інтерфейс в .exe файл необхідно використати, бібліотеку *pyinstaller*. Команда, яка дозволяє це зробити, виглядає наступним чином: *pyinstaller --onefile gui.pyw*.

Варто зазначити, що за замовчуванням, в каталозі з файлами, будуть уже скомпільовані файли .exe, до кожного інтерфейсу й потреба в повторній перекомпіляції може виникнути тільки при внесенні деяких змін в файлах .py.

6.2 Реалізація алгоритму шифрування на основі ступінчатої системи залишкових класів

Ця розробка зосереджена на реалізації алгоритму шифрування, який використовує ІСЗК [321, 322]. СЗК є фундаментальним математичним інструментом, який знаходить застосування в областях, таких як криптографія, обчислювальна техніка та цифрова обробка сигналів. Вона дозволяє представляти цілі числа у формі набору залишків від ділення цього числа на декілька взаємно простих модулів.

Ключовими перевагами використання ІСЗК у криптографії є здатність до паралельних обчислень та висока швидкість виконання операций, особливо в умовах обмежених ресурсів. Це робить її особливо привабливою для використання в застосунках, де важливі швидкість та ефективність, наприклад, в мобільних пристроях та вбудованих системах.

Програмна реалізація методу шифрування на основі ІСЗК включає кілька ключових етапів:

1. Вибір модулів. Найважливішим кроком є вибір взаємно простих модулів, які використовуються у системі, які повинні бути достатньо великими, щоб забезпечити високий рівень безпеки.
2. Перетворення даних. Вхідні дані (закодований текст або інша інформація) конвертуються у залишки за допомогою операцій ділення на модулі.
3. Шифрування. Процес шифрування здійснюється на основі запропонованого методу з використанням ступінчатої системи залишкових класів.
4. Конвертування даних. Після обробки залишків, результат шифрування конвертується назад у звичайну форму для зберігання або передачі.
5. Розшифрування. Процес розшифрування на кожному етапі ІСЗК здійснюється на основі Китайської теореми про залишки.

Ця програмна реалізація призначена для демонстрації потенціалу ІСЗК у криптографії та надає основу для розширення та адаптації методу під конкретні потреби та вимоги. Важливо підкреслити, що безпека шифрування сильно залежить від кількості рівнів ІСЗК, правильного вибору модулів, реалізації криптографічних алгоритмів.

Реалізація даного програмного продукту була виконана із використанням файлової системи як основного механізму для обробки та зберігання даних. Файли слугували первинним медіумом для імпорту вхідних даних, які підлягали шифруванню, а також для експорту результуючих зашифрованих даних. Цей підхід дозволив забезпечити високий рівень гнучкості та масштабованості у процесі шифрування, оскільки він сприяв легкій інтеграції з різноманітними джерелами та форматами даних.

Використання файлової системи також спростило інтерфейс користувача програмного продукту, надавши можливість легко вибирати та обробляти великі обсяги даних без необхідності їх попередньої ручної підготовки. Такий підхід є особливо важливим у контексті реалізації складних криптографічних операцій, де точність та надійність обробки даних мають критичне значення.

Файл Main.py є частиною програми на мові програмування Python, яка використовує бібліотеку PyQt5 для створення графічного інтерфейсу користувача (GUI). Код програми представлено в додатку А.

Розглянемо більш детальний опис даного модуля, який складається з певних етапів: імпорт бібліотек, створення віджетів та вікон, робота з графічним інтерфейсом, оголошення та налаштування компонентів, налаштування стилю та створення головного вікна програми.

Імпорт бібліотек здійснюється на основі модуля sys, який призначений для доступу до функцій і змінних Python, пов'язаних з системними операціями. Для створення віджетів та вікон використовується модуль PyQt5, який є основою функції PyQt5.QtWidgets.

Робота з графічним інтерфейсом, зокрема з кольорами та палітрою відбувається у PyQt5з застосуванням функції QtGui. Крім того модуль PyQt5 у PyQt5.QtCore призначенипй для роботи з основними функціями, такими як координати та події.

Імпорт інших власних модулів:

– CustomTabWidget –модуль, який містить власний віджет вкладок;

– DarkOrangePalette – модуль, який відповідає за палітру кольорів;

– Navbar –модуль з навігаційною панеллю;

– LayerMainLayout – модуль в якому налаштовується головний макет.

– ConfigDialog –модуль діалогового вікна конфігурації.

Клас MainWindow в повній мірі відповідає за головне вікно програми і він є підкласом QMainWindow.

У конструкторі відбувається налаштування діалогового вікна конфігурації ConfigDialog, після закриття якого викликається метод initUI для створення компоненти і встановлення взаємозв'язків між ними, таких як – навігаційна панель та віджет вкладок.

1. Оголошення та налаштування компонентів:

– main_widget – основний віджет, на якому розташовані всі інші елементи;

– Layout – вертикальний макет для розташування елементів;

– nav_layout – навігаційна панель, створена на основі класу NavigationBar;

– tab_widget – віджет вкладок на основі класу CustomTabWidget.

– palette – палітра кольорів, створена на основі класу DarkOrangePalette.

2. Налаштування стилю:

– використовується стиль "Fusion" для додатку PyQt5;

– встановлюється стилізація елементів інтерфейсу за допомогою CSS.

3. Створення об'єкта app типу QApplication:

– QApplication є головним об'єктом додатку PyQt5 і керує його життєвим циклом.

4. Створення головного вікна window типу MainWindow.

5. Відображення головного вікна за допомогою методу show().

6. Виклик методу exec_() об'єкта app для запуску головного циклу програми, який очікує події та реагує на них.

Отже, цей код відповідає за створення графічного інтерфейсу додатку на основі бібліотеки PyQt5, в якому є навігаційна панель, віджет вкладок та інші компоненти, які налаштовуються стилізацією за допомогою CSS. Після запуску програма очікує взаємодії користувача і відображає головне вікно з відповідними компонентами.

Файл Config.py призначений для задання тексту, який потрібно зашифрувати і кількості ключів на кожному рівні ІСЗК.

text = ""

keys_per_layer = 3

де text - текст для шифрування.

keys_per_layer = кількість ключів на шар

Файл crt.py складається з набору функцій для роботи з криптографічними операціями, зокрема: генерування ключів, шифрування та розшифрування повідомлень за допомогою ІСЗК. Фрагмент коду представлено у додатку Б.

Даний код є основою проекту, який дає змогу здійснювати генерування ключів, кодування символів, шифрування та розшифрування. Більш детальний опис основних функцій:

1. generate_keys_for_layers(layer, keys_per_layer) – функція генерує задану кількість ключів keys_per_layer для шару layer. Основні етапи цієї функції:

- поділ layer на 10;
- визначення діапазону (min та max) для генерування простих чисел на основі поділеного значення layer;
- генерування keys_per_layer простих чисел в заданому діапазоні та їх повернення у вигляді списку.

2. encrypt_message(message) – функція приймає текстове повідомлення message і повертає його зашифровану версію у вигляді числа. Основні етапи цієї функції:

- перетворення кожного буквенного символу тексту у велику літеру та обчислення відповідного числового значення для кожної літери (*A*=1, *B*=2, ..., *Z*=26).
- пробіл задається числовим значенням 40.
- повертається зашифроване повідомлення у вигляді одного цілого числа.

3. encrypt_number(number, keys) – функція приймає число number та список ключів keys і здійснює шифрування з використанням СЗК. Результатом є список зашифрованих значень для кожного ключа, тобто залишків.

4. chinese_remainder_theorem(modules, remainders) – функція здійснює перетворення згідно китайської теореми про залишки для згенерованих модулів і відповідних їм залишків. Основні етапи цієї функції:

- обчислюється *N* - добуток всіх модулів.
- здійснюється розрахунок проміжних значень n_i та x_i для кожного модуля та залишку;
- сумується result += remainder * Decimal(n_i) * Decimal(x_i) результат для кожного модуля та залишку.
- обчислюється кінцевий результат.

5. init_1_layer(encrypted_message, keys_per_layer) – функція ініціалізує перший шар шифрування, приймаючи зашифроване повідомлення encrypted_message та кількість ключів keys_per_layer. В її основі є функції generate_keys_for_layers – генерування ключів, encrypt_number –шифрування числа.

6. init_layer(layer, keys_per_layer) – функція ініціалізує наступні етапи шифрування на кожному з рівнів ІСЗК, приймаючи список layer і кількість ключів keys_per_layer.

Загалом, цей код дозволяє шифрувати повідомлення на основі використання ІСЗК з використанням генерованих ключів і обчислень. На рисунку 6.29 представлено результати роботи модуля, на першому етапі вводиться повідомлення, яке кодується згідно ASCII кодів, і вводиться кількість модулів для першо рівня.

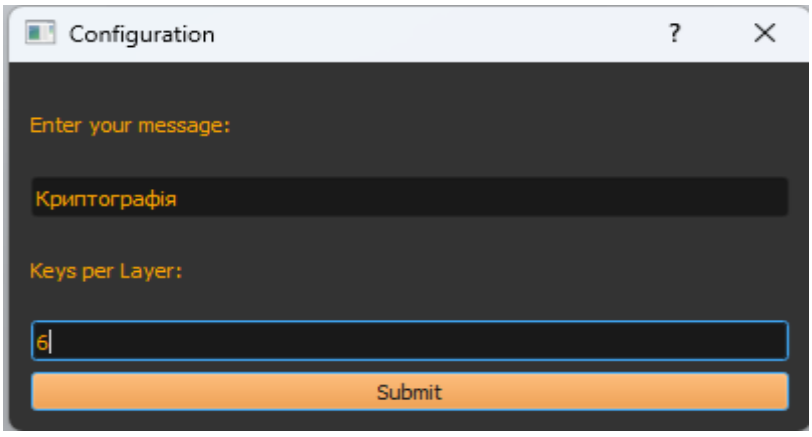


Рисунок 6.29 – Вікно програми для введення повідомлення і кількості модулів

Після цього здійснюється перехід у головне вікно програми де відбувається процес шифрування та розшифрування. На першому рівні виводяться залишки по відповідних взаємно простих модулях, які генеруються випадковим чином.

На рисунку 6.30 представлено інтерфейс реалізації програмного модуля, який відповідає за процес шифрування і розшифрування.

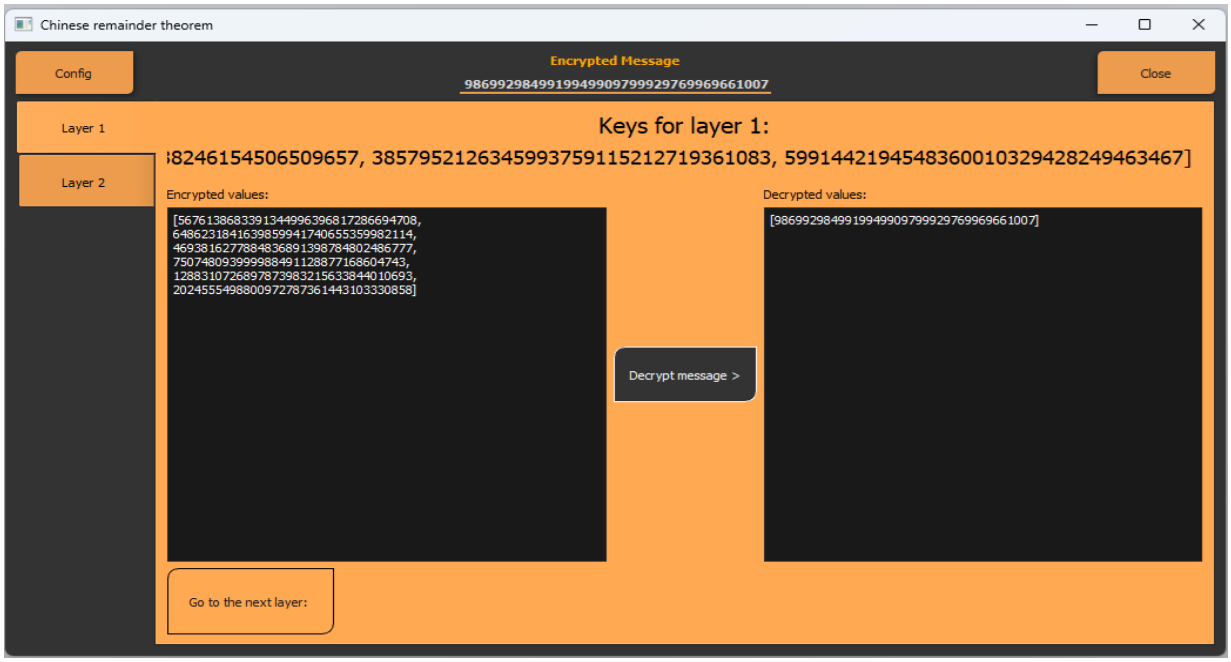


Рисунок 6.30 – Основне вікно програми

Для переходу на наступний рівень необхідно нажати функціональну кнопку «Go to the next layer», в якому відбувається процес шифрування і розшифрування згідно випадково згенерованих взаємно простих модулів (рис. 6.31).

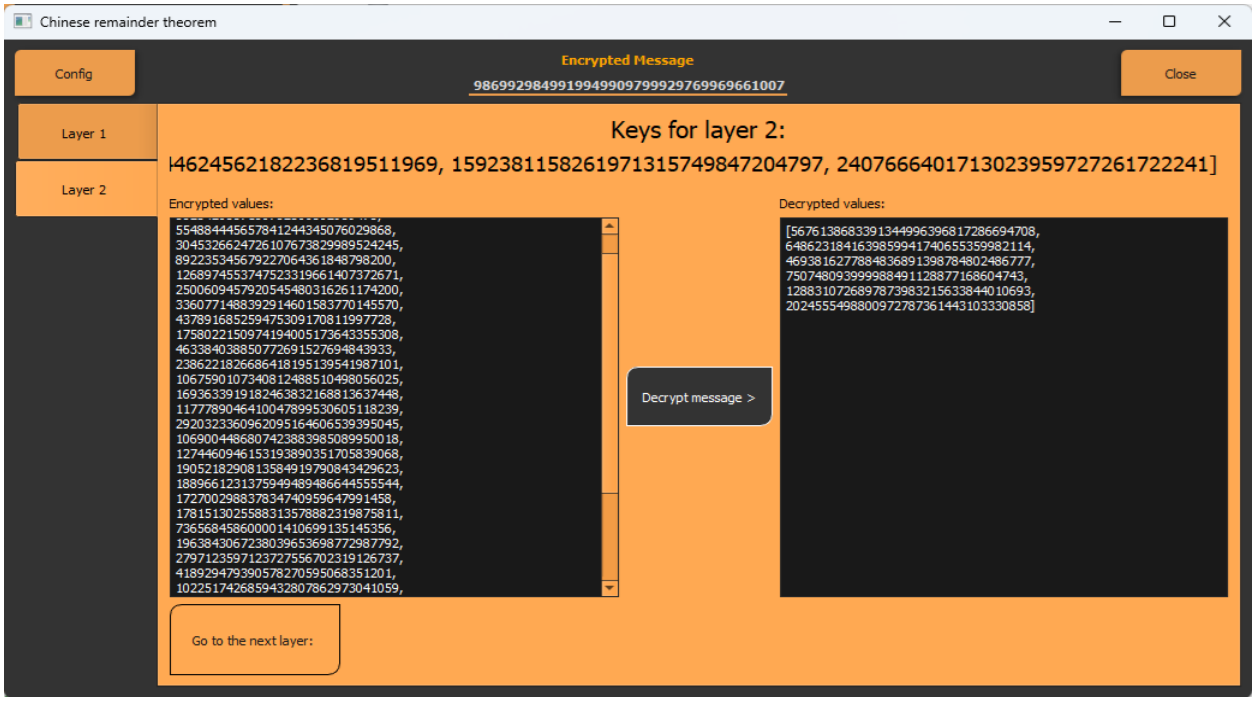


Рисунок 6.31 – Процес шифрування/розшифрування на другому рівні

Таким чином можна проводити шифрування і розшифрування в залежності від потреб щодо забезпечення рівня захищеності, який буде зростати зі збільшенням рівня симетричного крипто алгоритму в ІСЗК.

Файл ConfigDialog.py інкапсулює функціональність діалогового вікна конфігурації, реалізованого як довільний клас QDialog бібліотеки PyQt5. Цей модуль призначений для створення та налаштування графічного користувацького інтерфейсу, забезпечуючи інтерактивне середовище для введення користувачем конфігураційних параметрів. Використання PyQt5 потужної бібліотеки для створення графічних інтерфейсів на мові Python, дозволяє ефективно інтегрувати елементи управління та відображення даних у вигляді візуально привабливого та інтуїтивно зрозумілого інтерфейсу представлено в додатку В.

Розглянемо цей код детальніше:

1. ConfigDialog(QDialog) – клас є підкласом QDialog і відповідає за створення вікна діалогового вікна конфігурації.

2. У конструкторі класу __init__ виконуються такі дії:

– встановлюється заголовок вікна діалогового вікна ("Configuration");

– задається розмір вікна (400x200);

– встановлюється функція зворотнього виклику (callback_form), яка викликається після успішного введення даних;

– створюється вертикальний макет layout для розташування елементів інтерфейсу.

3. Створюються різні елементи інтерфейсу, такі як:

– label1 та label2 для відображення тексту;

– self.input1 та self.input2 – поля для введення тексту та кількості ключів на шарі;

– submit_button – кнопка "Submit" для відправлення даних.

4. Встановлюється стиль (CSS) для лейблів і вхідних полів.

5. З'єднано подію натискання на кнопку "Submit" з функцією submit, яка опрацьовує введені дані.

6. Функція submit отримує значення, введені користувачем в поля input1 та input2.

7. Виконуються перевірки введених даних на правильність:

- тексту у поле input1;
- цілого числа у поле input2;
- тексту, який містить тільки літери.

8. У випадку успішного проходження усіх перевірок активується accept, який викликає функцію зворотнього виклику callback, передаючи їй введені дані. Після цього процес завершується закриттям діалогового вікна.

9. Accept перевизначений для виклику функції зворотнього виклику та закриття діалогового вікна.

Таким чином, реалізований код формує інтерфейс діалогового вікна конфігурації, оснащеного текстовими полями для введення інформації та полем для зазначення кількості ключів у кожному шарі. Введені користувачем дані підлягають обробці відразу після активації команди "Submit".

Файл CustomTabWidget.py відповідає за розробку спеціалізованих віджетів вкладок у межах графічного інтерфейсу користувача на базі PyQt5 (додаток Г). Ці віджети надають можливість користувачам перемикатися між різними рівнями або секціями інтерфейсу. Основна функціональність вкладок полягає в налаштуванні вмісту для кожної окремої вкладки та додаванні нових вкладок за потребою.

Представимо більш детальний опис програмної реалізації файлу CustomTabWidget.py.

1. CustomTabWidget(VerticalTabWidget) – клас є підкласом VerticalTabWidget і відповідає за створення власного віджета вкладок.

2. У конструкторі __init__ виконуються наступні дії:

- викликається конструктор базового класу VerticalTabWidget;

– зберігається посилання на батьківський віджет (головне вікно) в self.parent;

– викликається метод createTabs для створення початкових вкладок;

– викликається метод createLayouts для створення макетів для кожної вкладки.

3. За допомогою createTabs створюється дві початкові вкладки (Layer 1 та Layer 2) та додає їх до віджета вкладок.

4. На основі використання createLayouts створюються макети для кожної існуючої вкладки, застосовуючи createLayoutForTab.

5. Add_tab дозволяє додати нову вкладку після поточної вкладки. Він створює нову вкладку та викликає метод createLayoutForTab для налаштування її макету.

6. Застосування createLayoutForTab дозволяє формувати макет для конкретної вкладки і включає в себе створення лейбла з назвою вкладки, отримання попереднього шару (якщо такий існує), виклик функції crt.init_layer для шифрування попереднього шару (або crt.init_1_layer, якщо це перший шар), розробку вмісту для вкладки за допомогою LayerMainLayout.

7. Next_tab перемикає віджет вкладок на наступну вкладку, якщо вона існує.

Отже, цей спеціалізований віджет вкладок забезпечує функціональність додавання та конфігурації нових вкладок, кожна з яких містить унікальний контент, та дозволяє користувачу ефективно переходити між цими вкладками у рамках основного вікна програмного забезпечення.

Файл DarkOrangePallete.py визначає клас DarkOrangePalette, який є підкласом класу QPalette з бібліотеки PyQt5. Даний клас слугує для створення кольорової палітри, яка використовується в різних віджетах графічного інтерфейсу користувача, дозволяючи визначати специфічні кольорові схеми для окремих елементів інтерфейсу.

from PyQt5.QtGui import QPalette, QColor

```
class DarkOrangePalette(QPalette):
  def __init__(self)-> None:
      super().__init__()
      self.setColor(QPalette.Window, QColor(51, 51, 51))
      self.setColor(QPalette.Button, QColor(255, 128, 0))
      self.setColor(QPalette.ButtonText, QColor(25, 25, 25))
      self.setColor(QPalette.Base, QColor(25, 25, 25))
      self.setColor(QPalette.AlternateBase, QColor(51, 51, 51))
```

Основна функціональність цього класу полягає у наступному:

1. Метод __init__ – конструктор класу, який викликає функцію створення об'єкта базового класу QPalette. В ньому встановлюються кольори для різних елементів інтерфейсу, використовуючи метод setColor.

- QPalette.Window – фоновий колір вікна;
- QPalette.Button – колір кнопок;
- QPalette.ButtonText – колір тексту на кнопках;
- QPalette.Base – колір фону базових елементів;
- QPalette.AlternateBase – колір фону альтернативних елементів.

В даному випадку кольори задані в форматі RGB (червоний, зелений, синій) за допомогою класу QColor. Кожен з яких може бути налаштований за потребою для створення власного оформлення інтерфейсу. У цьому випадку встановлені темні відтінки сірого та помаранчевого кольорів для кнопок.

Файл EncryptedMessageView.py визначає клас NavBarView, який є підкласом QVBoxLayout з бібліотеки PyQt5 і відповідає за створення та оновлення віджета навігаційної панелі, який відображає інформацію про зашифроване повідомлення.

```
from PyQt5.QtWidgets import QApplication,QLabel, QMainWindow,
QWidget, QVBoxLayout, QTabWidget, QTabBar, QPushButton, QLineEdit,
```

```
QLabel, QHBoxLayout, QStylePainter, QStyleOptionTab, QStyle, QProxyStyle,
QGraphicsDropShadowEffect, qApp
    from PyQt5.QtGui import QColor, QPalette
    from PyQt5.QtCore import QRect, QPoint, Qt
    import crt
    class NavBarView(QVBoxLayout):
        def __init__(self):
            super().__init__()
            label = QLabel("<h4><center>Encrypted Message</center></h4>")
            label.setObjectName("NavBarLabel")
            self.addWidget(label)
            label_bot  = QLabel("<h4> <center> set your message in config
window </center> </h4>")
            label_bot.setObjectName("NavBarText")
            self.label_bot = label_bot
            self.addWidget(self.label_bot)
        def update_label_bot_text(self, new_text, keys = None):
            text = crt.encrypt_message(new_text)
            self.label_bot.setText(f"<h4><center>{text}</center></h4>")
```

Ключове завдання цього класу визначається наступним чином:

1. __init__ – конструктор класу створює віджет навігаційної панелі, який складається з двох QLabel.

- перший QLabel (label) містить заголовок "Encrypted Message" та встановлює йому певний стиль за допомогою setObjectName;
- другий QLabel (label_bot) початково містить текст "set your message in config window" та має встановлене ObjectName.

2. Метод update_label_bot_text(new_text, keys=None) призначений для оновлення тексту другого QLabel (label_bot) з актуальним значенням.

Параметр new_text представляє собою новий текст для відображення. Він також приймає параметр keys, який наразі не використовується в цьому методі.

3. Метод crt.encrypt_message(new_text) використовується для шифрування нового тексту new_text.

Текст другого QLabel оновлюється з використанням методу setText, який відображає зашифрований текст у центрі навігаційної панелі.

Отже, цей клас NavBarView створює інтерфейс навігаційної панелі та надає можливість встановлення та оновлення тексту повідомлення, який автоматично шифрується та відображається у другому QLabel.

Файл ErrorHandler.py визначає клас WarningMessage, який є підкласом QMessageBox з бібліотеки PyQt5. Він використовується для створення сповіщень або вікон попередження з текстовим повідомленням та заголовком.

```
from PyQt5.QtWidgets import QPushButton,QMessageBox, QDialog, QVBoxLayout, QLabel, QLineEdit, QDialogButtonBox
class WarningMessage(QMessageBox):
    def __init__(self,title,text):
        super().__init__()
        self.setWindowTitle(title)
        self.setText(text)
        self.setStyleSheet("color: white; ")
        self.exec_()
```

Основне призначення даного класу полягає в наступному:

1. __init__ – конструктор класу приймає два параметри: title – відповідає за заголовок сповіщення; text - за текстове повідомлення.

2. Викликається конструктор базового класу QMessageBox і встановлюється:

- заголовок вікна за допомогою self.setWindowTitle(title);
- текст повідомлення з використанням self.setText(text);

– стиль для тексту повідомлення, де текст буде білого кольору: self.setStyleSheet("color: white; ").

3. Викликається метод self.exec_(), який відображає вікно попередження користувачу.

Отже, клас WarningMessage дозволяє створювати сповіщення з вказаним заголовком та текстовим повідомленням попереджень та інформацію про помилки та відображати їх на екрані користувача.

Файл LayerMainLayout.py містить два класи: BottomBorderLabel та LayerMainLayout, які використовуються для створення і відображення інтерфейсу користувача для кожного шару програми представлено в додатку Д.

Розглянемо більш детальний опис основних функцій файлу LayerMainLayout.py.

Клас BottomBorderLabel, який є субкласом QLabel, функціонує з метою генерації графічного компонента – мітки, доповненої оранжевою декоративною підкладкою в нижній частині. Основною особливістю цього класу є реалізація методу paintEvent, що дозволяє модифікувати графічне відображення мітки. В рамках цього методу відбувається створення об'єкта QPainter для виконання малювання, встановлення параметрів кольору та товщини лінії, після чого реалізується візуалізація оранжевої лінії на нижньому краю мітки. Для забезпечення відображення основного контенту мітки використовується метод базового класу.

LayerMainLayout, який успадковує властивості і методи від QVBoxLayout, призначений для створення та управління інтерфейсом кожного рівня в ієрархії вкладених вкладок. У конструкторі цього класу реалізовано наступні кроки: створення горизонтального макету hbox для розташування елементів інтерфейсу, формування міток з певним стилем, ініціація вертикальних макетів vvbox та vv2box для розміщення текстових областей, що відображають зашифровані та розшифровані дані відповідно, а також створення кнопок для активації процесів розшифрування та переходу

до наступного шару. Метод decrypt_config відповідає за процедуру розшифрування даних з подальшим їх відображенням, при цьому у випадку виявлення помилок активується WarningMessage. Метод setKeys служить для задання ключів кожного шару.

Загалом, клас LayerMainLayout є фундаментом для створення інтерактивного інтерфейсу, що забезпечує користувачам можливість введення, шифрування та розшифрування інформації на різних рівнях вкладених вкладок.

У файлі Navbar.py представлено два класи – NavBarButton і NavigationBar, що функціонують як основні компоненти для конструювання графічного інтерфейсу користувача, зокрема навігаційної панелі, в контексті програмного застосування (додаток Е).

1. Ініціалізація з параметрами. При створенні екземпляра класу, передається назва name та функція зворотного виклику callback, яка активується при взаємодії користувача з кнопкою.

2. Візуалізальна кастомізація. Кнопка має можливість зміни кольору при наведенні курсору, що реалізується через обробку відповідних подій миші.

3. Стилізація за допомогою стилів та палітри. Використання методів setObjectName, setAutoFillBackground, і setPalette дозволяє налаштувати зовнішній вигляд кнопки, використовуючи визначену палітру, наприклад DarkOrangePalette.

4. Інтеграція ефектів тіні. Застосування класу NavBarShadow з файлу ShadowEffect.py дозволяє створювати тіньові ефекти.

5. Обробка подій інтерфейсу. Використання перевизначеного методу event для керування взаємодіями користувача, зокрема наведенням та виходом курсору з області кнопки. Методи hoverEnterEvent та hoverLeaveEvent модифікують візуальні характеристики, як-от колір тіні, у відповідь на ці події.

Клас NavigationBar ідіграє ключову роль у створенні та управлінні навігаційною панеллю, що включає наступні аспекти:

1. Структура навігаційної панелі: Організація панелі з кнопками "Config", "Encrypted Message" та "Close".

2. Конструктор класу. Використання для створення кнопок "Config" та "Close", де "Config" відкриває конфігураційне вікно, а "Close" активує метод "Close" для закриття вікна..

3. Інтеграція додаткових компонентів. Включення об'єкта NavBarView відповідального за відображення зашифрованого повідомлення.

4. Розміщення компонентів у макеті. Використання для розташування елементів на навігаційній панелі з оптимальним візуальним розподілом.

Отже, клас NavigationBar реалізує комплексний механізм для створення функціональної та візуально привабливої навігаційної панелі, що включає інтерактивні кнопки для навігації в рамках програми, зокрема відкриття конфігураційних вікон, відображення інформації та закриття програми.

У файлі ShadowEffect.py визначається клас NavBarShadow, який є підкласом QGraphicsDropShadowEffect з бібліотеки PyQt5 (додаток Ж). Його основна функція - встановлення тіні для відображення під кнопками навігаційної панелі.

Клас NavBarShadow виконує роль у специфікації та адаптації візуальних ефектів тіні для елементів користувацького інтерфейсу, зокрема для кнопок у навігаційній панелі. Основні аспекти його реалізації включають:

- Ініціалізація через базовий клас. Конструктор класу ініціює об'єкт, спираючись на базовий клас QGraphicsDropShadowEffec. Це забезпечує інтеграцію стандартних функціональностей ефекту тіні у графічному інтерфейсі.
- Встановлення радіусу розмиття. За допомогою методу setBlurRadius, клас налаштовує радіус розмиття тіні, що дозволяє контролювати ступінь змазування та розпливчастості країв тіні. У даному контексті, радіус встановлено на значення 10, що вказує на певну м'якість країв тіні.

- Конфігурація кольору тіні. Використання методу setColor дозволяє налаштувати колір тіні. У цьому випадку обраний чорний колір (RGB значення 0, 0, 0) із ступенем прозорості 100, що забезпечує чіткість та визначеність тіні.
- Адаптивне зміщення тіні. Унікальна функція класу полягає у здатності змінювати параметри зміщення тіні на основі ідентифікатора (імені) кнопки. Наприклад, для кнопки з іменем "Close" тінь налаштовується так, щоб мати зміщення вліво і вниз на -2,2 пікселі, створюючи ефект глибини та візуальної відокремленості.

Таким чином, NavBarShadow репрезентує спеціалізований компонент інтерфейсу, який забезпечує додаткову гнучкість та естетичну варіативність при візуалізації елементів користувацького інтерфейсу, в даному випадку – кнопок навігаційної панелі. Через свої адаптивні характеристики, клас дозволяє індивідуалізувати візуальне представлення кожної кнопки, виходячи з її унікальної ролі та функціональності в інтерфейсі.

У файлі VerticalTabBar.py визначається два класи: TabBar та VerticalTabWidget, які використовуються для створення вертикальних вкладок у вашому інтерфейсі (додаток К).

TabBar є похідним класом від QTabBar, що застосовується для генерації вкладок у графічних користувацьких інтерфейсах. В його конструкторі не передбачено додаткових аргументів, замість цього відбувається інвокація конструктора суперкласу QTabBar.

Метод *tabSizeHint* у цьому класі модифіковано для адаптації розмірів вкладок, дозволяє динамічно збільшувати розміри вкладок удвічі, одночасно зберігаючи їх первісні пропорції.

Метод *paintEvent* реімплементовано для оптимізації візуалізації тексту на вкладках. Це досягається шляхом вертикального розміщення тексту, що забезпечує його читабельність та коректне відображення.

Клас *VerticalTabWidget*, що є нащадком *QTabWidget*, призначений для створення вертикально орієнтованих вкладок, розташованих з лівого боку

інтерфейсу. У його конструкторі викликається конструктор базового класу *QTabWidget*. Для задання відповідного розміщення вкладок використовується метод *setTabPosition* з параметром *QTabWidget.West*, який дозволяє розмістити вкладки ліворуч.

Таким чином, ці класи надають функціональність для створення вертикально розташованих вкладок із покращеною візуалізацією тексту та модифікованим розміром вкладок.

Для повідомлень різної довжини оцінено час шифрування та розшифрування на кожному з рівнів.

Очікувано, що із ростом розмірності повідомлення від 8 біт до 4096 біт час криптографічних перетворень збільшується приблизно в 100 разів (таблиця 6.2).

Таблиця 6.2

Час шифрування та розшифрування для повідомлень різної довжини на кожному з рівнів

Розмір тексту, біт	Час шифрування		Час розшифрування	
	I рівень (b_1–b_3)	II рівень (b_{11}–b_{33})	I рівень (b_1–b_3)	II рівень (b_{11}–b_{33})
8	7,799943E–06	4,300033E–06	2,659997E–05	1,419999E–05
16	1,279998E–05	7,799943E–06	5,199993E–05	2,020004E–05
32	1,630012E–05	1,369999E–05	4,129997E–05	2,959999E–05
64	2,010015E–05	1,389976E–05	5,489995E–05	4,520011E–05
128	2,790009E–05	2,320018E–05	1,246998E –04	9,370025E–05
256	5,609961E–05	6,760005E–05	6,430999E–04	2,128998E–04
512	1,146998E–04	9,070023E–05	5,221999E–04	4,703999E–04
1024	2,345002E–04	1,778999E–04	7,691999E–04	7,490998E–04
2048	4,413006E–04	3,627992E–04	2,385101E–03	2,479399E–03
4096	9,392999E–04	7,505005E–04	3,148402E–03	3,015401E–03

При однакових розрядностях час розшифровування більший, ніж час шифрування. Це пов’язано з використанням при розшифруванні обчислювально складних операцій КТЗ. Крім того, час криптографічних перетворень на другому рівні, на якому використовуються менші модулі, як правило, менший, ніж на першому.

6.3 Програмна реалізація криптоалгоритмів в поліноміальній системі залишкових класів.

Для реалізації проєкту обрано сучасний технологічний стек, який забезпечує високу продуктивність, зручність у розробці та підтримку кросплатформенності. За основу взято мову програмування C#, яка вирізняється своєю універсальністю та оптимальною продуктивністю для реалізації алгоритмів і логіки програмного забезпечення.

Проєкт базується на платформі .NET 7, яка забезпечує сучасні можливості для розробки додатків різного типу, включно з підтримкою кросплатформенності, що мінімізує залежність від операційних систем і сприяє зниженню витрат на адаптацію коду.

Для розробки графічного інтерфейсу використано Windows Presentation Foundation (WPF), що дозволяє створювати багатофункціональні та естетично привабливі віконні додатки. Завдяки застосуванню мови розмітки XAML, реалізується чіткий поділ між логікою програми та її інтерфейсною частиною, що сприяє спрощенню розробки та подальшої підтримки інтерфейсу.

Архітектурно проєкт реалізовано із застосуванням патерну Model-View-ViewModel (MVVM), який забезпечує чітке розділення між даними, бізнес-логікою та відображенням інформації. Компонент Model відповідає за обробку даних і реалізацію основної функціональності, View забезпечує відображення даних користувачеві, а ViewModel виступає проміжною ланкою, яка пов'язує модель із представленням і забезпечує їхню взаємодію. Така структура сприяє

підвищенню тестованості коду, його масштабованості та спрощенню додавання нової функціональності.

Обраний стек технологій і використаний патерн проєктування забезпечують створення ефективного, надійного та легко розширюваного програмного забезпечення. Такий підхід дозволяє оптимізувати процес розробки, підтримувати актуальність продукту та адаптувати його до змінних потреб користувачів [330].

6.3.1 Опис структурування кореневої директорії проєкту.

Для реалізації симетричного алгоритму шифрування в ІПСЗК була запропонована структура директорій, яка організована таким чином, щоб забезпечити ефективне управління кодом та спростити процес масштабування і підтримки програми. Головною директорією є PolynomialCypher, яка містить основні файли та піддиректорії, що реалізують основну функціональність. Всі компоненти проєкту організовані за принципом модульності, що дає змогу легко додавати нові функціональні можливості та зручніше працювати з кодом.

Директорія Commands відповідає за реалізацію командної логіки та взаємодії користувача з додатком, зокрема через створення динамічних команд та команд управління вікнами. Вона включає класи, які забезпечують інтерактивність, такі як DelegateCommand.cs, що дозволяє створювати команди за допомогою лямбда-виразів, і WindowControlCommands.cs, який керує діями вікон, такими як їх закриття чи згортання.

Components містить власні (кастомні) компоненти інтерфейсу користувача, серед яких важливу роль відіграють елементи, що надають зручність взаємодії, такі як заголовок вікна, реалізований у файлах Header.xaml/Header.xaml.cs.

Converters надає можливість перетворювати дані між різними форматами, забезпечуючи коректне відображення даних в інтерфейсі без необхідності змінювати бізнес-логіку. Ці класи виконують перетворення

чисел у текстові значення та допомагають адаптувати відображення даних у різних контекстах, зокрема для математичних виразів.

Основна бізнес-логіка програми реалізована в директорії Models, яка включає алгоритми шифрування та розшифрування, математичні моделі, а також параметри, необхідні для роботи алгоритмів. Це забезпечує високий рівень абстракції та дозволяє зручно працювати з даними на рівні алгоритмів та математичних операцій. В окремих піддиректоріях зберігаються класи для роботи з поліномами, параметрами алгоритмів та текстовими даними, що полегшує реалізацію логіки трансформацій.

Services надає додаткові сервіси для взаємодії з користувачем, зокрема через діалогові вікна, що дають змогу користувачеві отримувати зворотний зв'язок, переглядати результати шифрування та розшифрування, а також налаштовувати параметри алгоритмів.

Моделі представлення (ViewModels) забезпечують зв’язок між бізнес-логікою та графічним інтерфейсом програми. Вони містять усю необхідну логіку для обробки введених користувачем даних та взаємодії з інтерфейсними компонентами. Це дозволяє зберігати чіткий поділ між обробкою даних та їх відображенням, що робить програму зручною для розширення.

Директорія Views включає XAML файли, що реалізують графічний інтерфейс користувача, а також відповідні C#-класи для керування подіями інтерфейсу. Вона відповідає за створення усіх вікон програми, від головного до спеціалізованих діалогових вікон, що дозволяє користувачеві легко взаємодіяти з програмою.

Ресурси програми, такі як стилі та шаблони інтерфейсу, визначаються в App.xaml, тоді як App.xaml.cs містить код ініціалізації програми, включаючи запуск головного вікна. Така організація проекту забезпечує чітке розмежування відповідальностей, дозволяючи легко розширювати, підтримувати та модифікувати програму в майбутньому.

На рисунку 6.32 представлена структура директорій програмної реалізації симетричного методу шифрування в ІПСЗК.

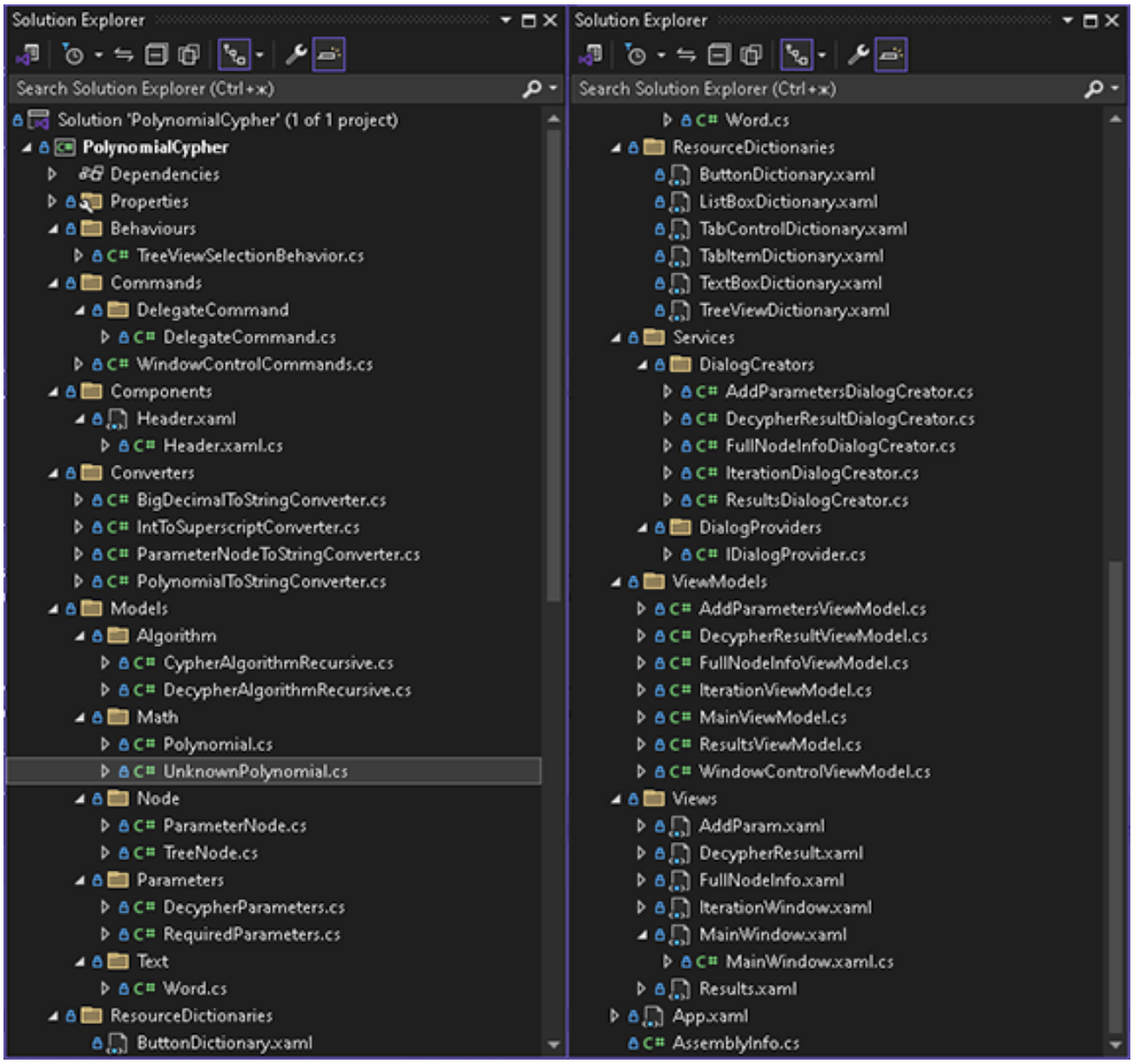


Рисунок 6.32 – Структура директорій програмної реалізації

6.3.2 Детальний опис реалізованих математичних та допоміжних класів в криптоалгоритмах в ІПСЗК

Представлена на рисунку 6.33 UML-діаграма показує взаємозв’язки між класами та методами у програмному забезпеченні, орієнтованому на роботу з поліномами, шифруванням та параметрами вузлів.

Дана UML-діаграма демонструє архітектуру програмного забезпечення для реалізації алгоритмів роботи з математичними об’єктами та параметрами в умовах ітеративних або шифрувальних обчислень. Вона є основою для створення ефективного інструменту з підтримкою багаторівневого управління вузлами, що важливо у проєктах з криптографії, обчислювальної математики чи симуляції складних моделей.

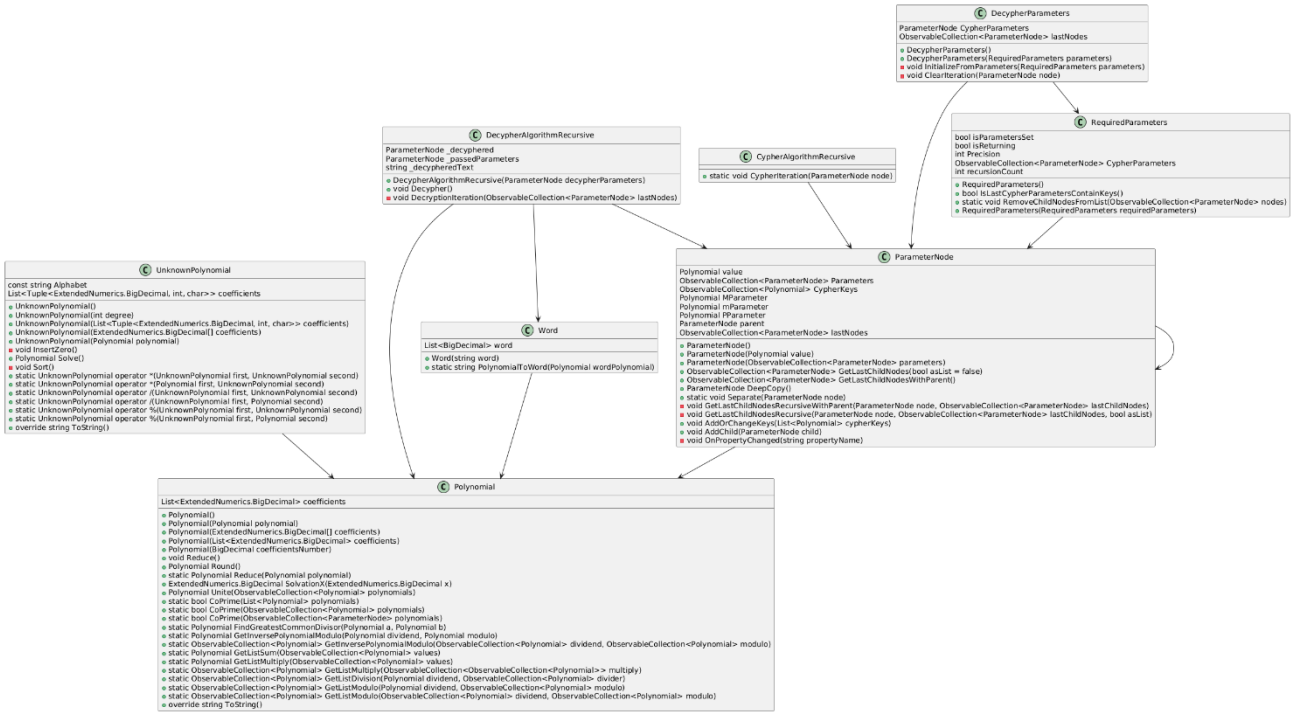


Рисунок 6.33 – UML-діаграма класів, необхідних для опрацювання алгоритму

На рисунку 6.34 представлено клас `Polynomial, який` забезпечує багатофункціональну основу для виконання операцій із поліномами, включаючи основні математичні дії, перевірку властивостей, а також обчислення обернених поліномів та залишків від ділення. Це важливо для реалізації обчислювальних алгоритмів у наукових дослідженнях, шифруванні та математичному моделюванні.

Клас Polynomial представляє математичний поліном і забезпечує широкий набір методів для виконання операцій із ним. Для обчислень із великими числами використовується бібліотека ExtendedNumerics.

1. Основні можливості класу

Клас дозволяє виконувати базові та розширені математичні операції над поліномами, зокрема додавання, множення, ділення, обчислення значення для заданого аргументу *x*, а також визначення найбільшого спільного дільника.

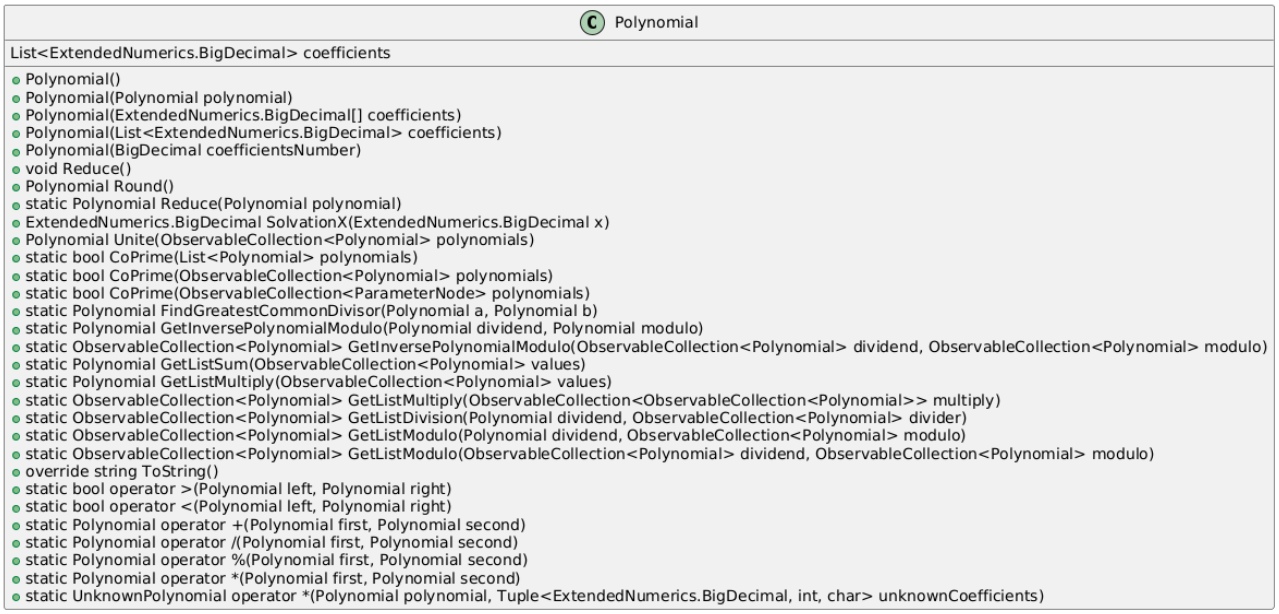


Рисунок 6.34 – UML-діаграма класу Polynomial

2. Конструктори

Polynomial надає кілька способів ініціалізації:

— Polynomial() - створює порожній об’єкт.

— Polynomial(Polynomial polynomial) - копіює існуючий поліном.

— Polynomial(ExtendedNumerics.BigDecimal[] coefficients) - ініціалізує поліном масивом коефіцієнтів.

— Polynomial(List<ExtendedNumerics.BigDecimal> coefficients) - приймає список коефіцієнтів для створення екземпляра.

— Polynomial(BigDecimal coefficientsNumber) - перетворює число на набір коефіцієнтів, формуючи поліном.

3. Методи для реалізації криптоалгоритмів у ПСЗК.

Обробка коефіцієнтів

- Reduce() - видаляє нульові коефіцієнти з кінця масиву.
- Round() — округлює всі коефіцієнти.

- static Reduce(Polynomial polynomial) - статичний метод для очищення від зайвих нульових коефіцієнтів.

Операції над поліномами

- SolvationX(BigDecimal x) - обчислює значення полінома при заданому *x*.
- Unite(ObservableCollection polynomials) - об’єднує кілька поліномів в один.
- static CoPrime(List polynomials) - перевіряє, чи є поліноми взаємно простими.
- static CoPrime(ObservableCollection polynomials) - аналогічно перевіряє взаємну простоту, але для всіх поліномів.
- static FindGreatestCommonDivisor(Polynomial a, Polynomial b) - знаходить найбільший спільний дільник двох поліномів.
- static GetInversePolynomialModulo(Polynomial dividend, Polynomial modulo) - визначає обернений поліном за модулем.
- static GetInversePolynomialModulo(ObservableCollection dividend, ObservableCollection modulo) - виконує аналогічну операцію для всіх поліномів.

4. Робота з колекціями

- static GetListSum(ObservableCollection values) - обчислює суму поліномів у колекції.
- static GetListMultiply(ObservableCollection values) - обчислює добуток колекції поліномів.
- static GetListMultiply(ObservableCollection<ObservableCollection> multiply) - множить кілька колекцій між собою.
- static GetListDivision(Polynomial dividend, ObservableCollection divider) - ділить поліном на всі елементи колекції.
- static GetListModulo(Polynomial dividend, ObservableCollection modulo) - визначає залишок від ділення полінома на кожен елемент колекції.

- static GetListModulo(ObservableCollection dividend, ObservableCollection modulo) - обчислює залишок від ділення колекції поліномів на іншу колекцію.

5. Перевантажені оператори

Для спрощення роботи з поліномами клас забезпечує такі оператори:

- > *i* < — порівняння двох поліномів за заданим критерієм.
- + — додавання.
- / — ділення.
- % — залишок від ділення.
- *** ** — множення.
- *** (*Tuple<BigDecimal, int, char>*)** — множення на невідомі коефіцієнти.

6. Перетворення в рядок

- ToString() — повертає текстове представлення полінома, зручне для відображення в інтерфейсі або логуванні.

Завдяки реалізованим методам і перевантаженим операторам клас Polynomial забезпечує високу гнучкість у роботі з математичними виразами.

Клас `UnknownPolynomial` реалізує функціональність для роботи з поліномами, у яких деякі або всі коефіцієнти невідомі. Цей клас надає інструменти для виконання основних операцій з поліномами, таких як множення, ділення, обчислення залишку та вирішення рівнянь (рисунок 6.35). Для забезпечення обчислень з високою точністю використовується бібліотека `ExtendedNumerics`.

Основна функціональність класу `UnknownPolynomial`

1. Конструктори

UnknownPolynomial() - ініціалізує порожній поліном.

UnknownPolynomial(int degree) - створює поліном заданого ступеня, де всі коефіцієнти мають значення 1.

UnknownPolynomial(List<Tuple<ExtendedNumerics.BigDecimal, int, char>> coefficients) - ініціалізує поліном зі списку коефіцієнтів, де кожен

коефіцієнт визначений як кортеж значення (BigDecimal), ступеня змінної та мітки змінної.

UnknownPolynomial(ExtendedNumerics.BigDecimal[] coefficients) - ініціалізує поліном з масиву коефіцієнтів, де індекси відповідають степеням змінної.

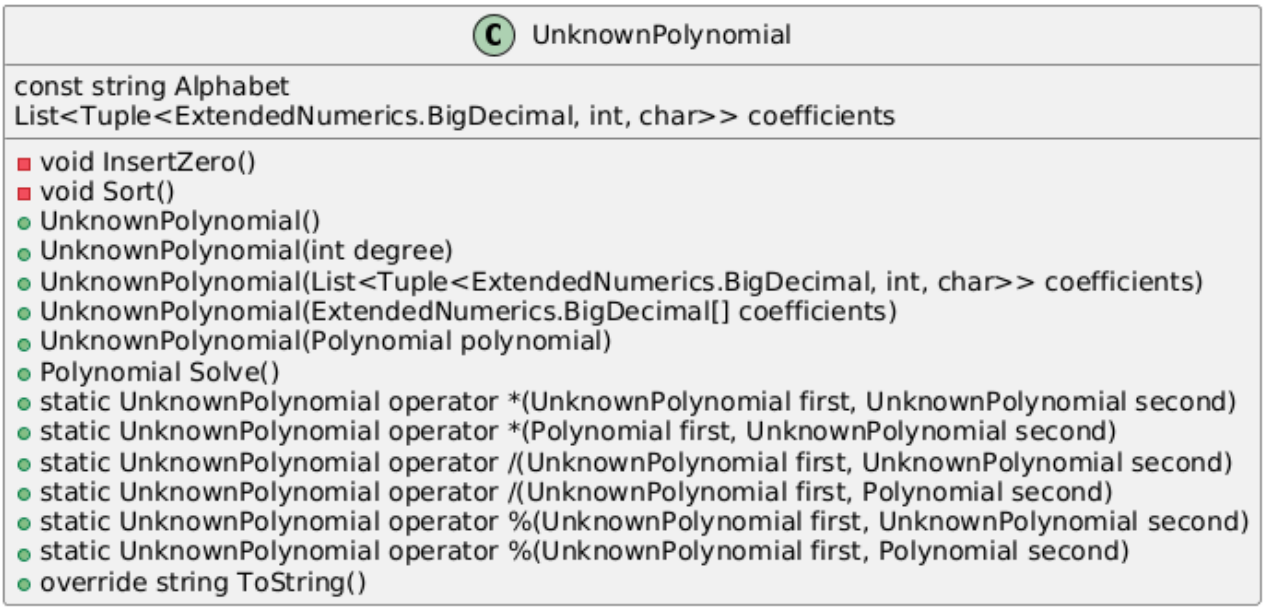


Рисунок 6.35 – UML-діаграма класу *UnknownPolynomial*

UnknownPolynomial(Polynomial polynomial) - ініціалізує об’єкт класу UnknownPolynomial на основі вже існуючого полінома (Polynomial).

2. Приватні методи

InsertZero() - вставляє коефіцієнти зі значенням 0 для всіх відсутніх степенів у поліномі, забезпечуючи коректність подання.

Sort() - забезпечує сортування коефіцієнтів полінома за зростанням степенів змінних для стандартизації внутрішнього представлення.

3. Публічні методи

Polynomial Solve() - реалізує метод невизначених коефіцієнтів для пошуку оберненого полінома шляхом обчислення їх можливих значень або еквівалентного представлення.

override string ToString() - формує текстове представлення полінома у вигляді математичного виразу з невідомими коефіцієнтами.

4. Реалізовані оператори

static UnknownPolynomial operator (UnknownPolynomial first, UnknownPolynomial second) - виконує множення двох поліномів із невідомими коефіцієнтами;

static UnknownPolynomial operator (Polynomial first, UnknownPolynomial second) - реалізує множення звичайного полінома (Polynomial) на поліном із невідомими коефіцієнтами.

static UnknownPolynomial operator /(UnknownPolynomial first, UnknownPolynomial second) - реалізує ділення одного полінома з невідомими коефіцієнтами на інший;

static UnknownPolynomial operator /(UnknownPolynomial first, Polynomial second) - виконує ділення полінома з невідомими коефіцієнтами на звичайний поліном.

static UnknownPolynomial operator %(UnknownPolynomial first, UnknownPolynomial second) - обчислює залишок від ділення одного полінома з невідомими коефіцієнтами на інший;

static UnknownPolynomial operator %(UnknownPolynomial first, Polynomial second) - визначає залишок від ділення полінома з невідомими коефіцієнтами на звичайний поліном.

Клас забезпечує гнучке представлення поліномів із підтримкою невідомих змінних, дозволяючи працювати з поліномами довільного ступеня та конфігурації коефіцієнтів. Оператори та методи класу оптимізовані для роботи з великими числами завдяки бібліотеці *ExtendedNumerics*.

Клас `ParameterNode` представляє структурний елемент, який використовується в алгоритмах шифрування та розшифрування (рисунок 6.36). Він забезпечує зберігання та обробку параметрів, таких як значення полінома, шифрувальні ключі та інші властивості. `ParameterNode` також

надає методи для управління деревоподібною структурою вузлів, включаючи додавання дочірніх елементів, отримання дочірніх вузлів і клонування об'єкта.

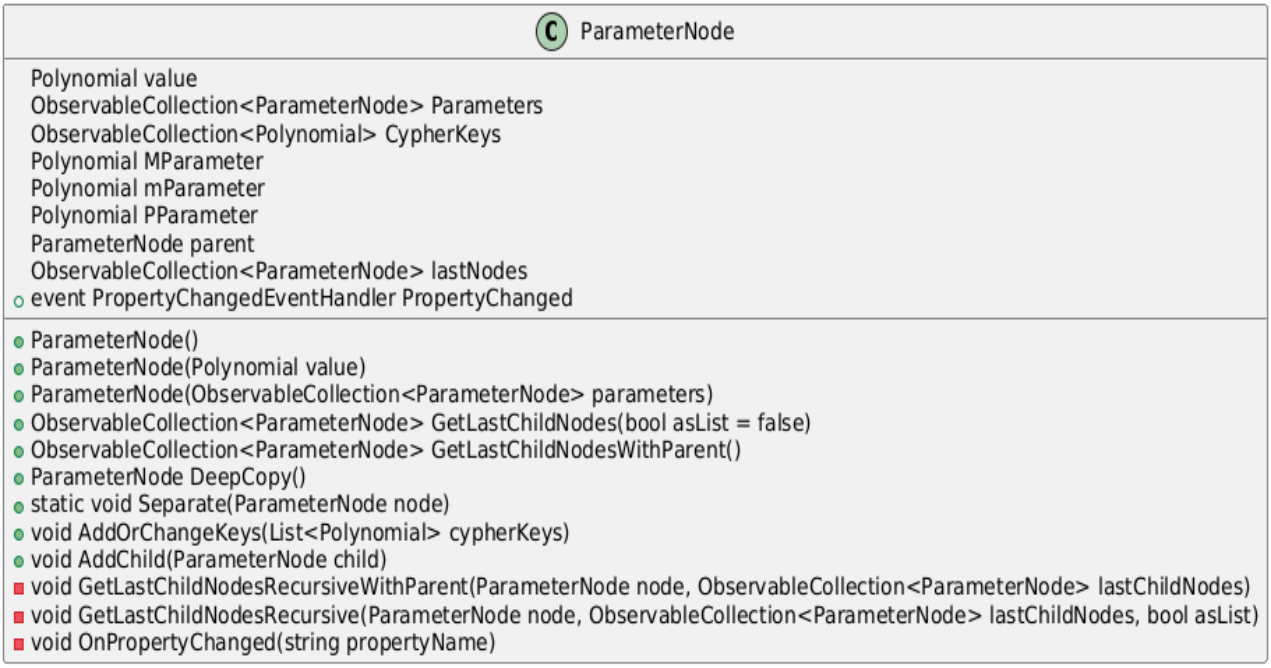


Рисунок 6.36 - UML-діаграма класу ParameterNode

1. Конструктори:

ParameterNode() – ініціалізує порожній вузол параметрів із типовими значеннями;

ParameterNode(Polynomial value) - створює вузол параметрів із заданим значенням полінома (Polynomial);

ParameterNode(ObservableCollection parameters) - ініціалізує вузол параметрів на основі переданої колекції параметрів (ObservableCollection).

2. Методи:

ObservableCollection GetLastChildNodes(bool asList = false) - повертає колекцію останніх дочірніх вузлів. Якщо параметр asList встановлено в true, результати повертаються як список;

ObservableCollection GetLastChildNodesWithParent() - повертає останні дочірні вузли разом із їхніми батьківськими вузлами, забезпечуючи повний контекст ієрархії;

ParameterNode DeepCopy() – створює глибоку копію вузла параметрів, включаючи всі дочірні елементи;

`static void Separate(ParameterNode node)` - статичний метод, який розділяє вхідний вузол `ParameterNode` на кілька вузлів на основі внутрішньої логіки (наприклад, розподілу параметрів);

void AddOrChangeKeys(List cypherKeys) – додає нові ключі шифрування до вузла або змінює існуючі. Використовується для динамічної зміни шифрувальних параметрів;

void AddChild(ParameterNode child) – додає новий дочірній вузол `child` до поточного вузла.

event PropertyChangedEventHandler PropertyChanged – подія, яка викликається при зміні будь-якої властивості вузла. Відповідає стандартам інтерфейсу *INotifyPropertyChanged*.

private void OnPropertyChanged(string propertyName) - викликає подію *PropertyChanged* для повідомлення про зміну значення властивості.

Клас *ParameterNode* є універсальним інструментом для роботи з параметрами в алгоритмах шифрування. Його підтримка деревоподібної структури та подій зміни властивостей дозволяє гнучко інтегрувати вузли параметрів у різноманітні системи. Особливістю реалізації є можливість роботи з колекціями параметрів і шифрувальними ключами, що робить його придатним для складних криптографічних задач. Крім того використання *ObservableCollection* забезпечує зручне відстеження змін у колекціях, а інтеграція з *INotifyPropertyChanged* динамічне оновлення властивостей вузла в UI або інших зв'язаних компонентах.

Клас *CypherAlgorithmRecursive* реалізує рекурсивний алгоритм шифрування (рисунок 6.37). Він містить метод для виконання однієї ітерації шифрування, яка створює нові вузли параметрів на основі заданого вузла.

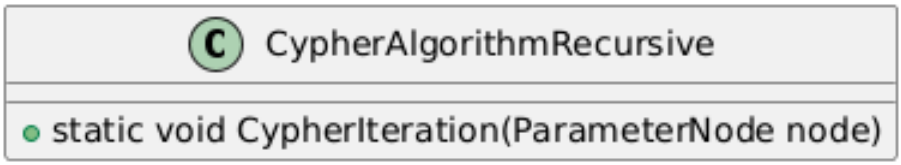


Рисунок 6.37 - UML-діаграма класу CypherAlgorithmRecursive

Метод `CypherIteration(ParameterNode node)` є статичним і реалізує одну ітерацію шифрування для заданого вузла параметрів. У рамках виконання цього методу здійснюється обчислення криптографічних параметрів `PParameter`, `MParameter` та `mParameter` для кожного дочірнього вузла. Отримані результати додаються до структури поточного вузла, розширюючи його ієрархію новими елементами.

Клас *DecypherAlgorithmRecursive* реалізує рекурсивний алгоритм розшифрування (рисунок 6.38). Він містить методи для виконання розшифрування, обчислення значення полінома та обробки вузлів параметрів.

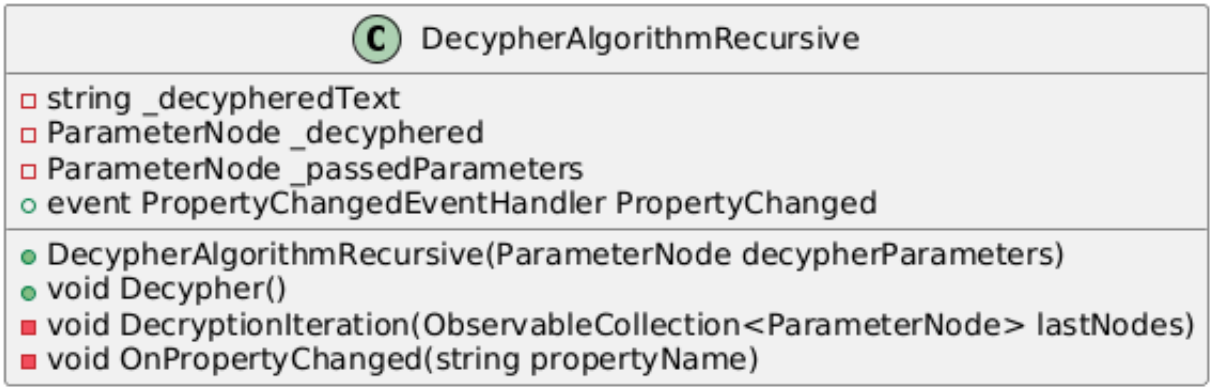


Рисунок 6.38 - UML-діаграма класу DecypherAlgorithmRecursive

Методи класу `DecypherAlgorithmRecursive`:

DecypherAlgorithmRecursive(ParameterNode decypherParameters) – конструктор, що ініціалізує алгоритм розшифрування з використанням переданих параметрів, представлених у вигляді об'єкта ParameterNode;

void Decypher() - метод, який реалізує виконання процесу розшифрування шляхом послідовної обробки параметрів;

private void DecryptionIteration(ObservableCollection lastNodes) – приватний метод, що виконує одну ітерацію розшифрування для заданої колекції вузлів *lastNodes*;

event PropertyChangedEventHandler PropertyChanged – подія, яка виникає при зміні властивостей об'єкта, забезпечуючи підтримку інтерфейсу *INotifyPropertyChanged* для сповіщення про зміни;

private void OnPropertyChanged(string propertyName) - приватний метод, який викликає подію *PropertyChanged* для інформування про зміну конкретної властивості, ідентифікованої через її ім'я.

Клас *RequiredParameters* визначає набір ключових параметрів, необхідних для виконання процесів шифрування та розшифрування (рисунок 6.39). Він включає властивості для зберігання характеристик, таких як точність обчислень, налаштування шифрування та інші пов’язані параметри. Крім того, клас надає методи для управління та маніпуляції цими параметрами, забезпечуючи їх інтеграцію у криптографічні алгоритми.

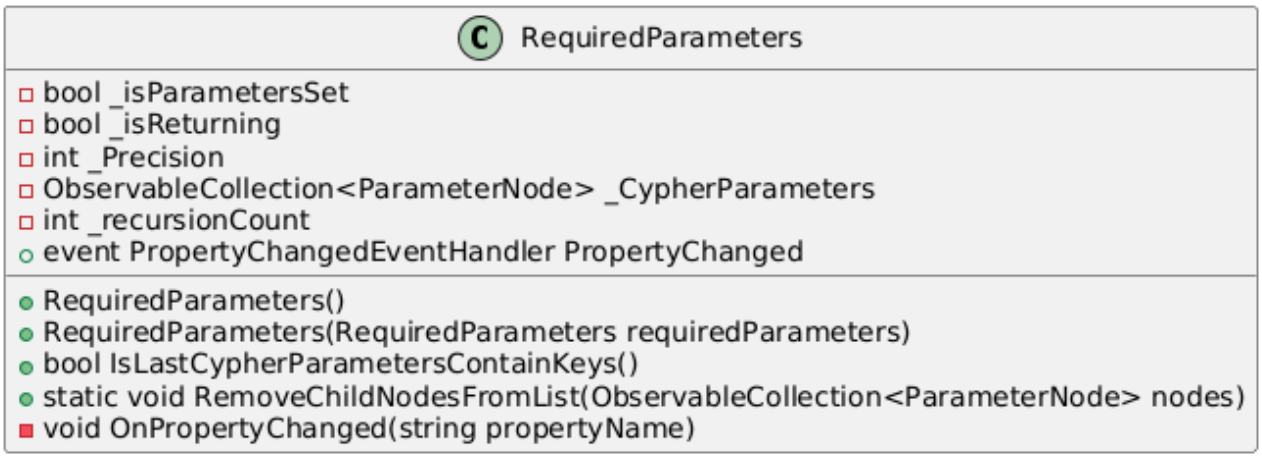


Рисунок 6.39 - UML-діаграма класу RequiredParameters

Методи класу *RequiredParameters*:

1. Конструктор *RequiredParameters()* - ініціалізує об'єкт класу з початковими значеннями параметрів. Використовується для створення екземпляра з дефолтними налаштуваннями, необхідними для подальшої роботи.

2. Конструктор *RequiredParameters(RequiredParameters requiredParameters)* – ініціалізує новий об'єкт на основі іншого екземпляра класу RequiredParameters. Це дозволяє створювати копії параметрів або передавати їх між компонентами системи.

3. Метод *IsLastCypherParametersContainKeys()* – виконує перевірку, чи містять останні встановлені параметри шифрування необхідні ключі. Метод забезпечує контроль над коректністю параметрів, необхідних для виконання шифрувальних операцій.

4. Статичний метод *RemoveChildNodesFromList(ObservableCollection nodes)* – видаляє дочірні вузли з переданої колекції вузлів. Метод використовується для очищення ієрархічної структури параметрів, зберігаючи лише необхідні елементи.

5. Подія *PropertyChanged* - викликається при зміні властивостей об'єкта. Ця подія дозволяє системі відслідковувати зміни параметрів і динамічно оновлювати пов'язані компоненти.

6. Метод *OnPropertyChanged(string propertyName)* - викликає подію PropertyChanged, вказуючи на зміну конкретної властивості за її іменем. Цей метод використовується для забезпечення узгодженості стану об'єкта з іншими частинами системи.

Клас *DecypherParameters* представляє параметри для розшифрування. Він містить властивості для зберігання параметрів шифрування та методи для ініціалізації та очищення вузлів параметрів (рисунок 6.40).

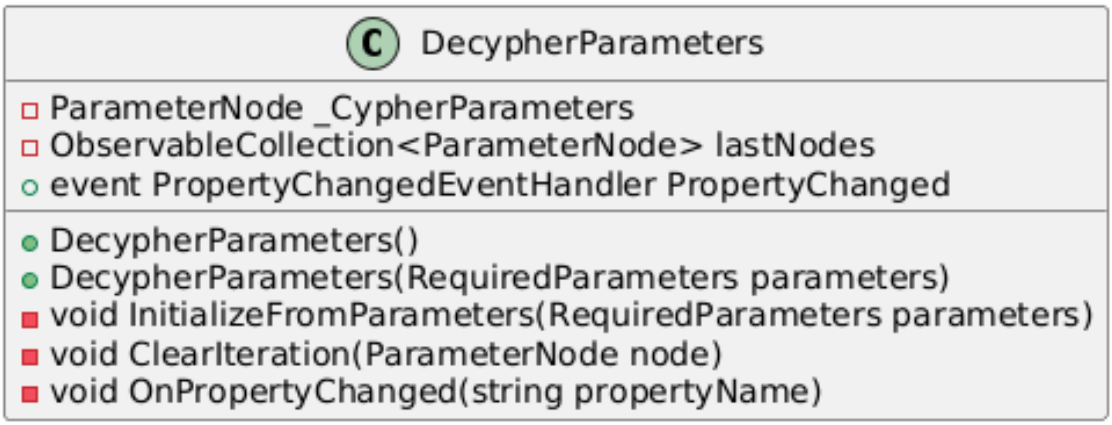


Рисунок 6.40 - UML-діаграма класу DecypherParameters

Методи класу *DecypherParameters*:

1. Конструктор *DecypherParameters()* - ініціалізує об'єкт класу з початковими значеннями параметрів, необхідних для виконання процесу розшифрування.

2. Конструктор *DecypherParameters(RequiredParameters parameters)* – створює екземпляр класу на основі об'єкта *RequiredParameters*. Використовується для ініціалізації параметрів розшифрування на базі існуючих налаштувань.

3. Метод *InitializeFromParameters (RequiredParameters parameters)* – здійснює ініціалізацію параметрів розшифрування шляхом їх копіювання або адаптації з об'єкта *RequiredParameters*. Забезпечує узгодженість параметрів між об'єктами.

4. Метод *ClearIteration(ParameterNode node)* – виконує очищення вузлів параметрів. Застосовується для видалення зайвих або неактуальних даних із дерева параметрів, що використовуються в процесі розшифрування.

5. Подія *PropertyChanged* - викликається при зміні властивостей об'єкта. Забезпечує інтеграцію з інтерфейсом *INotifyPropertyChanged*, дозволяючи відслідковувати зміни параметрів у режимі реального часу.

6. Метод *OnPropertyChanged(string propertyName)* – викликає подію *PropertyChanged* для повідомлення про зміну конкретної властивості об'єкта. Сприяє підтримці узгодженого стану об'єкта та інтеграції з іншими компонентами системи.

Клас *Word* представлений на рисунку 6.41 призначений для відображення слова як набору чисел типу *BigDecimal*. Він забезпечує функціональність для перетворення текстового рядка в поліном і зворотного перетворення полінома в текстовий рядок.

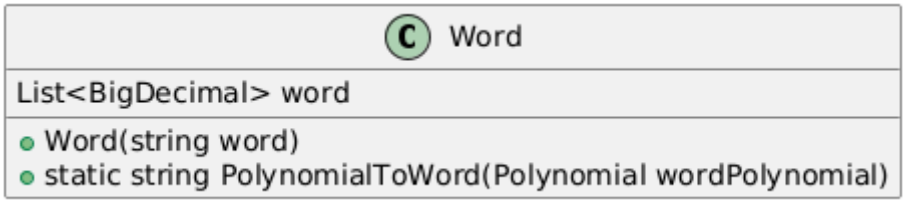


Рисунок 6.41 - UML-діаграма класу Word

Складовими методів класу `Word` є:

1. Конструктор *Word(string word)* – ініціалізує екземпляр класу *Word* на основі переданого текстового рядка. У процесі ініціалізації

кожен символ рядка перетворюється в числове значення типу *BigDecimal*, створюючи відповідний список чисел.

2. Статичний метод *PolynomialToWord(Polynomial wordPolynomial)* – виконує зворотне перетворення полінома в текстовий рядок. У цьому методі кожен коефіцієнт полінома інтерпретується як числове представлення символа, що дозволяє відновити вихідне слово.

Слід зазначити, що клас *Word* дозволяє легко переходити між текстовими даними та їх числовими представленнями у вигляді поліномів, що є корисним у контексті криптографічних та аналітичних задач. На рисунку 6.42 представлено діаграму активності, яка ілюструє послідовність дій користувача та програмного забезпечення під час виконання основних операцій. Вона відображає ключові етапи, зокрема ініціалізацію параметрів, виконання алгоритмів, обробку даних і завершення роботи.

Діаграма активності дозволяє формалізувати опис процесів у програмі, сприяючи розробці, аналізу й оптимізації її функціональності. Вона є важливим елементом моделювання в контексті сучасного розроблення програмних систем.

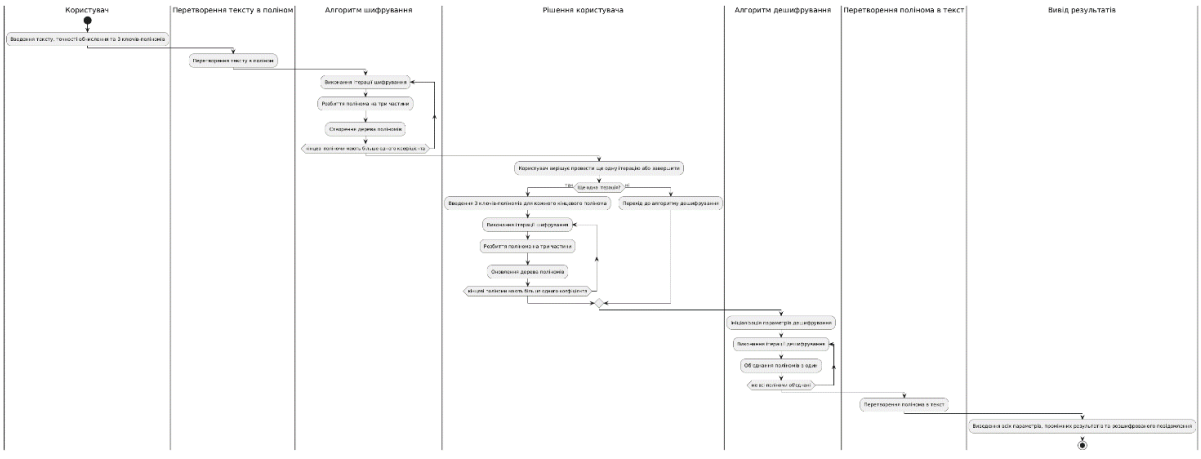


Рисунок 6.42 – Діаграма активності протягом роботи з програмою.

На рисунку 6.43 зображено стартове вікно програми, що реалізує алгоритм симетричного шифрування даних у ПСЗК. Основні елементи інтерфейсу:

1. Поле вводу тексту: розташоване у верхній частині вікна призначене для введення тексту, який буде перетворений у поліном. У прикладі введений текст "HPSM".

2. Поле налаштування точності (*Prec*): знаходиться праворуч від текстового поля призначене для встановлення кількості цифр точності при роботі з поліномами та великими числами. У прикладі значення точності становить 100.

3. Область відображення результатів: центральна частина вікна відведена для відображення поліномів-модулів, з допомогою яких проводиться шифрування в ПЗСК.

4. Кнопка "*Add/Change Parameters*": розташована внизу вікна призначена для додавання або редагування параметрів шифрування, таких як ключі або додаткові налаштування.

5. Кнопка "*Confirm*": знаходиться в нижньому правому куті використовується для підтвердження введених даних і запуску процесу шифрування.

Отже, це вікно дозволяє користувачеві налаштувати основні параметри шифрування, ввести текстові дані, перетворити їх у поліноми та розпочати процес шифрування в ІПСЗК. Інтуїтивно зрозумілий інтерфейс забезпечує зручність використання та дозволяє гнучко налаштовувати параметри відповідно до потреб користувача.

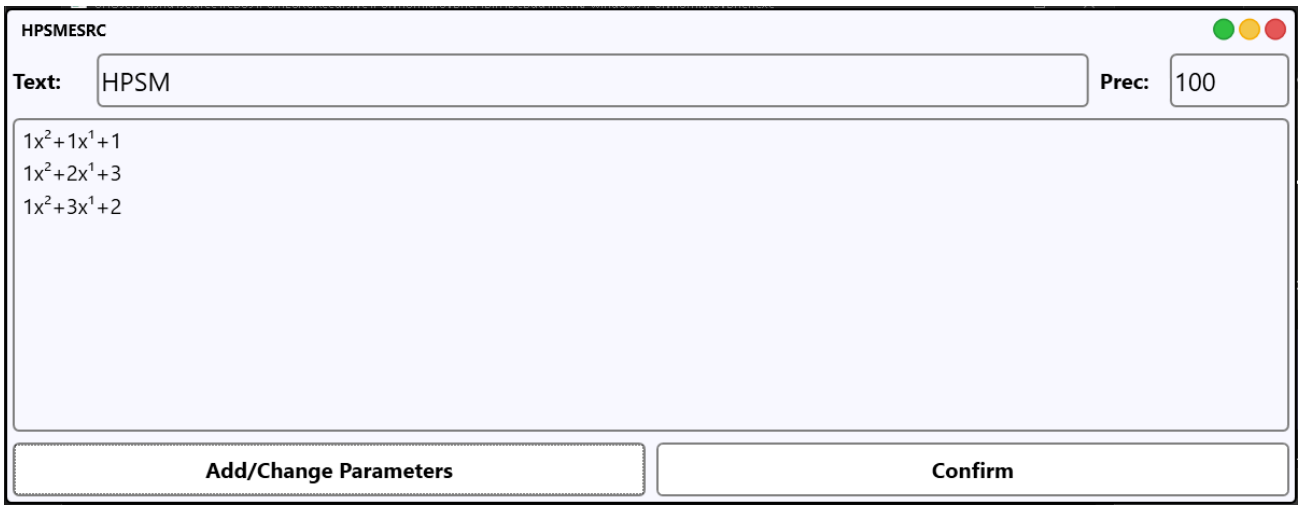


Рисунок 6.43 – Стартове вікно програми симетричного шифрування в ІПСЗК.

На рисунку 6.44 представлено інтерфейс вікна для введення модулів-ключів, які використовуються у процесі шифрування та розшифрування даних.

Основні елементи інтерфейсу:

1. Поля введення модулів: у вікні присутні три текстові поля, кожне з яких позначене як p(1), p(2) та p(3). Причому вводяться коефіцієнти поліномів-модулів починаючи з більшого показника степеня і закінчуючи нульовим. Для прикладу p(1) призначене для введення першого модуля з коефіцієнтами 1 1 1, що відповідає поліному x^2+x+1.

2. Кнопка "Add": розташована внизу вікна використовується для підтвердження введених модулів-ключів і додавання їх до системи.

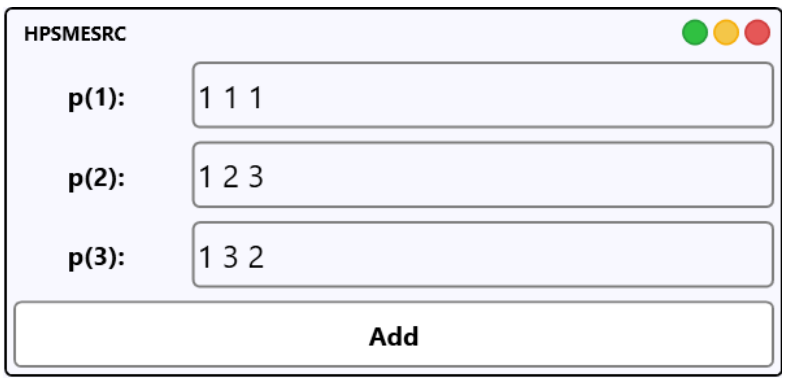


Рисунок 6.44 – Вікно додавання модулів-ключів у ІПСЗК.

Вікно забезпечує користувача можливістю задавати модулі-ключі, які є основними параметрами шифрування в ІПСЗК. Вони визначають структуру поліномів, що забезпечує криптографічну стійкість алгоритму.

Функціональні особливості:

- забезпечується зручний і зрозумілий інтерфейс для введення множин модулів;
- підтримується інтерактивний режим роботи, дозволяючи вносити зміни до набору параметрів;
- полегшується управління ключовими даними в межах криптографічної системи.

Процес функціонування програми розпочинається з введення користувачем необхідних параметрів. На початковому етапі задається текст для шифрування, рівень обчислювальної точності та три ключові значення коефіцієнтів поліномів-модулів. Ці вхідні дані слугують базисом для подальших криптографічних операцій.

Після отримання параметрів текст трансформується у відповідний поліном. Конвертація здійснюється за допомогою класу *Word*, який перетворює кожен символ тексту в числовий еквівалент, що виступає у ролі коефіцієнта полінома за принципом - перша літера в числовій формі відповідає коефіцієнту при найбільшому степені поліному n, відповідно друга – n-1 і т.д.. Сформований поліном стає основою для запуску процедури шифрування згідно запропоновнаного симетричного алгоритму в ПЗСК. На рисунку 6.45 представлено вікно програми обчислення значення $b(x)$ на другому ієрархічному рівні.

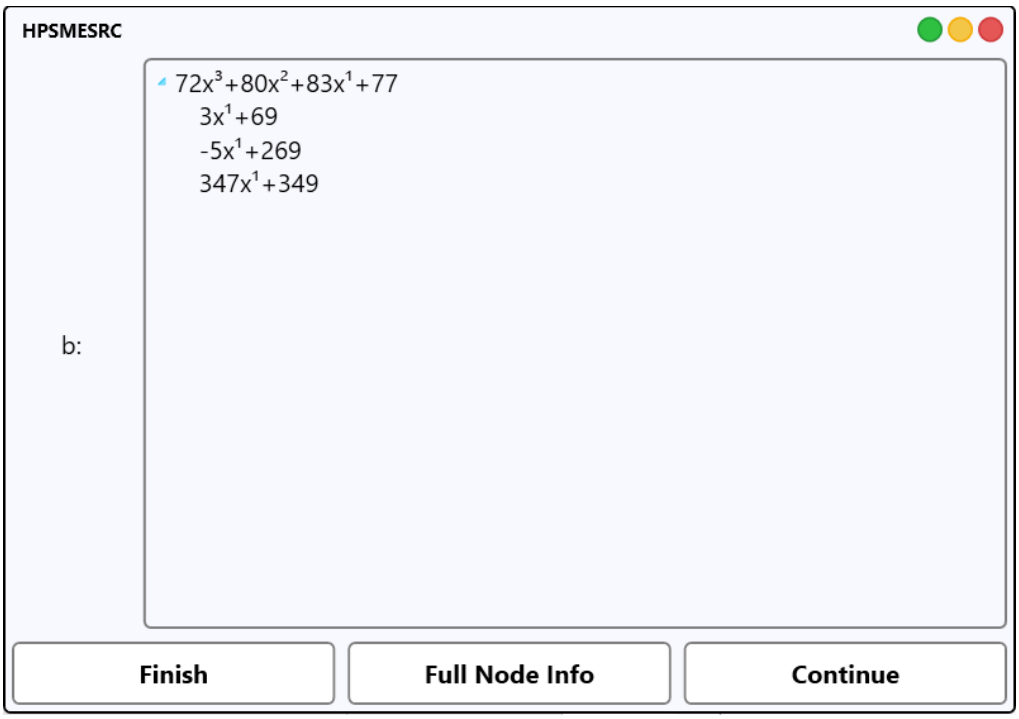


Рисунок 6.45 – вікно результатів ітерації

Рисунок 6.46 представляє інтерфейс програмного забезпечення на другому ієрархічному рівні для реалізації методу шифрування в ІПСЗК у процесі багатокрокових розрахунків.

1. Верхнє текстове поле ("*N*:") містить математичний вираз (вхідний текст для шифрування), який є початковою умовою, у даному випадку вказано поліном $N(x)=72x^3+80x^2+83x+7772$.

- Нижнє текстове поле ("*Last*:") - відображає результати обчислень на попередньому етапі або останньої ітерації. Тут наведено список поліномів $b_i(x)$, які можуть бути проміжними результатами:$3x+69$, $-5x+269$, $347x+349$.

2. Функціональні кнопки:

- "*Add/Change Parameters*" – дозволяє змінювати або додавати параметри для наступної ітерації, що є ключовим для налаштування параметрів моделювання.

- "*Full Node Info*" – забезпечує доступ до повної інформації про вузол, на якому здійснюються обчислення. Ця функція корисна для деталізованого аналізу процесу.

- "Next iteration" – ініціює перехід до наступного кроку ітераційного обчислення, що дозволяє реалізувати алгоритми поступового наближення до розв’язку.

Рисунок 6.46 – Вікно переходу до наступної ітерації

Процес шифрування виконується ітераційно. На кожному етапі вхідний поліном $N(x)$ поділяється на три частини за допомогою ключів-поліномів $p_i(x)$. Для кожного фрагмента відбувається обчислення нових значень основних параметрів $P_k(x)$, $m_{ik}(x)$, $M_{ik}(x)$, $b_{ik}(x)$, де k – ієрархічний рівень (рисунок 6.47), які формують вузли дерева. Ітерації тривають, поки кінцеві поліноми у структурі мають більше двох коефіцієнтів.

Після завершення кожної ітерації користувач обирає, чи продовжувати шифрування, чи завершити процес. У разі продовження вводяться три ключі-поліноми для кожного з кінцевих поліномів, і процедура повторюється. Якщо приймається рішення про завершення, система переходить до розшифрування.

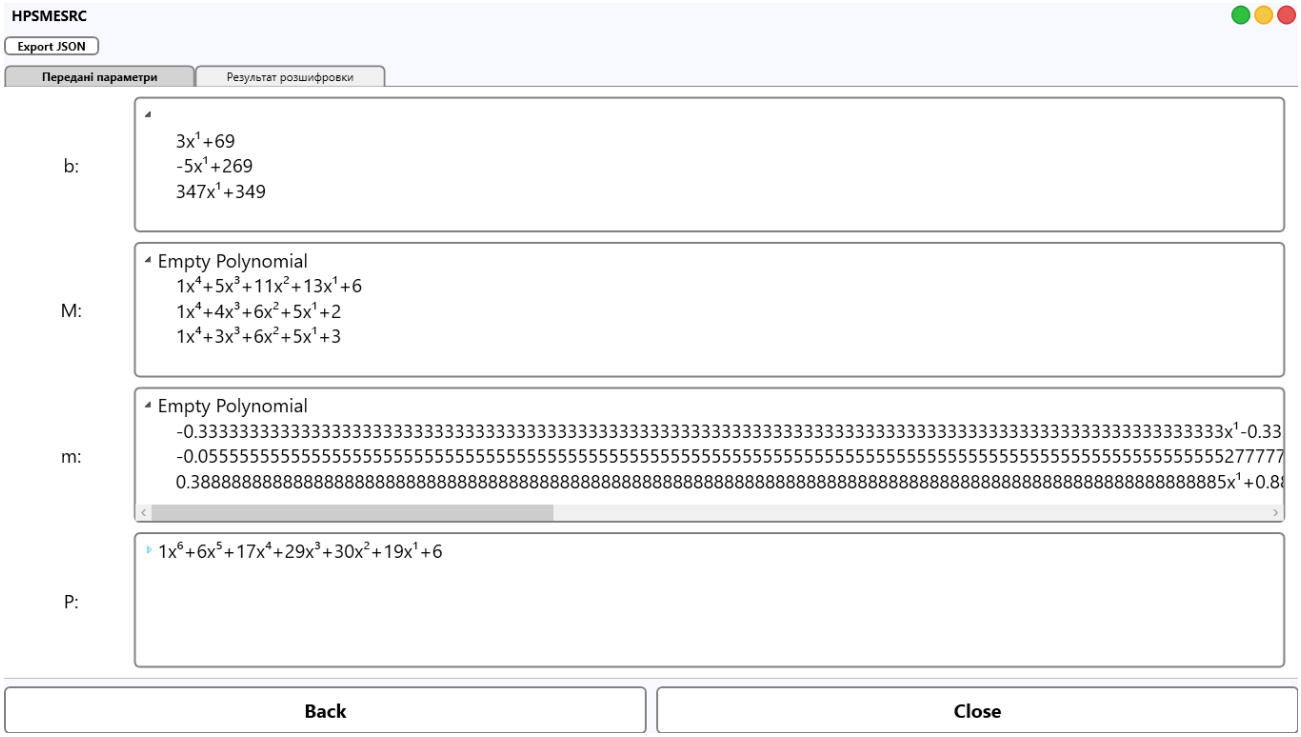


Рисунок 6.47 – Вікно відображення базових параметрів розшифрування

Розшифрування починається з налаштування параметрів, представлених об'єктом *DecypherParameters*. Під час ітераційного процесу відбувається поступове відновлення поліномів згідно КТЗ, починаючи з останнього ієрархічного рівня закінчуючи першим.

Відновлений поліном $N(x)$ першого ієрархічного рівня перетворюється в текст за допомогою використання класу *Word*, який забезпечує конвертацію числових коефіцієнтів у символи тексту, завершуючи процес розшифрування (рисунок 6.48).

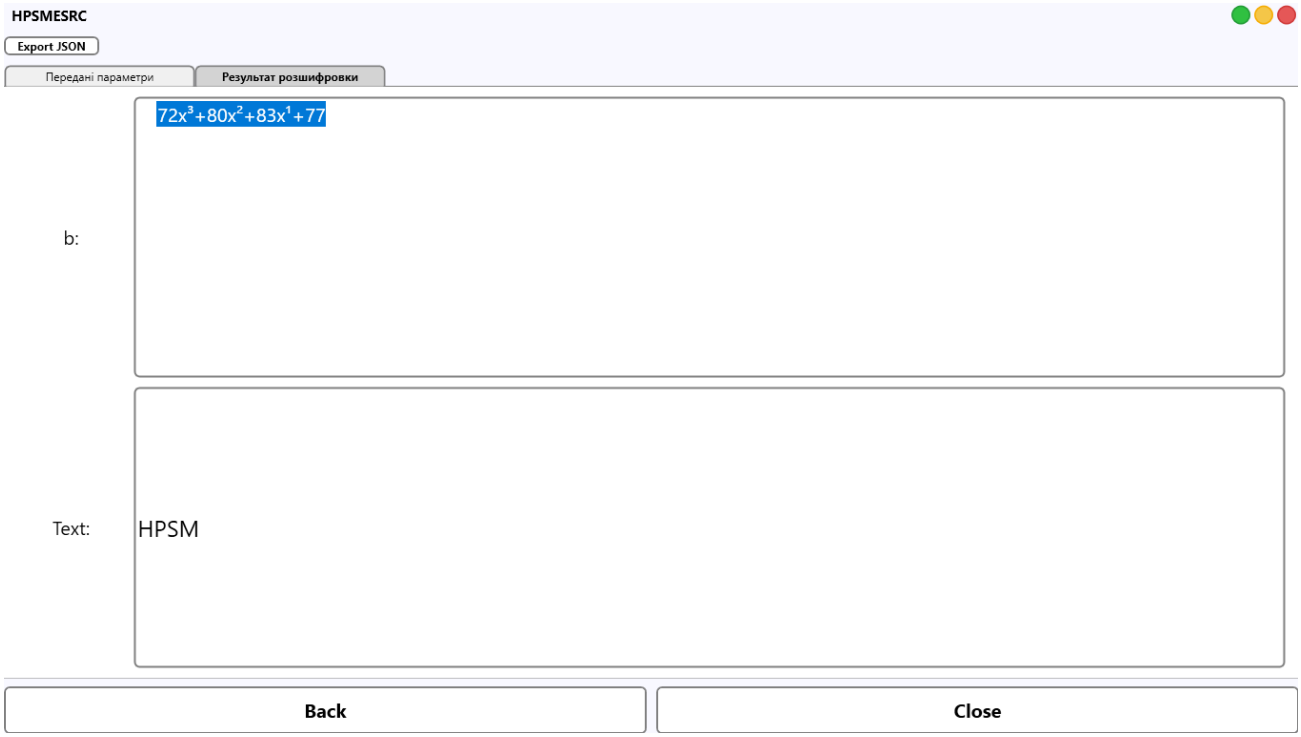


Рисунок 6.48 – вікно результатів розшифрування (друга сторінка)

Після завершення розшифрування користувач отримує доступ до всіх вхідних параметрів, проміжних обчислень і кінцевого повідомлення. Це забезпечує прозорість процесів і дає змогу оцінити результативність алгоритму.

Програма реалізує повний цикл криптографічної обробки тексту на основі поліномів, де користувач визначає кількість ітерацій та задає ключі для кожного етапу. Завдяки зручному інтерфейсу можна швидко вводити дані й аналізувати результати, що підвищує ефективність використання алгоритму для шифрування й розшифрування.

Виведення параметрів, проміжних результатів та кінцевого розшифрованого тексту дозволяє користувачам відстежувати весь процес і впевнитись у правильності обчислень.

Основні результати шостого розділу відображені у роботі [295, 321, 322, 330].

Висновки до шостого розділу

1. Здійснено повну програмну реалізацію симетричних та асиметричних криптоалгоритмів, розроблених у попередніх розділах, що підтвердило їхню працездатність, коректність і можливість практичного застосування в інформаційно-комунікаційних системах. Створене програмне забезпечення продемонструвало ефективність використання СЗК у задачах шифрування та розшифрування, забезпечивши значне прискорення обчислювальних процедур за рахунок властивостей паралелізму та декомпозиції.

2. Реалізація симетричних і асиметричних криптоалгоритмів засвідчила, що заміна операції множення операцією додавання в поєднанні з векторно-модульним представленням даних дає можливість зменшити часову складність ключових криптографічних перетворень. Показано, що алгоритми в ІСЗК можуть масштабуватися за кількістю рівнів, модулів та розрядністю, що відкриває можливість адаптації системи під відповідні вимоги безпеки та продуктивності. Реалізація алгоритмів у ІПСЗК підтвердила працездатність розроблених методів відновлення та пошуку обернених поліномів у криптографічних процедурах.

3. Окремо показано, що розроблене програмне забезпечення забезпечує прозорість криптографічних операцій завдяки виведенню всіх вхідних параметрів, проміжних результатів і кінцевих повідомлень. Це дає змогу користувачеві не лише виконувати шифрування і розшифрування, а й контролювати коректність кожного етапу, що особливо важливо під час тестування криптографічних систем і верифікації їх стійкості.

4. Узагальнення експериментальних результатів, отриманих у межах розділу, підтвердило, що програмна реалізація розроблених криптоалгоритмів у СЗК є ефективною, масштабованою та придатною до інтеграції у прикладні інформаційні системи різного призначення. Це забезпечує можливість практичного застосування створених криптометодів у системах захисту інформаційних потоків із підвищеними вимогами до швидкодії та стійкості.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-прикладну проблему, яка полягає у підвищенні ефективності захисту інформаційних потоків на основі заміни в алгоритмах шифрування операції множення операцією додавання, використання цілочисельної, МДФ, поліноміальної та ієрархічної СЗК для розробки нових криптографічних алгоритмів. При цьому отримано такі основні теоретичні й практичні результати і наукові висновки:

1. Проведено аналіз існуючих симетричних та асиметричних криптосистем, результати якого засвідчили їх обмеження щодо швидкодії, масштабованості та стійкості до криптоаналітичних атак, включно з квантовими. Обґрунтовано доцільність використання цілочисельної та поліноміальної систем залишкових класів, а також їх модифікацій для підвищення швидкодії, криптостійкості та потенціалу криптоалгоритмів. Результати проведеного аналізу дали можливість визначити завдання дисертаційного дослідження щодо розробки методів та засобів криптографічного захисту інформації на основі системи залишкових класів.

2. Розроблено та вдосконалено алгоритмічне забезпечення для асиметричних криптосистем Ель-Гамаля та Рабіна шляхом використання векторно-модульного методу модулярного множення та експоненціювання у системі залишкових класів, замінюючи операцію множення операцією додавання. Запропоновані підходи забезпечили зменшення часової складності криптографічних перетворень відповідно у 8 та 128 разів при розрядності вхідних параметрів 256 біт без зниження рівня криптостійкості.

3. Вперше розроблено симетричний криптоалгоритм у СЗК та МДФ СЗК, в якому шифрування реалізується шляхом представлення повідомлення у вигляді системи залишків. Розшифровування виконується на основі КТЗ. Встановлено умови (розрядність, кількість модулів), за яких алгоритм забезпечує вищий рівень стійкості в порівнянні з AES-256. Зокрема, при кількості модулів l=5, 6, 7, 8 з відповідними розрядностями k=139, 115, 98, 85 біт стійкість підвищується вдвічі.

4. Розроблено метод симетричного шифрування на основі КТЗ, в якому блок відкритого тексту розбито на підблоки (залишки), що менші за відповідні модулі. Відновлене із залишків десяткове число виступає шифротекстом. Розшифрування відбувається на основі пошуку залишків шифротексту за відповідними модулями. Це дозволило підвищити криптографічну стійкість до криптоаналітичних атак у 3 рази при кількості модулів k=5 та їх розрядності l=150 біт в порівнянні з AES-256 і пришвидшити процес розшифрування.

5. Розроблено метод пошуку оберненого полінома в кільці Z[x] на основі методу невизначених коефіцієнтів, що дозволило уникнути обчислення НСД двох поліномів і забезпечити підвищення швидкодії до 4,5 разів для степеня полінома 256 у порівнянні з класичним алгоритмом Евкліда.

6. Удосконалено метод відновлення полінома за його залишками в кільці Z[x], який не вимагає пошуку оберненого полінома та дозволяє повне розпаралелення процесу обчислень. Встановлено, що для кількості модулів 10 та степеня поліному 256 запропонований підхід підвищує швидкодію у 641 раз в порівнянні з алгоритмом Гарнера.

7. Вперше розроблено високопродуктивні симетричні та асиметричні криптоалгоритми на основі СЗК та МДФ СЗК шляхом довільної заміни базисних чисел в процесі шифрування на попарно взаємнопрості з відповідними модулями додаткові ключі, які дозволяють підвищити рівень стійкості до криптоаналітичних атак в 2 рази при кількості модулів k=5 та їх розрядності l=74 біти.

8. Вперше розроблено симетричний криптоалгоритм на основі поліноміальної СЗК, стійкість якого базується на комбінаційній складності криптоаналізу, що призводить до NP-повної задачі. Побудовано залежність криптостійкості від розмірності поля Галуа, степеня полінома та кількості модулів.

9. Вперше розроблено симетричний криптоалгоритм на основі ієрархічної СЗК, який має ступінчасту структуру з розподілом модулів по

рівнях. Показано, що така архітектура підвищує гнучкість алгоритму та забезпечує співставний або вищий криптозахист в порівнянні з AES-256. Встановлено, що при n=8, l=5, k =4 і при n=12, l=4, k =4 спостерігається підвищення криптостійкості до криптоаналітичних атак приблизно в 9 разів.

10. Вперше розроблено симетричний криптоалгоритм на основі ІПСЗК, який поєднує властивості ПСЗК та багаторівневої структури. Це дає змогу адаптивно керувати параметрами шифрування для досягнення заданих рівнів захисту та підвищити стійкість криптоалгоритму в 3,34 рази при степенях поліномів $n = 32$, $p = 2$, $l=3$ на третьому ієрархічному рівні в порівнянні з AES-256 та збільшити його швидкодію.

11. Вперше розроблено методологію криптографічного захисту інформації в системі залишкових класів, яка забезпечує комплексний підхід до розробки, реалізації та оптимізації запропонованих симетричних та асиметричних цілочисельних криптосистем, криптосистем на основі використання ПСЗК, векторно-модульних методів пошуку залишків, модулярного множення та експоненціювання, методу невизначених коефіцієнтів, методів відновлення поліному за його залишками на основі додавання добутку модулів або їх залишків, що дозволило забезпечити збільшення стійкості, зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного забезпечення та побудувати єдину стратегію криптографічного захисту інформаційних потоків на основі системи залишкових класів

12. Здійснена програмна реалізація запропонованих симетричних та асиметричних цілочисельних і поліноміальних криптосистем на основі використання СЗК, яка емпірично підтвердила переваги запропонованих методів і може бути підґрунтям для їх впровадження у сучасні кіберфізичні системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Карпінський М. П., Кінах Я. І., Войтенко, О. С., Паславський, В. Р., Якименко, І. З., Касянчук, М. М. Теоретичний аналіз інформаційної безпеки в комп'ютерних мережах. *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій "*, *2*, 2017. С.81-82.

2. Abood O. G., Guirguis S. K. A survey on cryptography algorithms. *International Journal of Scientific and Research Publications*, vol. 8, no. 7, pp. 410–415, 2018.

3. Mathur M., Kesarwani A. Comparison between DES, 3DES, RC2, RC6, BlowFish and AES. in *Proceedings of National Conference on New Horizons in IT-NCNHIT*, vol. 3, 2013, pp. 143–148

4. Schneier B. The BlowFish encryption algorithm. Link, 2008, [Online; Accessed 1 July 2020]. [Online]. Available: Link

5. Princy P. A comparison of symmetric key algorithms DES, AES, BlowFish, RC4, RC6: A survey. *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 6, no. 5, 2015, pp. 328–331.

6. Riman C., Abi-Char P. E. Comparative analysis of block cipher-based encryption algorithms: A survey. *Information Security and Computer Fraud*, vol. 3, no. 1, 2015, pp. 1–7.

7. Dixit P., Gupta A. K., Trivedi M. C., Yadav V.K. Traditional and hybrid encryption techniques: a survey. *Networking Communication and Data Knowledge Engineering*. Springer, 2018, pp. 239–248.

8. Bhanot R., Hans R. A review and comparative analysis of various encryption algorithms. *International Journal of Security and Its Applications*,vol. 9, no. 4, 2015, pp. 289–306.

9. Wahid M. N. A., Ali A., Esparham B., Marwan M. A comparison of cryptographic algorithms: Des, 3DES, AES, RSA and BlowFish for guessing attacks prevention. *J Comp Sci Appl Inform Technol*, vol. 3, no. 2, 2018, pp. 1–7.

10. Kaur G., Mahajan M. Evaluation and comparison of symmetric key algorithms. *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 2, no. 10, 2013, pp. 1960–1962.

11. Hendi A.Y., Dwairi M.O., Al-Qadi Z. A., Soliman M.S. A novel simple and highly secure method for data encryption-decryption. *InternationalJournal of Communication Networks and Information Security*, vol. 11, no. 1, 2019, pp. 232–238.

12. Tyagi N., Ganpati A. Comparative analysis of symmetric key encryption algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 8, 2014, pp. 63–70.

13. Nema P., Rizvi M.A. Critical analysis of various symmetric key cryptographic algorithms. *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 6, June 2015, pp.4301–4306.

14. Marwaha M., Bedi R.K., Singh A., Singh T. Comparative analysis of cryptographic algorithms. *International Journal of Advanced Engineering Technology*, 2013, pp. 16–18.

15. Nadeem A., Javed M.Y. A performance comparison of data encryption algorithms. *International Conference on Information and Communication Technologies*, 2005, pp. 84–89.

16. Sun P.J. Privacy protection and data security in cloud computing: a survey, challenges, and solutions. *IEEE Access*, vol. 7, 2019, pp. 147 420–147.

17. Tuchman W. A Brief History of the Data Encryption Standard. *USA: ACM Press/Addison-Wesley Publishing Co*. Ch. 16, 1997, pp. 275–280.

18. Mounika Jammula Comparative Study on DES and Triple DES Algorithms and Proposal of a New Algorithm Named Ternary DES for Digital Payments. *Asian Journal of Applied Science and Technology (AJAST).* Volume 6, Issue 1, January-March 2022, pp. 89-98, DOI: http://doi.org/10.38177/ajast.2022.6111.

19. Bufalo M., Bufalo D., Orlando G. A Note on the Computation of the Modular Inverse for Cryptography. Axioms 2021, 10, 116. DOI:10.3390/axioms10020116.

20. Adhie R.P., Hutama Y., Ahmar A.S., Setiawan M. Implementation cryptography data encryption standard (DES) and triple data encryption standard (3DES) method in communication system based near field communication (NFC). *Journal of Physics: Conference Series*, vol. 954, no. 1. IOP Publishing, 2018. DOI 10.1088/1742-6596/954/1/012009

21. Глинчук Л.Я., Грищанович Т.О., Ступінь А.П. Реалізація стандарту симетричного шифрування DES мовою програмування С та порівняння часу його роботи з відомими утилітами, *Кібербезпека:освіта, наука, техніка*, № 2 (14), 2021, с. 118-130.

22. Babar P. K., Bhope V. P. Design and implement dynamic key generation to enhance DES algorithm. *International journal for research in applied science & engineering technology*, 4, 2016, p. 465–468.

23. Zajac, P., Jókay M. Cryptographic properties of small bijective S-boxes with respect to modular addition. *Cryptogr. Commun*. 2020, 12, p. 947–963.

24. Лужецький В. А., Остапенко А.В. Аналіз алгоритмів симетричного блокового шифрування. *Інформаційні технології та комп'ютерна інженерія*. №3 (25), 2012, С. 55-65.

25. Білецький О.О., Білецький О.Я., Навроцький Д.А., Семенюк О.І. Програмно-моделюючий комплекс криптографічних AES подібних примітивів нелінійної підстановки. *Захист інформації*. Т. 16, № 1, 2014, С. 12–22.

26. Nover, H. Algebraic Cryptanalysis of AES: An Overview; University of Wisconsin: Madison, WI, USA, 2005.

27. Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N. Design of S-boxes defined with cellular automata rules. *In Proceedings of the Computing Frontiers Conference*, Siena, Italy, 15–17 May 2017; pp. 409–414.

28. Kuznetsov A., Wieclaw L., Poluyanenko N., Hamera L., Kandiy S., Lohachova Y. Optimization of a Simulated Annealing Algorithm for S-Boxes Generating. *Sensors*. 2022, p. 60-73.

29. Souravlias, D.; Parsopoulos, K.; Meletiou, G. Designing Bijective S-boxes Using Algorithm Portfolios with Limited Time Budgets. *Appl. Soft Comput*. 2017, 59, pp. 475–486.

30. Wang, J., Zhu, Y., Zhou, C., Qi, Z. Construction Method and Performance Analysis of Chaotic S-Box Based on a Memorable Simulated Annealing Algorithm. *Symmetry 2020*, 12, p. 2115.

31. Mariot, L., Picek, S., Leporati, A., Jakobovic, D. Cellular automata based S-boxes. *Cryptogr. Commun*. 2019, 11, pp. 41–62.

32. Tesař, P. A new method for generating high non-linearity S-boxes. *Radioengineering 2010*, 19, pp. 23–26.

33. Ubochi Chibueze Nwamouh, Bashir Olaniyi Sadiq , Kelechi Ukagwu John, A Comparative Analysis of Symmetric Cryptographic Algorithm as a Data Security Tool: A Survey. Stephen Nnamchi Ndubuisi Journal of Science and Technology Research. 5(3), 2023, pp. 144-168.

34. Tiessen T. Polytopic cryptanalysis. *Advances in Cryptology (EUROCRYPT-2016): Proceedings of the 35th International Conference*. V. 9665, N. Y., Springe, 2016, pp. 214–239.

35. M. Kasyanchuk, I. ч, S. Ivasiev, R. Shevchuk, L. Tymoshenko. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. *Proceedings of the International Conference "Advanced Computer Information Technology (ACIT 2018)*" (Ceske Budejovice, Czech Republic), pp. 232-235.

36. Mounika Jammula Comparative Study on DES and Triple DES Algorithms and Proposal of a New Algorithm Named Ternary DES for Digital Payments. *Asian Journal of Applied Science and Technology (AJAST).* Volume 6, Issue 1, January-March 2022, pp. 89-98, DOI: http://doi.org/10.38177/ajast.2022.6111.

37. Al-hazaimeh, O. M., Al-Shannaq , M. A., Bawaneh , M. J ., Nahar , K. M. Analytical Approach for Data Encryption Standard Algorithm. *International Journal of Interactive Mobile Technologies (iJIM)*, 17(14), 2023, pp. 126–143. https://doi.org/10.3991/ijim.v17i14.38641.

38. Lemaire E. Pretty Modular Symmetric Encryption (PMSE), compact algorithm for “embedded cryptography” with quite low computational cost. arXiv preprint arXiv:1905.08150 2019. https://hal.science/hal-02131858v1

39. Yakymenko I., Kasianchuk M., Gomotiuk, O., Ivasiev S., Basistyi P. Elgamal cryptoalgorithm on the basis of the vector-module method of modular exponentiation and multiplication. *Proceedings - 15th International Conference on Advanced Trends in Radioelectronics*, Telecommunications and Computer Engineering, TCSET 2020, 2020, pp. 926–929.

40. Задірака В.К., Олексюк О.С. Комп’ютерна криптологія. Тернопіль, Київ. 2002, 504 с.

41. Вербіцький О.В. Вступ до криптології. Львів: ВНТЛ, 1998, 247 с.

42. Stillwell J. Elements of Number Theory. Springer. 2010, 256 p.

43. Дичка І.А., Онай М.В. Способи знаходження мультиплікативного оберненого елемента в скінченних полях. Наук. техн. журн. «Наукові вісті НТУУ «КПІ». Вип. 2, 2015, С. 160-165.

44. Lorencz R. New Algorithm for Classical Modular Inverse. Cryptographic Hardware and Embedded Systems: International Workshop. 2002, pp. 57-70.

45. Sorenson J. Two fast GCD algorithms. Journal of Algorithms. 1994, pp. 110-144.

46. Vallee B. Dynamical analysis of a class of Euclidean algorithms. Theoretical Computer Science. V.297. 2003, pp. 447–486.

47. Vallee B. Dynamics of the binary Euclidean algorithm: functional analysis and operators. Algorithmica. Vol.22, №4, 1998, pp. 660–685.

48. Завало С.Т., Костарчук В.Н., Хацет Б.І. Алгебра і теорія чисел. К.: Вища школа, 1977. 400 с.

49. Papachristodoulou L., Batina L., Mentens N. Recent Developments in Side-Channel Analysis on Elliptic Curve Cryptography Implementations. *Springer International Publishing: Berlin*, Germany, 2017, pp. 49–76.

50. Dindayal Mahto, Dilip Kumar Yadav RSA and ECC: A Comparative Analysis. *International Journal of Applied Engineering Research ISSN 0973-4562*. Volume 12, Number 19, 2017, pp. 9053-9061

51. Yakymenko I.Z., Kasianchuk M.M., Ivasiev S.V., Melnyk A.M., Nykolaichuk Ya.M. Realization of RSA cryptographic algorithm based on vector-module method of modular exponention. *Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET–2018): Proceedings of the XIV–th International Conference*. L'viv–Slavske, 2018, pp.550-554.

52. Buhrow B. Parallel modular multiplication using 512-bit advanced vector instructions: RSA fault-injection countermeasure via interleaved parallel multiplication / Buhrow B., Gilbert B., Haider C. // Journal of Cryptographic Engineering.-2021.- № 2.- P.46-53.

53. M. Kasianchuk, I. Yakymenko, I. Pazdriy, A. Melnyk, S. Ivasiev Rabin's modified method of encryption using various forms of system of residual classes. *The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM-2017): Proceedings of the XIV International Conference.* Polyana-Svalyava, 2017, pp. 222-224.

54. Якименко І.З., Касянчук М.М., Волинський О.І., Пітух І.Р. Теорія алгоритмів RSA та Ель–Гамаля в розмежованій системі числення Радемахера–Крестенсона. *Вісник Хмельницького національного університету*. Технічні науки. №3, 2011, С. 265-273.

55. Касянчук М.М., Якименко І.З., Дубчак Л.О., Рендзеняк Н.А., Мандебура Н.М. Модифікований метод шифрування Рабіна з використанням різних форм системи залишкових класів. *Вісник Хмельницького національного університету*. Технічні науки. №1(245), 2017, С. 127-131.

56. Karpinski M., Rajba S., Zawislak S., Warwas K., Kasianchuk M., Ivasiev S., Yakymenko I. A method for decimal number recovery from its residues based on

the addition of the product modules. *In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1, 2019, pp. 13–17. https://doi.org/10.1109/IDAACS.2019.8924395.

57. Krasnobayev V.A., Koshman S.A. Method of realization of cryptographic RSA transformations on the basis of application of modular number system. *Biomedical Soft Computing and Human Sciences*. Vol. 17 (2), 2011, pp. 31–36.

58. Srivastava A., Mathur A. The Rabin cryptosystem and analysis in measure of chinese reminder theorem. *International Journal of Scientific and Research Publications*. Vol. 3 (6), 2013, pp. 1-4.

59. Ochoa-Jiménez E., Rivera-Zamarripa L., Cruz-Cortés N., Rodríguez-Henríquez F. Implementation of RSA signatures on GPU and CPU architectures. *IEEE Access 2020*, 8, 2020, pp. 9928–9941. https://doi.org/10.1109/ACCESS.2019.2963826.

60. Gbolagade K., Chaves R., Sousa L., Cotofana S. An improved RNS reverse converter for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set. *Circuits System: Proceedings of the IEEE International Symposium*. 2010, pp. 2103–2106.

61. Krasnobayev V.A., Koshman S.A. Method of realization of cryptographic RSA transformations on the basis of application of modular number system. *Biomedical Soft Computing and Human Sciences*. Vol. 17 (2), 2011, pp. 31–36.

62. Николайчук Я.М., Ивасьев С.В., Якименко И.З., Касянчук М.Н. Метод факторизации многоразрядных чисел на основе свойств квадратичности вычетов в системе остаточных классов. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2015. № 5(95). С. 45–45.

63. Николайчук Я.М., Федорович Ю.С. Теоретичні основи базисних перетворень СЗК. *Автоматика 2000*: *Матеріали наукової конференції*. Львів. 2000, С. 120.

64. Николайчук Я.М. Теорія джерел інформації. Тернопіль: ТзОВ: *Терн*ограф, 2010. 536 с.

65. Zheng Tian-Xiang Enhanced Rabin cryptosystem based on cubic congruence equation. *Journal of Computer Applications*. №7, 2009, pp. 121-129.

66. Yahia Awad, Abdul Nasser El-Kassar, Therrar Kadri Rabin Public-Key Cryptosystem in the Domain of Gaussian Integers. *International Conference on Computer and Applications (ICCA)*. 2018, pp. 1-340.

67. Касянчук М.М., Якименко І.З., Тимошенко Л.М., Івасьєв С.В., Николайчук Я.М. Векторно-модульний метод модулярного множення. Сучасні інформаційні та електронні технології: Матеріали Міжнародної науково-практичної конференції. Одеса. 2014, С. 152.

68. Івасьєв С.В. Методи та обчислювальні засоби рішення задач теорії чисел у базисах Радемахера – Крестенсона: дис. канд. техн. наук: 05.13.05. Тернопіль: ТНЕУ, 2016, 222 с.

69. Николайчук Я.М., Касянчук М.М., Якименко І.З., Долинюк Т.М. Теоретичні основи виконання модулярних операцій множення та експоненціювання в теоретико–числовому базисі Крестенсона–Радемахера. *Інформатика та математичні методи в моделюванні*. 2011. №2. С. 123–130.

70. Nykolaychuk Ya., Ivas'ev S., Yakymenko I., Kasianchuk M. Test of verification of multidigit numbers on simplicity on the basis of method of vector and modular multiplication. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2016*): Proceedings of the XIII–th International Conference. L'viv–Slavske. 2016. P.534-536 (Scopus).

71. Березовский А.И. О тестировании быстродействия алгоритмов и программ вычисления основных операций ассиметричной криптографии/ Березовский А.И., Задирака В.К., Шевчук Л.Б. //Кибернетика и системный анализ №5, 1999. – С. 59-66.

72. Zuras D., More on squaring and multiplying larges integers, IEEE Transactions on Computers,-1994.- Vol. 43, - № 8,- P. 899–908.

73. Kozaczko D., Ivasiev S., Yakymenko I., Kasianchuk M. Vector Module Exponential in the Remaining Classes System. *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2015)*. Warsaw, Poland. V.1., September, 2015, pp.161–163. DOI: 10.1109/IDAACS.2015.7340720.

74. T. Rajba, A. Klos-Witkowska, S. Ivasiev, I. Yakymenko, M. Kasianchuk. Research of Time Characteristics of Search Methods of Inverse Element by the Module. Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2017) (Bucharest, Romania). V.1. 2017. pp.82-85. DOI: 10.1109/IDAACS.2017.8095054.

75. Svoboda A. Rational numerical system of residual classes. Stroje Na Zpracovani Informaci. Vol. 1, 1957, pp. 33-48.

76. Svoboda A. The numerical system of residue classes in mathematical machine. Inform. processing. Vol. 2, 1960, pp. 81-100.

77. Краснобаев В.А., Кошман С.А., Мороз С.А., Курчанов В.Н., Янко А.С. Модели и методы обработки данных в системе остаточных классов. Харьков: ООО «В деле», 2017, 197 с.

78. Venturi D. Lecture Notes on Algorithmic Number Theory. *Springer-Verlag*, New-York, Berlin, 2009, 217 p.

79. Горбенко І.Д., Долгов В.І., Потій А.В., Федорченко В.Н. Аналіз каналів вразливості системи RSA. *Безпека інформації*. №2, 1995, С. 22-26.

80. Granger R., Kleinjung, T., Zumbrägel J. On the discrete logarithm problem in finite fields of fixed characteristic. *Trans. Am. Math. Soc*. 2018, pp. 3129–3145.

81. Gayoso Martínez, V., Hernández Encinas L., Martín Muñoz A., Durán Díaz R. Secure elliptic curves and their performance. *Log. J. IGPL*, 2019, 27, pp.277–238.

82. Silverman J.H., Tate J.T. Rational Points on Elliptic Curves. *Undergraduate Texts in Mathematics, Springer International Publishing: Cham, Switzerland*, 2015, pp.332. DOI 10.1007/978-3-319-18588-0

83. Verma S., Garg D. Improvement in rebalanced CRT RSA. *International Arab Journal of Information Technology*. Vol.12, No. 6, 2015, pp.524-532.

84. Fournaris A.P., Papachristodoulou L., Batina L., Sklavos N. Residue number system as a side channel and fault injection attack coun- termeasure in elliptic curve cryptography. *Design and Technology of Integrated Systems in Nanoscale Era (DTIS): Proceedings of the 2016 International Confere*nce. 2016, pp. 1–4.

85. Washington L. Elliptic Curves Number Theory and Cryptography. *Series Discrete Mathematics and Its Applications*, Chapman & Hall/CRC, 2008, 524 p.

86. Fournaris A.P., Papachristodoulou L., Batina L., Sklavos N. Secure and Efficient RNS Approach for Elliptic Curve Cryptography. *In Proceedings of the 6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)*, 2016, pp. 121–126.

87. Silverman J.H. The arithmetic of elliptic curves. New York: Springer. Vol. 106, 2007, pp. 17-40.

88. Coblitz N., Menezes A., Vanstone S. The state of elliptic cryptography. Design, Codes and Cryptography. Vol. 19, 2000, pp. 103–123.

89. Lenstra H.W. Factoring integers with elliptic curves. Ann.Math. Vol.126, 1987, pp. 649–674.

90. Guillermin N. A high speed coprocessor for elliptic curve scalar multiplications over Fp. Lecture Notes in Computer Science, Advances in Cryptology, Cryptographic Hardware and Embedded Systems. 2010, pp. 48–64.

91. Granger R., Kleinjung, T., Zumbrägel J. On the discrete logarithm problem in finite fields of fixed characteristic. *Trans. Am. Math. Soc*. 2018, pp. 3129–3145.

92. El Hassan El Kinani, Fatima Amounas Elliptic Curve Digital Signature Algorithm Using Boolean Permutation based ECC. *International Journal of*

Information and Network Security (IJINS). 1(3). Vol. 1, No. 3, August 2012, pp. 216-222.

93. Ambedkar B.R., Gupta A., Gautam P., Bedi S. An Efficient Method to Factorize the RSA Public Key Encryption. *Communication Systems and Network Technologies:* Proceeding of International Conference. 2011, pp. 108–111.

94. Ivas'ev S., Kasyanchuk M., Yakymenko I., Nykolaychuk Ya. Fundamental Backgrounds of the Discrete Logarithms Theory in the Rademacher–Krestenson's Basis. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2012*): Proceedings of the XI–th International Conference. L'viv–Slavske. 2012. P.93

95. Shehzad Ashraf Chaudhry, Mohammad Sabzinejad Farash, Husnain Naqvi, Muhammad Sher A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography. *Electronic Commerce Research*. Volume 16, 2016, pp. 113–139.

96. Igor Yakymenko, Mykhailo Kasianchuk, Vasyl Yatskiv, Ruslan Shevchuk, Vasyl Koval, Solomiya Yatskiv. Sustainability and Time Complexity Estimation of Cryptographic Algorithms Main Operations on Elliptic Curves. 2021 11th International Conference on Advanced Computer Information Technologies (ACIT). Pp. 494-498.

97. Якименко І.З., Касянчук М.М., Кімак В.Л. Теоретичні основи зменшення часової та апаратної складності систем захисту інформаційних потоків на основі еліптичних кривих з використанням теоретико-числового базису Радемахера-Крестенсона. *Вісник Національного університету «Львівська політехніка» «Комп'ютерні системи та мережі».* 2012. №745. С. 190–197.

98. Valach M. Origin of the code and system of remainder classes/ M.Valach, A. Svoboda// Stroje Na Zpracovani Informaci. Vol.3, 1955, pp. 121-134.

99. Biyashev R., Nyssanbayeva S., Kapalova N., Khakimov R., Modular models of the cryptographic protection of information. *Proc. International*

Conference on Computer Networks and Information Security (CNIS2015). 2015, pp. 393-398.

100. Biyashev R., Nyssanbayeva S., Kapalova N., Khakimov R.A., Development of a cryptographic protection system based on modular arithmetic. *Proc. XIII International Scientific and Practical Conference "IB-2013"*. Part I.- Taganrog: SFU publ.house, 2013, 215-220.

101. Nyssanbayeva S.E., Magzom M.M. Simulation of non- traditional encryption algorithm, Bulletin of KazNTU, No.4, 2015, pp. 596-599.

102. Николайчук Я.М. Коди поля Галуа: теорія та застосування. Тернопіль: ТзОВ: Терн граф, 2012, 576 с.

103. Givaki K., Hojabr R., Najafi M.H., Khonsari A., Gholamrezayi M.H., Gorgin S., Rahmati D. Using residue number systems to accelerate deterministic bit-stream multiplication. *In Proceedings of the 2019 IEEE 30th International Conference on Application-specific Systems*, Architectures and Processors (ASAP). IEEE, Vol. 2160, 2019, pp. 40–48. https://doi.org/10.1109/ASAP.2019.00-33.

104. Краснобаев В.А., Кошман С.О., Маврина М.О. Метод повышения достоверности контроля данных, представленных в системе остаточных классов. *Кибернетика и системний анализ*. Том 50 (6), 2014, С.167 –175.

105. Shevchuk, R., Yakymenko, I., Kasianchuk, M. Encryption Using Residue Number System: Research Trends and Future Challenges Proceedings - International Conference on Advanced Computer Information Technologies, ACIT2024, 2024, pp. 552–559 (Scopus).

106. Omondi A., Premkumar B. Residue number systems: theory and implementation. London: Imperial College Press, 2007. 296 p.

107. Narkiewicz W. Elementary and Analytic Theory of Algebraic Numbers. *Springer Berlin Heidelberg*, 2004. 712 p.

108. Wang Y., Song X., Aboulhamid M., Shen H. Adder based residue to binary number converters for $(2^n - 1, 2^n, 2^n + 1)$. IEEE Transactions on Signal Processing, 50(7), 2002, pp. 1772-1779. https://doi.org/10.1109/TSP.2002.1011216

109. Henk D.L. Hollmann, Ronald Rietman, Sebastiaan de Hoogh, Ludo Tolhuizen A Multi-layer Recursive Residue Number System. *IEEE International Symposium on Information Theory (ISIT) Date Added to IEEE Xplore*: 16 August 2018, pp.1460-1464, DOI: 10.1109/ISIT.2018.8437612.

110. Labafniya M., Eshghi M. Non-iterative RNS Division Algorithm. *I*nternational multiconference of engineers and computer scientists: Proceedings. Vol. 1, 2012, pp. 132-135.

111. Nakato Antoniou S., Rissner R. Irreducible polynomials in Int(Z). ITM Web of Conferences International Conference on Mathematics. Vol. 20, 2018, Article Number: 01004. https://doi.org/10.1051/itmconf/20182001004.

112. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. *J. Signal Process. Syst*. Vol. 56 (1), 2009, pp. 1–15.

113. Bajard J.-C., Imbert L., Plantard T. Arithmetic operations in the polynomial modular number system. *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, 2005, pp. 206–213. DOI: 10.1109/ARITH.2005.11.

114. Giorgi P. Parallel modular multiplication on multi-core processors. / Giorgi P., Imbert L., Izard T. // IEEE Symposium on Computer Arithmetic, Apr 2013, Austin, TX, United States. - P.135-142.

115. Krasnobayev V.A., Yanko A.S., Koshman S.A. Method for arithmetic comparison of data represented in a residue number system. *Cybernetics and Systems Analysis*. Vol. 52, №1, 2016, pp. 145–150.

116. Николайчук Я.М. Теорія цифрових перетворень мультибазисного супершвидкодіючого процесора. *Искусственный интеллект*. №4, 2008, С. 387-394.

117. Labafniya M. Eshghi M. RNS division algorithm for 2n-1 and 2n dividers. 22nd Iranian Conference on Electrical Engineering (ICEE): Proceedings. 2014. pp. 111-114.

118. Lu M. Chiang J.-S. A novel division algorithm for the residue number system. IEEE Transactions on Computers. Vol.41(8), 1992, pp. 1026-1032.

119. Bajard J.-C., Didier L.-S., Muller J.-M. A new Euclidean division algorithm for Residue Number Systems. J. VLSI Signal Processing. Vol. 19 (2), 1998, pp. 167–178.

120. Hitz M.A., Kaltofen E. Integer division in residue number systems. IEEE Trans. Comput. Vol. 44 (8), 1995, pp. 983–989.

121. Vivek N., Anusudha K. Design of RNS Based Addition Subtraction and Multiplication Units. International Journal of Engineering Trends and Technology. Vol. 10 (12), 2014, pp. 593-596.

122. Lalitha K.V., Sailaja V. High performance adder using Residue Number System. International Journal of VLSI and Embedded Systems. Vol. 05, 2014, pp. 1323-1332.

123. Vergos H. On the design of efficient modular adders. J. Circuits, Syst. and Comput. Vol. 14 (5), 2005, pp. 965–972285.

124. Jaberipur G., Parhami B., Nejati S. On building general modular adders from standard binary arithmetic components. Proc. Signals, Systems, and Computers: Proceedings of the 45th Asilomar Conference. 2011, pp. 6–9.

125. Краснобаев В. А., Загуменная Е.В., Мороз С.А., Жадан В.О. Математическая модель процесса табличной реализации операций алгебраического умноженя в классе вычетов. Радіоелектронні і комп'ютерні системи. Т.11 (2), 2012, С. 281-287.

126. Кошман С.А., Деренько Н.С. Метод реализации арифметических операций в модулярной арифметике на основе использования малоразрядных двоичных сумматоров. Радіоелектронні і комп'ютерні системи. Т.7 (26), 2007, С. 219-221.

127. Krasnobayev V.A., Yanko A.S., Koshman S.A. The method of error correction in the system of residual classes. *Nauka i studia*. Przemysl (Poland). №5 (136), 2015, pp. 51-62.

128. Angel M.A., Narendrakumar A. Improving system performance by using prefix adders in RNS. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*. Vol. 5 (9), 2016, pp. 1-5.

129. Harvey D. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*. Vol. 60, 2014, pp. 113–119.

130. Fadulilahi I.R., Bankas E.K., Ansuura J.B.A.K. Efficient Algorithm for RNS Implementation of RSA. *International Journal of Computer Applications*. Vol. 127 (5), 2015, pp. 14-19.

131. Hooshmand R., Aref M.R., Eghlidos T. Secret Key Cryptosystem Based on Non-systematic Polar Codes. *Wireless Pers Commun* 84, 2015, pp. 1345–1373. https://doi.org/10.1007/s11277-015-2691-9.

132. Biyashev R., Nyssanbayeva S., Kalimoldayev M., Magzom M. Development of an encryption algorithm based on nonpositional polynomial notations. *In Proceedings of the 2016 International Conference on Advanced Materials Science and Environmental Engineering*, 2016, pp. 241–243. https://doi.org/10.2991/amsee-16.2016.64.

133. Didier L.S., Dosso F.Y., Véron P. Efficient modular operations using the adapted modular number system. *Journal of Cryptographic Engineering*. 10, 2020, pp. 111–133. https://doi.org/10.1007/s13389-019-00221-7.

134. Singh N. An overview of Residue Number System. Devices, Circuits & Communication: Proceedings of the National Seminar. 2008, pp. 132-135.

135. Garner H.L., Harvey L.G. The Residue Number System. IRE Transactions on Electronic Computers. Vol. EC-8 (2), 1959, pp. 140-147.

136. Кошман С.А. Концепция реализации немодульных операций в модулярной системе счисления. Проблеми інформації: Матеріали другої міжнародної науково-технічної конференції. 2014, С. 94-95.

137. Kornerup P., Matula D.W. Finite Precision Number Systems and Arithmetic. Cambridge University Press. 2010, 699 p.

138. Tomczak T. Hierarchical residue number systems with small moduli and simple converters. *International Journal of Applied Mathematics and Computer Science*. Vol. 21 (1), 2011, pp. 173–192.

139. Барсов В. И., Сорока Л.С., Краснобаев В.А. Методология параллельной обработки информации в модулярной системе счисления. Харьков: УИПА, 2009, 268 с.

140. Anitha K., Arulananth T.S., Karthik R., Bhaskara P. Reddy Design and Implementation of Modified Sequential Parallel RNS Forward Converters. International Journal of Applied Engineering Research. Vol. 12 (16), 2017, pp. 6159-6163.

141. Reshadinezhad M., Samani F.K. A Novel Low Complexity Combinational RNS Multiplier Using Parallel Prefix Adder. International Journal of Computer Science. Vol. 10 (3), 2013, pp. 430-440.

142. Yang L.L., Hanzo L. A residue number system based parallel communication scheme using orthogonal signaling: Part II—Multipath fading channels. IEEE Trans. Veh. Technol. Vol. 51, 2002, pp. 1541-1553.

143. Akkal M., Siy P. A new mixed radix conversion algorithm MRC-II. J. Syst. Archit. V.53, 2007, pp.577–586.

144. Sousa L., Antao S., Martins P. Combining residue arithmetic to design efficient cryptographic circuits and systems. *IEEE Circuits Syst. Mag.* 2016, pp.6–32.

145. Esmaeildoust M., Schinianakis D., Javashi H., Stouraitis T., Navi, K. Efficient RNS implementation of elliptic curve point multiplication GF(p). *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* V.21, 2013, pp.1545–1549.

146. Chang C., Molahosseini A.S., Zarandi A.A.E., Tay T.F. Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications. IEEE Circuits Syst. Mag. V.15, 2015, pp.26–44.

147. Kaplun D., Butusov D., Ostrovskii V., Veligosha A., Gulvanskii V. Optimization of the FIR filter structure in finite residue field algebra. Electronics. V.7, 2018, 372 p.

148. Краснобаев В.А., Кошман С.О., Маврина М.О. Метод повышения достоверности контроля данных, представленных в системе остаточных классов. *Кибернетика и системний анализ*. Том 50 (6), 2014, С.167 –175.

149. Upadhyaya, Arun; Bhat, Shubha P.; Aithal, Ganesh. A Security Enhanced Image Encryption and Compression Using Residue Number System and Discrete Cosine Transform. In: *International Conference on Signal & Data Processing*. Singapore: Springer Nature Singapore, 2022. p. 419-434.

150. Andrijchuk V.A., Kuritnyk I.P., Kasyanchuk M.M., Karpinski M.P. Modern Algorithms and Methods of the Person Biometric Identification. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (*IDAACS–2005*): Proceedings of the Third IEEE Workshop. Sofia, Bulgaria, 2005, pp.403–406.

151. Hema V., Ganaga Durga M. Data Integrity Checking Based On Residue Number System and Chinese Remainder Theorem In Cloud. *International Journal of Innovative Research in Science, Engineering and Technology*. Vol.3 (3), 2014, pp. 2584-2588.

152. Kar A., Sur K., Godara S., Basak S., Mukherjee D., Sukla A.S., Das R., Choudhury R. Security in cloud storage: An enhanced technique of data storage in cloud using RNS. *In Proceedings of the IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, (New York, USA). 2016, pp. 1–4.

153. Chang C., Lee W., Liu Y., Goi B., Phan R.C.-W. Signature gateway: Offloading signature generation to IoT gateway accelerated by GPU. *IEEE Internet Things J.* V.6, 2019, pp.4448–4461.

154. Hu Zhengbing, Yatskiv V., Sachenko A. Increasing the Data Transmission Robustness in WSN Using the Modified Error Correction Codes on Residue Number System. *Elektronika ir Elektrotechnika*. Vol 21(1), 2015, pp. 76-81.

155. Roshanzadeh M., Saqaeeyan S. Error Detection & Correction in Wireless Sensor Networks By Using Residue Number Systems. *International Journal of Computer Network and Information Security*. №2, 2012, pp. 29-35.

156. Су Цзюнь, Яцкив В.В., Саченко А.А., Ху Чежньбин. Спецпроцессор кодирования изображений в системе остаточных классов. *Современные информационные и электронные технологии (СИЭТ-2012):* Труды МНПК. Одесса, 2012, С.95.

157. Sharoun A.O. Residue number system. Poznan university of technology academic journals. *Electrical Engineering*. №76, 2013, pp. 265-270.

158. Smyk R., Ulman Z., Czyżak M. Pipelined division of signed numbers with the use of residue arithmetic for small number range with the programmable gate array. *Electrical Engineering*. №76, 2013, pp. 117-126.

159. Patronik P., Piestrak S.J. Design of Reverse Converters for General RNS Moduli Sets $\{2^k, 2^n-1, 2^n+1, 2^{n-1}-1\}$. *IEEE Transactions on Circuits and Systems*. V.10 (1), 2014, pp. 143-148.

160. Patronik P., Piestrak S.J. Design of Reverse Converters for a New Flexible RNS Five-Moduli Set $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1, 2^{n-1}-1\}$. *Circuits Syst Signal Process.* V.36, 2017, pp.4593–4614.

161. Milanezi Junior J., da Costa J.P.C.L., Römer F., Miranda R.K., Marinho M.A.M., Del Galdo G. M-estimator based Chinese remainder theorem with few remainders using a kroenecker product based mapping vector. Digit. Signal Process. V.87, 2019, pp.60–74.

162. M. Kasyanchuk, "Theory and mathematical laws of perfect form of residual classes", Proceedings of the International Symposium “Problems of calculations optimization (PCO-XXXV)”, vol. 1, 2009, pp. 306-310.

163. Kasianchuk M. The concept of theoretical positions perfect shape transformation Krestenson and its practical application. Opto-Electronic Information and communication technology, vol. 2, no. 20, 2010, pp. 43-48.

164. Касянчук М.М., Якименко І.З., Паздрій І.Р., Николайчук Я.М. Аналітичний пошук модулів досконалої форми системи залишкових класів та

їх застосування в китайській теоремі про залишки. *Вісник Хмельницького національного університету*. Технічні науки. №1(221), Хмельницький 2015, С. 170-176.

165. Nykolaychuk Ya.M., Kasianchuk M.M., Yakymenko I.Z. Theoretical Foundations of the Modified Perfect form of Residue Number System. *Cybernetics and Systems Analysis*. Vol. 52, №2, 2016, pp. 219-223.

166. Касянчук М.М., Сидорчук Р.П. Алгоритми підбору модулів у системі залишкових класів. *Сучасні комп'ютерні інформаційні технології (ACIT–2011):* Матеріали І Всеукраїнської школи–семінару молодих вчених і студентів. Тернопіль, 2011, С.50–51.

167. Djath L., Bigou K., Tisserand A. Hierarchical Approach in RNS Base Extension for Asymmetric Cryptography. IEEE 26th Symposium on Computer Arithmetic (ARITH-2019) (Kyoto, Japan). 2019, pp.46-53.

168. Sousa L., Antao S., Martins P. Combining residue arithmetic to design efficient cryptographic circuits and systems. IEEE Circuits Syst. Mag. 2016, pp. 6–32.

169. Xiao H., Garg H., Hu J., Xiao G. New Error Control Algorithms for Residue Number System Codes. *Electronics and Telecommunications Research Institute*. Vol. 38 (2), 2016, pp.326-336.

170. Plantard T. Efficient word size modular arithmetic. *IEEE Transactions on Emerging Topics in Computing*. 9, 2021, pp. 1506–1518. https://doi.org/10.1109/TETC.2021.3073475.

171. Harvey D., Hoeven J. Faster polynomial multiplication over finite fields using cyclotomic coefficient rings. *Journal of Complexity*. Vol. 54, 2019, pp.101404. https://doi.org/10.1016/j.jco.2019.03.004.

172. Meredith M. Brandon. Polynomial Functions over Rings of Residue Classes of Integers. *Thesis, Georgia State University*, 2007. 48 p. doi: https://doi.org/10.57709/1059690

173. Ashok G., Kumar S.A., Kumari D.C. An Approach of Cryptosystem using Polynomials and Lucas Numbers. *Journal of Harbin Engineering University*. 44 (8), 2023, pp. 25–31.

174. Ashok G., Ashok Kumar S., Chaya Kumari D., Ramakrishna M. A type of public cryptosystem using polynomials and pell sequences. *Journal of Discrete Mathematical Sciences and Cryptography*. 25, 2022, pp.1951–1963. https://doi.org/10.1080/09720529.2022.2133237.

175. Alamsyah Bejo A., Adji T.B. The replacement of irreducible polynomial and affine mapping for the construction of a strong S-box. *Nonlinear Dynamics*. 93, 2018, pp. 2105–2118. https://doi.org/10.15294/sji.v7i1.24006.

176. Alamsyah, A. A novel construction of perfect strict avalanche criterion S-box using simple irreducible polynomials. *Scientific Journal of Informatics*. 7, 2020, pp. 10–22. https://doi.org/10.15294/sji.v7i1.24006.

177. Agarwal P., Singh A., Kilicman A. Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant. *Advances in Mechanical Engineering*. 10, 2018, 1687814018781638. https://doi.org/10.1177/1687814018781638.

178. Alshammari K.F., Mostafa A., Nashwan S. Avalanche analysis of variant polynomials for AES. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*. 12, 2021, pp. 2696–2703.

179. El-Kassar A.N., Haraty R.A., Awad Y., Debnath N.C. Modified RSA in the Domains of Gaussian Integers and Polynomials Over Finite Fields. *In Proceedings of the CAINE*. 2005, pp. 298–303.

180. Katz J., Sahai A., Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *In Proceedings of the Advances in Cryptology–EUROCRYPT 4642008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Istanbul, Turkey, April 13-17, 2008. Proceedings 27, 2008, pp. 146–162. https://doi.org/10.1007/978-3-540-78967-3_9.

181. Bajard J.C., Imbert L. A full RNS implementation of RSA. *IEEE Transactions on computers.* 53, 2004, pp. 769–774.

182. Calvez L. C., Azou S., Vilbe P. Variation on Euclid's algorithm for polynomials. *Electronics Letters*, vol. 33, no. 11, 1997, pp. 939-940.

183. Ivasiev S., Kasianchuk M., Yakymenko I., Shevchuk R., Karpinski M., Gomotiuk O. Effective algorithms for finding the remainder of multi-digit numbers. Proceedings of the International Conference "Advanced Computer Information Technology (ACIT-2019)" (Ceske Budejovice, Czech Republic). 2019, pp. 175-178. DOI: 10.1109/ACITT.2019.8779899.

184. Goupil and J. Palicot. Variation on variation on Euclid's algorithm. *IEEE Transactions on Signal Processing Letters*, vol. 11, no. 5, pp. 457-458, 2004.

185. Zadiraka V. K., Oleksyuk O.S. Computer arithmetic of multi decimal numbers. K., 2003, 264 p.

186. Nykolaychuk Ya. N., Volynskii O.I., Kulyna S.V. Theoretical foundations of construction and the structure of special processors in the Krestenson basis. *Vestn. Khmeln. Nats. Un-ta*, 1. No. 3, 2007, pp. 85–90.

187. Kasianchuk M., Yakymenko I., Pazdriy I., Zastavnyy O. Algorithms of findings of perfect shape modules of remaining classes system. *XIII International Conference "The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM-2015)"*, 23-25 February, 2015, Polyana-Svalyava (Zakarpattya), Ukraine, 2015, pp.168-171.

188. Liu L., Horng G., and Liu H. Y. Computing the modular inverse is as simple as computing the GCDs. *International Journal of Finite Fields and Their applications*, vol. 14, 2008, pp. 65-75.

189. Xiao, F., Lu, D. & Wang, D. Solving Multivariate Polynomial Matrix Diophantine Equations with Gröbner Basis Method. J Syst Sci Complex 35, 2022, pp. 413–426. https://doi.org/10.1007/s11424-021-0072-x.

190. Milne J.S. Algebraic Number Theory. Version 3.08. MilneANT. 2020, 166 p. https://www.jmilne.org/math/CourseNotes/ANTc.pdf.

191. Zheng Y., Wang Q., Wei W. On Inverses of Permutation Polynomials of Small Degree Over Finite Fields. *IEEE Transactions on Information Theory*, vol. 66, no. 2, 2020, pp. 914-922. doi: 10.1109/TIT.2019.2939113.

192. Nyokabi G. J., Salleh M., Mohamad I. NTRU inverse polynomial algorithm based on circulant matrices using gauss-jordan elimination. *2017 6th ICT International Student Project Conference (ICT-ISPC)*, Johor, Malaysia, 2017, pp. 1-5. doi: 10.1109/ICT-ISPC.2017.8075326.

193. Kroó, J. Szabados. Inverse polynomial mappings and interpolation on several intervals. *Journal of Mathematical Analysis and Applications*. Volume 436, Issue 2, 2016, pp 1165-1179. https://doi.org/10.1016/j.jmaa.2015.12.032.

194. Gonzalez-Cardel M.F., Diaz-Uribe R. An analysis on the inversion of polynomials. Rev. mex. fís. vol.52, n.2, 2006, pp.163-171. https://www.scielo.org.mx/pdf/rmfe/v52n2/v52n2a9.pdf.

195. Chun-Myoung Park. A Study on Constructing the Inverse Element Generator over GF(3m). Internatinal Journalof Kimics. VOL. 8, NO. 3, JUNE 2010, pp. 317-322.

196. Khan A.G., Basharat S., Riaz M.U. Analysis of asymmetric cryptography in information security based on computational study to ensure confidentiality during information exchange. *International Journal of Scientific & Engineering Research*. Vol. 9, Iss. 10, 2018, pp. 992–999. https://doi.org/10.13140/RG.2.2.30495.61602.

197. Peter R. Turner Residue polynomial systems. *Theoretical Computer Science*. Vol.279 (1-2), 2002, pp. 29–49. https://doi.org/10.1016/S0304-3975(00)00425-4

198. Takagi T., Naito S. Construction of RSA Cryptosystem over the Algebraic Field Using Ideal Theory and Investigation of Its Security. *Electronics and Communications in Japan (Part III Fundamental Electronic Science)*. Vol. 83 (8), 2000, pp. 19-29. DOI:10.1002/(SICI)1520-6440(200008)83:83.3.CO;2-S.

199. Karpinski M., Rajba S., Zawislak S., Warwas K., Kasianchuk M., Ivasiev S., Yakymenko I. A method for decimal number recovery from its residues based on

the addition of the product modules. *In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1, 2019, pp. 13–17. https://doi.org/10.1109/IDAACS.2019.8924395.

200. Nykolaychuk Ya.M., Kasianchuk M.M., Yakymenko I.Z. Theoretical Foundations for the Analytical Computation of Coefficients of Basic Numbers of Krestenson's Transformation. *Cybernetics and Systems Analysis*. Vol. 50 (5), 2014, pp. 649-654. DOI:10.1007/s10559-014-9654-0.

201. Hayder R.H. H-Rabim Cryptosystem. Journal of Mathematics and Statistics. Vol. 10 (3), 2014, pp. 304-308.

202. Ivasiev S., Kasyanchuk M., Yakymenko I., Gomotiuk O., Shylinska I., Bilovus L. Algorithmic Support for Rabin Cryptosystem Implementation Based on Addition. Advanced Computer Information Technology (ACIT–2020): Proceedings of the International Conference. Deggendorf (Germany). 2020, pp. 779-782.

203. El-Kassar A.N., Haraty R.A., Awad Y.A. Rabin Public-key Cryptosystemin Rings of Polinomias Over Finite Fields. The International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04, December 2004). At: Cairo, Egypt.2004. URL:
https://www.researchgate.net/publication/323934925_RABIN_PUBLICEY_CRYPTOSYSTEM_IN_RINGS_OF_POLYNOMIALS_OVER_FINITE_FIELDS

204. El-Kassar A. N., Rizk M., Mirza N. M., Awad Y. A. El-Gamal public key cryptosystem in the domain of Gaussian integers. *Int. J. Appl. Math*.Vol. 7, No. 4, 2001, pp. 405-412.

205. El-Kassar A. N., Haraty R.A. ElGamal Public-Key Cryptosystem Using Reducible Polynomials Over a Finite Field. *IASSE 2004*, pp. 189-194.

206. Gafitoiu I. B. Polynomial based RSA. Linnaeus University. Sweden. 2015, p. 34
URL:http://www.diva-portal.se/smash/get/diva2:823505/FULLTEXT01.pdf

207. Lawnik M., Kapczynski A. The application of modified Chebyshev polynomials in asymmetric cryptography. *Computer Science*. №20(3). 2019, pp. 367-381. DOI:10.7494/csci.2019.20.3.3307.

208. Vairachilai S., Kavithadevi M.K., Gnanajeyaraman R. Public Key Cryptosystems Using Chebyshev Polynomials Based on Edge Information. Proceedings of the 2014 World Congress on Computing and Communication Technologies, Trichirappalli, India, 2014, pp. 243-245. DOI: 10.1109/WCCCT.2014.21.

209. Li Z.-H., Cui Y.-D., Xu H.-M. Fast algorithms of public key cryptosystem based on Chebyshev polynomials over finite field. The Journal of China Universities of Posts and Telecommunications. Vol. 18 (2), 2011, pp. 86-93, https://doi.org/10.1016/S1005-8885(10)60049-0.

210. Rajesh Pratap Singh R. H. Permutation Polynomials and their Applications in Cryptography: Permutation polynomials and multivariate public key cryptography. *LAP LAMBERT Academic Publishing : Paperback*, 2012, 92 p.

211. Emmanuel Thomé. Square Root Algorithms for the Number Field Sieve. *WAIFI 2012: Arithmetic of Finite Fields*, 2012, pp. 208-224.

212. Migliore V., Real M. M., Lapotre V., Tisserand A., Fontaine C., Gogniat G. Fast polynomial arithmetic for Somewhat Homomorphic Encryption operations in hardware with Karatsuba algorithm. 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 2016, pp. 209-212. doi: 10.1109/FPT.2016.7929535.

213. Jayet-Griffon C., Cornelie M.-A., Maistri P., Elbaz-Vincent P., Leveugle R. Polynomial multipliers for fully homomorphic encryption on FPGA. 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Riviera Maya, Mexico, 2015, pp. 1-6. doi: 10.1109/ReConFig.2015.7393335.

214. Kaustubh Shivdikar, Gilbert Jonatan, Evelio Mora, Neal Livesay, Rashmi Agrawal, Ajay Joshi, José L. Abellán, John Kim, David Kaeli Accelerating Polynomial Multiplication for Homomorphic Encryption on GPUs.

2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED).2022, P. 61-72. DOI Bookmark: 10.1109/SEED55351.2022.00013

215. Jianhua Wu, Hai Liu, Xishun Zhu Image encryption based on permutation polynomials over finite fields Optica Applicata, Vol. L, No. 3, 2020, pp. 357-376. DOI: 10.37190/oa200303

216. Bejo Alamsyah A., Adji T. B. The replacement of irreducible polynomial and affine mapping for the construction of a strong S-box. *Nonlinear Dynamics*, vol. 93, no. 4, 2018, pp. 2105–2118.

217. Kouther Fahad Alshammaria, Ayman Mostafaa, Shadi Nashwana. Avalanche Analysis of Variant Polynomials for AES. *Turkish Journal of Computer and Mathematics Education*. Vol.12 No.14, 2021, pp.2696- 2703.

218. Chu J., Benaissa M. Error detecting AES using polynomial residue number systems. *Microprocessors and Microsystems*. Vol. 37, Issue 2, 2013, pp. 228-234. https://doi.org/10.1016/j.micpro.2012.05.010.

219. EL-KASSAR, Abdul Nasser, et al. Modified RSA in the Domains of Gaussian Integers and Polynomials Over Finite Fields. In: *CAINE*. 2005. p. 298-303.

220. Амербаев В.М. Теоретические основы машинной арифметики. Алма–Ата: Наука, 1976, 324 с.

221. Beletsky A., Kovalchuk A., Novikov K., Poltoratskyi D. Algorithm for the synthesis of irreducible polynomials of linear complexity. *Ukrainian Information Security Research Journal.* 22, 2020, pp. 74–87.

222. Daniel Panario, Alfredo Viola. Analysis of Rabin's Polynomial Irreducibility Test Special Issue:Analysis of Algorithms Dedicated to Don Knuth, Volume19, Issue3-4, October - December 2001, pp. 525-551.

223. Yakymenko I., Kasanchuk M., Ivasiev S., Shevchuk R., Batko Yu., Vasylkiv V. Method for Determining Prime and Relatively Prime Numbers of 2^n+k Type Based on the Periodicity Property. *Proceedings of the 10th International Conference "Advanced Computer Information Technology (ACIT 2020)"*, (Deggendorf, Germany). 2020, pp. 751-754.

224. Adki V., Hatkar S. A Survey on Cryptography Techniques. International Journal of Advanced Research in Computer Science and Software Engineering. Vol. 6 (6), 2016, pp. 469-475.

225. Tot Ivan A., Bajčetič Jovan B., Jovanovič Boriša C., Trikoš Mladen B. Bogičevič Dušan Lj, Gajič Tamara M. Biometric standards and methods. *Vojnotehnicki glasnik/Military Technical Courier*, vol. 69, no. 4, 2021. DOI: https://doi.org/10.5937/vojtehg69-32296.

226. Balajishanmugam, V. High-performance computing based on residue number system: a review. In 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1, pp. 639-647.

227. Okeyinka A. Computational Speeds Analysis of RSA and ElGamal Algorithms. *Proceedings of the World Congress on Engineering and Computer Science (WCECS 2015)*, San Francisco (USA), V. I, 2015, pp. 237-242.

228. Stallings W. Cryptography and Network Security: Principles and Practice. *5th Prentice Hall Press Upper Saddle River*, NJ, USA, 2010, 719 p.

229. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. *CRC Press*, 2003, 780 p.

230. Amalraj A. J., Raybin Jose J. J. A survey paper on cryptography techniques International Journal of Computer Science and Mobile Computing. Vol. 5, Issue. 8, 2016, pp.55 – 59.

231. Valarmathy N., Vishnupriya P. Network Security and Cryptography Techniques. *Networkihg and Communication Enginee*ring. Vol.9, №9, 2017, pp. 229-231.

232. Song Y. Cryptanalytic attacks on RSA. *Springer Science and Business Media*, Inc., 2008, 255 p.

233. Yakymenko I., Kasyanchuk M., Nykolaychuk Ya. Matrix Algorithms of Processing of the Information Flow in Computer Systems Based on Theoretical and Numerical Krestenson's Basis. *Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET–2010): Proceedings of the X–th International Conference*. L'viv–Slavske. 2010, P.241.

234. Shoup V. A Computational Introduction to Number Theory and Algebra. *Cambridge University Press*, 2005, 517 p.

235. Panario D. Combinatorial and algebraic aspects of polynomials over finite fields. Tech. Rep. 306/97, Department of Computer Science, University of Toronto, 1997. PhD Thesis.

236. Karpiński M., Ivasiev S., Yakymenko I., Kasianchuk M., Gancarczyk T. Advanced method of factorization of multi-bit numbers based on Fermat's theorem in the system of residual classes. *Proc. of 16th International Conference on Control, Automation and Systems (ICCAS–2016)*. Gyeongju, Korea. V.1, October, 2016, pp.1484–1486.

237. Elia Michele, Piva Matteo, Schipani Davide The Rabin cryptosystem revisitet. *Applicable Algebra in Engineering, Communication and Computing*, 26(3), 2015, pp.251-275. https://doi.org/10.5167/uzh-128310.

238. D.G. Cantor, E. Kaltofen, On fast multiplication of polynomials over arbitrary algebras, *Acta Inform*, 28, 1991, pp. 693–701.

239. Dasgupta S., Papadimitriou C., Vazirani U. Algorithms. McGraw-Hill Science, Engineering, 2006, 336 p.

240. Касянчук М.М., Якименко І.З., Івасьєв С.В. Криптосистема Рабіна на основі операції додавання. *Математичне та комп'ютерне моделювання: Технічні науки.* 2019. В.19. С.145-150.

241. Kasianchuk, M., Yakymenko, I., Yatskiv, S., Gomotiuk, O., Bilovus, L.The Method of Joint Execution of the Basic Operations of the Rabin Cryptosystem CEUR Workshop Proceedings, 2023, 3373, pp. 425–436.

242. Beuchat J.-L. Some Modular Adder and Multipliers for Field Programmable Gate Arrays. *Parallel and Distributed Processing: IEEE Proceedings of International Symposium.* Vol.17, 2010, pp.8-11.

243. Wu H. Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis. *IEEE Transactions on Computers*. Vol.51 (7), 2002, pp. 151-155.

244. Hardy G.H., Wright E.M., Wiles A. An Introduction to the Theory of Numbers. *Oxford University Press*. 2008, 656 p.

245. Jeffrey H., Jill P., Joseph H. An Introduction to Mathematical Cryptography. *Berlin: Springer*, 2008. 540 p.

246. Zhengbing Hu., Dychka I., Onai M., Bartkoviak A. The Analysis and Investigation of Multiplicative Inverse Searching Methods in the Ring of Integers Modulo M. *International Journal of Intelligent Systems and Applications (IJISA)*. Vol. 8, №11, 2016, P. 9-18.

247. Verkhovsky B. Enhanced Euclid Algorithm for Modular Multiplicative Inverse and Its Application in Cryptographic Protocols. *IJCNS* , *3*, 2010, pp.901–906.

248. Hoffstein J., Pipher J., Silverman J. An Introduction to Mathematical Cryptography. *Springer Science+Business Media*, LLC, 2008, 524 p.

249. Marc Joye GCD-Free Algorithms for Computing Modular Inverses. Joye Marc, Paillier Pascal. CHES 2003: Cologne, Germany. 2003, pp. 243-253.

250. Bufalo M., Bufalo D., Orlando G. A Note on the Computation of the Modular Inverse for Cryptography. Axioms 2021, 10, 116. DOI:10.3390/axioms10020116.

251. І. З. Якименко, М. М. Касянчук, С. В. Івасьєв Криптосистема Рабіна на основі операції додавання // Математичне та комп'ютерне моделювання. Серія: Технічні науки № 19, 2019. 145–150 с.

252. Patel R.A., Benaissa M., Boussakta S. Fast parallel-prefix architectures for modulo 2^n-1 addition with a single representation of zero. *IEEE Transactions on Computers.* 56, 2007, pp. 1484–1492.

253. M. Elia, D. Schipani. On the Rabin signature. *J. Discrete Math. Sci. Cryptogr*., Vol. 16, no.6, 2013, pp.367-378.

254. Igor Yakymenko, Inna Shylinska, Mykhailo Kasianchuk, Lesia Bilovus, Oksana Gomotiuk Algorithmic Support for Rabin Three-Modular Cryptosystem Based on the Operation of Addition. *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT*). 2020, pp. 328-331.

255. Kalimoldayev M., Tynymbayev S., Magzom M. Software-hardware facilities for cryptosystems based on polynomial RNS. Problems of Informatics. N 4, 2018, pp. 73–84.

256. Yongnan Li, Limin Xiao, Aihua Liang, Yao Zheng, Li Ruan. Fast Parallel Garner Algorithm for Chinese Remainder Theorem. *9th International Conference on Network and Parallel Computing (NPC),* Sep 2012, Gwangju, South Korea. 2012, pp.164-171.

257. Mykhailo Kasianchuk, Ihor Yakymenko, Mikolaj Karpinski, Ruslan Shevchuk, Volodymyr Karpinskyi, Inna Shylinska. Theoretical Bases for Reducing the Time Complexity of the Rabin Cryptosystem. Conference on Computer Science and Information Technologies. 2020. Pp. 628-639.

258. Якименко І.З., Касянчук М.М., Волинський О.І., Пітух І.Р. Теорія алгоритмів RSA та Ель–Гамаля в розмежованій системі числення Радемахера–Крестенсона. *Вісник Хмельницького національного університету.* Технічні науки. №3, 2011, С. 265-273.

259. Kasianchuk M. M. The theory and mathematical regularities of perfect form of residue number system. in: Trans. Intern. Symp. *Problems of optimization of computations (POC–XXXV).* Vol. 1, Kyiv–Katsiveli, 2009, pp. 306–310.

260. Якименко І. З. Удосконалення реалізації криптоалгоритму Ель-Гамаля на основі системи залишкових класів. *Інформатика та математичні методи в моделюванні*, 2018, 8, № 1. С. 69-77.

261. Якименко І.З., Касянчук М.М., Кінах Я.І., Власюк І.М., Суслін В.В. Удосконалення реалізації асиметричних криптоалгоритмів на основі системи залишкових класів. Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп’ютерні інформаційні технології» (ACIT), 2018. С. 79.

262. Kasianchuk, M., Yakymenko, I., Yatskiv, V., Karpinski, M., Yatskiv, S. Method of Multi-Bit Numbers Multiplication in Residue Number System for Asymmetric Cryptosystems. CEUR Workshop Proceedings, 2022, 3156, pp. 365–377. https://ceur-ws.org/Vol-3156/paper27.pdf .

263. Касянчук М. М., Якименко І. З., Івасьєв С. В., Мандебура Н. М., Неміш В. М. Дослідження часових характеристик апаратної реалізації методів пошуку оберненого елемента за модулем. Вісник Хмельницького національного університету. Технічні науки. 2017. № 6 (255). С. 191–197.

264. Rajba T., Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of Time Characteris-tics of Search Methods of Inverse Element by the Module. Proceedings of the 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2017). 2017. Vol. 1. P. 82–85.

265. Николайчук Я. М., Касянчук М.М, Якименко І. З., Івасьєв С. В. Ефективний метод модулярного множення в теоретико-числовому базисі Радемахера–Крестенсона. Вісник Національного університету "Львівська політехніка". Комп'ютерні системи та мережі. - 2014. - № 806. - С. 195-199. - Режим доступу: http://nbuv.gov.ua/UJRN/VNULPKSM_2014_806_31.

266. Zuras D., More on squaring and multiplying larges integers, IEEE Transactions on Computers,-1994.- Vol. 43, - № 8,- P. 899–908.

267. Березовский А.И. О тестировании быстродействия алгоритмов и программ вычисления основных операций ассиметричной криптографии/ Березовский А.И., Задирака В.К., Шевчук Л.Б. //Кибернетика и системный анализ №5, 1999. – С. 59-66.

268. Giorgi P. Parallel modular multiplication on multi-core processors. / Giorgi P., Imbert L., Izard T. // IEEE Symposium on Computer Arithmetic, Apr 2013, Austin, TX, United States. - P.135-142.

269. Buhrow B. Parallel modular multiplication using 512-bit advanced vector instructions: RSA fault-injection countermeasure via interleaved parallel multiplication / Buhrow B., Gilbert B., Haider C. // Journal of Cryptographic Engineering.-2021.- № 2.- P.46-53.

270. Dimauro Giovanni et al., RNS architectures for the implementation of thediagonal function. Information processing letters 73, vol. 5, 2000, pp. 189-198.

271. Iakymenko I., Kasianchuk M., Kinakh I., Karpinski M. Construction of distributed thermal or piezoelectric sensor based on residue systems. *Przeglad Elektrotechniczny*. 2017. №1. P. 290-294.

272. David Vigilant RSA with CRT: A new cost-effective solution to thwart fault attacks. Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008, pp 130-145. DOI:10.1007/978-3-540-85053-3_9.

273. Aho A.V., Hopcroft J.E., Ullman J.D. The Design and Analysis of Computer Algorithms. *Wesley Publishing Company*, 1974, 480 p.

274. Parthasarathy S. Multiplicative inverse in mod(m). *Algologic Technical Report*. №1, 2012, pp. 1-3.

275. Якименко І. Алгоритми побудови модифікованої досконалої форми системи залишкових класів. *Спеціалізовані комп'ютерні технології в інформатиці: Колективна монографія*. Під ред. В.Задіраки, Я.Николайчука. Тернопіль: Бескиди, 2017. С. 580-604.

276. Mykhailo Kasianchuk, Ihor Yakymenko, Vasyl Yatskiv, Stepan Ivasiev, Andriy Sverstiuk. Same Bit-Size Moduli Formation of Residue Number System for Application in Asymmetric Cryptography. IntelITSIS 2021. pp. 301-308.

277. Касянчук М.М., Якименко І.З., Івасьєв С.В., Маслияк Б.О. Метод розширення набору модулів модифікованої досконалої форми системи залишкових класів. *Математичне та комп'ютерне моделювання: Технічні науки.* 2017. В.15. С.73-78.

278. Kasianchuk M.N., Nykolaychuk Ya.N., Yakymenko I.Z. Theory and Methods of Constructing of Modules System of the Perfect Modified Form of the System of Residual Classes. *Journal of Automation and Information Sciences*. Vol.48, №8, 2016, pp.56-63.

279. Ananda Mohan P.V. Residue Number Systems: Theory and Applications. *Birkhäuser,* 2016, 351 p.

280. М.М. Касянчук, І.З. Якименко, Я.М. Николайчук Симетричні криптоалгоритми у системі залишкових класів. Cybernetics & Systems Analysis, 2021, Vol 57, Issue 2, p.184-189.

281. Stanislaw Zawislak, Mykhailo Kasianchuk, Igor Iakymenko, Daniel Jan carczyk Methods of Crypto-stable Symmetric Encryption in the Residual Number System. *Procedia Computer Science*. Volume 207, 2022, pp. 128-137.

282. Swidan O., Zilic Z. A. Direct residue-to-analog conversion scheme based on Chinese remainder theorem. *In: 2010 IEEE International Conference on Electronics*, Circuits, and Systems, 2010, pp. 687–690.

283. Bogdanov A., Khovratovich D., Rechberger C. Biclique cryptanalysis of the full AES. *ASIACRYPT-2011*. LNCS. V. 7073, 2011, pp. 344–371.

284. Dobbertin Hans, Knudsen Lars, Robshaw Matt. The Cryptanalysis of the A Brief Survey. *AES 2004*, LNCS 3373, 2005, pp. 1–10.

285. Schoinianakis D. Residue arithmetic systems in cryptography: a survey on modern security applications. J. Cryptogr. Eng., vol. 10, no. 3, 2020, pp. 249–267.

286. Nykolaychuk Ya.M., Yakymenko I.Z., Vozna N.Ya., Kasianchuk M.M. Residue Number System Asymmetric Cryptoalgorithms. *Cybernetics and Systems Analysis*. Vol. 58, No. 4, 2022, pp.611-618. https://doi.org/10.1007/s10559-022-00494-7.

287. Suárez-Albela M., Fraga-Lamas P., Fernández-Caramés T.M. A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices. *Sensors 2018*. 18, 3868; https://doi.org/10.3390/s18113868.

288. Ahmed Muhammad Hussein, Sadeq Taha Abdulazeez. The Existence of a Polynomial Inverse Integrating Factors and Studies About the Limit Cycles for Cubic, Quartic and Quintic Polynomial Systems. *Baghdad Science Journal*. 2021, pp. 322-329. DOI: http://dx.doi.org/10.21123/bsj.2021.18.2.0322

289. Moreno J., Saiz A. Inverse functions of polynomials and its applications to initialize the search of solutions of polynomials and polynomial systems. Numer Algor 58, 2011, pp. 203–233. https://doi.org/10.1007/s11075-011-9453-x.

290. Yuki Katagiri, Kazuaki Iwamura, Yosuke Nakanishi, Sachio Takano, Ryohei Suzuki. Arbitrary polynomial chaos expansion and its application to power flow analysis-Fast approximation of probability distribution by arbitrary polynomial expansion. *Journal of Physics: Conference Series* 1780, 2021, 012025. doi:10.1088/1742-6596/1780/1/012025.

291. J. B. Lasserre, "Inverse polynomial optimization," *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, USA, 2011, pp. 2794-2799. doi: 10.1109/CDC.2011.6160364.

292. William D. Banks, Daniel Lieman, Igor E. Shparlinski, Van Thuong To. Cryptographic Applications of Sparse Polynomials over Finite Rings. *ICISC 2000*, pp. 206-220

293. Krausz Markus, Land Georg, Richter-Brockmann Jan, Güneysu Tim. Efficiently Masking Polynomial Inversion at Arbitrary Order. *Cryptology ePrint Archive*, Paper 2022/707. https://ia.cr/2022/707.

294. Drucker Nir, Gueron Shay, Kostic Dusan. Fast polynomial inversion for post quantum QC-MDPC cryptography. Information and Computation, Volume 281, 2021, 104799, ISSN 0890-5401, https://doi.org/10.1016/j.ic.2021.104799.

295. Kasianchuk M., Yakymenko I., Nykolaychuk Y. Symmetric Cryptoalgorithms in the Residue Number System. *Cybernetics and Systems Analysis*. Vol. 57(2), 2021, pp. 329–336. https://doi.org/10.1007/s10559-021-00358-6.

296. Paliouras V., Skavantzos A., Stouraitis T. Low power convolvers using the Polynomial Residue Number System. *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, Phoenix-Scottsdale, AZ, USA, 2002, pp. II-II, doi: 10.1109/ISCAS.2002.1011461.

297. Chu Junfeng, Benaissa Mohammed. GF(2m) multiplier using Polynomial Residue Number System. *IEEE Asia-Pacific Conference on Circuits and*

Systems, Proceedings, APCCAS. 2009, pp.1514 - 1517. 10.1109/APCCAS.2008.4746320.

298. Boucher D., Ulmer F. Linear Codes using Skew Polynomials with Automorphisms and Derivations. *Designs, codes and cryptography*. 70(3), 2014, pp. 405–431.

299. Puchinger Sven, Wachter-Zeh Antonia. Fast operations on linearized polynomials and their applications in coding theory. *Journal of Symbolic Computation*. Volume 89, 2018, pp. 194-215, https://doi.org/10.1016/j.jsc.2017.11.012.

300. Dadkhahi H., Gotchev A., Egiazarian K. Inverse polynomial reconstruction method in DCT domain. *EURASIP J. Adv. Signal Process*. 2012, Article number 133. 2012. https://doi.org/10.1186/1687-6180-2012-133.

301. Jassim Wissam, Raveendran P., Mukundan Ramakrishnan. New orthogonal polynomials for speech signal and image processing. *Signal Processing IET*. Vol. 6 (8), 2012, pp. 713-723. DOI: 10.1049/iet-spr.2011.0004.

302. Ashraphijuo M., Madani R., Lavaei J. Inverse function theorem for polynomial equations using semidefinite programming. *2015 54th IEEE Conference on Decision and Control (CDC)*, Osaka, Japan, 2015, pp. 6589-6596, doi: 10.1109/CDC.2015.7403257.

303. Mudassir Shams, Naila Rafiq, Nasreen Kausar, Shams Forruque Ahmed, Nazir Ahmad Mir, Suvash Chandra Saha. Inverse Family of Numerical Methods for Approximating All Simple and Roots with Multiplicity of Nonlinear Polynomial Equations with Engineering Applications Mathematical Problems in Engineering. Volume 2021, Article ID 3124615, 9 p. https://doi.org/10.1155/2021/312461.

304. Kasianchuk M., Yakymenko I., Ivasiev S. Theoretical foundations for creating five modular modified perfect form of the system of residual classes, in *Inżynier XXI Wieku: VII Międzynarodowa Konferencja studentow oraz doktorantow, 08.12.2017: monografia*, 1st ed., Vol.2., Bielsko – Biała (Poland): Akademia Techniczno-Humanistyczna w Bielsku-Białej, 2017, pp. 123-130. Chapter in monograph.

305. Cheon J.H., Hong S., Lee C., Son Y. Polynomial functional encryption scheme with linear ciphertext size. *Cryptology ePrint Archive* 2018. https://ia.cr/2018/585

306. R. Shevchuk, I. Yakymenko, M. Karpinski, I. Shylinska and M. Kasianchuk, "Finding the inverse of a polynomial modulo in the ring Z[x] based on the method of undetermined coefficients", Computer Science, vol. 25, no. 2, 2024.

307. M. Freed. RSA Encryption Using Polynomial Rings. *Point Loma Nazarene University*. 2018, p. 17.

308. Mahatab K., Sampath K. Chinese remainder theorem for cyclotomic polynomials in Z[X]. *Journal of* Algebra. Vol. 435, 2015, pp. 223-262.

309. Yakymenko, M. Kasianchuk, I. Shylinska, R. Shevchuk, V. Yatskiv, M. Karpinski. Polynomial Rabin Cryptosystem Based on the Operation of Addition. *Proceedings of the 12th International Conference "Advanced Computer Information Technology* (ACIT-2022)" 2022, pp. 345–350. DOI: 10.1109/ACIT54803.2022.9913089.

310. Harvey D., Hoeven J., Scheidler R., Sorenson J. Faster integer multiplication using short lattice vectors. *Proceedings of the Thirteenth Algorithmic Number Theory Symposium*, Open Book Series 2, Mathematical Sciences Publishers, Berkeley (2019), pp. 293-310, https://doi.org/10.2140/obs.2019.2.293 .

311. Gashkov S.B., Sergeev I. S. Complexity of computation in finite fields. *Fundamentalnaya i prikladnaya matematika*. Vol. 17 (2011/2012), No. 4, 2012, pp. 95—131.

312. Zadiraka V.K., Kudin A.M. New models and methods for estimating the cryptographic strength of information security systems. *Cybernetics and Systems Analysis.* Vol. 53, N 6, 2017, P. 978–985. https://doi.org/10.1007/s10559-017-9999-2.

313. Yakymenko, I., Karpinski, M., Shevchuk, R., Kasianchuk, M. Symmetric Encryption Algorithms in a Polynomial Residue Number System. Journal of Applied Mathematics., 2024, 2024, pp. 1-12. DOI:10.1155/2024/4894415

314. Kapalova N., Dyusenbayev D. Security analysis of an encryption scheme based on nonpositional polynomial notations. *Open Engineering 2016*, 6. https://doi.org/10.1515/eng-2016-0034.

315. Bajard J.C., Imbert L., Plantard T. Arithmetic operations in the polynomial modular number system. In Proceedings of the 17th IEEE Symposium on Computer Arithmetic (ARITH'05). IEEE, 2005, pp. 206–213. https://doi.org/10.1109/ARITH.2005.11.

316. Shivdikar K., Jonatan G., Mora E., Livesay N., Agrawal R., Joshi A., Abellán J.L., Kim J., Kaeli D. Accelerating polynomial multiplication for homomorphic encryption on GPUs. *In Proceedings of the 2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*, 2022, pp. 61–72. https://doi.org/10.1109/SEED55351.2022.00013.

317. Jastrzebska Magdalena On the Properties of the Möbius Function. January 2006. Formalized Mathematics 14(1). 2006, pp. 29-36. DOI:10.2478/v10037-006-0005-0

318. C. D. Godsil An Introduction to the Moebius Function. URL: http://www.kmu.gov.ua/control/uk/photogallery/gallery?gall eryId=15725757& (дата звернення: 15.11.2017). https://discuss.codechef.com/questions/72953/a-dance-with-mobius-function

319. Schneier B. The BlowFish encryption algorithm. Link, 2008, [Online; Accessed 1 July 2020]. [Online]. Available: Link.

320. Sedgewick Robert, Wayne Kevin. Algorithms, Fourth Edition. *Addison-Wesley Professional*. 2020, 956 p.

321. Yakymenko, I., Kasianchuk, M., Martyniuk, O., & Martyniuk, S. (2025). A Symmetric Cryptoalgorithm Based on a Hierarchical Residue Number System. *International Journal of Computing*, 24(1), pp. 92-101. https://doi.org/10.47839/ijc.24.1.3880

322. Yakymenko, I., Martyniuk, O., Martyniuk, S., Yakymenko, Y., Kasianchuk, M. Hierarchical Encryption in a Residual Number System Proceedings

- International Conference on Advanced Computer Information Technologies, ACITThis link is disabled., 2024, pp. 496–499 (Scopus).

323. Kasyanchuk M., Karpinski M., Kochan R., Karpinskyi V., Litawa G., Shylinska I., Yakymenko I. Developing Symmetric Encryption Methods Based on Residue Number System and Investigating Their Cryptosecurity. *IACR Cryptol.* ePrint Arch. Report 2020/589 https://www. semanticscholar.org/paper/ De-veloping-Symmetric-Encrypti-on-Methods-Based-On-Kasian-chuk-Karpinski /96b75ca8c900 1122a4b8b110e5bc21c9e858fb1d

324. I. Yakymenko, M. Kasianchuk, I. Shylinska A Method for Polynomial Recovery from its Resudues Based on Addition in Z[x] Ring. Informatics and Mathematical Methods in Simulation Vol.14 (2024), No. 4, pp. 305-313. DOI 10.15276/imms.v14.no4.305

325. Tadeusz Tomczak. Hierarchical residue number systems with small moduli and simple converters. *International Journal of Applied Mathematics and Computer Science*. Vol. 21 (2011), ISSUE 1, March 2011, pp.173-192.

326. Bajard J.-C., Marrez J., Plantard T., Véron P. On Polynomial Modular Number Systems over Z/pZ. *Advances in Mathematics of Communications (in Press)*. 2022. DOI: 10.3934/amc.2022018. URL: https://arxiv.org/abs/2001.03741

327. Bard, G.V. Algebraic Cryptanalysis; Springer US: Boston, MA, USA, 2009; ISBN 978-0-387-88756-2.

328. Nover, H. Algebraic Cryptanalysis of AES: An Overview; University of Wisconsin: Madison, WI, USA, 2005.

329. Якименко І.З. Методологія криптографічного захисту інформації на основі цілочисельної, модифікованої досконалої та поліноміальної систем залишкових класів, *Матеріали XIV Міжнародної науково-технічної конференції ITSec-2025 Безпека інформаційних технологій*, 22-24 травня 2025, м. Тернопіль (Україна), с. 224-228.

330. Shevchuk, R., Karpinski, M., Kasianchuk, Yakymenko, I., M., Melnyk, A., Tykhyi, R. Software for Improve the Security of Kubernetes-based CI/CD

Pipeline. Proceedings - International Conference on Advanced Computer Information Technologies, ACIT2023, 2023, pp. 420–425.

331. Пат. 160091 Україна МПК G06F7/04. Пристрій порівняння даних, представлених у непозиційній системі залишкових класів / Николайчук Я.М., Якименко І.З., Івасьєв С.В, Грига В.М. № u202404328 заявл. 03.09.2024; опубл. 06.08.2025, Бюл. № 32/2025.

Додаток А

Документи, що підтверджують впровадження результатів дисертаційної роботи

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи
Західноукраїнського національного університету
_____ Віктор ОСТРОВЕРХОВ
«__» _____ 2025 р.

АКТ

впровадження у навчальний процес результатів дисертаційної роботи Якименка Ігоря Зіновійовича «Методи та засоби криптографічного захисту інформації на основі системи залишкових класів», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що результати, отримані в дисертаційній роботі Якименка Ігоря Зіновійовича, зокрема: побудова трьохмодульної криптосистеми Рабіна, алгоритмічне забезпечення для реалізації криптосистем Ель-Гамаля на основі векторно-модульного методу та системи залишкових класів (СЗК), розробка симетричних та асиметричних цілочисельних криптосистем, криптосистем на основі китайської теореми про залишки, криптосистем з використанням поліноміальних систем залишкових класів, векторно-модульних методів пошуку залишків, симетричних методів шифрування/розшифрування інформаційних потоків в ієрархічній цілочисельній та поліноміальній СЗК, методу невизначених коефіцієнтів, методів відновлення поліному за його залишками на основі додавання добутку модулів або їх залишків, програмні рішення методів шифрування/ розшифрування в СЗК та ПСЗК використовуються в навчальному процесі факультету комп'ютерних інформаційних технологій Західноукраїнському національному університеті при підготовці бакалаврів та магістрів спеціальності «Кібербезпека та захист інформації» при викладанні дисциплін «Криптографія», «Основи кібербезпеки», «Дослідження і проектування систем захисту інформації», при підготовці на спеціальності «Комп'ютерна інженерія» при викладанні дисциплін «Захист інформації в комп'ютерних системах», «Програмно-апаратні засоби захисту інформації».

Завідувач кафедри кібербезпеки
д.т.н., проф. Василь ЯЦКІВ

Завідувач кафедри компютерної інженерії
к.т.н., доц. Леся ДУБЧАК

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Західноукраїнського національного університету,
д.т.н., проф. Микола ДИВАК
«__» грудня 2025 р.

АКТ

про використання результатів докторської дисертації Якименка Ігоря Зіновійовича на тему «Методи та засоби криптографічного захисту інформації на основі системи залишкових класів».

Комісія у складі завідувача кафедри кібербезпеки факультету комп'ютерних інформаційних технологій, д.т.н., проф. Яцківа В.В. (голова комісії), начальника науково-дослідної частини, д.е.н., проф. Семанюк В.З., начальника відділу прогнозування і маркетингу Кушніра О.Р. склали цей акт про те, що дослідження та результати дисертаційної роботи Якименка І.З. використані під час виконання науково-дослідних робіт на кафедрах комп'ютерної інженерії та кібербезпеки факультету комп'ютерних інформаційних технологій з безпосередньою участю автора, а саме:

1) науково-дослідної роботи на тему: «Паралельні методи та засоби реалізації алгоритмів захисту інформації в комп'ютерних мережах з використанням математичного апарату еліптичних кривих» (виконавець), номер державної реєстрації 0109U000035, у якій автором розроблено методи оцінки значень еліптичних кривих;

2) науково-дослідної роботи на тему: "Опрацювання багаторозрядних чисел в системі залишкових класів" (виконавець), номер державної реєстрації 0115U001607, у якій автором розроблено методи модулярного множення та експоненціювання багаторозрядних чисел та здійснено оцінку їх часової складності;

3) науково-дослідної роботи на тему: "Розробка теоретичних засад методів формування та цифрового опрацювання даних у розподілених спеціалізованих комп'ютерних системах" (виконавець), номер державної реєстрації 0112U008458, у якій автором проведено дослідження часової складності розроблених алгоритмів опрацювання багаторозрядних чисел;

4) науково-дослідної роботи на тему: "Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж" (виконавець), номер державної реєстрації 0117U000414, у якій автором розроблено методи побудови наборів модулів модифікованої досконалої форми системи залишкових класів;

5) науково-дослідної роботи на тему: «Стійкі до криптоаналізу методи та засоби шифрування в поліноміальних системах» (керівник), номер державної реєстрації 0123U104713, у якій автором розроблено методи та засоби шифрування в поліноміальних системах залишкових класів та проведено дослідження їх складностей.

Голова комісії:
Завідувач кафедри кібербезпеки — Василь ЯЦКІВ

Члени комісії:
Начальник науково-дослідної частини — Віта СЕМАНЮК

Начальник планово-фінансового відділу — Олексій КУШНІР

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Якименка Ігоря Зіновійовича на тему "Методи та засоби криптографічного захисту інформації на основі системи залишкових класів"

Цей акт складено про те, що результати дисертаційної роботи Якименка Ігоря Зіновійовича на тему "Методи та засоби криптографічного захисту інформації на основі системи залишкових класів" розглянуті і схвалені для можливого застосування при плануванні заходів із покращення стану інформаційної безпеки..

Разом з дисертантом проводилися експериментальні дослідження програмної реалізації криптоалгоритмів на основі системи залишкових класів (СЗК): симетричного методу шифрування у СЗК, ієрархічного симетричного криптоалгоритму на основі СЗК, поліноміального симетричного криптоалгоритму в СЗК, асиметричних алгоритмів шифрування у СЗК (модифікацій криптосистем Рабіна та Ель-Гамаля).

Для забезпечення високого рівня захисту та підвищення швидкодії криптоалгоритмів рекомендуються до впровадження методи оптимізації криптографічних обчислень: метод невизначених коефіцієнтів для пошуку оберненого полінома в кільці Z[x], методи відновлення полінома в кільці Z[x] на основі додавання добутку модулів, векторно-модульні методи пошуку залишків, модулярного множення та експоненціювання.

Запропоновані підходи та результати отримані в дисертаційній роботі дозволять підвищити рівень захисту інформації підприємства завдяки застосуванню нових криптографічних алгоритмів на основі СЗК. Оптимізація обчислювальних процесів при виконанні криптографічних операцій дозволилить зменшити навантаження на серверне обладнання та прискорити

обробку захищених даних, що також сприятиме розширенню функціональних можливостей системи захисту інформації підприємства за рахунок впровадження різних типів криптоалгоритмів (симетричних та асиметричних). Забезпечено гнучкість налаштувань параметрів криптосистеми в залежності від необхідного рівня захисту та доступних обчислювальних ресурсів.

Отже, результати дисертаційної роботи Якименка І.З. мають практичну цінність для підвищення рівня інформаційної безпеки та оптимізації обчислювальних процесів при виконанні криптографічних операцій. Розроблені методи та програмні засоби рекомендуються для подальшого використання та розвитку в системі захисту інформації.

Заступник начальника Управління
кібербезпеки та цифрового розвитку-
начальник відділу цифрової трансформації
Міністерство енергетики України **Віталій БАЗАЛИЦЬКИЙ**

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Якименка Ігоря Зіновійовича на тему "Методи та засоби криптографічного захисту інформації на основі системи залишкових класів"

Цей акт складено про те, що результати дисертаційної роботи Якименка Ігоря Зіновійовича на тему "Методи та засоби криптографічного захисту інформації на основі системи залишкових класів" розглянуті і схвалені для можливого застосування при плануванні заходів із покращення стану інформаційної безпеки..

Разом з дисертантом проводилися експериментальні дослідження програмної реалізації криптоалгоритмів на основі системи залишкових класів (СЗК): симетричного методу шифрування у СЗК, ієрархічного симетричного криптоалгоритму на основі СЗК, поліноміального симетричного криптоалгоритму в СЗК, асиметричних алгоритмів шифрування у СЗК (модифікацій криптосистем Рабіна та Ель-Гамаля).

Для забезпечення високого рівня захисту та підвищення швидкодії криптоалгоритмів рекомендуються до впровадження методи оптимізації криптографічних обчислень: метод невизначених коефіцієнтів для пошуку оберненого полінома в кільці Z[x], методи відновлення полінома в кільці Z[x] на основі додавання добутку модулів, векторно-модульні методи пошуку залишків, модулярного множення та експоненціювання.

Запропоновані підходи та результати отримані в дисертаційній роботі дозволять підвищити рівень захисту інформації підприємства завдяки застосуванню нових криптографічних алгоритмів на основі СЗК. Оптимізація обчислювальних процесів при виконанні криптографічних операцій дозволилить зменшити навантаження на серверне обладнання та прискорити

обробку захищених даних, що також сприятиме розширенню функціональних можливостей системи захисту інформації підприємства за рахунок впровадження різних типів криптоалгоритмів (симетричних та асиметричних). Забезпечено гнучкість налаштувань параметрів криптосистеми в залежності від необхідного рівня захисту та доступних обчислювальних ресурсів.

Отже, результати дисертаційної роботи Якименка І.З. мають практичну цінність для підвищення рівня інформаційної безпеки та оптимізації обчислювальних процесів при виконанні криптографічних операцій. Розроблені методи та програмні засоби рекомендуються для подальшого використання та розвитку в системі захисту інформації.

Заступник начальника Управління
кібербезпеки та цифрового розвитку-
начальник відділу цифрової трансформації
Міністерство енергетики України **Віталій БАЗАЛИЦЬКИЙ**

АКЦІОНЕРНЕ ТОВАРИСТВО
«ТЕРНОПІЛЬОБЛЕНЕРГО»
Код ЄДРПОУ 00130725
вул. Енергетична, 2, м. Тернопіль, Україна, 46007
тел.: +38 (0352) 52-50-13, 52-15-03
e-mail: kanc@toe.com.ua
www.toe.com.ua

№ 5291/24 від 17.12.2025

ДОВІДКА

Даним листом засвідчується, що у АТ «Тернопільобленерго» використовує результати дисертаційної роботи Якименка Ігоря Зіновійовича «Методи та засоби криптографічного захисту інформації на основі системи залишкових класів», що стосуються розроблення та оптимізації сучасних криптографічних методів на основі системи залишкових класів (СЗК).

У процес впровадження включено такі результати наукової роботи:

- програмні модулі для прискореної обробки інформаційних потоків із застосуванням цілочисельної, поліноміальної та ієрархічної форм СЗК;
- розроблені методи симетричного та асиметричного шифрування з використанням СЗК, що забезпечують збільшення швидкодії процесів криптоперетворень при збереженні високої криптостійкості;
- алгоритмічні рішення для оптимізації відновлення числових та поліноміальних структур за їх залишками, адаптовані для застосування у системах енергетичного сектору;
- технічні рекомендації щодо інтеграції криптографічних компонентів в інформаційно-телекомунікаційну інфраструктуру підприємства.

Впроваджені напрацювання використані для модернізації системи захисту інформації, підвищення надійності передачі службових даних та зменшення ризиків несанкціонованого доступу до критично важливих інформаційних ресурсів підприємства.

В.о. Генерального директора **Володимир ГУМЕН**

==========

ТОВ "Науково виробнича фірма "Інтеграл"

ЄДРПОУ 32220131,

ІПН 322201319180, номер свідоцтва 100010157

==========

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи

Якименка Ігоря Зіновійовича

на тему:

«Методи та засоби криптографічного захисту інформації на основі системи залишкових класів»

Цим актом підтверджується, що результати наукового дослідження Якименка Ігоря Зіновійовича були впроваджені у практичну діяльність ТОВ "Науково виробнича фірма "Інтеграл" у межах розробки та модернізації корпоративної системи криптографічного захисту інформації. У процесі реалізації інноваційних підходів, викладених у дисертації, нами було адаптовано низку авторських алгоритмів, зокрема симетричні та асиметричні криптографічні механізми, побудовані на використанні системи залишкових класів, поліноміальних структур і модифікованих математичних формулювань, що дозволяють уникати операцій міжрозрядного переносу і суттєво підвищують швидкодію.

Під час впровадження було інтегровано поліноміальний симетричний алгоритм з двоключовою схемою шифрування, що дозволило розширити функціональні можливості захисту даних у внутрішньокорпоративному середовищі, а також модернізовано механізми генерації ключів у відповідності до математичного апарату модифікованої досконалої форми СЗК. Запропоновані підходи забезпечили суттєве зменшення навантаження на серверні ресурси, зокрема при виконанні повторюваних криптографічних

операцій, а також підвищили опірність до атак типу аналізу частот, обраного шифртексту та факторизаційного зламу.

Окремо відзначено доцільність використання запропонованого методу пошуку оберненого полінома в кільці Z[x], що дозволив оптимізувати процес побудови зворотних перетворень у внутрішніх криптографічних процедурах. У ході внутрішнього тестування підтверджено ефективність використання векторно-модульних методів модулярного експоненціювання в асиметричних алгоритмах, що реалізовані у форматі програмних бібліотек та інтегровані в систему електронного документообігу підприємства.

Упровадження здійснювалося згідно з внутрішнім наказом № 03-07/2025 від 7 березня 2025 року та погоджено технічною комісією компанії. Результати підтверджено актами випробувань програмного забезпечення, журналами криптографічного аудиту та аналітичними висновками про зменшення обчислювальних витрат при збереженні високого рівня криптостійкості. Запропоновані рішення включено до корпоративного реєстру схвалених засобів захисту інформації.

Таким чином, наукові розробки, здійснені Якименком І.З., знайшли практичне застосування в інфраструктурі інформаційної безпеки ТОВ "Науково виробнича фірма "Інтеграл" і використовуються як у щоденній роботі компанії, так і в рамках впровадження комплексних рішень для клієнтів із секторів фінансів, телекомунікацій і державного управління.

Директор ТОВ "НВФ "ІНТЕГРАЛ" О.С.Колос

ЄДРПОУ 32220131, ІПН 322201319180, номер свідоцтва 100010157
Поштова адреса: м. Тернопіль, вул. Текстильна, 32.

ЗАВОД
РЕМПОБУТТЕХНІКА
товариство з обмеженою відповідальністю

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи

Якименка Ігоря Зіновійовича

на тему:

«Методи та засоби криптографічного захисту інформації на основі системи залишкових класів»

ТОВ ЗАВОД "РЕМПОБУТТЕХНІКА" підтверджує, що результати дисертаційного дослідження Якименка Ігоря Зіновійовича були впроваджені в діяльність підприємства з метою підвищення ефективності та надійності криптографічного захисту інформації в умовах зростаючих кіберзагроз. Зокрема, були використані розроблені автором симетричні та асиметричні криптографічні алгоритми на основі цілочисельної, модифікованої досконалої, поліноміальної та ієрархічної систем залишкових класів. Запропоновані підходи дозволили реалізувати нову архітектуру криптографічних перетворень, яка передбачає заміну операцій множення на додавання, що забезпечило зменшення часової складності криптографічних операцій при збереженні або підвищенні рівня криптостійкості.

У рамках впровадження алгоритмів та методів, розроблених у дисертації, було оновлено програмне ядро криптографічної платформи підприємства та впроваджено окремі компоненти у системи захисту інформації клієнтів з критичною IT-інфраструктурою. Зокрема, реалізовано криптоалгоритми, що використовують ієрархічну СЗК для захисту міжрівневих інформаційних потоків, а також асиметричні алгоритми шифрування Рабіна з використанням векторно-модульного підходу та поліноміальної арифметики. Це дало змогу підвищити продуктивність криптографічних операцій на низькоресурсних пристроях на 25–30%, а також значно посилити захист даних від класичних і

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ ЗАВОД "РЕМПОБУТТЕХНІКА"
Україна, 46010, м. Тернопіль, вул. Текстильна, 32, тел. (0352) 283756 факс (0352) 283627 моб. 0672082663

комбінованих атак за рахунок використання систем числення, які ускладнюють математичне моделювання криптоаналітиком.

Розроблені автором підходи знайшли застосування також у внутрішній документації підприємства як методологічна основа для розробки технічних вимог до нових засобів захисту інформації. На підставі проведених експериментальних досліджень та аудиту продуктивності криптографічних компонентів підтверджено доцільність і ефективність впровадження зазначених рішень у практичну діяльність організації. Впровадження здійснювалося у межах технічного завдання № ЦКБ/04-25 від 10 лютого 2025 року, результати якого затверджені технічною радою підприємства.

Таким чином, результати дисертаційного дослідження Якименка І.З. не лише підтверджують свою наукову і практичну значущість, а й стали основою для створення інноваційних продуктів у сфері прикладної криптографії, що відповідають актуальним вимогам інформаційної безпеки в умовах сучасного кіберпростору.

Директор Л. П. Слободянюк

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ ЗАВОД "РЕМПОБУТТЕХНІКА" УКРАЇНА ЗАВОД РЕМПОБУТТЕХНІКА 03056917 ТЕРНОПІЛЬ

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ ЗАВОД "РЕМПОБУТТЕХНІКА"
Україна, 46010, м. Тернопіль, вул. Текстильна, 32, тел. (0352) 283756 факс (0352) 283627 моб. 0672082663

Додаток Б

Лістинги кодів програмних модулів

Програмний модуль Main.py

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QTabWidget, QTabBar, QPushButton, QLineEdit, QLabel, QHBoxLayout, QStylePainter, QStyleOptionTab, QStyle, QProxyStyle, QGraphicsDropShadowEffect, qApp
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtCore import QRect, QPoint, Qt
import CustomTabWidget
import DarkOrangePalette
from Navbar import NavigationBar
from LayerMainLayout import LayerMainLayout
from ConfigDialog import ConfigDialog
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        dialog = ConfigDialog(self.initUI)
        dialog.exec_()
            def initUI(self, text = None, keys_per_layer = None):
        self.setWindowTitle("Chinese remainder theorem")
        self.resize(1000, 800)
        ## Create components
        self.keys_per_layer = keys_per_layer
        main_widget = QWidget(self)
        layout = QVBoxLayout(main_widget)
        self.nav_layout = NavigationBar(self)
        self.nav_layout.encypted_message_layout.update_label_bot_text(text,keys_per_layer)
```

```
    tab_widget = CustomTabWidget.CustomTabWidget(self)
    palette = DarkOrangePalette.DarkOrangePalette()
          ## Set relations
           self.setPalette(palette)
    layout.addLayout(self.nav_layout)
           layout.addWidget(tab_widget)
          self.setCentralWidget(main_widget)
    if __name__ == "__main__":
app = QApplication(sys.argv)
# Set the dark-orange style for the application
app.setStyle("Fusion")
palette = DarkOrangePalette.DarkOrangePalette()
    app.setStyleSheet("""
         #ConfigButton {
       background-color: rgb(236,156,76);
       border-top-left-radius: 5px;
       border-bottom-right-radius: 5px;

       width:100%;
       height: 40%
    }
    #ConfigButton:hover {
       background-color: rgb(255,169,82);
    }
    #AddLayerButton {
       background-color: rgb(255,169,82);
       border: 1px solid black;
       border-top-left-radius: 10px;
       border-bottom-right-radius:10px;
       padding:10px;
```

```
    width:120%;
    height: 40%;
}
#AddLayerButton:hover {
    background-color: rgb(236,156,76);
}
#NavBarLabel {
    color:orange;
}
#NavBarText {
    color:rgba(255,255,255,0.8);
    border: 2px solid rgb(236,156,76);
    border-top-style: none;
    border-left-style: none;
    border-right-style: none;
}
#TextAreaLayer {
    color: white;
}
#DecryptButton {
    background-color: rgb(51, 51, 51);
    border : 1px solid white;
    border-top-left-radius: 10px;
    border-bottom-right-radius: 10px;
    color:white;
    width:120%;
    height:50%
}
#DecryptButton:hover{
    background-color: grey;
```

```
    }
    #LayerNavBarNew {
        font-size:18px;
        padding-bottom:10px;
        background-color:rgb(255,169,82);
        border: none;
    }
            """)
app.setPalette(palette)
# Apply the shadow effect to the buttons
window = MainWindow()
window.show()
sys.exit(app.exec())
```

Програмний модуль crt.py

```
import random
from decimal import Decimal
import sympy
import numpy as np
import warnings
warnings.filterwarnings("error")

def generate_keys_for_layers(layer, keys_per_layer)->dict:
    layer = layer[0]/10
    def generate_prime(min_value, max_value):
        # Generate a random number within the specified range
        number = random.randint(min_value, max_value)

        # Find the next prime number greater than or equal to the generated number
        prime = sympy.nextprime(number)

        return prime
    min = int(layer/1000)
    max = int(layer/100)
    print(min)
    print(max)
    keys = [generate_prime(min,max) for i in range(keys_per_layer)]
    # print(keys)


    return keys
```

```
def encrypt_message( message: str) -> list:

    encrypted_numbers = []
    for char in message:
        if char.isalpha():
            char = char.upper()
            number = ord(char) - 64
            encrypted_numbers.append(str(number))
        if char == " ":
            number = 40
            encrypted_numbers.append(str(number))

    return int("".join(encrypted_numbers))

def encrypt_number( number:int, keys: list)->list:

    return  [number % mod for mod in keys]

def chinese_remainder_theorem(modules, remainders):
    # Step 1: Calculate N
    N = 1
    for module in modules:
        N *= module

    result = 0
    # Step 2-5: Calculate x_i and the result
    for module, remainder in zip(modules, remainders):
        n_i = N // module
        x_i = pow(n_i, -1, module)
        try:
```

```
            result += remainder * n_i  * x_i
        except RuntimeWarning:
            result += remainder * Decimal(n_i)  * Decimal(x_i)
    # Step 6: Calculate the final result
    return result % N

def init_1_layer(encrypted_message, keys_per_layer):
    encrypted_message = int(encrypted_message)
    keys_per_layer = int(keys_per_layer)
    keys = generate_keys_for_layers([encrypted_message], keys_per_layer)
    layer = encrypt_number(encrypted_message, keys)
    return layer,keys

def init_layer(layer , keys_per_layer):
    new_layer = []

    keys_per_layer = int(keys_per_layer)
    keys = generate_keys_for_layers(layer , keys_per_layer)
    for i in layer:

        new_value = encrypt_number(int(i), keys)
        new_layer.append(new_value)

    reshaped_list = [element for sublist in new_layer for element in sublist]

    return  reshaped_list, keys
```

Програмний модуль ConfigDialog.py

```
from PyQt5.QtWidgets import QPushButton,QMessageBox, QDialog, QVBoxLayout, QLabel, QLineEdit, QDialogButtonBox
from ErrorHandler import WarningMessage

class ConfigDialog(QDialog):

    def __init__(self, callback_form):
        super().__init__()
        self.setWindowTitle("Configuration")
        self.resize(400,200)
        self.callback = callback_form
        layout = QVBoxLayout()

        label1 = QLabel("Enter your message:")
        label1.setStyleSheet("color: orange;")
        self.input1 = QLineEdit()
        self.input1.setPlaceholderText("Input your text here")
        self.input1.setStyleSheet("color:orange;")
        label2 = QLabel("Keys per Layer:")
        label2.setStyleSheet("color: orange;")
        self.input2 = QLineEdit()
        self.input2.setPlaceholderText("Input len of keys per layer")
        self.input2.setStyleSheet("color:orange;")
        submit_button = QPushButton("Submit")
        submit_button.clicked.connect(self.submit)
        submit_button.setObjectName("SubmitButton")  # Set object name for the CSS styling
```

```
        layout.addWidget(label1)
        layout.addWidget(self.input1)
        layout.addWidget(label2)
        layout.addWidget(self.input2)
        layout.addWidget(submit_button)

        self.setLayout(layout)


    def submit(self):
        input1_value = self.input1.text()
        input2_value = self.input2.text()

        # Perform necessary actions with the input values
        print("Input 1:", input1_value)
        print("Input 2:", input2_value)
        if input1_value == "":
            WarningMessage("Invalid Input", "Message cannot be empty")
            return
        elif input2_value == "":
            WarningMessage("Invalid Input", "Keys per layer cannot be empty")
            return
        elif not input1_value.replace(" ","").isalpha():
            WarningMessage( "Invalid Input", "Enter your message should
contain only letters.")
            return
        elif not input2_value.isdigit():
            WarningMessage("Invalid Input","Keys per Layer should be an
integer.")
```

```
        return

    self.accept(input1_value, input2_value)

def accept(self, input_value, input2_value = None) -> None:
    self.callback(input_value, input2_value)
    return super().accept()
```

Програмний модуль CustomTabWidget.py

```
import typing
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QTabWidget, QTabBar, QPushButton, QLineEdit, QLabel, QHBoxLayout, QStylePainter, QStyleOptionTab, QStyle, QProxyStyle, qApp
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtCore import QRect, QPoint, Qt
from LayerMainLayout import LayerMainLayout
from VerticalTabBar import VerticalTabWidget
from bs4 import BeautifulSoup

import crt

class CustomTabWidget(VerticalTabWidget):

    def __init__(self, parent) -> None:
        super().__init__()
        self.parent = parent
        self.createTabs()
        self.createLayouts()

    def createTabs(self):

        tab1 = QWidget()
        tab2 = QWidget()

        self.addTab(tab1, "Layer 1")
        self.addTab(tab2, "Layer 2")
```

```
def createLayouts(self):

    for i in range(self.count()):
        tab = self.widget(i)
        self.createLayoutForTab(tab,i)

def add_tab(self):
    # Create a new tab and add it to the tab widget
    new_tab = QWidget()
    tab_number = self.count()
    cur_index = self.currentIndex()
    if (cur_index == self.count()-1):

        self.addTab(new_tab, f"Layer {tab_number + 1}")
        self.createLayoutForTab(new_tab,tab_number )

    self.next_tab()

def createLayoutForTab(self,tab, tab_number):
        label = QLabel(f"Layer {tab_number+1}: {tab}")
        label.setAlignment(Qt.AlignCenter)
        prev_tab = self.widget(tab_number-1)
        content = None
        if prev_tab:
            layer = prev_tab.layout().layer
            print(self.parent.keys_per_layer)
            layer, keys = crt.init_layer(layer,self.parent.keys_per_layer )
        else:
```

```
                html_string =
self.parent.nav_layout.encypted_message_layout.label_bot.text()
                soup = BeautifulSoup(html_string, 'html.parser')
                text = soup.get_text()
                integer_value = int(text)

                layer ,keys = crt.init_1_layer(integer_value,
self.parent.keys_per_layer)

            content = [layer, keys]
            tab1_layout = LayerMainLayout(tab, tab_number+1, self, content)

        def next_tab(self):
            cur_index = self.currentIndex()
            if cur_index < len(self)-1:
                self.setCurrentIndex(cur_index+1)
```

Програмний модуль LayerMainLayout.py

```
import sys
from PyQt5.QtWidgets import QApplication, QScrollArea,
QLineEdit,QVBoxLayout, QWidget, QHBoxLayout, QTextEdit, QPushButton,
QLabel
from PyQt5.QtGui import QColor, QPainter, QPen
from PyQt5.QtCore import Qt
import numpy as np
import crt
from ErrorHandler import WarningMessage

class BottomBorderLabel(QLabel):
    def paintEvent(self, event):
        painter = QPainter(self)
        painter.setPen(QPen(QColor("orange"), 2, Qt.SolidLine))
        painter.drawLine(0, self.height() - 2, self.width(), self.height() - 2)
        super().paintEvent(event)


class LayerMainLayout(QVBoxLayout):

    def __init__(self, parent, i, parent_layout, content=None):
        super().__init__(parent)
        # Create the label with the text "Keys for layer 2:"s
        hbox = QHBoxLayout()

        label1 = QLabel(f'<center>Keys for layer {i}:</center>')
        label1.setStyleSheet("font-size:20px;")
        self.addWidget(label1)
        self.keys = None
```

```
self.layer = None
if content:
    self.layer = content[0]
    self.keys = content[1]
# Create the second label with a bottom orange border

label2 = QLineEdit()
label2.setReadOnly(True)
label2.setText(f" {self.keys} ")
label2.setObjectName("LayerNavBarNew")
label2.setAlignment(Qt.AlignCenter)
self.label2 = label2
self.addWidget(self.label2)

vvbox = QVBoxLayout()
# Create the first text area
label_encr = QLabel("Encrypted values:")
label_encr.setObjectName("LayerLabel")
text_area1 = QTextEdit()
text_area1.setReadOnly(True)
text_area1.setText(f"{self.layer}")
text_area1.setObjectName("TextAreaLayer")
self.text_area1 = text_area1
vvbox.addWidget(label_encr)
vvbox.addWidget(self.text_area1)
hbox.addLayout(vvbox)

# Create the button with a right-skewed arrow
buttond = QPushButton('Decrypt message >' )
buttond.setObjectName("DecryptButton")
```

```
    buttond.clicked.connect(self.decrypt_config)
    # button.setFixedSize(30, 30)  # Set a fixed size for the button
    hbox.addWidget(buttond)

    # Create the second text area
    vv2box = QVBoxLayout()
    label_dec = QLabel("Decrypted values:")
    label_dec.setObjectName("LayerLabel")
    text_area2 = QTextEdit( )
    text_area2.setReadOnly(True)
    text_area2.setObjectName("TextAreaLayer")
    self.text_area2 = text_area2
    vv2box.addWidget(label_dec)
    vv2box.addWidget(self.text_area2)
    hbox.addLayout(vv2box)
    hhbox = QHBoxLayout()
    button = QPushButton("Go to the next layer: ")
    button.clicked.connect(parent_layout.add_tab)
    button.setObjectName("AddLayerButton")
    hhbox.addWidget(button)
    hhbox.addStretch(1)
    self.addLayout(hbox)
    self.addLayout(hhbox)

def decrypt_config(self):
    integers = []
    for i in self.layer :
        integers.append(int(i))

    integers = np.array(integers).reshape(-1, len(self.keys))
```

```
            prev_layer = []
            print("Keys", self.keys)
            print(integers)
            try:
                for i in integers:

                    prev_layer_value = crt.chinese_remainder_theorem(self.keys,
i.tolist())
                    prev_layer.append(prev_layer_value)
            except ValueError as e :
                WarningMessage("Error",f"{e}")

            self.text_area2.setText(f"{prev_layer}")

        def setKeys(self, keys):

            self.label2.setText(keys)
```

Програмний модуль Navbar.py

```
from PyQt5.QtWidgets import  QPushButton, QLabel,QVBoxLayout, QHBoxLayout, QGraphicsDropShadowEffect
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtCore import Qt, QObject, QEvent
from ConfigDialog import ConfigDialog
from DarkOrangePalette import DarkOrangePalette
from ShadowEffect import NavBarShadow
from EncryptedMessageView import NavBarView
class NavBarButton(QPushButton):
    def __init__(self, name, callback):
        QPushButton.__init__(self, name)
        self.clicked.connect(callback)
        self.setMouseTracking(True)
        self.setObjectName("ConfigButton")
        self.setAutoFillBackground(True)
        self.setPalette(DarkOrangePalette())
        shadow_effect = NavBarShadow(name)
        self.shadow_effect = shadow_effect
        self.setGraphicsEffect(self.shadow_effect)

    def event(self, event):
        if event.type() == QEvent.HoverEnter:
            self.hoverEnterEvent(event)
        elif event.type() == QEvent.HoverLeave:
            self.hoverLeaveEvent(event)

        return super().event(event)

    def hoverEnterEvent(self, event):
```

```
            self.shadow_effect.setColor(QColor(0,162, 232, 100))  # Adjust the
shadow color for hover effect

        def hoverLeaveEvent(self, event):
            self.shadow_effect.setColor(QColor(0, 0, 0, 100))  # Reset the shadow
color

    class NavigationBar(QHBoxLayout):
        def __init__(self, main_layout):
            super().__init__()

            self.setObjectName("NavBar")
            palette = DarkOrangePalette()
            # Create the "Config" button
            self.parent_l = main_layout
            config_button = NavBarButton("Config",self.open_config_dialog)
            self.config_button = config_button
            self.addWidget(self.config_button)
            self.encypted_message_layout = NavBarView()
            self.addStretch(1)
            self.addLayout(self.encypted_message_layout)
            self.addStretch(1)
            close_button = NavBarButton("Close",main_layout.close)

            self.close_button = close_button
            self.addWidget(self.close_button)
               def open_config_dialog(self):
            dialog = ConfigDialog(self.parent_l.initUI)
            dialog.exec_()
```

Програмний модуль ShadowEffect.py

```
import typing
from PyQt5 import QtCore
from PyQt5.QtWidgets import  QPushButton, QVBoxLayout, QHBoxLayout, QGraphicsDropShadowEffect
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtCore import Qt, QObject, QEvent

class NavBarShadow(QGraphicsDropShadowEffect):

    def __init__(self, name ) -> None:
        super().__init__()

        self.setBlurRadius(10)
        self.setColor(QColor(0, 0, 0, 100))
        if name == "Close":

            self.setOffset(-2, 2)
        else :
            self.setOffset(2, 2)
```

Програмний модуль VerticalTabBar.py

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget,
QVBoxLayout, QTabWidget, QTabBar, QPushButton, QLineEdit, QLabel,
QHBoxLayout, QStylePainter, QStyleOptionTab, QStyle, QProxyStyle, qApp
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtCore import QRect, QPoint, Qt

class TabBar(QTabBar):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    def tabSizeHint(self, index):
        s = QTabBar.tabSizeHint(self, index)
        if s.width() < s.height():
            s.transpose()
        s.scale(s.width() * 2, s.height() * 2, Qt.KeepAspectRatio)
        return s

    # Make text visible adequately
    def paintEvent(self, event):
        painter = QStylePainter(self)
        style_option = QStyleOptionTab()
        for i in range(self.count()):
            self.initStyleOption(style_option, i)
            painter.drawControl(QStyle.CE_TabBarTabShape, style_option)
            painter.save()
            s = style_option.rect.size()
            s.scale(s.width() * 2, s.height() * 2, Qt.KeepAspectRatio)
            rect = QRect(QPoint(), s)
            rect.moveCenter(style_option.rect.center())
```

```
            style_option.rect = rect
            center = self.tabRect(i).center()
            painter.translate(center)
            painter.rotate(90)
            painter.translate(center*-1)
            painter.drawControl(QStyle.CE_TabBarTabLabel, style_option)
            painter.restore()
class VerticalTabWidget(QTabWidget):
    def __init__(self):
        super().__init__()
        self.setTabBar(TabBar(self))
        self.setTabPosition(QTabWidget.West)
```