МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

КАСЯНЧУК МИХАЙЛО МИКОЛАЙОВИЧ

УДК 004.056.53

ДИСЕРТАЦІЯ

МЕТОДИ ОПРАЦЮВАННЯ БАГАТОРОЗРЯДНИХ ЧИСЕЛ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ НА ОСНОВІ МОДУЛЯРНОЇ АРИФМЕТИКИ

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 _____ Касянчук М.М.

Наукові консультанти:

Николайчук Ярослав Миколайович, доктор технічних наук, професор, завідувач кафедри спеціалізованих комп'ютерних систем Тернопільського національного економічного університету, академік Міжнародної академії інформатики

Карпінський Микола Петрович, доктор технічних наук, професор, завідувач кафедри інформатики та автоматики Університету в Бєльсько-Бялій (Польща)

Тернопіль - 2020

АНОТАЦІЯ

Касянчук М.М. Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.21 – «Системи захисту інформації». – Тернопільський національний економічний університет, Тернопіль, Національний авіаційний університет, Київ, 2020.

Основними операціями найбільш поширених асиметричних криптосистем RSA, Рабіна, Ель-Гамаля є модулярне множення, модулярне експоненціювання та пошук мультиплікативного оберненого елемента за модулем. Вони характеризуються значною обчислювальною складністю. Крім того, при розрядностях ключа порядку 1024 біт проявляються недоліки двійкової системи числення (велика розрядність, наявність міжрозрядних переносів, строга послідовність при виконанні обчислень). Це сповільнює виконання арифметичних операцій у позиційних системах числення.

Одним із напрямків підвищення швидкодії виконання операцій модулярного множення та експоненціювання є використання векторно-модульних методів, у яких порівняно невисока часова складність поєднується з простотою програмної та апаратної реалізацій.

Іншим шляхом підвищення швидкодії сучасних обчислювальних систем є розпаралелення процесу обробки інформації. Цією властивістю володіє непозиційна система залишкових класів. Її успішно можна застосовувати для додавання, віднімання, піднесення до степеня, множення цілих багаторозрядних чисел, що є важливо для асиметричної криптографії. Одним з недоліків системи залишкових класів, який сповільнив її розвиток, є складність при переведенні чисел у позиційну систему числення. Спростити цю процедуру дозволяє досконала та модифікована досконала форми системи залишкових класів, що є перспективним, проте найменш розвиненим

підходом при опрацюванні багаторозрядних чисел в асиметричних криптосистемах

Дисертаційна робота присвячена вирішенню актуальної науково-практичної проблеми підвищення ефективності опрацювання багаторозрядних чисел на основі використання векторно-модульних операцій модулярного множення та експоненціювання, досконалої та модифікованої досконалої форм системи залишкових класів для зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах.

Для цього необхідно розв’язати наступні задачі:

– проаналізувати сучасні методи, алгоритми та засоби опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики з метою визначення перспектив для підвищення їх ефективності за допомогою матрично- та векторно-модульних методів та використання різних форм системи залишкових класів;

– удосконалити векторно-модульний метод модулярного множення та експоненціювання;

– розробити алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного алгоритму модулярного множення та модулярного експоненціювання;

– розробити методи пошуку оберненого елемента за модулем та здійснити їх програмну і апаратну реалізацію;

– розробити метод пошуку мультистепеневої функції за модулем;

– розробити метод пошуку модулів системи залишкових класів, які дозволяють аналітично обчислювати коефіцієнти базисних чисел;

– розробити методи побудови системи модулів досконалої та модифікованої досконалої форм системи залишкових класів;

– удосконалити метод Ферма для факторизації багаторозрядних чисел;

– розробити програмну реалізацію операції множення в цілочисельній та модифікованій досконалій формі системи залишкових класів, сумісного виконання алгоритму Евкліда та множення, методів модулярного множення, експоненціювання, методів факторизації багаторозрядних чисел;

– розробити трьохмодульну криптосистему Рабіна та її VHDL-модель;

– розробити методологію опрацювання багаторозрядних чисел в асиметричних криптосистемах.

Аналіз джерел показав, що для найбільш поширених асиметричних криптосистем гостро стоїть питання підвищення швидкодії опрацювання багаторозрядних чисел. В той же час, векторно-модульний метод модулярного множення та експоненціювання, а також використання різних форм системи залишкових класів не набули значного поширення для асиметричних криптосистем.

Відповідно до цього, розроблено методи пошуку оберненого елемента за модулем та виконання китайської теореми про залишки на основі додавання модуля та додавання залишку, які за рахунок використання модулярних операцій додавання модуля або залишку дають можливість розпаралелити процес пошуку оберненого елемента за модулем і, відповідно, зменшити часову складність даної операції при її використанні в асиметричних криптосистемах.

Також розроблений метод пошуку мультистепеневої функції за модулем, який за рахунок двократного використання функції Ейлера, виконання арифметичних дій над операндами, меншими від заданого модуля, та переходу до лінійної конгруенції, дозволяє уникнути виконання операції модулярного експоненціювання багаторозрядних чисел, відповідно зменшуючи часову складність.

Запропоновано метод пошуку набору модулів системи залишкових класів, який за рахунок обчислення коефіцієнтів базисних чисел на основі аналітичних виразів при відновленні десяткового числа із системи залишкових класів забезпечує уникнення громіздкої операції знаходження

мультиплікативного оберненого елемента за модулем, відповідно зменшуючи часову складність та збільшуючи швидкодію обчислювальних систем.

Розроблені методи побудови наборів модулів досконалої форми системи залишкових класів на основі дробових перетворень та факторизації, які за рахунок уникнення виконання операції пошуку мультиплікативного оберненого елемента за модулем та множення на нього дозволяють зменшити часову та апаратну складності при переведенні чисел із системи залишкових класів в десяткову систему числення.

Розроблені методи побудови трьох- та багатомодульної модифікованої досконалої форми системи залишкових класів, які за рахунок використання аналітичних виразів, отриманих на основі дробових перетворень, факторизації, теореми Вієта, розв'язку систем конгруенцій, дозволяють зменшити розрядність операндів під час проміжних обчислень, уникнути виконання операції пошуку оберненого елемента за модулем та множення на нього і, відповідно, зменшити часову складність при відновленні десяткового числа із системи залишкових класів. Обґрунтовано доцільність використання модифікованої досконалої форми системи залишкових класів в асиметричних криптосистемах замість існуючої цілочисельної форми.

Удосконалено метод Ферма для факторизації багаторозрядних чисел, який за рахунок заміни операції добування квадратного кореня та піднесення до квадрату на кожній ітерації на обчислювально простіші операції додавання і віднімання дає можливість зменшити розрядності операндів, спростити процедуру пошуку факторизованих чисел та підвищити швидкодію обчислень для множників різної розрядності.

Розроблено трьохмодульну криптосистему Рабіна, яка за рахунок узагальнення методів побудови модифікованої досконалої форми системи залишкових класів та їх використання дозволила підвищити швидкодію процесів шифрування та розшифровування блоків відкритого тексту в порівнянні із звичайною цілочисельною формою та розширити блок шифрування без зменшення стійкості криптосистеми.

Розроблено методологію опрацювання багаторозрядних чисел, яка за рахунок використання матрично- та векторно-модульних методів пошуку залишку, модулярного множення та експоненціювання, знаходження оберненого елемента на основі додавання модуля, а також використання досконалої та модифікованої досконалої форм системи залишкових класів дозволяє забезпечити зменшення обчислювальної складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення та побудувати єдину стратегію опрацювання багаторозрядних чисел в асиметричних криптосистемах.

Проведені експериментальні дослідження підтвердили достовірність теоретичних положень та практичних розробок дисертаційного дослідження. Результати досліджень впроваджені або плануються до впровадження (підтверджено відповідними актами) в Управлінні Державної служби спеціального зв'язку та захисту інформації України в Тернопільській області (від 8.07.2019 р.), Управлінні Державної служби України з надзвичайних ситуацій в Тернопільській області (від 8.07.2019 р.), ТзОВ НВФ «Інтеграл» (№2507-01 від 25.07.2019 р.), ТзОВ ТКБР «Стріла» (№288 від 19.07.2019 р.), компанії «CONNECT» (ФОП Яконюк Р.А.) (від 10.07.2019 р.), науковому процесі Громадської організації «Міжнародна академія інформації» (від 10.07.2019 р.), використані при виконанні п'яти науково-дослідних робіт у Тернопільському національному економічному університеті (ТНЕУ) (від 11.07.2019 р.), у навчальному та науковому процесах Університету у Бельсько-Бялій (Польща) (від 16.07.2019 р.), факультету комп'ютерних інформаційних технологій ТНЕУ (від 9.07.2019 р.), фізико-математичного факультету Тернопільського національного педагогічного університету ім. В.Гнатюка (№928-33/03 від 10.07.2019 р.), Академії ГУСПОЛ (Чеська Республіка) (від 10.07.2019 р.).

Теоретичні результати роботи відображені в розроблених нових методах, методології та здійснених програмних і програмно-апаратних 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 Тернопільського національного економічного 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), «Світлодіодне підсвічування зовнішньої реклами», «Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж» (Державний реєстраційний номер 0117U000414).

Ключові слова: асиметричні криптосистеми, модулярна арифметика, система залишкових класів, багаторозрядні числа, досконала та модифікована досконала форми, векторно-модульний метод, китайська теорема про залишки, мультиплікативний обернений елемент, факторизація.

ABSTRACT

Kasianchuk M. Methods of processing multi-digit numbers in asymmetric cryptosystems based on modular arithmetic. – Qualifying scientific work as a manuscript.

Thesis for a Doctor of Technical Science degree in specialty 05.13.21 – «Information security systems». – Ternopil National Economic University, Ternopil, National Aviation University, Kyiv, 2020.

The basic operations of the most common asymmetric RSA cryptosystems, Rabin, El-Gamal are modular multiplication, modular exponentiation, and multiplicative inverted element search by module. They are characterized by considerable computational complexity. In addition, when the bit of the order of

1024 bits, there are several disadvantages of the binary number system including high bit, the presence of inter-bit transfers, strict sequence when performing calculations. The abovemention slows down the arithmetic operations in positional numerical systems. One of the ways to increase the speed of performing modular multiplication and exponential operations is to use vector-modular methods, in which the relatively low time complexity is combined with the simplicity of software and hardware implementations.

Another way to improve the performance of modern computing systems is to parallel the process of information processing. This property has a non-positional residual number system. It can be successfully used to add, subtract, elevate, multiply integers, which is important for asymmetric cryptography. One of the disadvantages of the residual number system, which has slowed its development, is the difficulty in translating numbers into a positional numbering system. This procedure allows to simplify and modify the perfect form of the residue number system, which is a promising but least developed approach for processing multi-digit numbers in asymmetric cryptosystems.

The dissertation is devoted to the solution of the actual scientific and practical problem of increasing the efficiency of processing multi-digit numbers based on the use of vector-modular operations of modular multiplication and exponential, perfect and modified perfect forms of a residual number system for reducing the time complexity, increasing the speed in algorithmic cryptosystems.

To reach the aim, the following tasks were setted :

– to analyze modern methods, algorithms and tools of processing multi-digit numbers in asymmetric cryptosystems on the basis of modular arithmetic in order to determine the prospects for increasing their efficiency by means of matrix- and vector-modular methods and use of various forms of the residual number system;

– to improve the vector-modular method of modular multiplication and exponentialization;

– to develop the algorithmic support for implementation of RSA and El-Gamal cryptosystems on the basis of vector-modular algorithm of modular multiplication and modular exponential;

– to develop methods for finding the inverted element by module and to carried out their software and hardware implementation;

– to develop the method of searching for a multistage function by module;

– to develop a method of finding the modules of the residual number system that allow to calculate the coefficients of basis numbers analytically;

– to develop methods for building a system of modules of the perfect and modified perfect forms of the residual number system;

– improve the Fermat method for factorization of multi-digit numbers;

– to develop software implementation of multiplication operation in integer and modified perfect form of residual number system, joint implementation of Euclid and multiplication algorithm, modular multiplication methods, exponentials, multiples factorization methods;

– develop a three-module Rabin cryptosystem and its VHDL model;

– to develop a methodology for processing of multi-digit numbers in asymmetric cryptosystems.

Analysis of the references showed that for the most widespread asymmetric cryptosystems there is an issue of increasing the speed of processing of multi-digit numbers. At the same time, the vector-modular method of modular multiplication and exponentialization, as well as the use of various forms of the residual number system, have not become wide poopularity for asymmetric cryptosystems.

Accordingly to that, methods for finding the inverted element by modulus and the implementation of the Chinese residual theorem based on adding a module and adding a residue have been developed, which, through the use of modular operations of adding a module or a remainder, make it possible to parallelize the

process of finding a rotated element by a module and, accordingly, to reduce the time the complexity of this operation when used in asymmetric cryptosystems.

A method of searching for a multistage function by a module has also been developed, which, by doubling the use of the Euler function, performing arithmetic operations on operands smaller than a given module, and moving to linear congruence, avoids the operation of modular exponentiality of multi-digit numbers, reducing the time complexity accordingly.

A method of searching for a set of modules of a residual number system is proposed, which, by calculating the coefficients of basis numbers on the basis of analytical expressions, when recovering a decimal number from the residual number system, avoids the cumbersome operation of finding a multiplicative inverted element in the module, thus reducing the time complexity.

Methods for building sets of modules of the perfect form of a residual number system based on fractional transformations and factorization are developed.

Methods of construction of three- and multimodular modified perfect form of a residual number system have been developed, which, by using analytical expressions obtained on the basis of fractional transformations, factorization, the theorem of Viet, the solution of congruence systems, allow to reduce the discharge of operands in intermediate calculations finding the inverse of the module and multiplying by it and, accordingly, reduce the time complexity when restoring the decimal number from the residual number system. The expediency of using the modified perfect form of the residual number system in asymmetric cryptosystems instead of the existing integer form is substantiated.

The Fermat method for factorization of multiple digits has been improved, which, by replacing the square root extraction and squaring it at each iteration with computationally simple addition and subtraction operations, makes it possible to reduce the operands of the operands, to simplify the procedure for finding the factorized numbers and the velocity multipliers.

A three-module Rabin cryptosystem has been developed which, by generalizing methods of constructing a modified perfect form of a residual number system and using them, has allowed to increase the speed of encryption and decryption of open-text blocks compared to the usual integer form and to expand the encryption block.

The methodology of processing of multi-digit numbers has been developed, which due to the use of matrix and vector-modular methods of finding the residual, modular multiplication and exponentiality, finding the inverted element on the basis of adding a module, as well as using the perfect and modified perfect forms of the residual number system, allows to reduce increase the speed of algorithms, specialized software and hardware and build a unified strategy for processing multiple row numbers in asymmetric cryptosystems.

The conducted experimental studies confirmed the validity of theoretical provisions and practical developments of the dissertation research.

The research results are implemented or planned to be implemented (confirmed by appropriate acts) in the Office of the State Service for Special Communication and Information Protection of Ukraine in Ternopil region (from 8.07.2019), the Office of the State Emergency Service of Ukraine in Ternopil region (from 8.07), NPF Integral LLC (№2507-01 from 25.07.2019), TKBR LLC Arrow (№288 from 19.07.2019), CONNECT (FAK Yakonyuk RA) (from 10.07.2019), a scientific process of the Public organization «International Academy of Information» (from 10.07.2019), used in the performance of five research works at the Ternopil National Economic University (TNEU) (from 11.07.2019), at instrumental and scientific processes of the University of Bielsko-Biala (Poland) (from 16.07.2019), Faculty of Computer Information Technology TNEU (from 9.07.2019), physics and mathematics faculty of Ternopil V. Hnatiuk National Pedagogical University (№928-33/03 from 10.07.2019), GUSPOL Academy (Czech Republic) (from 10.07.2019).

The theoretical results of the work are reflected in the developed new methods, methodologies and implemented software implementations. The results

of the dissertation work were reflected in the following reports of research of the state budgetary and economic contracts of the departments of computer engineering, specialized computer systems and cybersecurity of Ternopil National Economic University: «Parallel methods and means of realization of algorithms of information protection in computer networks apparatus of elliptic curves» (State registration number 0109U000035), «Processing of multi-digit numbers in the residual number system» (State registration number 0115U001607), «Development of theoretical principles of methods of forming and digital processing of data in distributed specialized computer systems» (State registration number 0112U008458), «Lighting for outdoor», «Theoretical Foundations and Hardware for Improving the Performance of Wireless Sensor Networks» (State Registration Number 0117U000414).

Keywords: asymmetric cryptosystems, modular arithmetic, residue number cystem, multi-digit numbers, perfect and modified perfect forms, vector-modular method, Chinese remainder theorem, multiplicative inverse element, factorization.

Список основних публікацій здобувача

1. Касянчук М. М. Досконала форма системи залишкових класів: методи побудови та застосування (Монографія). Тернопіль: Економічна думка (ТНЕУ), 2019. 224 с.
2. 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.
3. 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.
4. 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.

5. Касянчук М. 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. Р. 11.
6. 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.
7. 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*. 2014. Vol. 50, № 5. P. 649-654 (Scopus).
8. Nykolaychuk Ya.M., Kasianchuk M.M., Yakymenko I.Z. Theoretical Foundations of the Modified Perfect form of Residue Number System. *Cybernetics and Systems Analysis*. 2016. Vol. 52, №2. P. 219-223 (Scopus).
9. 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*. 2016. Vol.48, №8. P.56-63 (Scopus).
10. 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 (Scopus).
11. Касянчук М. Построение модифицированной совершенной формы системы остаточных классов с использованием факторизации.

Радиоэлектроника, информатика, управление. 2017. Vol.42, №3. P.53-59 (Web of Science).

12. Николайчук Я.М., Ивасьев С.В., Якименко И.З., Касянчук М.Н. Метод факторизации многоразрядных чисел на основе свойств квадратичности вычетов в системе остаточных клас сов. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2015. № 5(95). С. 45–45.
13. Николайчук Я.Н., Ивасьев С.В., Якименко И.З., Касянчук М.Н. Метод компактной кодировки простых многоразрядных чисел в двоичной системе исчисления. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2016. №5 (96). С. 21-23.
14. Касянчук М.Н. Экспериментальное исследование программной реализации системы остаточных классов и ее модифицированной совершенной формы. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2017. №5 (97). С. 53-57.
15. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах/ *Захист інформації*. 2019. Т.21, №2. С. 65- 73.
16. 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.
17. Якименко І.З., Тимошенко Л.М., Касянчук М.М. Вибір параметрів еліптичних кривих у задачах шифрування інформаційних потоків. *Сучасна спеціальна техніка*. 2018. № 2. С. 63–71.
18. 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.

19. 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.
20. 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.
21. 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.
22. 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.
23. Касянчук М.М. Побудова модифікованої досконалої форми системи залишкових класів на основі розв'язку систем конгруенцій. *Науковий збірник Національного лісотехнічного університету України*. 2016. Т.26, №7. С. 372-377.
24. Касянчук М.М. Побудова трьохмодульної модифікованої досконалої форми системи залишкових класів на основі розв'язку квадратного рівняння. *Інформатика та математичні методи в моделюванні*. 2016. Т.6, №1. С. 19–25.
25. 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.

26. 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.
27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. Касянчук М.М., Якименко І.З., Волинський О.І., Пітух І.Р. Теорія алгоритмів RSA та Ель–Гамаля в розмежованій системі числення

Радемахера–Крестенсона. *Вісник Хмельницького національного університету. Технічні науки.* 2011. №3. С. 265-273.

33. Касянчук М.М., Якименко І.З., Николайчук Я.М. Теорія алгоритмів пошуку найбільшого спільного дільника у базисі Крестенсона. *Вісник Тернопільського національного технічного університету.* 2011. Т.16, №1. С. 154–161.
34. Касянчук М.М. Концепція теоретичних положень досконалої форми перетворення Крестенсона та його практичне застосування. *Оптико-електронні інформаційно-енергетичні технології*. 2010. №2 (20). С. 43-47.
35. Касянчук М.М., Николайчук Я.М., Якименко І.З. Теорія алгоритмів перетворень китайської теореми про залишки в матрично розмежованому базисі Радемахера–Крестенсона. *Вісник Національного університету «Львівська політехніка» «Комп'ютерні системи та мережі»*. 2010. №688. С. 118–124.
36. Яциковська У.О. Касянчук М.М., Трембач Р.Б. Удосконалена система захисту комп'ютерної мережі на підставі асиметричного шифрування. *Вісник Східноукраїнського національного університету імені Володимира Даля*. 2009. №6(136). Ч.1. С. 57–60.
37. 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. Ceske Budejovice (Czech Republic). 2019. P. 175-178 (Scopus).
38. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. *Advanced Computer Information Technology* (*ACIT–2018*): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235 (Scopus).
39. 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. P.550-554 (Scopus).

40. Rajba T. Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of Time Characteristics of Search Methods of Inverse Element by the Module. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (*IDAACS–2017*): Proceedings of the 2017 IEEE 9th International Conference. Bucharest, Romania. V.1. September, 2017. P.82–85 (Scopus).
41. Kasianchuk M. Yakymenko I., Pazdriy I., Melnyk A., Ivasiev S. 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. P.222-224 (Scopus).
42. 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. P.1484–1486 (Scopus).
43. 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, Telecommunica-tions and Computer Science* (*TCSET–2016*): Proceedings of the XIII–th International Conference. L'viv–Slavske. 2016. P.534-536 (Scopus).
44. 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. P.161–163 (Scopus).

45. 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. P.168-171 (Scopus).
46. 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 (Scopus).
47. Kasjanchuk M., Yakymenko I., Ivasiev S., Nykolaychuk Ya. Fundamental theoretical and algorithmic principles of the applied tasks decision of theory of numbers and construction of the high-performance special processors on their basis. *The Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2011*): Proceedings of the XI International Conference. Polyana-Svalyava. 2011. P.168-169 (Scopus).
48. 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 (Scopus).
49. Grynchyshyn T., Yakymenko I., Nykolaychuk Ya., Kasyanchuk M. The Theoretical Basis of Bisignal Formation of Information Flow in Computer Systems with Open Optical Signals. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2010*): Proceedings of the X–th International Conference. L'viv–Slavske. 2010. P.222 (Scopus).
50. 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. P.403–406 (Scopus).

51. Карпінський М., Кінах Я., Яциковська У., Якименко І., Касянчук М. Удосконалення архітектури комп'ютерної мережі для програмної реалізації криптоаналітичних алгоритмів. *Інформаційні моделі, системи та технології*: Матеріали V науково-технічної конференції. Тернопіль. 2018. С. 93.
52. Николайчук Я.М., Якименко І.З., Івасьєв С.В., Касянчук М.М. Метод збереження простих великорозрядних чисел у базисі Радемахера. *Питання оптимізації обчислень* (*ПОО-XXXVII*): Матеріали Міжнародної молодіжної математичної школи. Київ: Інститут кібернетики імені В.М. Глушкова НАН України. 2015. С. 159-161.
53. Касянчук М.М., Якименко І.З., Николайчук Я.М., Івасьєв С.В. Векторно-модульний метод множення багаторозрядних чисел в базисі Радемахера-Крестенсона. *Захист інформації і безпека інформаційних систем – 2014*: Матеріали Міжнародної конференції. Львів. 2014. С. 53-54.
54. Задірака В.К., Николайчук Я.М., Касянчук М.М. Теоретичні основи та високопродуктивний алгоритм обчислення мультистепеневої функції в базисі Крестенсона. *Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління* (*ПНМК-2010*): Матеріали Міжнародної проблемно-наукової міжгалузевої конференції. 2010. Т.1, В.6. С. 30-32.
55. Kasianchuk M. Conception of theoretical bases of the accomplished form of Krestenson's transformation and its practical application. *Advanced Computer Systems and Network: Design and Application* (*ACSN–2009*): Proceedings of the 4–th International Conference. L'viv. 2009. P.299–301.
56. Касянчук М.М. Теорія та математичні закономірності досконалої форми системи залишкових класів. *Питання оптимізації обчислень* (*ПОО–XXXV*): Праці Міжнародного симпозіуму. Київ–Кацивелі. 2009. Т.1. С. 306–310.

ЗМІСТ

ВСТУП …… 25

Розділ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ОПРАЦЮВАННЯ БАГАТОРОЗРЯДНИХ ЧИСЕЛ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ…….…… 36

1.1 Теоретичні основи алгебри і теорії чисел для застосування в асиметричних криптосистемах …….. 36

1.2 Аналіз найбільш поширених асиметричних криптосистем…….. 44

1.3 Аналіз найбільш поширених методів факторизації...…… 55

1.4 Теоретичні основи СЗК……. 64

1.5 Сучасний стан та напрями підвищення ефективності використання СЗК при опрацюванні багаторозрядних чисел в асиметричних криптосистемах…… 77

Висновки до першого розділу…… 90

Розділ 2. МЕТОДИ ВИКОНАННЯ МОДУЛЯРНИХ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ……..... 92

2.1 Структурна схема організації взаємодії розроблених методів для асиметричних криптосистем…… 92

2.2 Метод пошуку залишку в розмежованій системі числення……... 94

2.3 Методи модулярного множення …… 96

2.4 Методи модулярного експоненціювання……..… 104

2.5 Методи пошуку оберненого елемента за модулем на основі додавання модуля та залишку ……..…… 109

2.6 Алгоритмічне забезпечення криптосистеми RSA на основі векторно-модульного методу модулярного експоненціювання та множення…… 112

2.7 Алгоритмічне забезпечення криптосистеми Ель-Гамаля

на основі векторно-модульного методу модулярного експоненціювання та множення……. 115
2.8 Алгоритм Евкліда у розмежованій системі числення…… 119
2.9 Китайська теорема про залишки на основі додавання добутку модулів……. 121
2.10 Метод обчислення мультистепеневої функції…… 125
2.11 Вдосконалення методу факторизації Ферма…… 127
Висновки до другого розділу…… 130
Розділ 3. ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ…… 132
3.1 Перспективи використання різних форм СЗК в асиметричних криптосистемах…….…… 132
3.2 Теоретичні основи аналітичного пошуку коефіцієнтів базисних чисел СЗК……..……... 136
3.3 Метод побудови ДФ СЗК на основі дробових перетворень…….. 141
3.4 Узагальнення методу дробових перетворень.…… 145
3.5 Побудова ДФ СЗК методом факторизації…… 149
3.5.1 Часткові випадки…….. 151
3.6 Застосування ДФ СЗК у китайській теоремі про залишки 154
Висновки до третього розділу…… 156
Розділ 4. МЕТОДИ ПОБУДОВИ ТРЬОХМОДУЛЬНОЇ МОДИФІКОВАНОЇ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ…….. 158
4.1 Метод перемноження модулів……..…. 158
4.2 Побудова трьохмодульної МДФ СЗК на основі факторизації…….. 160
4.3 Розробка трьохмодульної МДФ СЗК за допомогою теореми Вієта…… 164

4.4 Трьохмодульна МДФ СЗК на основі розв'язку систем конгруенцій……... 169
4.5 Метод побудови системи модулів МДФ СЗК з використанням послідовності Фібоначчі…… 175
4.6 Побудова трьохмодульної МДФ СЗК для багаторозрядних чисел…… 177
4.7 Приклад побудови трьохмодульної МДФ СЗК для багаторозрядних чисел…… 182
Висновки до четвертого розділу……..… 187
Розділ 5. МЕТОДИ ПОБУДОВИ БАГАТОМОДУЛЬНОЇ МОДИФІКОВАНОЇ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ…..…….. 188
5.1 Метод побудови багатомодульної МДФ СЗК на основі факторизації…….. 188
5.2 Приклад побудови чотирьохмодульної МДФ СЗК……. 190
5.3 Побудова та дослідження п'ятимодульної МДФ СЗК…... 196
5.4 Метод розширення набору модулів МДФ СЗК…….. 210
5.5 Застосування МДФ СЗК в технічних системах…….. 215
Висновки до п'ятого розділу…….. 225
Розділ 6. ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПОШУКУ МОДУЛІВ РІЗНИХ ФОРМ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ ТА ЇХ ЗАСТОСУВАННЯ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ……. 227
6.1 Програмна модель побудови ДФ СЗК……. 227
6.2 Програмна реалізація методу побудови МДФ СЗК…….. 232
6.3 Побудова трьохмодульної криптосистеми Рабіна на основі різних форм СЗК…….. 239
6.4 HDL-модель трьохмодульної криптосистеми Рабіна….. 244
Висновки до шостого розділу…… 252

Розділ 7. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ. МЕТОДОЛОГІЯ ОПРАЦЮВАННЯ БАГАТОРОЗРЯДНИХ ЧИСЕЛ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ …… 253

7.1 Програмний застосунок для реалізації методів модулярного експоненціювання……. 253

7.2 Дослідження сумісного виконання алгоритму Евкліда та множення у криптосистемі Рабіна……. 259

7.3 Експериментальне дослідження часових характеристик методів факторизації……..…….. 266

7.4 Часові характеристики програмної реалізації методів пошуку оберненого елемента за модулем…… 271

7.5 Дослідження апаратної реалізації методів пошуку оберненого елемента за модулем……... 275

7.6 Часові характеристики трьохмодульної криптосистеми Рабіна……. 286

7.7 Дослідження програмної реалізації множення у СЗК та МДФ СЗК…….. 288

7.8 Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах……. 295

Висновки до сьомого розділу……. 299

ВИСНОВКИ…… 301

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…….. 305

Додаток А. Документи, що підтверджують впровадження результатів дисертаційної роботи……. 343

Додаток Б. Лістинги програмних кодів…… 356

ВСТУП

Обгрунтування вибору теми дослідження. На даний час інтенсивна комп’ютеризація усіх видів людської діяльності приводить до повсякденного використання телекомунікаційних систем та мереж. Представлена у цифровому вигляді інформація повинна бути надійно захищена від різних видів загроз, таких як несанкціонований доступ, модифікація, руйнування, підробка електронного цифрового підпису тощо. Це досягається засобами криптографічного захисту.

Важливу роль під час захисту інформаційних потоків відіграють асиметричні криптосистеми. Вони усувають основний недолік симетричних криптосистем: необхідність надійного каналу обміну ключем. Значний вклад в розвиток асиметричної криптографії зробили такі зарубіжні та вітчизняні вчені: У. Діффі, М. Хеллманн, Р. Меркле, Р. Райвест, А. Шамір, Л. Адлеман, М. Рабін, Т. Ель-Гамаль, М. Манассі, О. Василенко, Е. Маховенко, І.Горбенко та інші.

Основними операціями в найпоширеніших асиметричних криптосистемах RSA, Рабіна, Ель-Гамаля, є модулярне множення та модулярне експоненціювання. Однак методи, які найчастіше використовуються для їх реалізації (стандартний, Шенхаге-Штрасена, Монтгомері, Карацуби-Офмана), характеризуються значною часовою та/або апаратною складністю. Пришвидшити процес виконання обчислень дозволяють матрично- та векторно-модульні методи, але вони не набули значного поширення для асиметричних криптосистем.

Крім того, в зв’язку із збільшенням довжини ключа все більше стали проявлятися недоліки двійкової системи числення, а саме, її багаторозрядність, строго послідовна структура, наявність міжрозрядних переносів тощо, які в значній мірі сповільнюють виконання арифметичних операцій.

Найперспективнішим шляхом підвищення швидкодії сучасних обчислювальних систем є розпаралелення процесу обробки інформації. Цією властивістю володіє система залишкових класів (СЗК). Вона дозволяє істотно поліпшити параметри обчислювальних систем у порівнянні з пристроями, побудованими на тій же фізико-технологічній базі і які працюють у позиційних системах числення. Значний теоретичний та прикладний внесок у розвиток СЗК та її застосування в обчислювальній техніці зробили такі вчені: І. Акушський, Д. Юдицький, В. Амербаєв, В. Торгашев, В. Краснобаєв, Я. Николайчук, В. Яцків, М. Червяков, О. Фінько, А. Омонді (A. Omondi), Б. Премкумар (B. Premkumar), А. Молахоссеіні (A. Molahosseini), А.Мохан (A.Mohan) та інші.

Хоча СЗК не позбавлена недоліків (труднощі при виконанні операцій ділення, порівняння, визначення переповнення розрядної сітки), однак її успішно можна застосовувати для додавання, віднімання, піднесення до степеня, множення цілих багаторозрядних чисел, що є важливо для асиметричної криптографії. Безсумнівна перевага СЗК - це можливість виконання операцій над числами, які менші за вибрані модулі, розпаралелення процесу обчислень та відсутність міжрозрядних переносів.

Ще одним з недоліків СЗК, який сповільнив її розвиток, є складність при переведенні чисел із СЗК у позиційну систему числення. Відомі підходи 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. Уникнути виконання цієї операції дозволяє досконала (ДФ) та модифікована досконала форми (МДФ) СЗК, що є перспективним, проте найменш розвиненим підходом використання СЗК при опрацюванні багаторозрядних чисел в асиметричних криптосистемах, оскільки на даний час відсутні методи побудови системи модулів, які задовольняють умови ДФ та МДФ СЗК.

Таким чином, на сучасному етапі розвитку науки і техніки в процесі функціонування асиметричних криптосистем існує таке об’єктивне

протиріччя між потребою у збільшенні довжини ключа, який забезпечує високу стійкість до криптоаналізу та конфіденційність даних, з одного боку, та збільшенням часової складності, зменшенням швидкодії відповідного програмного та апаратного забезпечення при реалізації асиметричних криптосистем.

Враховуючи викладене, *актуальною науково-технічною проблемою* є підвищення ефективності опрацювання багаторозрядних чисел на основі використання векторно-модульних методів модулярного множення та експоненціювання, ДФ та МДФ СЗК для зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах.

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 Тернопільського національного економічного 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), «Світлодіодне підсвічування зовнішньої реклами», «Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж» (Державний реєстраційний номер 0117U000414).

Мета і завдання дослідження.

Метою досліджень є розробка методів, засобів та методології опрацювання багаторозрядних чисел в асиметричних криптосистемах для зменшення обчислювальної складності, підвищення швидкодії алгоритмів,

спеціалізованого програмного і апаратного забезпечення. Для досягнення поставленої мети у дисертаційній роботі необхідно розв’язати наступні задачі:

1) проаналізувати сучасні методи, алгоритми та засоби опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики з метою визначення перспектив підвищення їх ефективності на основі матрично- та векторно-модульних методів, використання різних форм СЗК;

2) удосконалити векторно-модульний метод модулярного множення та експоненціювання;

3) розробити алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного алгоритму модулярного множення та модулярного експоненціювання;

4) розробити методи пошуку оберненого елемента за модулем та здійснити їх програмну і апаратну реалізацію;

5) розробити метод пошуку мультистепеневої функції за модулем;

6) розробити метод пошуку модулів СЗК, які дозволяють аналітично обчислювати коефіцієнти базисних чисел;

7) розробити методи побудови системи модулів ДФ та МДФ СЗК;

8) удосконалити метод Ферма для факторизації багаторозрядних чисел;

9) розробити програмну реалізацію операції множення в цілочисельній та МДФ СЗК, сумісного виконання алгоритму Евкліда та множення, методів модулярного множення, експоненціювання, методів факторизації багаторозрядних чисел;

10) розробити трьохмодульну криптосистему Рабіна та її VHDL-модель;

11) розробити методологію опрацювання багаторозрядних чисел в асиметричних криптосистемах.

Об'єкт дослідження – процеси опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики.

Предмет дослідження – методи, алгоритми та засоби для зменшення обчислювальної складності при опрацюванні багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики.

Методи дослідження. Проведені дослідження базуються на використанні математичних основ алгебри і теорії чисел (для розробки методів побудови ДФ та МДФ СЗК, реалізації КТЗ), теорії алгоритмів (для оцінки часової складності відомих та розроблених методів), методів криптографії (для розробки трьохмодульної криптосистеми Рабіна), програмування (для реалізації методів побудови ДФ та МДФ СЗК, дослідження модулярного експоненціювання, факторизації та пошуку оберненого елемента), схемотехнічного проектування (для дослідження методів пошуку оберненого елемента та трьохмодульної криптосистеми Рабіна), теорії множин (для побудови методології опрацювання багаторозрядних чисел в асиметричних криптосистемах), статистики (для обробки експериментальних результатів).

Наукова новизна отриманих результатів полягає в наступному:

1) отримали подальший розвиток методи модулярного множення та експоненціювання в асиметричних криптосистемах, які за рахунок використання матрично- та векторно-модульних методів характеризуються меншою часовою та апаратною складністю в порівнянні з відомими;

2) вперше розроблено методи пошуку оберненого елемента за модулем та виконання китайської теореми про залишки на основі додавання модуля та додавання залишку, які за рахунок використання модулярних операцій додавання модуля або залишку дають можливість розпаралелити процес пошуку оберненого елемента за модулем і, відповідно, зменшити часову складність даної операції при її використанні в асиметричних криптосистемах;

3) вперше розроблено метод пошуку мультистепеневої функції за модулем, який за рахунок двократного використання функції Ейлера, виконання арифметичних дій над операндами, меншими від заданого модуля,

та переходу до лінійної конгруенції, дозволяє уникнути виконання операції модулярного експоненціювання багаторозрядних чисел, відповідно зменшуючи часову складність;

4) вперше розроблено метод пошуку набору модулів системи залишкових класів, який за рахунок обчислення коефіцієнтів базисних чисел на основі аналітичних виразів при відновленні десяткового числа із системи залишкових класів забезпечує уникнення громіздкої операції знаходження мультиплікативного оберненого елемента за модулем, відповідно зменшуючи часову складність та збільшуючи швидкодію обчислювальних систем;

5) вперше розроблено методи побудови наборів модулів досконалої форми системи залишкових класів на основі дробових перетворень та факторизації, які за рахунок уникнення виконання операції пошуку мультиплікативного оберненого елемента за модулем та множення на нього дозволяють зменшити часову та апаратну складності при переведенні чисел із системи залишкових класів в десяткову систему числення;

6) вперше розроблено методи побудови трьох- та багатомодульної модифікованої досконалої форми системи залишкових класів, які за рахунок використання аналітичних виразів, отриманих на основі дробових перетворень, факторизації, теореми Вієта, розв’язку систем конгруенцій, дозволяють зменшити розрядність операндів під час проміжних обчислень, уникнути виконання операції пошуку оберненого елемента за модулем та множення на нього і, відповідно, зменшити часову складність при відновленні десяткового числа із системи залишкових класів. Обґрунтовано доцільність використання модифікованої досконалої форми системи залишкових класів в асиметричних криптосистемах замість існуючої цілочисельної форми;

7) удосконалено метод Ферма для факторизації багаторозрядних чисел, який за рахунок заміни операції добування квадратного кореня та піднесення до квадрату на кожній ітерації на обчислювально простіші операції додавання і віднімання дає можливість зменшити розрядності операндів,

спростити процедуру пошуку факторизованих чисел та підвищити швидкодію обчислень для множників різної розрядності;

8) вперше розроблено трьохмодульну криптосистему Рабіна, яка за рахунок узагальнення методів побудови модифікованої досконалої форми системи залишкових класів та їх використання дозволила підвищити швидкодію процесів шифрування та розшифровування блоків відкритого тексту в порівнянні із звичайною цілочисельною формою та розширити блок шифрування без зменшення стійкості криптосистеми;

9) вперше розроблено методологію опрацювання багаторозрядних чисел, яка за рахунок використання матрично- та векторно-модульних методів пошуку залишку, модулярного множення та експоненціювання, знаходження оберненого елемента на основі додавання модуля, а також використання досконалої та модифікованої досконалої форм системи залишкових класів дозволяє забезпечити зменшення обчислювальної складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення та побудувати єдину стратегію опрацювання багаторозрядних чисел в асиметричних криптосистемах.

Практичне значення одержаних результатів.

1. Розроблено алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу модулярного множення та модулярного експоненціювання, які характеризується меншою обчислювальною складністю шифрування/розшифрування в порівнянні з відомими.

2. Здійснено програмну та апаратну реалізацію методів пошуку оберненого елемента за модулем, що дозволило виявити переваги запропонованих методів над існуючими.

3. Розроблено алгоритмічне забезпечення для пошуку модулів ДФ та МДФ СЗК на основі дробових перетворень, факторизації, теореми Вієта, розв’язку систем конгруенцій, що дозволило уникнути виконання операції оберненого елемента за модулем та множення на нього при переведенні

чисел з СЗК в десяткову систему числення, та обґрунтовано доцільність використання МДФ СЗК в асиметричних криптосистемах.

4. Програмно реалізовано та досліджено виконання множення в цілочисельній та МДФ СЗК, сумісне виконання алгоритму Евкліда та множення у криптосистемі Рабіна, методів модулярного експоненціювання в криптосистемі RSA, методів факторизації, що дозволило показати зменшення обчислювальної складності при використанні запропонованих методів.

5. Побудовано VHDL-модель трьохмодульної криптосистеми Рабіна на основі цілочисельної та МДФ СЗК, отримано відповідні часові характеристики, які показують зменшення часової складності при використанні МДФ СЗК.

Результати досліджень впроваджені або плануються до впровадження (підтверджено відповідними актами) в Управлінні Державної служби спеціального зв'язку та захисту інформації України в Тернопільській області (від 8.07.2019 р.), Управлінні Державної служби України з надзвичайних ситуацій в Тернопільській області (від 8.07.2019 р.), ТзОВ НВФ «Інтеграл» (№2507-01 від 25.07.2019 р.), ТзОВ ТКБР «Стріла» (№288 від 19.07.2019 р.), компанії «CONNECT» (ФОП Яконюк Р.А.) (від 10.07.2019 р.), науковому процесі Громадської організації «Міжнародна академія інформації» (від 10.07.2019 р.), використані при виконанні п'яти науково-дослідних робіт у Тернопільському національному економічному університеті (ТНЕУ) (від 11.07.2019 р.), у навчальному та науковому процесах Університету у Бельсько-Бялій (Польща) (від 16.07.2019 р.), факультету комп'ютерних інформаційних технологій ТНЕУ (від 9.07.2019 р.), фізико-математичного факультету Тернопільського національного педагогічного університету ім. В.Гнатюка (№928-33/03 від 10.07.2019 р.), Академії ГУСПОЛ (Чеська Республіка) (від 10.07.2019 р.).

Особистий внесок. Наукові положення, які містяться в дисертації, отримані здобувачем особисто. У друкованих працях, опублікованих у співавторстві, автору належить: [2, 47] – запропоновано методи опрацювання багаторозрядних чисел в СЗК, [3] – обґрунтовано застосування СЗК в

криптографічних засобах захисту інформації, [4, 37] – запропоновано метод пошуку залишку багаторозрядних чисел за модулем, [6] – запропоновано метод побудови п’ятимодульної МДФ СЗК, [7] – запропоновано метод підбору модулів для аналітичного обчислення коефіцієнтів базисних чисел, [8, 9] – запропоновано методи побудови МДФ СЗК, [10] – проведено чисельні розрахунки для побудови термо- або п’єзоелектричного сенсора на основі СЗК, [12, 13, 38, 42, 52] – проведено аналітичні розрахунки методів факторизації багаторозрядних чисел на основі властивостей квадратичних лишків в СЗК та компактного кодування багаторозрядних двійкових чисел відповідно, [15] – розроблена методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах, [16] – розроблено алгоритмічне забезпечення криптосистеми Рабіна на основі операції додавання, [17, 30] - розроблено математичне забезпечення для вибору параметрів еліптичних кривих при шифруванні, [18, 19, 40] – запропоновано методи та здійснено інтерпретацію експериментальних чисельних результатів пошуку оберненого елемента за модулем, [20] – запропоновано метод розширення набору модулів МДФ СЗК, [21] – запропоновано метод сумісного виконання алгоритму Евкліда та множення, [22, 41] – запропоновано трьохмодульний метод шифрування Рабіна з використанням різних форм СЗК, [25, 28, 31, 44, 53] – запропоновано методи модулярного експоненціювання та модулярного множення, [26] – проведено чисельні дослідження алгоритму пошуку символів Якобі, [27, 34, 45] – запропоновано методи аналітичного пошуку модулів ДФ СЗК, [29, 49] – розроблено алгоритми опрацювання інформаційних потоків у комп’ютерних системах, [32] – запропоновано метод побудови криптоалгоритмів RSA і Ель-Гамаля в розмежованій СЗК, [33, 35] – розроблено алгоритми пошуку найбільшого спільного дільника та перетворень КТЗ на основі СЗК, [36, 51] – запропоновано метод захисту комп’ютерної мережі на основі асиметричного шифрування та апарату еліптичних кривих, [39] – розроблено алгоритм реалізації криптоалгоритму RSA на основі векторно-модульного методу модулярного множення та

експоненціювання, [43] – проведено чисельні розрахунки перевірки багаторозрядних чисел на простоту, [46] – запропоновано метод пошуку дискретного логарифма в СЗК, [48] – розроблено матричні алгоритми опрацювання інформаційних потоків, [50] – проведено огляд сучасних алгоритмів і методів біометричної ідентифікації особи, [54] – розроблено метод обчислення мультистепеневої функції.

Апробація результатів дисертації. Основні результати дисертаційної роботи доповідались і обговорювались на таких міжнародних та вітчизняних конференціях, школах, семінарах, як: проблемно-наукова міжгалузева конференція «Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління», Україна (2008-2012 роки); Міжнародна конференція «Теоретичні та прикладні аспекти побудови програмних систем» (2008), Міжнародний симпозіум «Питання оптимізації обчислень» Інституту кібернетики імені В.М. Глушкова НАН України (2009, 2013, 2015, 2017 роки), Всеукраїнська школа–семінар молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології» (Тернопіль) (2011-2019 роки), Міжнародна науково-практична конференція «Сучасні інформаційні та електронні технології» (Одеса) (2014 рік), Міжнародна конференція «Захист інформації і безпека інформаційних систем» (Львів) (2014 рік), VII Міжнародна школа-семінар «Теорія прийняття рішень» (Ужгород) (2014 рік), проблемно–наукова міжгалузева конференція «Юриспруденція та проблеми інформаційного суспільства» (Тернопіль-Бучач), Міжнародна науково-технічна конференція «Актуальні задачі сучасних технологій» (Тернопіль) (2017 рік), науково-технічна конференція «Інформаційні моделі, системи та технології» (Тернопіль) (2018 рік), Міжнародна науково-технічна конференція «Безпека інформаційних технологій» (Україна-Єгипет) (2019), IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (2005, 2015, 2017, 2019 роки), International Conference «Advanced Computer Systems and Network: Design and Application» (2009 рік),

International Conference «Modern Problems of Radio Engineering, Telecommunications and Computer Science» (L'viv) (2010, 2012, 2014, 2016, 2018 роки), International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (L'viv) (2011, 2015, 2017 роки), International Conference on Control, Automation and Systems (Korea) (2016), Inter University Conference of Students, PhD Students and Young Scientists «Engineer of XXI Century» (Poland) (2016, 2017 роки), International Conference «Advanced Computer Information Technology» (Czech Republic) (2018, 2019).

Публікації. За матеріалами дисертації опубліковано 76 друкованих праць (56 основних з яких наведено в авторефераті), з яких 30 статей у фахових виданнях (5 одноосібних), в тому числі 5 індексовано у наукометричних базах Scopus та Web of Science (1 одноосібна), 3 - у періодичних виданнях іноземних держав (1 одноосібна); 40 - у матеріалах та тезах доповідей конференцій (5 одноосібних), з яких 14 проіндексовані наукометричними базами Scopus та Web of Science; 1 авторська та розділи у 5 колективних монографіях.

Обсяг і структура дисертації. Дисертація складається з анотації, змісту, вступу, семи розділів, загальних висновків, списку використаних джерел, додатків та має 287 сторінок основного тексту, 92 рисунки, 84 таблиці, 38 сторінок додатків. Список літератури загалом містить 346 найменувань і займає 38 сторінок. Загальний обсяг роботи 380 сторінок.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ОПРАЦЮВАННЯ БАГАТОРОЗРЯДНИХ ЧИСЕЛ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ

У першому розділі проаналізовано та досліджено сучасний стан опрацювання багаторозрядних чисел в асиметричних криптосистемах, визначено переваги та недоліки розглянутих методів, а також напрямки підвищення ефективності використання СЗК в асиметричних криптосистемах

1.1 Теоретичні основи алгебри і теорії чисел для застосування в асиметричних криптосистемах

Фундаментальною теоретичною основою сучасних асиметричних криптосистем [1-6] є алгебра і теорія чисел, або, точніше, теорія лишків чи конгруенцій [7-12]. Цілі числа a та b називаються порівнянними (або конгруентними) за модулем z, якщо різниця чисел $a - b$ націло ділиться на z. Співвідношення між a, b і z запишемо у вигляді:

$$a \equiv b(\mathrm{mod} z) \qquad (1.1)$$

Запис mod z буде означати, що $z \geq 1$. Запис (1.1) називається конгруенцією.

Відповідно до визначення, запис $a \equiv 0(\mathrm{mod} z)$ означає, що a націло ділиться на z.

Приклади:

$101 \equiv 17(\mathrm{mod} 21)$, тому що 101 – 17 = 84, а $84 \vdots 21$ або $135 \equiv 11(\mathrm{mod} 4)$, тому що числа 135 та 11 при діленні на 4 дають остачу 3.

Число a конгруентне числу b тоді й тільки тоді, коли a й b мають однакові залишки при діленні на z, тому як визначення конгруенцій можна

взяти наступне: цілі числа a й b називаються конгруентними за модулем z, якщо залишки від ділення цих чисел на z рівні.

Основні властивості конгруенцій:

1) рефлексивність: $a \equiv a(\mathrm{mod} z)$;

2) симетричність: якщо $a \equiv b(\mathrm{mod} z)$, то $b \equiv a(\mathrm{mod} z)$;

3) транзитивність: якщо $a \equiv b(\mathrm{mod} z)$, $b \equiv c(\mathrm{mod} z)$, то $a \equiv c(\mathrm{mod} z)$;

4) якщо $a \equiv b(\mathrm{mod} z)$ і k_0 – довільне ціле число, то $k_0 a \equiv k_0 b(\mathrm{mod} z)$;

5) якщо $k_0 a \equiv k_0 b(\mathrm{mod} z)$ й НСД (k_0, z) = 1, то $a \equiv b(\mathrm{mod} z)$;

6) якщо $a \equiv b(\mathrm{mod} z)$ й k_0 – довільне натуральне число, то $k_0 a \equiv k_0 b(\mathrm{mod} k_0 z)$;

7) якщо $k_0 a \equiv k_0 b(\mathrm{mod} k_0 z)$, де k_0 і z – довільні натуральні числа, то $a \equiv b(\mathrm{mod} z)$;

8) якщо $a \equiv b(\mathrm{mod} z)$, $c \equiv d(\mathrm{mod} z)$, то $a + c \equiv b + d(\mathrm{mod} z)$ й $a - c \equiv b - d(\mathrm{mod} z)$;

9) якщо $a \equiv b(\mathrm{mod} z)$, $c \equiv d(\mathrm{mod} z)$, то $ac \equiv bd(\mathrm{mod} z)$;

10) якщо $a \equiv b(\mathrm{mod} z)$, то при будь-якому цілому $k > 0$, $a^k \equiv b^k(\mathrm{mod} z)$;

11) будь-який доданок лівої або правої частини конгруенції можна перенести із протилежним знаком в іншу частину.

Теорема про ділення з остачею: розділити число $a \in Z$ на число $b \in Z$, $b \neq 0$, з остачею, означає знайти пару цілих чисел q та r, таких, що виконуються наступні умови:

$$a = b\gamma + r, 0 \leq r < |b|. \qquad (1.2)$$

Легко доводиться, що для будь-яких цілих чисел a та $b \neq 0$ ділення з остачею можливо і числа γ і r визначаються однозначно.

Один із методів виконання арифметичних операцій над даними цілими числами грунтується на простих положеннях теорії чисел. Ідея цього методу

полягає в тому, що цілі числа представляються в одній з непозиційних систем – СЗК, у якій замість операцій над цілими числами оперують із залишками від ділення цих чисел на обрані заздалегідь взаємно прості числа – модулі $p_1, p_2,..., p_j$. Найчастіше числа $p_1, p_2,..., p_j$ вибирають із множини простих чисел.

Нехай $A \equiv \alpha_1 (\mathrm{mod}\, p_1)$, $A \equiv \alpha_2 (mod\ p_2)$,…, $A \equiv \alpha_j (\mathrm{mod}\, p_j)$...

Важливо відзначити, що при цьому немає ніякої втрати інформації за умови, що $A < p_1 p_2 ... p_j = P$, тому що завжди, знаючи $(\alpha_1, \alpha_2,..., \alpha_j)$ можна відновити саме число *A.* Тому кортеж $(\alpha_1, \alpha_2,..., \alpha_j)$можна розглядати як один зі способів подання цілого числа *A* в комп'ютері – модулярне подання або подання в СЗК.

Мультиплікативно оберненим елементом до числа *a* у модулярній арифметиці є таке число *b*, що виконується конгруенція [13, 14]:

$$ab \bmod z = 1. \qquad (1.3)$$

Умовою існування мультиплікативно оберненого елемента є рівність 1 найбільшого спільного дільника (НСД) чисел *a* і *b*, тобто числа *a* і *b* повинні бути взаємно прості. Якщо ця умова не виконується, то мультиплікативно обернений елемент до *a* не існує.

Методи пошуку мультиплікативного оберненого елемента можна розділити на дві великі категорії [15-17]: методи, що не ґрунтуються на методах пошуку НСД, і методи, які є похідними від методів пошуку НСД (рис. 1.1).

Найбільш поширеними методами, які відносяться до першої групи, є повний перебір всіх можливих варіантів (або брутальна атака) та метод на основі функції Ейлера.

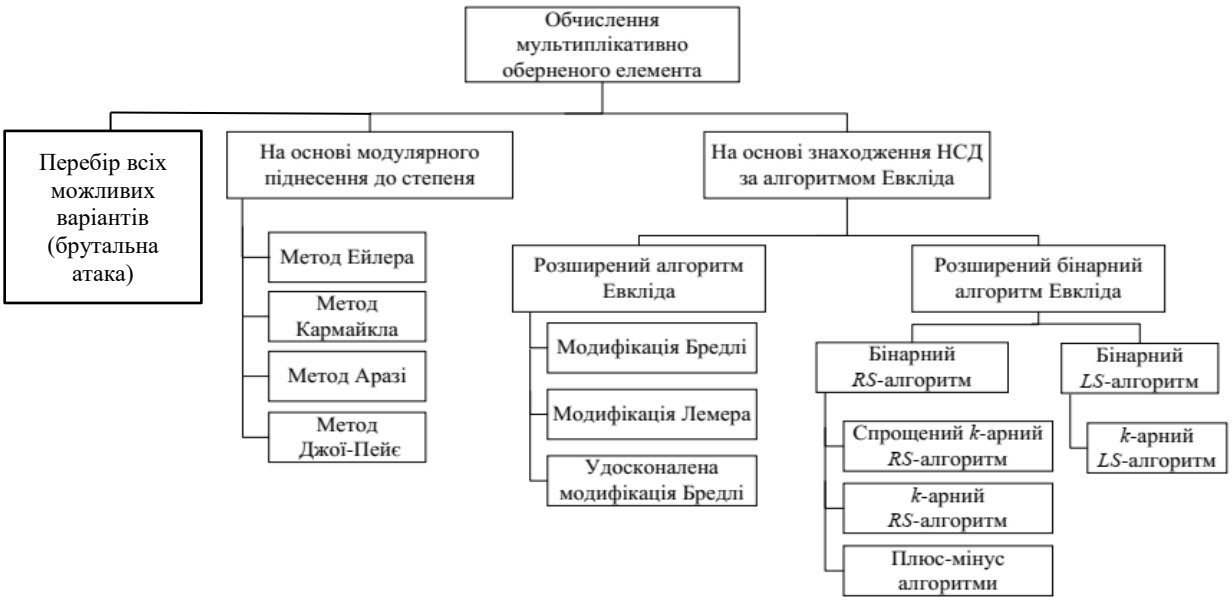


Рис.1.1 - Класифікація методів пошуку оберненого елемента

Під брутальною атакою розуміється метод рішення математичних задач, при якому складність повного перебору (брутальної атаки) залежить від кількості всіх можливих варіантів вирішення задачі. Цей метод відноситься до класу методів пошуку рішення задач із вичерпуванням можливих варіантів розв'язку системи, є найпростішим і водночас найзатратнішим. Він характеризується високою обчислювальною складністю, так як повний перебір вимагає значних часових затрат.

Функція Ейлера φ(*z*), де *z* - натуральне число, це цілочисельна функція, яка рівна кількості натуральних чисел, не більших за *z* і взаємно простих з ним. Функцію Ейлера можна подати у вигляді так званого добутку Ейлера:

$$\varphi(z) = z \prod_{p_0 | z} \left(1 - \frac{1}{p_0}\right), \tag{1.4}$$

де p_0 – просте число.

При використанні теореми Ейлера $a^{\varphi(z)} \bmod z = 1$ отримується: $a^{\varphi(z)-1} \bmod z = a^{-1} \bmod z$. Дана процедура передбачає виконання операції модулярного експоненціювання, що може призвести до переповнення

розрядної сітки процесора та ускладнює пошук оберненого елемента за модулем для багаторозрядних чисел.

Найбільш ефективними та поширеними є методи другої категорії, зокрема пошук мультиплікативного оберненого елемента за модулем за допомогою розширеного алгоритму Евкліда. Для цього спочатку потрібно записати прямий алгоритм Евкліда, згідно якого для будь–якого $z>a=r_0$, де z і a – цілі числа, виконується така система рівнянь:

$$\begin{aligned}
&z = r_0 \cdot q_1 + r_1,\ q_1 = a,\ 0 \le r_1 < r_0;\\
&r_0 = r_1 \cdot q_2 + r_2,\ 0 \le r_2 < r_1;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&r_{j-3} = r_{j-2} \cdot q_{j-1} + r_{j-1},\ 0 \le r_{j-1} < r_{j-2};\\
&r_{j-2} = r_{j-1} \cdot q_j + r_j,\ 0 \le r_j < r_{j-1};\\
&r_{j-1} = r_j \cdot q_{j+1} + 0.
\end{aligned} \tag{1.5}$$

Оскільки a і z є взаємно простими, то r_j=1. Далі для реалізації розширеного алгоритму Евкліда описану процедуру необхідно повторити в зворотньому порядку:

$$\begin{aligned}
1 = r_j &= r_{j-2} - q_j r_{j-1} = r_{j-2} - q_j\left(r_{j-3} - q_{j-1} r_{j-2}\right) = r_{j-2} - q_j r_{j-3} + q_j q_{j-1} r_{j-2} =\\
&= -q_j r_{j-3} + \left(1 + q_j q_{j-1}\right) r_{j-2} = -q_j r_{j-3} + \left(1 + q_j q_{j-1}\right)\left(r_{j-4} - q_{j-2} r_{j-3}\right) = \ldots
\end{aligned} \tag{1.6}$$

Даний процес продовжується до тих пір, поки не отримається вираз $v \cdot z + t \cdot r_0 = 1$, де величина $b = t \bmod z = a^{-1} \bmod z$ і буде шуканим оберненим елементом. В табл. 1.1 наведено приклад використання розширеного алгоритму Евкліда, а на рис. 1.2 - його блок-схема.

Даний метод характеризується великою кількістю ділень з остачею, перемножень та підстановок, хоч він і володіє найменшою часовою складністю у порівнянні з двома іншими.

Таблиця 1.1 - Пошук оберненого елемента $41^{-1} \bmod 157$ за допомогою розширеного алгоритму Евкліда

Алгоритм Евкліда	Розширений алгоритм Евкліда
157=41·3+34 41=34·1+7 34=7·4+6 7=6·1+1	1=7-1·6=7-1·(34-4·7)=-1·34+5·7=-1·34+5·(41-1·34)=5·41-6·34= =5·41-6·(157-3·41)=-6·157+23·41; $41^{-1} \bmod 157=23$

Методи пошуку оберненого елемента на основі алгоритму Евкліда можна розділити на два класи [13, 18] (рис. 1.1):

– методи, що ґрунтуються на класичному алгоритмі Евкліда [19];

– методи, що засновані на бінарному алгоритмі Евкліда [20].

Методи другого класу є більш ефективними, так як не використовують обчислювально-витратних операцій ділення на довільне число. Ці методи використовують лише елементарні операції, такі як додавання, віднімання та ділення на 2, що є еквівалентним зсуву на один двійковий розряд праворуч.

Поширеним застосуванням розширеного алгоритму Евкліда є КТЗ - один з найдавніших, але надзвичайно важливий на даний час обчислювальний алгоритм.

Ще у першому столітті нашої ери китайський математик Сунь–Цзи запропонував цікаву загадку, якою було покладено початок модулярній арифметиці: потрібно було знайти число, що при діленні на 3 дасть в остачі 2, на 5 – 3, на 7 – 2. Крім того, він в частковому випадку показав еквівалентність розв’язку системи модулярних рівнянь та розв’язку одного модулярного рівняння.

Протягом майже двох тисяч років КТЗ постійно вдосконалювалася і розвивалася. Зокрема, у XIII столітті інший китайський математик Цань Цзю–шао розв’язав наведену вище задачу.

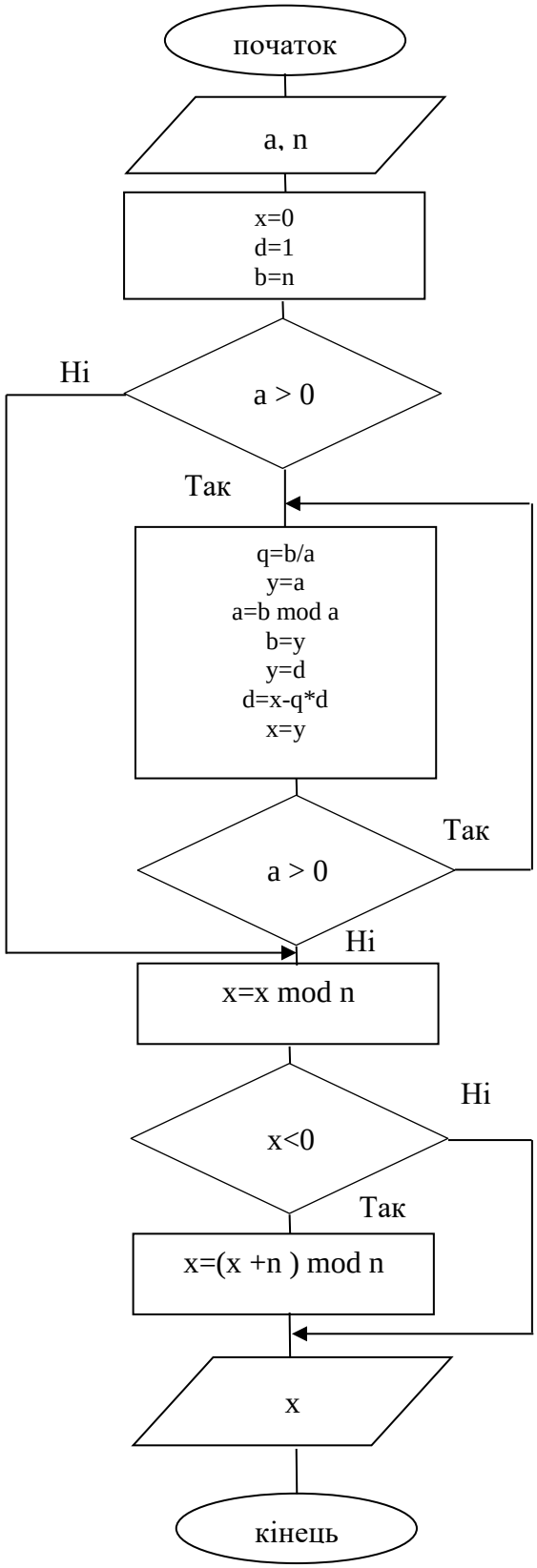


Рис. 1.2 - Блок-схема розширеного алгоритму Евкліда

У XVIII столітті німецький математик Л.Ейлер навів загальне формулювання і доведення КТЗ, а К.–Ф. Гаус істотно розвинув його в своїх знаменитих „Арифметичних дослідженнях” [21].

І, нарешті, у середині XX століття чеські вчені М.Валах і А.Свобода запропонували використати древню китайську ідею на новому технічному рівні, створивши перші модулярні електронно–обчислювальні машини „Епос” і „Епос–2” [22].

Слід зазначити, що на даний час існує декілька еквівалентних формулювань КТЗ. Найбільш поширене з них таке [9]: якщо натуральні числа $p_1, p_2, \ldots, p_j$ попарно взаємно прості, то для будь–яких цілих $r_1, r_2, \ldots, r_j$, таких що $0 \le r_i < p_i$ існує число A, яке при діленні на p_i дає залишок r_i при всіх $i=1, 2, ..., j$; більше того, якщо існує два таких числа A_1 та A_2, то $A_1 \bmod P = A_2 \bmod P$, де $P = \prod_{i=1}^{j} p_i$.

Дана теорема представляється переважно у вигляді такої системи порівнянь:

$$\begin{cases} A \bmod p_1 = r_1 \\ A \bmod p_2 = r_2 \\ \ldots\ldots\ldots\ldots \\ A \bmod p_i = r_i \\ \ldots\ldots\ldots\ldots \\ A \bmod p_j = r_j. \end{cases} \qquad (1.7)$$

Шукане число обчислюється за формулою:

$$A = \left(\sum_{i=1}^{j} P_i m_i r_i \right) \bmod P, \qquad (1.8)$$

де $P_i = \frac{P}{p_i} = p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_j$, $m_i = P_i^{-1} \bmod p_i$.

Як було сказано вище, пошук мультиплікативного оберненого елемента, необхідний для реалізації КТЗ, характеризується значною обчислювальною складністю в зв’язку з неможливістю розпаралелення процесу обчислень. Причому всі операції повинні виконуватися над дуже великими числами, що може призвести до переповнення розрядної сітки сучасних потужних обчислювальних засобів.

1.2 Аналіз найбільш поширених асиметричних криптосистем

Відомо, що усім симетричним криптосистемам, у яких шифрування та розшифровування відбувається за допомогою одного і того самого ключа, притаманні такі основні недоліки [23, 24]:

- принциповою є надійність каналу передачі ключа другому учаснику секретних переговорів. Інакше кажучи, ключ повинен передаватися по секретному каналу;

- до служби генерації ключів пред'являються підвищені вимоги, обумовлені тим, що для j абонентів при схемі взаємодії "кожен з кожним" потрібно $j(j\text{-}1)/2$ ключів, тобто залежність кількості ключів від кількості абонентів є квадратичною.

Для вирішення перерахованих вище проблем симетричного шифрування призначені системи з асиметричним шифруванням або шифруванням з відкритим ключем (рис. 1.3), що використовують властивості функцій з секретом, розроблених Діффі і Хеллманом [25].

Ці системи характеризуються наявністю у кожного абонента двох ключів: відкритого і закритого (секретного). При цьому відкритий ключ відкрито передається всім учасникам секретних переговорів. Таким чином, вирішуються дві проблеми:

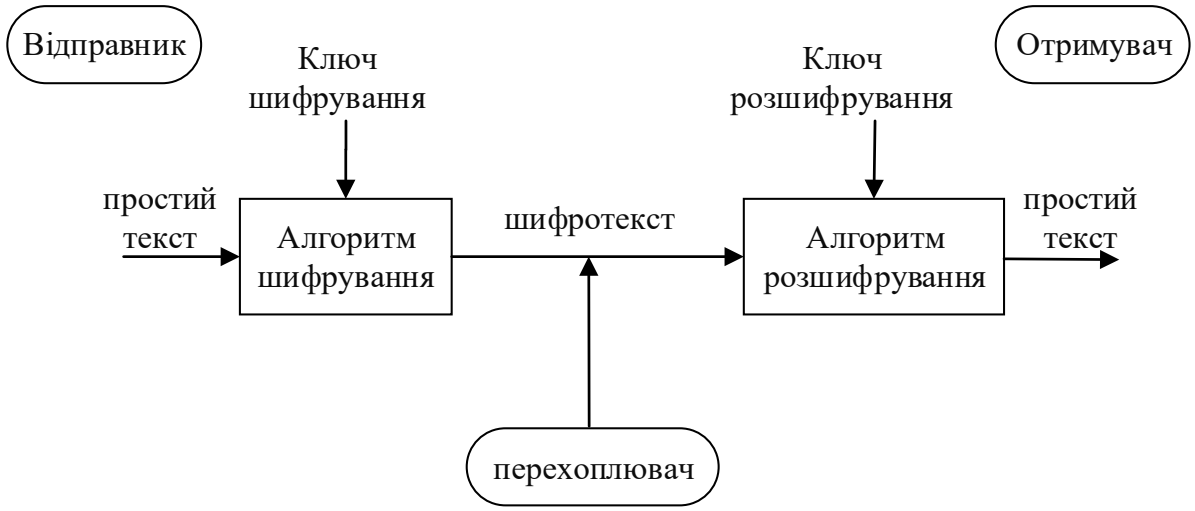


Рис. 1.3 – Схема асиметричних криптосистем

- немає потреби в секретній доставці;

- відсутня квадратична залежність кількості ключів від кількості користувачів - для *j* користувачів потрібно 2*j* ключів.

Першим шифром, розробленим на принципах асиметричного шифрування, є шифр RSA. Він названий так за першими літерами прізвищ його винахідників: Рона Райвеста, Аді Шаміра і Леонарда Адлемана - засновників компанії RSA Data Secutity [26]. RSA - не тільки найпопулярніший з асиметричних шифрів, але, мабуть, взагалі найвідоміший.

Математичне обгрунтування RSA таке: пошук дільників дуже великого натурального числа, яке є добутком двох простих чисел – дуже трудомістка процедура. Відповідно, за допомогою відкритого ключа дуже складно обчислити відповідний йому таємний ключ. Схема криптоалгоритму RSA представлена на рис. 1.4.

Шифр RSA всебічно вивчений і визнаний стійким при достатній довжині ключів. Наприклад, 512 біт для забезпечення стійкості не вистачає, а 1024 біти вважається прийнятним варіантом. З ростом потужності процесорів при даній довжині ключа RSA втрачає стійкість до атаки повним перебором,

однак це дозволяє застосувати більш довгі ключі, що в свою чергу підвищить стійкість шифру.

Шифр працює за алгоритмом, який включає в себе генерацію ключів, шифрування та розшифрування.

Для того, щоби згенерувати пари ключів, виконуються такі дії:

- вибирається два великих простих числа p та q;
- шукається їх добуток $n=p \cdot q$;
- знаходиться функція Ейлера $\varphi(n)=(p-1)(q-1)$;
- вибирається ціле число e таке, що $1<e<\varphi(n)$ та e - взаємно просте з $\varphi(n)$, тобто НСД$(e, \varphi(n))=1$;
- на основі розширеного алгоритму Евкліда шукається число d таке, що $ed \pmod{\varphi(n)}=1$ або $d=e^{-1} \pmod{\varphi(n)}$.

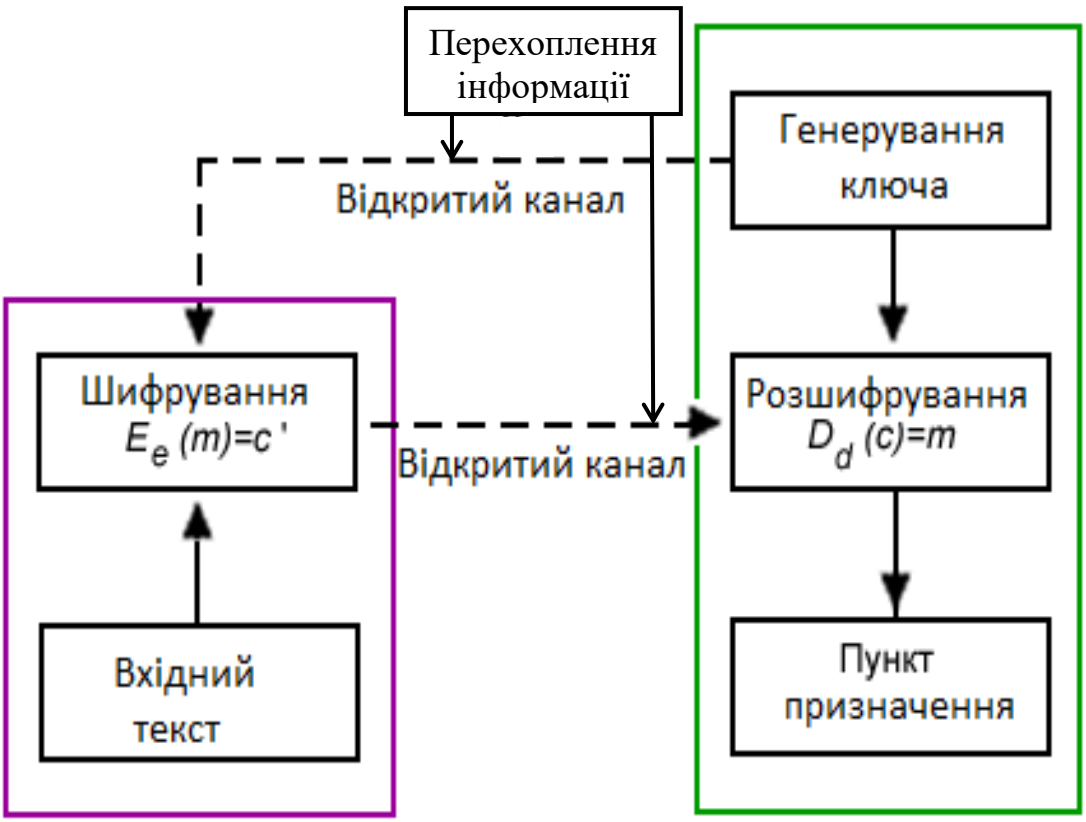


Рис. 1.4 – Криптоалгоритм RSA

Число n називається модулем, числа e та d - відповідно відкритою та секретною експонентами. Пара чисел (n,e) є відкритою частиною ключа, а пара (n, d) - секретною. Числа p і q після генерації ключів можна знищити, однак у жодному разі не можна розкрити.

Для шифрування повідомлення припустимо, що перший абонент хоче відправити другому повідомлення A. Для початку він за допомогою узгодженого протоколу перетворення, відомого як доповняльні схеми або таблиці перетворення, перетворює A в ціле число a так, щоб $0 \le a \le n$. Опісля він обчислює зашифрований тест A', використовуючи відкритий ключ другого абонента e, за допомогою рівняння:

$$A' = a^e \bmod n. \tag{1.9}$$

Це може бути зроблено досить швидко, навіть у випадку, коли текст перевищує 500-бітну розрядність.

Розшифрування відбувається наступним чином:

$$a = (A')^d \bmod n. \tag{1.10}$$

Відповідно при виконанні даної операції відновлюєтсья вихідне повідомлення:

$$(A')^d \equiv (a^e)^d \equiv a^{ed} \pmod{n}, \tag{1.11}$$

З умови

$$ed \equiv 1 \pmod{\varphi(n)}, \tag{1.12}$$

випливає твердження, що $ed \equiv k_0\varphi(n) + 1$ для деякого цілого k_0, отже:

$$a^{ed} \equiv a^{k_0\varphi(n)+1} \pmod{n}. \quad (1.13)$$

Згідно з теоремою Ейлера:

$$a^{\varphi(n)} \equiv 1 \pmod{n}, \quad (1.14)$$

тому

$$a^{k_0\varphi(n)+1} \equiv a \pmod{n}, (A')^d \equiv a \pmod{n}. \quad (1.15)$$

Для шифрування необхідно знати пару чисел *e, n*, для дешифрування - *d, n*. Перша пара - відкритий ключ, друга - закритий. Знаючи відкритий ключ, можна обчислити значення закритого ключа. Необхідною проміжною дією цього перетворення є знаходження множників *p* і *q*, для чого потрібно розкласти *n* на множники - ця процедура займає дуже багато часу. Саме з величезною обчислювальною складністю пов'язана криптостійкість RSA [27].

Криптосистема Рабіна [28, 29] стала першою асиметричною криптосистемою, яка грунтується на складності знаходження квадратичного лишку. Вона, як і будь-яка асиметрична криптосистема, використовує відкритий та закритий ключі. Відкритий ключ використовується для шифрування повідомлень, він може бути відкрито опублікований. Закритий ключ потрібний для розшифрування і тільки має бути відомий одержувачам зашифрованих повідомлень. Схема шифрування згідно криптосистеми Рабіна наведена на рис. 1.5.

Для генерації ключів вибираються два випадкових числа *p* та *q* із урахуванням таких вимог:

- числа мають бути великими;
- числа мають бути простими;
- має виконуватися умова: $p \equiv q \equiv 3 \bmod 4$.

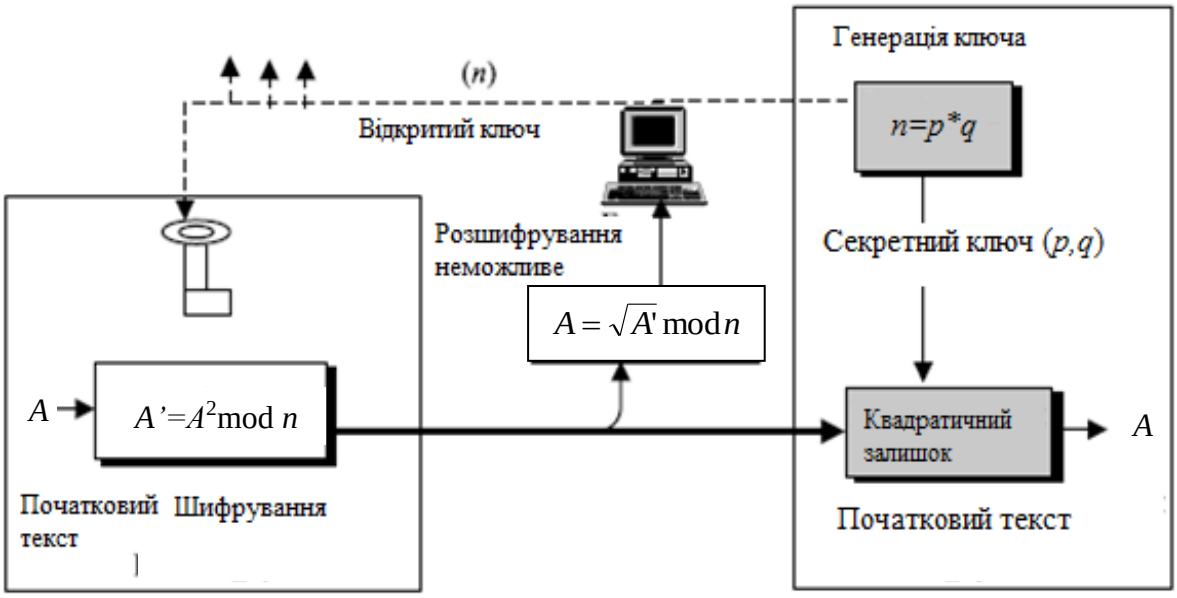


Рис. 1.5 - Схема шифрування згідно криптосистеми Рабіна

Виконання таких вимог істотно прискорює процедуру знаходження коренів за модулем p і q. Далі обчислюється число $n=p \cdot q$, тоді число n - відкритий ключ; числа p і q - закритий.

Початкове повідомлення A (текст) шифрується з допомогою відкритого ключа - числа n - за такою формулою:

$$A' = A^2 \bmod n. \qquad (1.16)$$

Завдяки використанню операції піднесення до квадрату за модулем швидкість шифрування системи Рабіна більша, як швидкість шифрування криптосистемою RSA, навіть коли в останньому випадку значення експоненти вибрати невелике.

При розшифруванні криптограми A' для зручності вводяться додаткові допоміжні величини c_1 і c_2:

$$c_1 = A' \bmod p; \quad c_2 = A' \bmod q. \qquad (1.17)$$

Для знаходження A необхідно знайти квадратичні лишки c_1, c_2 відповідно за модулями p і q:

$$x^2 \equiv c_1 (\mathrm{mod}\, p),\ y^2 \equiv c_2 (\mathrm{mod} q). \qquad (1.18)$$

В результаті можна записати чотири системи порівнянь:

$$\begin{cases} A_1 \equiv x (\mathrm{mod}\, p); \\ A_1 \equiv y (\mathrm{mod} q); \end{cases} \begin{cases} A_2 \equiv x (\mathrm{mod}\, p); \\ A_2 \equiv -y (\mathrm{mod} q); \end{cases} \begin{cases} A_3 \equiv -x (\mathrm{mod}\, p); \\ A_3 \equiv y (\mathrm{mod} q); \end{cases} \begin{cases} A_4 \equiv -x (\mathrm{mod}\, p); \\ A_4 \equiv -y (\mathrm{mod} q). \end{cases} \qquad (1.19)$$

Одне з рішень (1.19), яке грунтується на використанні КТЗ, буде шуканим повідомленням A.

Розшифрування тексту, крім правильного, приводить ще до трьох хибних результатів. Це і є основною незручністю криптосистеми Рабіна, а також одним із факторів, що перешкоджали тому, щоби вона знайшла широке практичне використання.

Якщо вихідний блок являє текстове повідомлення, то визначення правильного варіанту не є важким. Однак коли повідомлення є потоком випадкових бітів (зокрема, для генерації ключів чи цифрового підпису), то тоді визначення потрібного тексту стає проблемою. Одним із способів усунути цей недолік є додавання перед шифруванням до повідомлення відомого заголовка або якоїсь мітки [30].

Однак у класичній криптосистемі Рабіна блок відкритого тексту обмежується величиною відкритого ключа ($A<n$). Тому для досить довгих повідомлень потрібно кожен блок шифрувати окремо.

Приблизно такою ж обчислювальною складністю, як і факторизація, володіє операція дискретного логарифмування у скінченному полі, на якій ґрунтується асиметрична криптосистема Ель-Гамаля (Elgamal). Вона включає в себе алгоритм шифрування та алгоритм цифрового підпису.

Схема була запропонована Тахером Ель-Гамалем у 1985 році [31] і являється одним із удосконалених варіантів алгоритму Діффі-Хеллмана, які використовуються для шифрування та забезпечення аутентифікації.

Алгоритм виконується в такій послідовності:

1) генерується випадкове просте число p;

2) вибирається ціле число g - первісний корінь p;

3) вибирається випадкове ціле число a таке, що $1<a<p-1$;

4) обчислюється $h=g^a \bmod p$;

5) відкритий ключ становлять три числа (p, g, h), таємний ключ - число a.

Повідомлення A, яке повинно бути менше від числа p, шифрується таким чином:

1) вибирається сесійний ключ – випадкове ціле число r таке, що $1<r<p-1$;

2) вираховуються числа $c_1=g^r \bmod p$ і $c_2=h^rA \bmod p$.

Пара чисел (c_1, c_2) є шифротекстом.

Одним з недоліків криптосистеми Ель-Гамаля є те, що довжина шифротексту вдвічі більша від вихідного повідомлення A.

Знаючи закритий ключ a, із шифртексту (c_1, c_2) можна обчислити вихідне повідомлення за такою формулою:

$$A=c_2(c_1^{-1})^a \bmod p. \qquad (1.20)$$

При цьому дуже легко перевірити, що $(c_1^{-1})^a \bmod p=g^{-ra} \bmod p$ і тому $c_2(c_1^{-1})^a \bmod p=(h^rA)g^{-ra} \bmod p=(g^{ra}A)g^{-ra} \bmod p=A \bmod p$.

Для практичних обчислень більше підходить наступна формула:

$$A= c_2(c_1^{-1})^a \bmod p= c_2(c_1)^{p-1-a} \bmod p. \qquad (1.21)$$

Схема шифрування на основі криптоалгоритму Ель-Гамаля представлена на рис. 1.6.

Оскільки у схему Ель-Гамаля вводиться випадкова величина *r*, то цей шифр можна назвати шифром багатозначної заміни. У зв'язку з випадковістю вибору числа *r* таку схему називають ще схемою ймовірнісного шифрування. Ймовірнісний характер шифрування є перевагою для криптосистеми Ель-Гамаля, тому що у таких схемах спостерігається більша стійкість у порівнянні зі схемами із певним процесом шифрування.

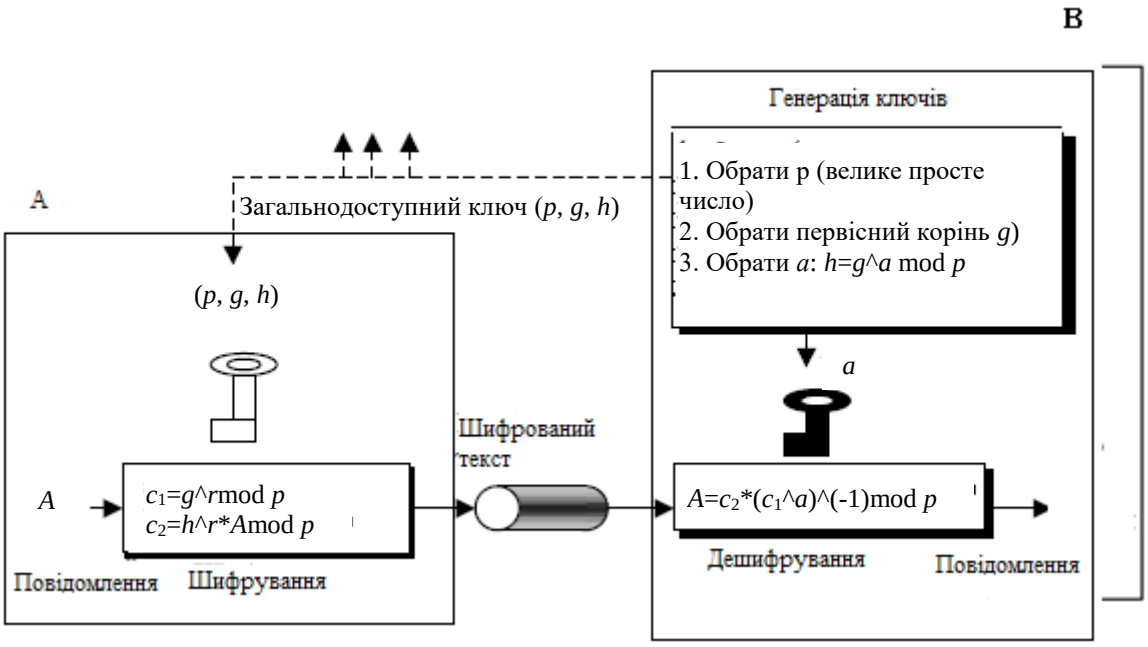


Рис. 1.6 – Схема шифрування на основі криптоалгоритму Ель-Гамаля

Недоліком криптосистеми Ель-Гамаля є подвоєння довжини зашифрованого тексту у порівнянні із початковим текстом. Для схеми ймовірнісного шифрування саме повідомлення *A* і ключ однозначно не визначають шифротекст. В схемі Ель-Гамаля необхідно використовувати різні значення випадкової величини *r* для шифрування різних повідомлень *A* та *A*'. Якщо використовувати однакові *r*, то тоді для відповідних шифротекстів (c_1, c_2) і (c_1', c_2') виконується співвідношення $c_2(c_2')^{-1} = A(A')^{-1}$. З цього виразу можна легко обчислити *A*', якщо відомо *A*.

На даний час криптосистеми із відкритим ключем вважаються найбільш перспективними. До них належить також схема Ель-Гамаля, криптостійкість якої ґрунтується на обчислювальній складності проблеми дискретного логарифмування.

Крім того, існує велика кількість інших алгоритмів, які базуються на схемі Ель-Гамаля, зокрема: DSA, ECDSA, KCDSA, схема Шнорра тощо. В табл. 1.2 представлено порівняльні характеристики деяких асиметричних криптоалгоритмів [32]. Також заслуговують на увагу криптоалгоритми, які використовують арифметику еліптичних кривих, визначених над простими полями Галуа [33, 34].

Однак усі операції у проаналізованих криптоалгоритмах відбуваються в ПСЧ (зокрема, множення у двійковій або десятковій системах числення характеризуються обчислювальною складністю $O(n_0^2)$, де n_0 – розрядність операндів), у яких відсутня можливість розпаралелення процесу обчислень, що значно збільшує час їх виконання.

Одним з підходів для підвищення швидкодії пошуку результату модулярного множення є метод Карабуци-Оффмана, однак практично він майже не використовується через складність його реалізації. В [35] описано алгоритм Шенхаге – Штрассена з використанням швидкого перетворення Фур'є, який потребує $O(n_0 \log(n_0) \log(\log(n_0)))$ бітових операцій. В [36] запропоновано алгоритм модулярного множення з обчислювальною складністю $n_0 \cdot \log_2 n_0$.

В циклі робіт [37-43] запропоновано матрично- та векторно-модульні методи виконання основних арифметичних операцій для знаходження залишку, модулярного множення та модулярного піднесення до степеня на основі двійкової системи числення з використанням розмежованої форми СЗК, які характеризуються порівняно невисокою часовою складністю (табл. 1.3) [44], що поєднується з простотою їх програмної та апаратної реалізацій в порівнянні з іншими.

Таблиця 1.2 - Порівняльні характеристики деяких асиметричних криптоалгоритмів

Алго-ритм	Ключ	Призна-чення	Крипостій-кість, MIPS	Примітки
RSA	До 4096 біт	Шифру-вання і підпис	$2{,}7 \cdot 10^{28}$ для ключа 1300 біт	Грунтується на труднощах факторизації великих чисел; один із перших асиметричних алгоритмів. Міститься в багатьох стандартах
ElGa-mal	До 4096 біт	Шифрування і підпис	При однаковій довжині ключа криптостійкості рівна RSA, тобто $2{,}7 \cdot 10^{28}$для ключа 1300 біт	Грунтується на важкій задачі обрахунку дискретних логарифмів у скінченому полі; дозволяє без зниження стійкості швидко генерувати ключі. Використовується у алгоритмі цифрового підпису DSA-стандарту DSS
DSA	До 1024 бит	Тільки підпис		Грунтується на труднощах дискретного логарифмування у скінченному полі; являється держ. стандартом США; застосовується для секретних та несекретних комунікацій; розробником є АНБ.
ECDS A	До 4096 бит	Шифру-вання і підпис	Крипостійкість та швидкість роботи кращі, ніж у RSA	Сучасний напрямок. Розробляється багатьма провідними математиками

Таблиця 1.3 – Часова складність алгоритмів модулярного множення

Назва алгоритму	Часова складність
Стандартний	n_0^2
Шенхаге-Штрасена	$n_0 \cdot \log_2 n_0 \log_2(\log_2 n_0)$
Карацуби	$n_0^{\log_2 3} = n_0^{1,585}$
Монтгомері	$n_0 \cdot \log_2 n_0$
Матрично-модульний	$n_0 \cdot \log_2 n_0$
Векторно-модульний	$(n_0 \cdot \log_2 n_0)/2$

1.3 Аналіз найбільш поширених методів факторизації

Як було показано вище, на складності обчислювальної задачі факторизації великих чисел ґрунтується, зокрема, стійкість асиметричних криптосистем Рабіна та RSA [45-49]. Факторизацією натурального числа називається його розклад у добуток простих множників. Існування та єдиність (із точністю до порядку проходження множників) такого розкладу випливає із основної теореми арифметики [50].

Алгоритми факторизації, в залежності від складності, можна розбити на дві групи (рис. 1.7) [51, 52]. Перша група – це є експоненціальні алгоритми, складність яких залежить експоненціально від довжини вхідних параметрів (тобто від довжини самого числа у бінарному представленні). Для позначення їх складності прийнята *O*-нотація, яка дозволяє враховувати у функції лише найбільш значущі елементи, відкидаючи другорядні.

Друга група - субекспоненціальні алгоритми, які мають більшу за поліноміальну складність, але меншу, ніж експоненціальна. Для позначення їх складності прийнята *L*-нотація.

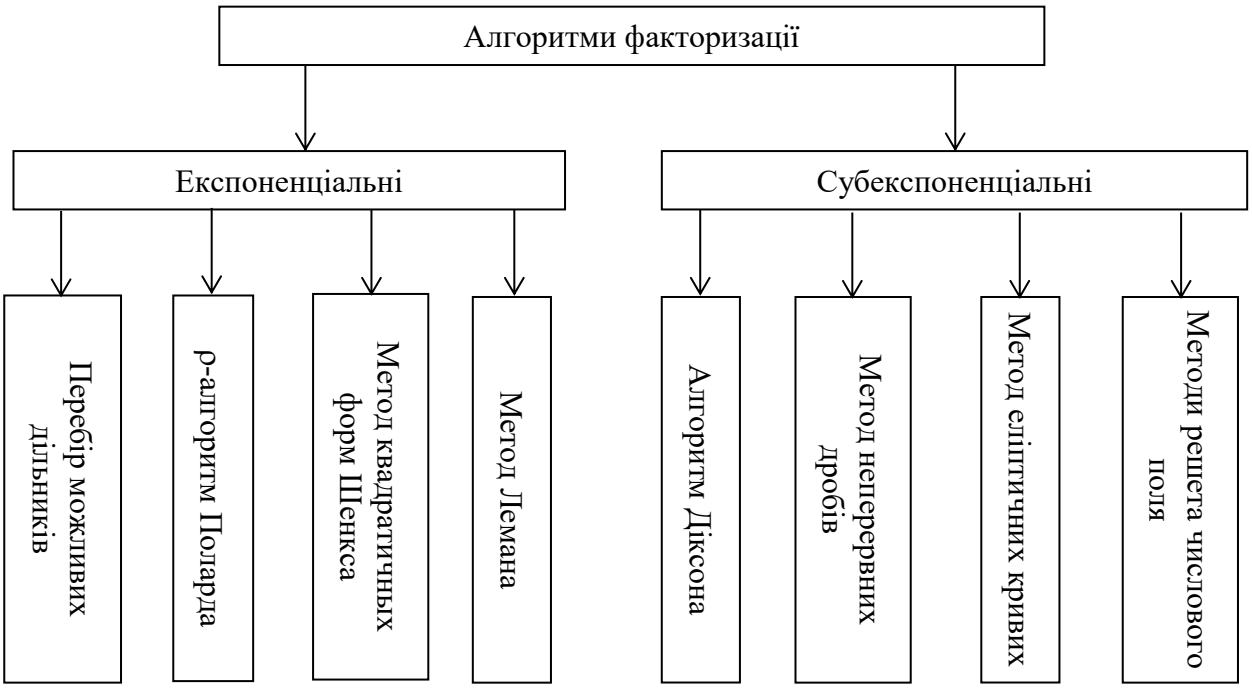


Рис. 1.7 – Класифікація алгоритмів факторизації

Усі ці методи є досить трудомісткі, тому для чисел великої довжини вимагають значних обчислювальних ресурсів. Але теоретичне обгрунтування необхідної складності таких обчислень чи, іншими словами, існування високих нижніх оцінок не доведено, тобто на класичному комп'ютері питання про існування алгоритму факторизації із поліноміальною складністю для виконання факторизації є однією із важливих відкритих проблем сучасної теорії чисел [53].

Перебір дільників (пробне ділення) – це є алгоритм факторизації чи тестування простоти числа шляхом повного перебору усіх можливих потенційних дільників. Зазвичай перебір дільників полягає в переборі всіх простих чисел від 2 до квадратного кореня з числа, що факторизується і в обчисленні залишку від ділення останнього на кожне з цих чисел. Якщо залишок дорівнює нулю, то існує дільник. Після досягнення квадратного кореня і неможливості знайти дільник число, що факторизується оголошується простим [54-56].

У практичних завданнях даний алгоритм застосовується рідко через його велику асимптотичну складність, проте його застосування виправдане у

разі, коли перевіряються числа відносно невеликі, оскільки даний алгоритм досить легко реалізується.

Найбільш поширеним на даний час є метод Ферма [57-59], який ґрунтується на пошуку пар натуральних чисел A і B, для яких виконується рівність $n=A^2-B^2$, де $n=p\cdot q$ – відоме ціле число, що є добутком двох невідомих простих чисел p і q, які необхідно знайти. Для цього шукається $m=\left\lceil\sqrt{n}\right\rceil$ і далі обчислюється параметр $q(t)=(m+t)^2-n$, де $t=1, 2, 3, \ldots$, до тих пір, поки деяке значення $q(t)$ не буде рівним повному квадрату якогось числа, наприклад, B^2. Тоді відповідно $A^2=(m+t)^2$ і шуканий розклад $n=p\cdot q= A^2-B^2= =(A-B)(A+B)$.

Суттєвим недоліком даного методу є значна обчислювальна складність, що обумовлена виконанням великого числа обчислювально складних операцій у позиційній системі числення над багаторозрядними числами, які включають операції піднесення до степеня і добування квадратного кореня над числами, складність яких експоненційно зростає, починаючи із розрядності числа, що факторизується. Однак він є алгоритмічно найпростіший та характеризується формалізацією границь кількості ітерацій, піддається розпаралеленню, не є рекурентним і ймовірнісним, характеризується обмеженим числом типів операцій.

Таким чином, як і метод пробного ділення, алгоритм Ферма має експоненційну оцінку і неефективний для розкладання великих чисел. Метод Ферма можна поліпшити, спочатку виконавши пробне ділення числа n на числа від 2 до деякої константи, виключивши тим самим малі дільники, і тільки потім виконати пошук методом Ферма. Крім того, Ферма-подібні алгоритми відзначаються ефективністю при умові рівності невеликої різниці між множниками.

Узагальнення методу Ферма було запропоновано у [60], де замість пар чисел, які задовольняють співвідношенню $t^2-t_0^2=n$, шукаються пари чисел, що задовольняють більш загальному виразу $t^2\equiv t_0^2 \bmod n$, оскільки досить багато

чисел, які одержуються за формулою $q(t)=(m+t)^2-n$, розкладаються на прості множники.

ρ-алгоритм Джона Полларда, запропонований ним у 1974 році [61], грунтується на алгоритмі Флойда для пошуку довжини циклу у послідовності та деяких наслідках із парадоксу днів народжень. Числова послідовність зациклюється, починаючи с деякого числа. Цикл может бути представлений у вигляді грецької букви ρ, що і послужило назві сімейства алгоритмів. Алгоритм найбільш ефективний при факторизації складених чисел з досить малими множниками в розкладі.

Істотним недоліком алгоритму є необхідність зберігати велику кількість попередніх значень, тому на даний час існує декілька підходів до його покращення.

Метод факторизації цілих чисел, заснований на застосуванні квадратичних форм [62], розроблений Даніелем Шенксом у 1975 році як розвиток методу факторизації Ферма. Для 32-розрядних комп'ютерів алгоритми, що грунтуються на цьому методі, є безумовними лідерами з алгоритмів факторизації для чисел порядку 10^{10}-10^{18} і, ймовірно, такими і залишаться. Даний алгоритм може розділити практично будь-яке складене 18-значне число за час, менший, ніж мілісекунда. Алгоритм є надзвичайно простим, красивим і ефективним. Крім того, методи, що базуються на даному алгоритмі, використовуються як допоміжні при розкладанні дільників великих чисел типу чисел Ферма.

Один з варіантів алгоритму (SQUFOF) [63] використовує групу класів бінарних квадратичних форм з позитивним дискримінантом. У ньому також відбувається знаходження амбігової форми і розкладання дискримінанту на множники. Алгоритм працює з цілими числами, що не перевищують $2\sqrt{n}$. Серед алгоритмів факторизації з експоненційної складністю SQUFOF вважається одним з найефективніших.

Алгоритм Діксона [64] використовує у своїй основі ідею Лежандра, яка полягає у пошуку пари цілих чисел x та y таких, що $x^2 \equiv y^2 \bmod n$ і $x \neq \pm y \bmod n$. Метод Діксона є узагальненням методу Ферма.

У теорії чисел факторизація методом неперервних дробів (CFRAC) є алгоритмом загального вигляду, придатним для факторизації будь-якого цілого числа і який не спирається на спеціальні форми або властивості. Він був винайдений Д. Г. Лемером і Р. Е. Поверсом [65] в 1931-му році і перероблений в алгоритм для комп'ютера Міхаелем А. Моррісоном і Джоном Бріллхартом в 1975-му році [66]. Метод неперервних дробів грунтується на алгоритмі Діксона. Він використовує неперервний дріб, що сходиться до $\sqrt{jn}$, де $j \in N$.

Метод квадратичного решета для факторизації великих чисел розроблений С. Померанцем в 1981 році [67-71]. Довгий час він перевершував інші методи факторизації цілих чисел загального вигляду, що не мають простих дільників, порядок яких значно менший $\sqrt{n}$ (для чисел, що мають прості дільники, набагато менші від $2\sqrt{n}$, швидшим є метод факторизації на еліптичних кривих). Квадратичне решето являє собою різновид методу факторних баз (узагальнення методу факторизації Ферма). Цей метод вважається другим за швидкістю (після загального методу решета числового поля). До цих пір він є найшвидшим для цілих чисел до 100 десяткових цифр і влаштований значно простіше, ніж загальний метод решета числового поля. Це є універсальний алгоритм факторизації, оскільки час його виконання залежить виключно від розміру числа, що факторизується, а не від його особливої структури і властивостей.

Алгоритм намагається знайти такі квадрати чисел, що рівні за модулем n (число, яке факторизується), що часто приводить до факторизації n. Алгоритм працює у два етапи: етап збору даних, де він збирає інформацію, що може привести до рівності квадратів, та етап обробки даних, де він поміщає усю зібрану інформацію у матрицю та обробляє її для отримання

рівності квадратів. Перший етап легко може бути розпаралелений на багато процесів, але другий етап вимагає великих обсягів пам'яті і його розпаралелити важко.

Один із простих методів пошуку рівних квадратів полягає у тому, щоби вибрати випадкове число, піднести його до квадрату та сподіватися, що залишок від ділення на n буде квадратом будь-якого іншого числа. Наприклад, $80^2 \bmod 5959=441=21^2$. Цей спосіб знаходить квадрати лише у рідкісних випадках для великих n, але коли він дійсно знайде ці числа, то тоді можна вважати, що факторизація завершена. Цей метод схожий на метод факторизації Ферма, а також є модифікацією методу розкладання на множники Діксона.

У методі факторизації Ферма окремо підбирається число a, щоб $a^2 \bmod n$ було квадратом. Але підібрати таке число важко. В квадратичном решеті обчислюється $a^2 \bmod n$ для деяких a, і потім знаходиться така підмножина, добуток елементів якого є квадратом. Це приведе до порівняння квадратів.

Наприклад, $41^2 \bmod 1649=32$, $42^2 \bmod 1649=115$ та $43^2 \bmod 1649=200$. Жоден з отриманих результатів не є квадратом, але результат добутку $(32)\cdot(200)=6400=80^2$. З іншого боку, розглянувши попередній добуток за mod 1649, можна знайти, що $(32)\cdot(200) = (41^2)\cdot(43^2) = ((41)\cdot(43))^2$. Оскільки $(41)\cdot(43) \bmod 1649=114$, то отримується порівняння квадратів: $114^2 \equiv 80^2 \pmod{1649}$.

Але як вирішити проблему, фіксуючи множину чисел і знаходячи підмножину, добуток елементів якої є квадратом? Почнемо з поняття вектора показників степенів. Наприклад, є число 504. Його розклад на прості множники має наступний вигляд: $504 = 2^3 3^2 5^0 7^1$. Його можна уявити у вигляді вектора показників степенів (3, 2, 0, 1), який фіксує степені простих чисел 2, 3, 5 та 7, які беруть участь у розкладі. Число 490 по аналогії мало б вектор (1, 0, 1, 2). Множення чисел - це те ж саме, що і поелементне

додавання їх векторів показників степенів, тобто добуток 504·490 описується вектором (4, 2, 1, 3).

Далі, число є квадратом, якщо кожен елемент в його векторі показників степенів парний. Наприклад, при додаванні векторів (3, 0, 0, 1) і (1, 2, 0, 1) отримується (4, 2, 0, 2). Це говорить про те, що добуток чисел 56·126 є квадратом. Насправді, непотрібно шукати точні значення, які одержуються у векторі, до тих пір, доки кожен елемент у результуючому векторі буде парний. З цієї причини можна брати кожен елемент по mod 2 і виконувати додавання елементів по mod 2: (1, 0, 0, 1) + (1, 0, 0, 1) = (0, 0, 0, 0).

Отже, задача прийняла такий вигляд: задано множину векторів (0,1), потрібно знайти таку підмножину, яка доповнюється до нульового вектора при використанні додавання за mod 2. Це є завдання лінійної алгебри, тобто потрібно знайти лінійно залежні вектора. З теореми лінійної алгебри випливає, що доти, поки кількість векторів більша, ніж кількість елементів в кожному векторі, то така залежність має існувати. Можна ефективно знаходити лінійно залежні вектори, наприклад, помістивши вихідні вектори як стовпці матриці і потім використовувати метод Гауса, який легко пристосувати для роботи із цілими числами за mod 2. Як тільки лінійно залежні вектори будуть знайдені, потрібно просто перемножити числа, відповідні цим векторах і отримати шуканий квадрат.

Однак піднесення до квадрату множини випадкових чисел по mod n призводить до великої кількості різних простих множників, довгих векторів і великих матриць. Щоб позбутися від цієї проблеми, потрібно спеціально шукати числа такі, що $a^2 \bmod n$ має тільки прості невеликі множники (такі числа називаються гладкими). Їх знайти складніше, але використання таких чисел дозволяє уникнути великих векторів та матриць [72].

Алгоритм факторизації натурального числа із використанням еліптичних кривих володіє субекспоненціальним часом виконання [73-78]. Він третій за швидкістю роботи після загального методу решета числового поля та методу квадратичного решета.

На практиці цей метод часто використовується для виявлення (або відкидання) невеликих простих дільників числа. Якщо отримане після роботи алгоритму число все ще є складним, то інші співмножники є великими числами. При збільшенні кількості кривих шанси знайти простий дільник зростають, але залежність очікуваної кількості еліптичних кривих від кількість цифр у шуканому дільнику експоненційна.

Спеціальний метод решета числового поля (SNFS) є методом факторизації цілих чисел спеціального виду, наприклад, чисел Мерсенна. З нього був отриманий загальний метод решета числового поля, що є найбільш ефективним алогритмом факторизації великих цілих чисел [79-81].

SNFS грунтується на більш простому методі раціонального решета і працює наступним чином. Нехай n число для факторизації. Аналогічно методу раціонального решета, алгоритм SNFS може бути розбитий на два етапи:

1) знаходження числа мультиплікативних співвідношень факторної бази, більшого, ніж число елементів у факторній базі;

2) перемноження співвідношень між собою таким чином, щоб степені отриманих добутків були однаковими, тобто отримання співвідношень виду $a^2 \equiv b^2 \pmod{n}$. Це в свою чергу призведе до факторизації числа n: n=НСД ($a+b$, n)×НСД ($a-b$, n). У разі, якщо отриманий розклад є тривіальним (тобто $n=n\times1$), то слід шукати інші добутки співвідношень, що задовольняють даній умові. Цей крок ідентичний кроку в методі раціонального решета і є завданням лінійної алгебри, на відміну від першого кроку, який в цьому методі є більш ефективним.

Даний метод також можна використовувати, крім чисел Мерсенна, для чисел, які представлені у вигляді полінома з невеликими коефіцієнтами. Справа в тому, що спеціальний метод решета числового поля виробляє просіювання для двох числових полів. Ефективність методу сильно залежить від розміру певних елементів в цих полях. Якщо число можна представити у вигляді полінома з маленькими коефіцієнтами, то числа, з якими проводяться

обчислення, набагато менші чисел, з якими доводиться мати справу, якщо такого полінома не існує [82-86].

Метод решета числового поля (і спеціальний, і загальний) можна уявити як удосконалення більш простого методу - методу раціонального решета або методу квадратичного решета [87]. Подібні їм алгоритми вимагають знаходження гладких чисел порядку $\sqrt{n}$. Розмір цих чисел експоненціально зростає із зростанням n. Метод решета числового поля, у свою чергу, вимагає знаходження гладких чисел субекспоненціального відносно n розміру. Завдяки тому, що ці числа менші, ймовірність того, що такого розміру число виявиться гладким, вища, що і є причиною ефективності методу решета числового поля. Для досягнення прискорення обчислення в рамках методу проводяться у числових полях, що ускладнює алгоритм у порівнянні із більш простим раціональним решетом.

Основні принципи загального решета числового поля:

1) метод факторизації Ферма для факторизації натуральних непарних чисел n, який полягає в пошуку таких цілих чисел t і t_0, що $t^2-t_0^2=n$ і, відповідно, приводить до розкладу $(t-t_0)(t+t_0)=n$;

2) знаходження підмножини у множині цілих чисел, добуток яких – квадрат;

3) формування факторної бази;

4) просіювання виконується подібно решету Ератосфена (звідки метод і отримав свою назву). Решетом служать прості числа факторної бази і їх степені. При просіюванні число не «викреслюються», а діляться на число з решета. Якщо в результаті число виявилося одиницею, то воно гладке;

5) основна ідея полягає в тому, щоб замість перебору чисел і перевірки діляться їх квадрати по модулю n на прості числа з факторної бази, перебираються прості числа з бази і відразу для всіх чисел виду t^2-n перевіряється, діляться вони на це просте число або його степінь.

У [88] запропоновано метод розкладання на множники добутку двох простих чисел на основі розв’язання задачі дискретного логарифмування.

Показано, що запропонований метод і метод Ферма є еквівалентними за обчислювальною складністю, однак число ітерацій для методу дискретного логарифмування в $0{,}5\log_2 N$ менше, ніж для методу Ферма.

У [89] показано, що при наявності дуже близьких розв'язків найбільш ефективним є метод Ферма, при відносно невеликому множнику – метод еліптичних кривих, при малих множниках – методи Шенкса і ρ-алгоритм Джона Полларда. В загальному випадку потрібно використовувати метод квадратичного решета.

1.4 Теоретичні основи СЗК

Усяка обчислювальна структура тісно пов'язана із системою числення, у якій вона працює. Під системою числення розуміється сукупність прийомів для позначення (або запису) чисел, чи точніше, спосіб кодування (або подання) елементів деякої кінцевої моделі дійсних чисел словами одного або більше алфавітів. Кодування являє собою ін'єктивне відображення діапазону системи числення на декартовий добуток його алфавітів, тобто $F : D \to A,$ де $A = A_1 \times A_2 \times ... \times A_j$, тобто відображення F елементу $x \in D$ ставить у відповідність кодове слово $(x_1, x_2, ..., x_j)$, де $x_i = A_i (i = \overline{1,n})$ – i–а цифра, j – довжина коду. За допомогою зворотного відображення F^{-1}, яке називається декодуванням, також можна визначити систему числення.

У будь–якій кодовій системі повинні виконуватись наступні вимоги [90-92]:

– можливість подання у даній системі будь–якої величини у розглянутому діапазоні, який заздалегідь визначено;

– одиничність подання – будь–яка кодова комбінація відповідає одному й тільки одному числу в заданому діапазоні;

– простота виконання операцій із числами в даній системі числення.

Отже, коди чисел являють собою імена числових об'єктів, які становлять числовий діапазон. Діапазони як моделі дійсних чисел повинні із максимально доступною повнотою та простотою відбивати властивості числової множини.

Усяке подання чисел робочого діапазону є лише складеним елементом відповідної машинної арифметики і не може розглядатися окремо від неї [93-94]. Арифметичні властивості тої або іншої системи числення насамперед визначаються характером міжрозрядних зв'язків, які з'являються у ході виконання операцій над кодовими словами. Дослідження показують [95-98], що в рамках звичайної ПСЧ значного прискорення виконання операцій домогтися неможливо. Це пояснюється тим, що в ПСЧ значення розряду будь–якого числа, крім молодшого, що є результатом двомісної арифметичної операції, залежить не тільки від значення однойменних операндів, але й від усіх молодших розрядів, тобто ПСЧ має строго послідовну структуру. Однак сьогодні перевага віддається обчислювальним структурам, що володіють здатностями до паралельної обробки інформації [99-106]. Цими особливостями володіють непозиційні коди із паралельною структурою, що дозволяють реалізувати ідею розпаралелювання операцій на рівні виконання елементарних арифметичних операцій. Ця думка зародилася у середині 50–х років минулого століття, коли чеські вчені М. Валах та Л. Свобода в своїх дослідженнях в області непозиційних систем числення [22, 107, 108] розглядали подання чисел у вигляді набору залишків від ділення на обрані натуральні модулі – основи системи. Подібну систему числення і стали називати СЗК або модулярною системою числення (МСЧ). Слідом за чеськими вченими можливість застосування цієї системи в обчислювальних системах розглянута в дослідженнях таких зарубіжних та вітчизняних учених: І. Акушського [109, 110], Д. Юдіцького [111], В. Торгашова [112], В. Амербаєва [113, 114], М. Червякова [115-117], В. Краснобаєва [118-121], Я. Николайчука [122-125], A. Omondi [126], Б. Premkumar [127-128], A. Mohan [129-130], О. Фінько [131], В. Яцківа [132].

Нехай задані додатні взаємно прості числа $p_1, p_2, ..., p_j$, які називають основами або модулями системи. Позначимо $P = \prod_{i=1}^{j} p_i$. Ця величина характеризує величину діапазона системи [111, 126, 130]. Під СЗК розуміють таку непозиційну систему числення, у якій ціле невід'ємне число *N* можна представити як набір залишків від ділення цього числа на обрані основи системи, тобто

$$A = (\alpha_1, \alpha_2, ..., \alpha_n),\ \alpha_i = A - \left[\frac{A}{p_i}\right] \cdot p_i, \left(i = \overline{1, j}\right). \qquad (1.22)$$

Можливість такого подання числа визначається теоремою про ділення з остачею в кільці цілих чисел [133]: якщо $A \in Z, p \in Z, p \neq 0$, то існують єдині $q_0 \in Z, r_0 \in Z$, такі, що $A = q_0 p + r_0, 0 \leq r_0 < |p|, q_0 = \left[\frac{A}{p}\right]$.

Нескладно помітити, що кожна остача r_i виходить незалежно від інших та містить інформацію про все число.

Установити взаємно–однозначну відповідність між цілими числами з діапазону [0, *P*) і їхніми залишками дозволяє КТЗ [134-135].

Можливість застосування СЗК в обчислювальних алгоритмах обумовлює наявністю певного ізоморфізму між математичними операціями над цілими числами й відповідними операціями над системою цілих невід'ємних залишків по окремих модулях. Причому додавання, множення та піднесення до цілого додатного степеня будь–яких цілих додатних чисел повністю ідентичні відповідним операціям, що виконуються над системою залишків [111].

Нехай операнди *A* і *B*, а також результати операцій додавання та множення *A* + *B* та *A*·*B* представлені відповідно залишками α_i, β_i, γ_i, δ_i за

модулями $p_i, (i = \overline{1, j})$, причому ці два числа та результати перебувають в діапазоні [0, *P*), тобто

$$\begin{aligned} A &= (\alpha_1, \alpha_2, ..., \alpha_j), \\ B &= (\beta_1, \beta_2, ..., \beta_j), \\ A + B &= (\gamma_1, \gamma_2, ..., \gamma_j), \\ A \cdot B &= (\delta_1, \delta_2, ..., \delta_j), \end{aligned} \tag{1.23}$$

і $0 \le A < P$, $0 \le B < P$, $0 \le A + B < P$, $0 \le A \cdot B < P$.

Вирази (1.23) можна переписати у такому вигляді:

$$\gamma_i = \alpha_i + \beta_i (\mathrm{mod}\, p_i); \ \delta_i = \alpha_i \beta_i (\mathrm{mod}\, p_i) \tag{1.24}$$

$$\gamma_i = \alpha_i + \beta_i - \left[\frac{\alpha_i + \beta_i}{p_i} \right] p_i, \ \delta_i = \alpha_i \beta_i - \left[\frac{\alpha_i \beta_i}{p_i} \right] p_i. \tag{1.25}$$

Справедливість таких правил виконання арифметичних операцій в СЗК випливає безпосередньо з властивостей конгруенцій.

Дійсно, на підставі (1.22) $\gamma_i = A + B - \left[\frac{A + B}{p_i} \right], (i = \overline{1, j})$.

З представлення *A* та *B* за теоремою про ділення з остачею випливає, що $A = s_{1i} p_i + \alpha_i$, $B = s_{2i} p_i + \beta_i$, $(i = \overline{1, j})$, $s_{1i} \in Z, s_{1i} \ge 0, s_{2i} \in Z, s_{2i} \ge 0$.

Тоді $A + B = (s_{1i} + s_{2i}) p_i + \alpha_i + \beta_i, \left[\frac{A + B}{p_i} \right] = s_{1i} + s_{2i} + \left[\frac{\alpha_i + \beta_i}{p_i} \right] p_i$.

Звідси $\gamma_i = \alpha_i + \beta_i - \left[\frac{\alpha_i + \beta_i}{p_i} \right] p_i$.

У випадку множення $\delta_i = AB - \left[\frac{AB}{p_i} \right] p_i$. Тоді

$$AB = s_{1i}s_{2i}p_i^{\,2} + (\alpha_i s_{2i} + \beta_i s_{2i})p_i + \alpha_i\beta_i,\ \left[\frac{AB}{p_i}\right] = s_{1i}s_{2i}p_i + \alpha_i s_{2i} + \beta_i s_{1i} + \left[\frac{\alpha_i\beta_i}{p_i}\right].$$

Отже, $\delta_i = \alpha_i\beta_i - \left[\frac{\alpha_i\beta_i}{p_i}\right]p_i, (i = \overline{1,n})$.

Приклади.

Дані: $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$. $A = (0, 0, 3, 4)$, $B = (1, 1, 2, 0)$.

Знайти: $A+B$, $A - B$, AB.

Рішення: $P = \prod_{i=0}^{n} p_i = 2 \cdot 3 \cdot 4 \cdot 7 = 210.$

$A+B = (0, 0, 3, 4) + (1, 1, 2, 0) = (1, 1, 0, 4)$.

$AB = (0, 0, 3, 4) \cdot (1, 1, 2, 0) = (0, 0, 1, 0)$.

$A - B = (0, 0, 3, 4) - (1, 1, 2, 0) = (1, 2, 1, 4)$.

На відміну від ПСЧ, у якій число A представляється у вигляді

$$A = A_n N^n + A_{n-1}N^{n-1} + ... + A_0 N^0 = \sum_{i=0}^{n} A_i N^i, \qquad (1.26)$$

де N – основа ПСЧ, значення числа в модулярному коді не залежить від місця розташування кожного розряду в його представленні, а залежить від значення основи відповідного розряду. Тому модулярний код є непозиційним.

Отже, виконання арифметичних операцій в модулярному коді відбувається незалежно по кожному із модулів, що і вказує на паралелізм цієї системи. Така обставина визначає порозрядне виконання операцій. Ця властивість позбавляє від необхідності «позичати» чи «переносити» одиницю старшого розряду, що приводить до появи кодів із паралельною структурою. Це дозволяє розпаралелити алгоритми при виконанні арифметичних операцій [117, 136-140].

Переведення чисел із ПСЧ у СЗК за допомогою виразу (1.22) пов'язаний з реалізацією операції ділення, яка характеризується значною

обчислювальною складністю [141-148], тому використання даного методу є неефективним.

Отже, операції додавання й множення над числами, представленими у СЗК, зводяться до відповідних операцій над числами цього подання [149-154]. Це відноситься і до піднесення до степеня, до обчислення значень многочлена тощо. Операція віднімання в СЗК замінюється додаванням з адитивною інверсією від'ємного числа. Усі ці операції є модульні, тобто вони не вимагають позиційних характеристик чисел, які опрацьовуються.

Дослідження СЗК виявили цілий ряд переваг:

1) максимальний паралелізм [155-158]. Для оцінки рівня паралелізму системи числення вводиться спеціальний показник

$$\prod(v) = \frac{n(v)}{\lambda}, \tag{1.27}$$

де λ – довжина коду системи,

$n(v)$ – кількість порозрядних показників паралелізму $\pi_1, \pi_2, ..., \pi_j$, не менших заданого порога $v\left(\frac{1}{j} \le v \le 1\right)$, причому $\pi_i = 1 - \frac{n_i}{j} (i = \overline{1, j})$, n_i – максимально можливе число пар цифр (x_j, y_j), які здійснюють вплив на значення суми $Z = X + Y$ в ході її формування мовою даного коду. Для СЗК показник паралелізму приймає максимально можливе значення 1. Це говорить про відсутність міжрозрядних зв'язків у числі, записаному в СЗК;

2) малоразрядність залишків [139, 159-163]. Тому через малу кількість всеможливих кодових комбінацій з'являється можливість побудови табличної арифметики. При цьому більшість операцій перетворюються в однотактові, що здійснюється простою вибіркою з таблиць. У міру вдосконалювання технології виробництва запам'ятовувальних пристроїв з високою щільністю запису інформації, що становлять технічну систему табличного методу обчислень, інтерес до СЗК неухильно зростає;

3) реалізація принципу конвейєрної обробки інформації [164-167]. Це означає, що при виконанні обчислень модульні й наступні за ним операції вдається сполучити за часом тоді, коли чергові операції залежать від результатів поточних операцій, що ще не закінчилися. Таким чином, алгоритми модулярної арифметики мають конвеєрну структуру;

4) висока точність, надійність, здатність до самокорекції [168-179]. Причому в СЗК можна побудувати непозиційні коди, які виявляють та виправляють помилки, що є повністю арифметичними, тобто у цих кодах інформативна та контрольна частини щодо будь-якої операції рівноправні. Така особливість надає можливість варіювати коригувальною здатністю коду за рахунок зміни точності обчислень.

Звичайно, ця система теж не позбавлена недоліків. До них відноситься неможливість візуально порівнювати числа [180-182], відсутність ознак виходу результатів за межі діапазону [183-186], обмеженість дії системи сферою цілих додатних чисел [126, 130, 187-189], одержання у всіх випадках точного результату операції [190-192], що виключає можливість безпосереднього округлення, а також труднощі виконання немодульних операцій [193-196]. Але вони не є непереборними.

У СЗК до модульних відносяться операції додавання, віднімання і множення. Аналіз під час використання арифметики СЗК вказує, що усі їх ланки мають однакову структуру, типовим елементом якої є послідовність виду

$$V = \left| \sum_{i=1}^{j} \left| F_i(\alpha_i) \right|_{p_i} \right|_{p_i}, \qquad (1.28)$$

де $F_i(\alpha_i)$ – цілочисельна функція залишка α_i за деяким модулем;

p_i – основа СЗК;

$|...|_{p_i}$ – операція визначення найменшого залишку за модулем p_i.

До немодульних операцій відносяться операції, при яких значення того або іншого результату розряду залежить від його всіх або декількох розрядів вихідного числа.

Пристрої, що реалізують немодульні операції, досить чітко розділяються на два типи [197-198].

Прикладом пристрою першого типу є пристрій згортки, який забезпечує обчислення:

$$V = \left| \sum_{i=1}^{j} \left| A_i Q_i \right|_{p_i} \right|_{p_i}, \qquad (1.29)$$

де A_i – значення i-го розряду вихідного числа, представленого у ПСЧ;

Q_i – ваговий коефіцієнт.

Пристрої згортки широко використовуються у цифрових системах, які функціонують у СЗК і представляють істотну, а часом і основну частину устаткування, призначену для реалізації ряду способів виконання операцій – переводу чисел із ПСЧ у СЗК, ділення довільних чисел і деяких інших. Крім того, такі пристрої знаходять застосування й у цифрових системах, що функціонують у ПСЧ.

Прикладом пристроїв другого типу є пристрої позиційного перетворення [199], що забезпечують одержання характеристик, які вказують на приналежність числа, представленого в СЗК, тому або іншому інтервалу діапазона можливого представлення чисел.

Математичною основою для пристроїв першого типу є визначення найменших невід’ємних залишків, які визначаються згортками вихідного числа за кожним модулем. Для визначення згорток за кожним модулем необхідно перевести число з ПСЧ в СЗК.

Переведення числа в СЗК можна здійснити методом ділення. Однак через операцію ділення технічна реалізація такого методу неефективна для

машинного використання, крім того, даний метод вимагає застосування арифметичного пристрою в ПСЧ.

Розглянемо метод переведення числа з ПСЧ в СЗК, що не містить операції ділення і який називаний методом безпосереднього підсумовування модульних значень розрядів позиційного числа.

Нехай число X записане у позиційній системі числення із основою N, тобто

$$X = A_j N^j + A_{j-1} N^{j-1} + ... + A_0 N^0 \text{ або } X = \sum_{i=0}^{j} A_i N^i, \qquad (1.30)$$

де $0 \le A_i \le N-1$.

Представимо степені основи N^i і коефіцієнти A_i у СЗК з основами $p_1, p_2, ..., p_j$, тоді:

$$N^i = (B_1^{(i)}, B_2^{(i)}, ..., B_n^{(i)}),\ A^i = (A_1^{(i)}, A_2^{(i)}, ..., A_n^{(i)}). \qquad (1.31)$$

Підставивши (1.31) в (1.22), можна одержати:

$$X = (\sum_{i=0}^{j-1} A_1^{(i)} B_1^{(i)} \bmod p_1, \sum_{i=0}^{j-1} A_2^{(i)} B_2^{(i)} \bmod p_2, ..., \sum_{i=0}^{j-1} A_j^{(i)} B_j^{(i)} \bmod p_j). \qquad (1.32)$$

Таким чином, для утворення числа X у СЗК потрібно k констант, що є степенями p_i й $p_i - 1$ констант, що відповідають значенням A_i.

Маючи в пам'яті процесора масив з $j + p_i - 1$ констант, все переведення може бути здійснений процесором, що працює в СЗК [200-201].

Розглянутий метод є основою широкого вибору можливих апаратурних реалізацій цифрових перетворювачів ПСЧ – СЗК, які розрізняються між собою як по складу й кількості використовуваних елементів, так і по

швидкості перетворення інформації. Відомі у літературі цифрові перетворювачі ПСЧ – СЗК, що ґрунтуються на даному методі і аналіз яких дозволив зробити важливі висновки про те, що одним з істотних недоліків подібних перетворювачів є більші апаратурні витрати при переведенні чисел великої розрядності й низька швидкодія. Підвищені вимоги, пов'язані зі зменшенням апаратурних засобів і збільшенням швидкості обробки інформації, привели до необхідності глибокого вивчення питань розробки ефективних алгоритмів. Для вирішення цієї задачі пропонуються два методи. Розглянемо перший метод. Він грунтується на теоремі, що є основою даного методу перетворення чисел із ПСЧ у СЗК як апаратурними, так і програмними способами: нехай число X записане у позиційній системі числення із основою N і якщо $X_j = \sum_{i=0}^{n_0} A_i^{(j)} C_i^j$, де $C_i \equiv N^i (\mathrm{mod}\, p_i)$, p_i – просте число, n_0 – число розрядів p_i (при i = 1, 2,…, n_0), то $X \equiv X_j (\mathrm{mod}\, p_j)$ і $X_j < X$.

Розглянемо другий метод, що знайшов широке застосування в літературі. Назвемо його методом послідовного множення й підсумовування. Суть методу полягає в наступному. Нехай число записане у вигляді (1.14). Інакше цей вираз можна записати так:

$$X = (...(A_j N + A_{j-1})N + A_{j-2})N + ... + A_1)N + A_0 \equiv X_1 \bmod p_j = -\alpha_j \bmod p_j$$

$$\begin{gathered}(A_j N + A_{j-1})N \equiv X_j (\mathrm{mod}\, p_j), \\ (X_j \bmod P_j + A_{j-2})N \equiv X_{j-1} (\mathrm{mod}\, p_j), \\ (X_3 \bmod p_j + A_1)N \equiv X_2 (\mathrm{mod}\, p_j), \\ X_2 \, mod \, p_j + A_0 \equiv X_1 \, mod \, p_j = \alpha_j \, mod \, p_j .\end{gathered} \qquad (1.33)$$

Тобто значення числа X у СЗК за модулем p_j утвориться шляхом множення старшого розряду на основу системи числення N, потім

підсумовування отриманого результату із значенням наступного розряду за модулем p_j, потім множення отриманого результату на основу *N* за модулем p_j. Такі послідовні множення й підсумовування за модулем виконуються доти, поки при підсумовуванні не буде додане значення молодшого розряду.

Слід зазначити, що розглянутий метод дозволяє реалізувати досить економічні стосовно апаратурних витрат цифрові пристрої перетворення інформації [202].

СЗК володіє однією особливістю, яку можна відносити до недоліків цієї системи: не можна визначити візуально величину числа, яке представлене у СЗК, а отже й неможливе виконання таких операцій, як порівняння чисел, визначення знаку числа. Один зі шляхів вирішення цієї проблеми полягає у перетворенні чисел із СЗК у ПСЧ. Оцінимо існуючі способи переведення як традиційні (метод ортогональних базисів; переведення числа в узагальнену позиційну систему, так і нові (інтервальні методи переведення).

В основі методу ортогональних базисів відновлення числа по його залишкам при переведенні із СЗК у ПСЧ лежить КТЗ (див. п. 1.1).

Нехай взаємно прості основи СЗК $p_1, p_2, ..., p_j$; $P = \prod_{i=1}^{j} p_i$ – діапазон системи. З вибором системи визначаються її основні константи – базиси B_i, $i = \overline{1, j}$. Задача переведення числа $A=(\alpha_1, \alpha_2, ..., \alpha_j)$ у ПСЧ полягає у визначенні таких чисел B_i, $i = \overline{1, j}$, щоб $A = \sum_{i=1}^{j} p_i B_i$. Для однозначного визначення p_i на базиси системи B_i накладається ряд обмежень і показується, що даною властивістю володіють такі базиси:

$B_1 = (1, 0, 0, \ldots, 0, 0)$, $B_2 = (0, 1, 0, \ldots, 0, 0)$, ..., $B_j = (0, 0, 0, \ldots, 0, 1)$,

які називаються ортогональними.

Тоді у випадку ортогональних базисів $P_i = \alpha_i$, $i = \overline{1, j}$. Ортогональні базиси визначаються по формулі

$$B_i = \frac{m_i P}{p_i} = m_i P_i \ , \ i = \overline{1, j}, \tag{1.34}$$

де

$$P_i = \frac{P}{p_i} = p_1 \cdot p_2 \cdot \ldots \cdot p_{i-1} \cdot p_{i+1} \cdot \ldots \cdot p_j \ , \tag{1.35}$$

m_i – цілі позитивні числа, що називаються вагами базису, їх визначають із порівнянь

$$P_i m_i \ (\mathrm{mod}\, p_i) = 1. \tag{1.36}$$

Тоді за КТЗ число $A = (\alpha_1, \alpha_2, \ldots, \alpha_j) = \sum_{i=1}^{j} \alpha_i B_i (\mathrm{mod} P)$.

Таким чином, якщо знайдені ортогональні базиси для системи основ, то для переведення числа $A = (\alpha_1, \alpha_2, \ldots, \alpha_j)$ досить обчислити $\sum_{i=1}^{j} \alpha_i B_i$ й ввести цю суму в діапазон $[0;\ P)$ відніманням величини, кратної P, тобто

$$A = \left| \sum_{i=1}^{j} \alpha_i B_i \right|_p = \sum_{i=1}^{j} \alpha_i B_i - r_A P \ , \tag{1.37}$$

де r_A – ранг числа A, який показує, скільки разів треба відняти величину діапазону P з отриманого числа, щоб повернути його в діапазон.

Розглянемо приклад. Нехай дана система основ $p_1 = 2,\ p_2 = 3,\ p_3 = 5,\ p_4 = 7,\ p_5 = 11$, об'єм діапазону $P = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. перевести число $A = (1, 2, 1, 4, 7)$ у позиційну систему.

Обчислимо ортогональні базиси. Для цього знайдемо величини P_i:

$$P_1 = \frac{P}{p_1} = 1155,\ P_2 = \frac{P}{p_2} = 770,\ P_3 = \frac{P}{p_3} = 462,\ P_4 = \frac{P}{p_4} = 330,\ P_5 = \frac{P}{p_5} = 210.$$

Шукаємо ваги базисів:

$$1155\, m_1 \equiv 1 (\mathrm{mod}\ 2),\ m_1 \equiv 1 (\mathrm{mod}\ 2).$$

$$770\, m_2 \equiv 1 (\mathrm{mod}\ 3),\ m_2 \equiv 2 (\mathrm{mod}\ 3).$$

$$462\, m_3 \equiv 1 (\mathrm{mod}\ 5),\ m_3 \equiv 3 (\mathrm{mod}\ 5).$$

$$330\, m_4 \equiv 1 (\mathrm{mod}\ 7),\ m_4 \equiv 1 (\mathrm{mod}\ 7).$$

$$210\, m_5 \equiv 1 (\mathrm{mod}\ 11),\ m_5 \equiv 1 (\mathrm{mod}\ 11).$$

Тоді одержуємо самі базиси:

$$B_1 = m_1 \cdot P_1 = 1 \cdot 1155 = 1155,$$

$$B_2 = m_2 \cdot P_2 = 2 \cdot 770 = 1540,$$

$$B_3 = m_3 \cdot P_3 = 3 \cdot 462 = 1386,$$

$$B_4 = m_4 \cdot P_4 = 1 \cdot 330 = 330,$$

$$B_5 = m_5 \cdot P_5 = 1 \cdot 210 = 210.$$

Обчислимо величину числа A:

$$A = \left| 1 \cdot 1155 + 2 \cdot 1540 + 1 \cdot 1386 + 4 \cdot 330 + 7 \cdot 210 \right|_{2310} = \left| 8411 \right|_{2310} = 1481.$$

Оскільки ортогональні базиси B_i повністю визначаються вибором основ системи, тому вони можуть бути обчислені заздалегідь, причому один раз.

Недоліком розглянутого методу є те, що доводиться мати справу із великими числами B_i і, крім того, дії додавання та множення треба виконувати в ПСЧ, а отриманий результат необхідно вводити у діапазон відніманням величини, кратної P.

1.5 Сучасний стан та напрями підвищення ефективності використання СЗК при опрацюванні багаторозрядних чисел в асиметричних криптосистемах

З вищесказаного випливає, що використання СЗК в комп'ютерних системах дозволяє істотно підвищити швидкодію реалізації цілочисельних арифметичних операцій [203-204]. Наприклад, в [205] проведено розрахунок та порівняльний аналіз продуктивності комп'ютерної системи обробки цілочисельних даних, представлених в СЗК. На основі порівняльної оцінки продуктивності системи 15Э1235 (розробка алгоритму вибору шляху для комутації повідомлень) та літературних джерел [206] було отримано табл. 1.4 з часом виконання відповідних операцій над 32-бітними словами в ПСЧ та СЗК.

Таблиця 1.4 – Характеристики системи 15Э1235 в ПСЧ та СЗК

№	Тип операції	К-сть операцій	Час виконання операцій (ум.од.)	
			ПСЧ	СЗК
1	Звертання до ОЗП	360	1080	1080
2	Звертання до ПЗП	100	300	300
3	Додавання	1314	9198	275
4	Множення	1600	320000	320
5	Порівняння	63	441	6

Крім того, показано, що продуктивність системи під час роботи з алгоритмами вибору шляху у ПСЧ становить всього 3 еталонних задачі за відносну умовно вибрану одиницю часу, а при використанні СЗК - 500 еталонних задач за відносну умовну одиницю часу, тобто майже в 170 разів більше.

В табл. 1.5 показано, що використання технологій паралельної обробки даних, зокрема СЗК, забезпечує більш високу надійність (за ймовірністю безвідмовної роботи), ніж при використанні двійкової ПСЧ, при меншій кількості додатково введеного обладнання [206].

Таблиця 1.5 – Показники продуктивності та надійності системи 15Э1235 в ПСЧ та СЗК

Показник	ПСЧ	СЗК
Продуктивність (ум.од.)	3	500
Надійність (ймовірність безвідмовної роботи)	0,966	0,9999
Відносна к-сть обладнання (ум.од.)	64	60
К-сть додаткового обладнання (%)	100	87,5

Відповідно, в [118] стверджується, що продуктивність модулярної комп’ютерної системи може бути в десятки або й сотні разів більшою, ніж у ПСЧ при тій самій тактовій частоті.

В циклі робіт [207-210] запропоновано та досліджено ефективний метод для паралельного виконання високоточних арифметичних операцій над багаторозрядними числами на основі СЗК. Його суть полягає в перетворенні вихідних операндів в групи незалежних чисел меншої розрядності з наступною паралельною обробкою елементів цих груп на багатоядерних обчислювачах (рис. 1.8). При цьому можна здійснити налаштування базису обчислень як під роботу з числами конкретної розрядності, так і під конкретну архітектуру системи, що дає можливості оптимізації обчислювального процесу.

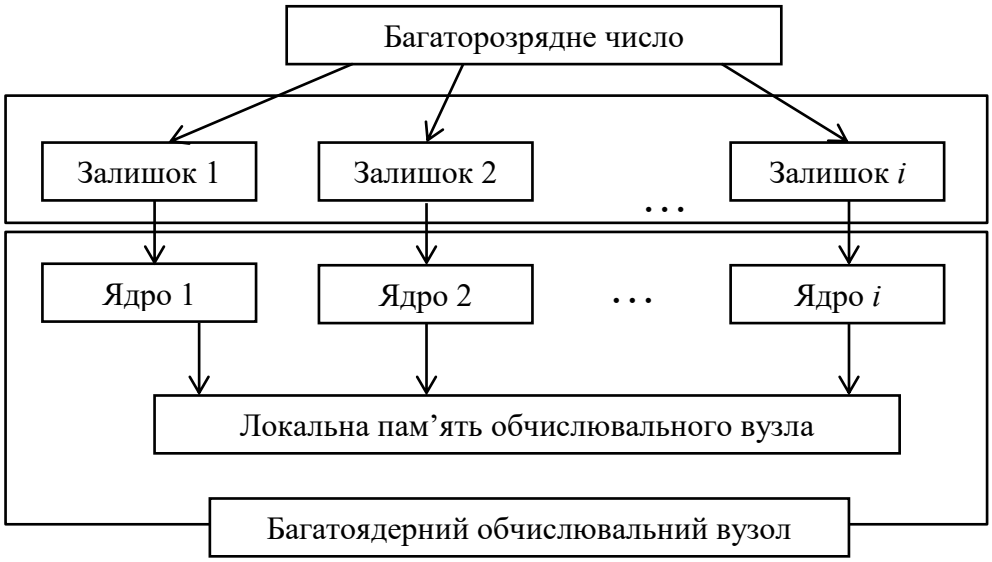


Рис. 1.8 – Схема розподілу багаторозрядного числа між ядрами обчислювального вузла

В ході експерименту обчислювався добуток матриць розмірністю 700x700, елементами якої були цілі числа розрядністю від 32 до 248 біт, на основі послідовного та паралельного алгоритмів множення. Схеми розподілу масивів при паралельному алгоритмі для СЗК та ПСЧ представлені відповідно на рис. 1.9 та рис. 1.10.

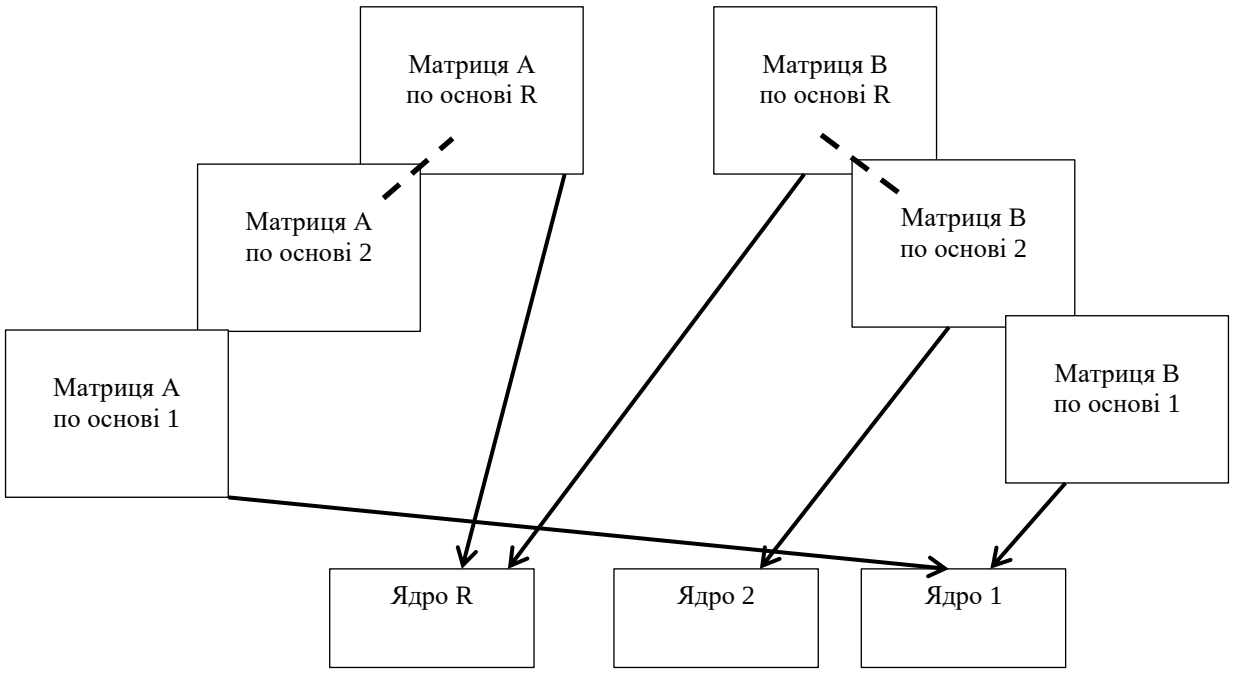


Рис. 1.9 – Схема розподілу масивів у паралельному алгоритмі СЗК

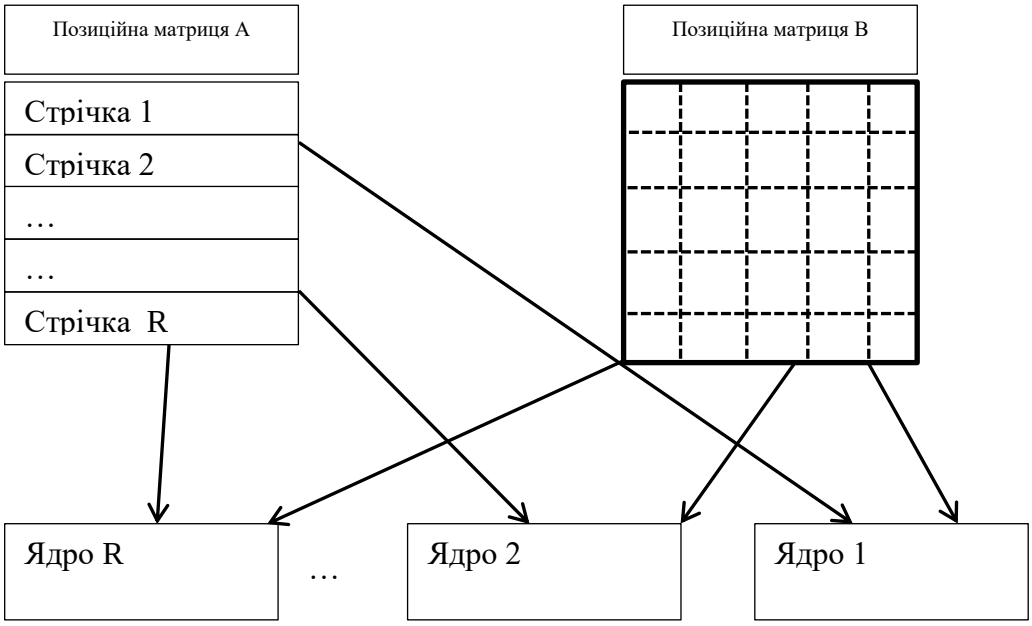


Рис. 1.10 – Схема розподілу масивів у паралельному алгоритмі ПСЧ

Графіки залежності часу множення від розрядності числових даних та прискорення, яке досягається при використанні СЗК, наведені відповідно на рис. 1.11 та рис. 1.12. Видно, що вже при 32-бітних операндах час множення матриць з СЗК значно (приблизно в 40 разів при паралельному алгоритмі) менший часу множення матриць в двійковій системі числення. При збільшенні розрядності чисел прискорення з використанням СЗК постійно зростає і для 248-бітних елементів досягає 125 разів. Це пояснюється тим, що із збільшенням розрядності позиційні методи істотно сповільнюються, а в СЗК час обчислень не залежить від розрядності в силу замикання арифметичних операцій відносно кільця лишків по вибраних модулях.

Одним із шляхів підвищення швидкодії обчислювачів, які працюють у СЗК, є вибір спеціалізованих наборів модулів, від чого істотно залежить час виконання як модульних, так і немодульних операцій. Тому в СЗК запропоновано багато різних наборів модулів різного виду та різної кількості для визначених застосувань, які істотно впливають на всі частини апаратної реалізації, включаючи прямі перетворювачі, модульні арифметичні канали, зворотні перетворювачі [211-213].

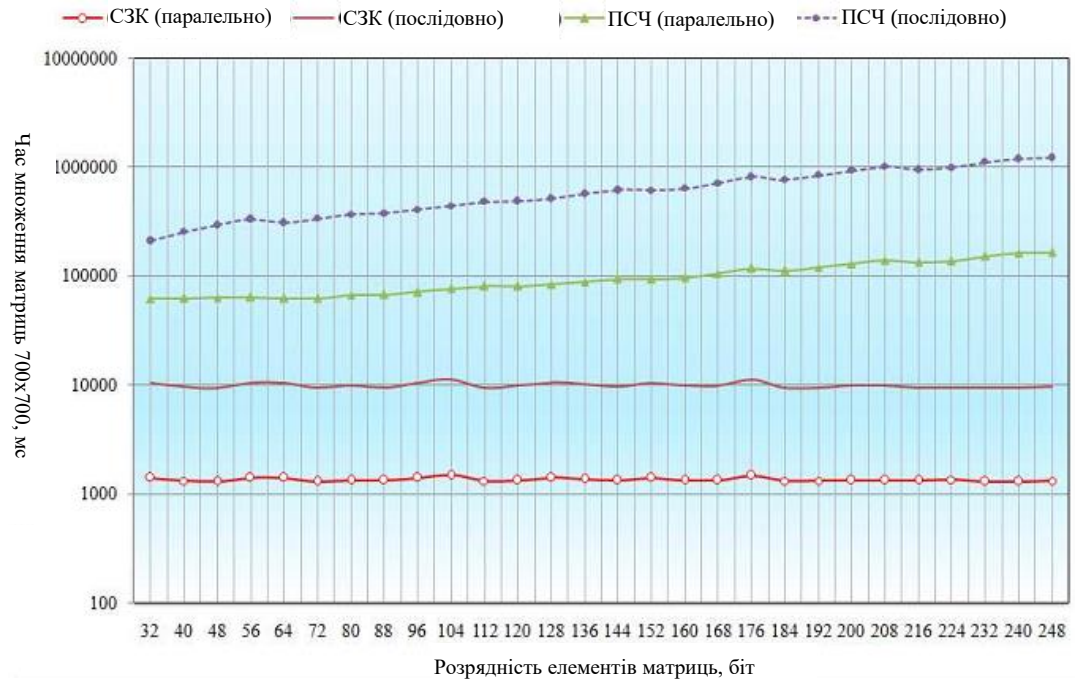


Рис. 1.11 – Залежність часу множення матриць від розрядності елементів

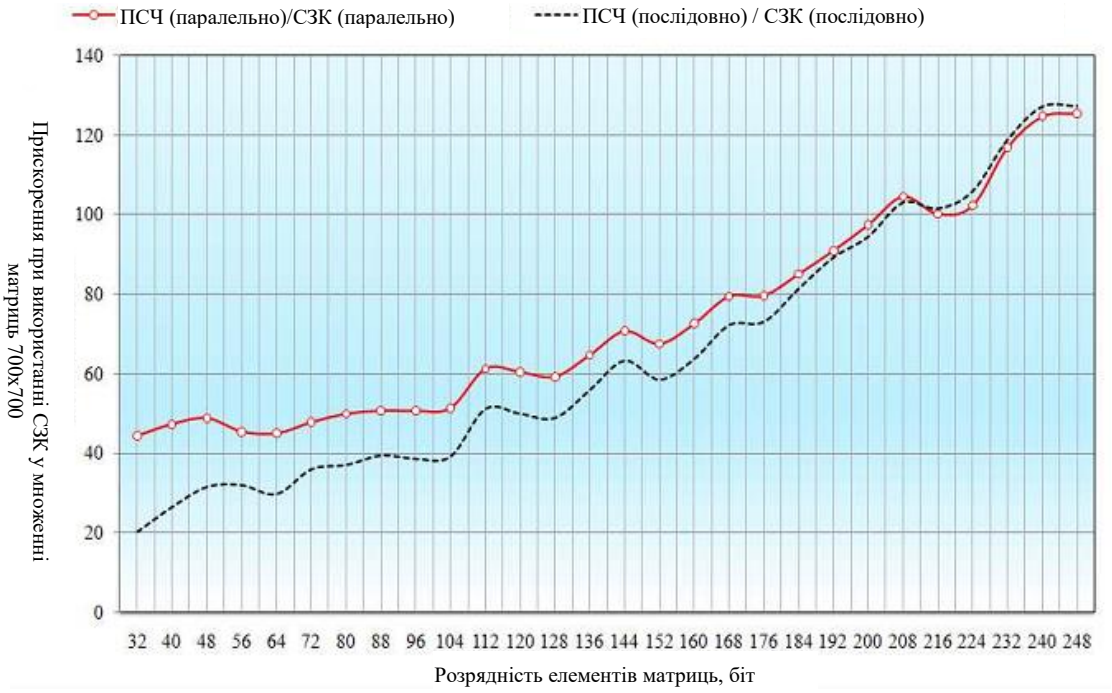


Рис. 1.12 – Залежність прискорення СЗК від розрядності елементів

Наприклад, для цифрової обробки сигналів вимагається менше модулів, ніж для криптографії. Тому в переважній більшості робіт розглядаються модулі виду 2^k, $2^k \pm 1$, що дає можливість раціонального використання регістрів розрядної сітки [214-216]. Найгірший модуль, тобто з

найбільшою складністю виконання (може бути найбільшим або модулем комплексного типу), визначає загальний параметр прямого перетворювача або арифметичного каналу.

Крім того, реалізація схем для модулів 2^k+1 набагато складніша, ніж для 2^k або 2^k-1. Затримка зворотнього перетворення для популярних модулів СЗК представлена в [217-220]. У табл. 1.6 наведено деякі набори спеціалізованих модулів і вказано їх характеристики [221].

Прямий перетворювач, а також арифметичні канали складаються з незалежних схем для кожного модуля, тому для більшої кількості модулів це призводить до простіших схем реалізації. Набір з трьох модулів типу (2^k-1, 2^k, 2^k+1), (2^k, $2^{k-1}-1$, 2^k-1) і (2^k, $2^{k+1}-1$, 2^k-1) називається збалансованим і забезпечує обмежений динамічний діапазон та паралелізм.

Варто зазначити, що існують деякі схожі набори модулів до тих, які визначені у [241]. Для розширення паралелізму можна використовувати чотири модулі типу (2^k-1, 2^k, 2^k+1, $2^{k+1}+1$), (2^k-1, 2^k, 2^k+1, $2^{k+1}-1$), (2^k-1, 2^k, 2^k+1, $2^{k-1}-1$), (2^k-3, 2^k-1, 2^k+1, 2^k+3), а також 5 модулів - (2^k-1, 2^k, 2^k+1, $2^{k-1}-1$, $2^{k+1}-1$). Ці типи модулів називають арифметичними, оскільки вони є результатом ефективних арифметичних операцій.

На відміну від прямих перетворювачів та модульних арифметичних каналів, зворотне перетворення та складні операції СЗК вимагають складних і немодулярних паралельних структур. Збалансовані набори модулів, які дуже популярні для арифметичної частини СЗК, не придатні для зворотних перетворювачів, оскільки вони призводять до складних мультиплікативних операцій і, відповідно, до зниження продуктивності зворотного перетворювача. Тому для забезпечення швидкого зворотного перетворення пропонуються такі набори (конверсійні): (2^k, $2^{2k}-1$, $2^{2k}+1$), (2^k-1, 2^k, 2^k+1, $2^{2k}+1$), (2^k-1, 2^k, 2^k+1, $2^{2k+1}-1$), (2^k-1, 2^k+1, 2^{2k}, $2^{2k}+1$) і (2^k-1, 2^k+1, 2^{2k}, $2^{2k+1}-1$). Серед них четвертий набір має найкращий зворотний перетворювач, що актуально для програм, які часто потребують зворотного перетворення.

Таблиця 1.6 - Набори модулів СЗК та їх характеристики

Літ-ра	Набір модулів	Рік	Характеристика
[222]	2^k-1, 2^k, 2^k+1	1967	Конверсійний
[223]	$2k$-1, $2k$, $2k$+1	1992	Неефективний
[224]	2^{2k}+1, 2^k+1, 2^k-1	1997	Конверсійний
[225]	2^k-1, 2^k, 2^{k-1}-1	1998	Арифметичні
[226]	2^k-1, 2^k, 2^{k+1}-1	1999	Арифметичний
[227]	2^k-1, 2^k, 2^{2k+1}-1	2008	Арифметичний
[228]	2^{2k}-1, 2^k, 2^{2k} +1	2008	Конверсійний
[229]	2^α, 2^β-1, 2^β+1	2008	Гнучкий, конверсійний
[230]	2^k-1, 2^k, 2^k+1, 2^{k+1}+1	1999	Арифметичний
[231]	2^k-1, 2^k, 2^k+1, 2^{k+1}-1	2000	Арифметичний
[214]	2^k-1, 2^k, 2^k+1, 2^{2k}+1	2003	Конверсійний
[232]	2^k-1, 2^k+1, 2^k-3, 2^k+3	2004	Збалансований
[233]	2^k-1, 2^k+1, 2^{2k}-2, 2^{2k+1}-3	2008	Великий діапазон
[215]	2^k-1, 2^k+1, 2^{2k}, 2^{2k}+1	2010	Конверсійний
[215]	2^k-1, 2^k, 2^k+1, 2^{2k+1}-1	2010	Арифметичний
[234]	2^k-1, 2^k+1, 2^{2k}, 2^{2k+1}-1	2010	Великий діапазон, арифметичний
[235]	2^α, 2^k-1, 2^k+1, 2^{k+1}+1	2014	Арифметичний
[235]	2^α, 2^k-1, 2^k+1, 2^{k-1}-1	2014	Арифметичний
[236]	2^k-1, 2^k, 2^k+1, 2^k-$2^{(k+1)/2}$+1, 2^k+$2^{(k+1)/2}$+1	2005	Конверсійний
[237]	2^k-1, 2^k, 2^k+1, 2^{k-1}-1, 2^{k+1}-1	2007	Збалансований, арифметичний
[238]	$2^{k/2}$-1, 2^k, $2^{k/2}$+1, 2^k+1, 2^{2k-1}-1	2009	Незбалансований
[239]	2^k-1, 2^k, 2^k+1, 2^k-$2^{(k+1)/2}$+1, 2^k+$2^{(k+1)/2}$+1, $2^{k\pm1}$+1	2013	Дуже великий діапазон і паралелізм
[239]	2^k-1, $2^{k+\beta}$, 2^k+1, 2^k-$2^{(k+1)/2}$+1, 2^k+$2^{(k+1)/2}$+1, $2^{k\pm1}$+1	2013	Дуже великий діапазон, гнучкий
[240]	$2^{k+\beta}$, 2^k-1, 2^k+1, 2^k-β_1, 2^k+β_1, …, 2^k-β_i, 2^k+β_i	2013	Дуже великий діапазон, збалансований

З іншого боку, третій набір із заміною 2^{2k+1}-1 на 2^{2k}+1 є найкращим для додатків, які потребують великої кількості додавань та множень.

У наборі з більшою кількістю модулів (2^k-1, $2^{k+\beta}$, 2^k+1, 2^k-$2^{(k+1)/2}$+1, 2^k+$2^{(k+1)/2}$+1 $2^{k\pm1}$+1) використовується незбалансований модуль $2^{k+\beta}$ для збільшення динамічного діапазону. Крім того, вводиться також узагальнення цього модуля, встановленого за допомогою модулів форм $2^k\pm\beta$. Цей тип модулів особливо підходить для криптографічних додатків, що вимагають великої кількості модулів з ефективними арифметичними операціями, а також продуктивним зворотним перетворювачем. Важливим моментом при цьому є розробка спеціальних арифметичних схем, що призводить до збільшення ефективності.

Отже, для кожного конкретного застосування з вказаними арифметичними операціями, апаратними компонентами та обмеженнями необхідно вибирати відповідний набір модулів.

У табл. 1.7 [242] представлено результати апаратної реалізації СЗК на інтегральній схемі спеціального призначення (ASIC) для різних модулів (Converter-1: 2^k-1, 2^k+1, 2^{2k}, 2^{2k+1}-1 та Converter-2: 2^k-1, 2^k+1, 2^{2k}, 2^{2k}+1) для k=4, 8, 12 і 16 на базі технології TSMC. Досліджувалася площа чіпа (мкм^2), його корисна площа (мкм^2), часова затримка (нс), потужність (мВт).

Видно, що зворотний перетворювач для Converter-2 має кращий параметр затримки, ніж Converter-1, який характеризується швидшим арифметичним блоком СЗК та прямим перетворювачем. Таким чином, використання певного набору модулів залежить від поставленої задачі.

Отже, відповідний набір модулів та арифметика СЗК при апаратній реалізації дозволяє суттєво зменшити споживання енергії, особливо в мультиплікаторах [243] та мультиплікаторах-накопичувачах [244-245]. Проте реалізація інших арифметичних операцій (наприклад, ділення, виявлення знаків, порівняння чисел, виявлення переповнення розрядної сітки) ускладнює обчислення в СЗК і його слід уникати.

Таблиця 1.7 – Результати дослідження апаратної реалізації СЗК

Параметри схеми	Converter-1				Converter-2			
k	4	8	12	16	4	8	12	16
Площа чіпа, мкм^2	4639	9858	13710	17901	5040	9006	13478	17793
Корисна площа, мкм^2	1575,7	3223,8	4347,7	5638	1591,9	2787,5	4098,2	5353,9
Часова затримка (нс)	0,693	1,056	1,33	1,736	0,5	0,732	0,91	1,093
Потужність (мВт)	3,615	4,528	5,485	5,761	4,446	5,42	6,382	6,997

Тому СЗК особливо корисні в алгоритмах, домінуючими операціями в яких є множення та додавання [246]. Наприклад, в [243, 247] запропонована програма обробки цифрових зображень на основі СЗК, в [245, 248] описуються фільтри з обмеженим імпульсним відгуком на основі СЗК. Огляд потенціалу та застосунків СЗК представлений в [249-250]. Загальна схема апаратної реалізації СЗК показана на рис. 1.13.

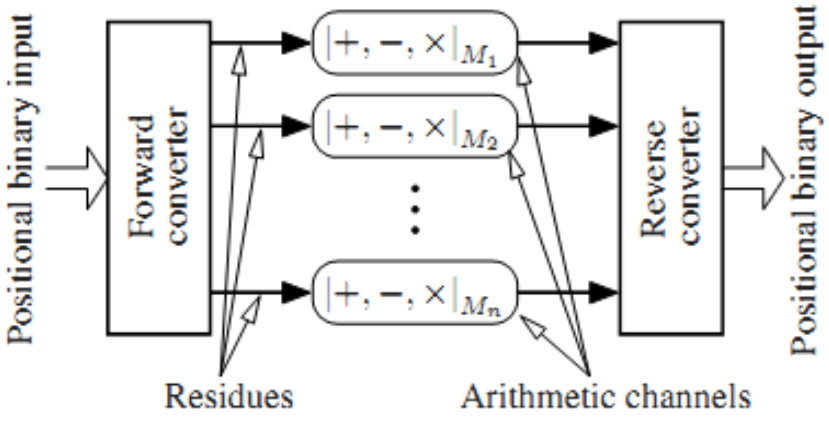


Рис. 1.13 - Загальна схема апаратної реалізації СЗК

Схема складається з трьох основних частин: прямого перетворювача, арифметичних каналів СЗК та зворотного перетворювач. Перехідний перетворювач переводить вхідні двійкові числа в набір значень залишків. Додавання, віднімання та множення виконуються за допомогою незалежних схем, що виконують обчислення за відповідним модулем. Кінцевою частиною схеми є зворотний перетворювач, який обчислює вихідне двійкове число із залишків, що отримуються в кожному арифметичному каналі. Перетворювачі вимагають відповідних витрат на апаратне забезпечення, тому використання СЗК виправдане лише тоді, якщо економія арифметичних каналів перевищує вартість конвертації.

Однією з основних проблем в СЗК є вибір індивідуальної системної бази, щоб отримати відповідний динамічний діапазон, швидкість та складність схеми. Для необхідного динамічного діапазону необхідно знайти компроміс між швидкістю арифметичних операцій та складністю перетворення. Малі модулі дозволяють створювати невеликі та швидкі арифметичні одиниці, але кількість модулів в базі СЗК і, отже, складність перетворювачів зростають. З іншого боку, арифметичні операції для великих модулів можуть бути досить повільними. Поєднання малих модулів, великого динамічного діапазону та простих перетворювачів можливе в ієрархічних СЗК, у яких значення усіх або лише деяких залишків представлено в СЗК з динамічними знаками, меншими за діапазон головної системи [111, 251, 252]. Система, яка використовується для відображення значень залишків, називається нижчим рівнем СЗК, тоді як система, числа якої представлені на наступному рівні, називається СЗК більш високого рівня. Таким чином отримується багаторівневі ієрархічна СЗК. Найвища система в ієрархії називається верхнім рівнем СЗК. Динамічні діапазони СЗК нижчого рівня можна вибрати двома способами. У першому підході [111, 251] динамічні діапазони нижнього рівня досить великі, щоб відобразити проміжні результати. Наприклад, для множення діапазон нижнього рівня СЗК повинен дорівнювати квадрату модуля вищого рівня. Перевага цього

рішення полягає в тому, що ті ж модулі можуть бути використані для відображення різних значень залишків. Наприклад, для множення при найвищому рівні СЗК (17, 19, 20, 21) максимальні значення залишків будуть 16, 18, 19 і 20. Тоді динамічні діапазони для нижнього рівня повинні бути як мінімум $16^2+1=257$, $18^2+1=325$, $19^2+1=362$ і $20^2+1=401$. Таким чином, СЗК (3, 4, 5, 7) з діапазоном 420 можна використовувати для всіх чотирьох чисел.

Отже, можна побудувати ієрархічну СЗК з діапазоном $17\cdot19\cdot20\cdot21>217$ з 3-бітними модулями. На жаль, головним недоліком цього рішення є швидке зростання динамічних діапазонів СЗК нижчого рівня. Крім того, перетворювачі між рівнями повинні використовуватися після невеликої кількості арифметичних операцій.

У другому підході [252] база СЗК верхнього рівня вибирається з чисел, що розкладаються на малі множники. Нижчий рівень СЗК будується з коефіцієнтів для відповідних модулів. У цьому методі діапазон нижчого рівня дорівнює модулям від базового вищого рівня. Таким чином, виконання обчислень у нижчому СЗК ідентичні з їх формуванням по модулю вищих рівнів. Перетворення між послідовними рівнями може бути здійснене один раз для всіх арифметичних операцій, що призводить до низьких витрат на апаратне забезпечення. Проте основним недоліком цієї ідеї є труднощі з виявленням базової СЗК найвищого рівня. У роботі [252] ця база вибирається з модулів 2^{2k}-1, а нижній рівень – з 2^k-1, 2^k +1). Це дозволяє реалізовувати перетворювачі та арифметичні одиниці як прості структури. Однак деякі залишки можуть бути близькими до динамічного діапазону системи, тому переваги втрачаються.

У роботі [161] запропоновано новий метод побудови бази ієрархічної СЗК. База включає в себе модулі $2^k\pm1$, які розкладаються на малі множники. Цей підхід дозволяє побудувати перетворювачі вводу/виводу як дворівневі схеми, представлені на рис. 1.14. У верхньому рівні здійснюються перетворення між великими числами (близькими до системного діапазону) та залишками за модулем $2^k\pm1$.

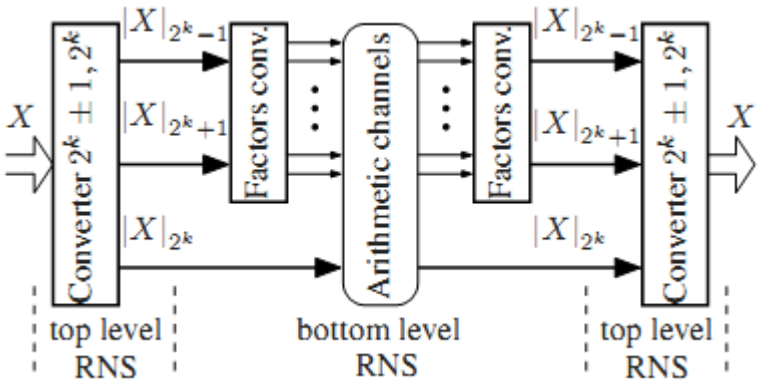


Рис. 1.14 - Структура апаратної реалізації ієрархічної СЗК

У нижньому рівні проводиться перетворення між залишками за модулями $2^k \pm 1$ та модулями, що є множниками $2^k \pm 1$. Завдяки ефективному виконанню операцій за модулем $2^k \pm 1$ для великих чисел область затримки критичного шляху запропонованих дворівневих перетворювачів невелика. Крім того, арифметичні операції виконуються над невеликими числами, і, отже, суматори та перемножувачі можуть бути реалізовані відносно простими та швидкими схемами.

В зв'язку із збільшенням об'ємів обчислень в останні роки почав проявлятися інтерес дослідників до застосування СЗК в сучасній асиметричній криптографії [253-256]. Зокрема, в [257-260] представлено безпечні та ефективні підходи для застосування СЗК в криптографії на еліптичних кривих. Вони є особливо ефективні як протидія від атак по побічному каналі витоку та під час введення несправностей в роботу обчислювальної системи. У [261-266] 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 серії робіт [267-272] 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.

Іншим напрямком підвищення швидкодії асиметричних криптосистем є використання нових підходів до виконання арифметичних операцій модулярного множення та експоненціювання. В [273] запропонована організація паралельного виконання модулярного експоненціювання і встановлено, що двопотоковий паралелізм при цьому є найбільш доцільною формою. Для практичної реалізації такої можливості запропонована організація прискореного обчислення модулярної експоненти на двоядерному процесорі. Теоретично і експериментально доведено, що розроблена організація забезпечує практично дворазове прискорення виконання операції модулярного експоненціювання для розрядностей 2048 та 4096. Значніше прискорення виконання модулярного експоненціювання може бути досягнуто при переході на рівень операцій процесорного множення [274] та хмарних технологій [275-276].

В [277-279] показано, що застосування СЗК в методі Монтгомері є ефективним способом збільшення швидкодії модулярного множення, однак його часова складність залишається високою при опрацюванні багаторозрядних чисел.

Але в усіх розглянутих підходах виникає необхідність обчислення оберненого елемента, що при використанні цілочисельної форми істотно зменшує швидкість відновлення десяткового числа із СЗК. 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 працях [280-287] розроблено методи побудови ДФ СЗК, у [288-293] - МДФ СЗК, які дозволяють уникнути пошуку оберненого елемента за модулем та множення на нього. В табл. 1.8 наведено основні характеристики цілочисельної форми, ДФ та МДФ СЗК [294].

Таблиця 1.8 – Характеристики форм СЗК

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

Незважаючи на існуючі ефективні алгоритми опрацювання інформаційних потоків у СЗК, що реалізуються на основі програмно-апаратних спецпроцесорів та забезпечують захист від помилок і певний рівень захисту інформації від несанкціонованого доступу, потрібно їх додаткове дослідження, аналіз і ефективне застосування нових методів та алгоритмів при застосуванні різних форм СЗК в асиметричних криптосистемах.

Висновки до першого розділу

1. Проведено аналіз теоретичних основ алгебри і теорії чисел, визначено основні властивості конгруенцій для їх застосування в асиметричних криптосистемах. Встановлено, що існуючі алгоритми пошуку мультиплікативного оберненого елемента та реалізації КТЗ характеризується значною обчислювальною складністю в зв’язку з неможливістю розпаралелення процесу обчислень та виконанням операцій над багаторозрядними числами, що може призвести до переповнення розрядної сітки сучасних потужних обчислювальних засобів

2. Проаналізовано найбільш поширені асиметричні криптосистеми (RSA, Рабіна, Ель-Гамаля), основними операціями у яких є модулярне

множення, модулярне експоненціювання та пошук оберненого елемента за модулем. Вони характеризуються значною обчислювальною складністю і при розрядностях ключа порядку 1024 біт проявляються недоліки двійкової арифметики (велика розрядність, наявність міжрозрядних переносів, строга послідовність при виконанні обчислень), що сповільнює виконання арифметичних операцій.

3. Здійснена систематизація найбільш відомих методів факторизації, на складності якої ґрунтується криптоаналіз переважної більшості криптографічних алгоритмів з відкритим ключем. Відомі на даний час експоненціальні та субекспоненціальні методи досить трудомісткі, тому вимагають значних обчислювальних ресурсів для багаторозрядних чисел.

4. Проведено аналіз теоретичних основ СЗК, на основі якого визначено її основні переваги над позиційними системами числення (можливість розпаралелення процесу обчислень, відсутність міжрозрядних переносів) та недоліки (труднощі при виконанні ділення та порівняння чисел, визначенні переповнення допустимого діапазону), обґрунтовано необхідність використання СЗК в асиметричних криптосистемах.

5. Проаналізовано сучасний стан та напрями підвищення ефективності використання спеціалізованих наборів модулів при виконанні операцій модулярного множення та модулярного експоненціювання для їх застосування в асиметричних криптосистемах. Відмічено, що багатьох задачах використання СЗК разом з паралельними обчисленнями дозволяє значно збільшити продуктивність обчислювальних систем.

Основні результати першого розділу відображені у роботах [37-43, 256, 264-266, 280-294].

2 МЕТОДИ ВИКОНАННЯ МОДУЛЯРНИХ АРИФМЕТИЧНИХ ОПЕРАЦІЙ

У другому розділі представлено схему адміністративного алгоритму застосування розроблених методів в асиметричних криптосистемах, а також проведені теоретичні дослідження виконання модулярних операцій над багаторозрядними числами.

2.1 Структурна схема організації взаємодії розроблених методів для асиметричних криптосистем

Напрямками підвищення швидкості опрацювання багаторозрядних чисел в асиметричних криптосистемах є зменшення обчислювальної складності при виконанні арифметичних операцій на основі векторно-модульного методу та використання ДФ і МДФ СЗК. На рис. 2.1 представлено схему адміністративного алгоритму, в якому показано застосування розроблених методів в асиметричних криптосистемах RSA, Рабіна та Ель-Гамаля.

Адміністративний алгоритм асиметричних криптосистем виконує адміністративні функції координації та управління для використання модулів 2 (Алгоритми виконання модулярних операцій) та 5 (Форми СЗК) для асиметричних криптосистем, відповідно модулів 1 (Криптосистема Ель-Гамаля), 3 (Криптосистема RSA) та 4 (Криптосистема Рабіна).

Модуль 2 включає в себе підмодулі 2.1-2.7 (Алгоритм Евкліда, матрично- та векторно модульні методи модулярного множення та модулярного експоненціювання, Пошук оберненого елемента, Обчислення мультистепеневої функції, КТЗ та удосконалення методу факторизації Ферма).

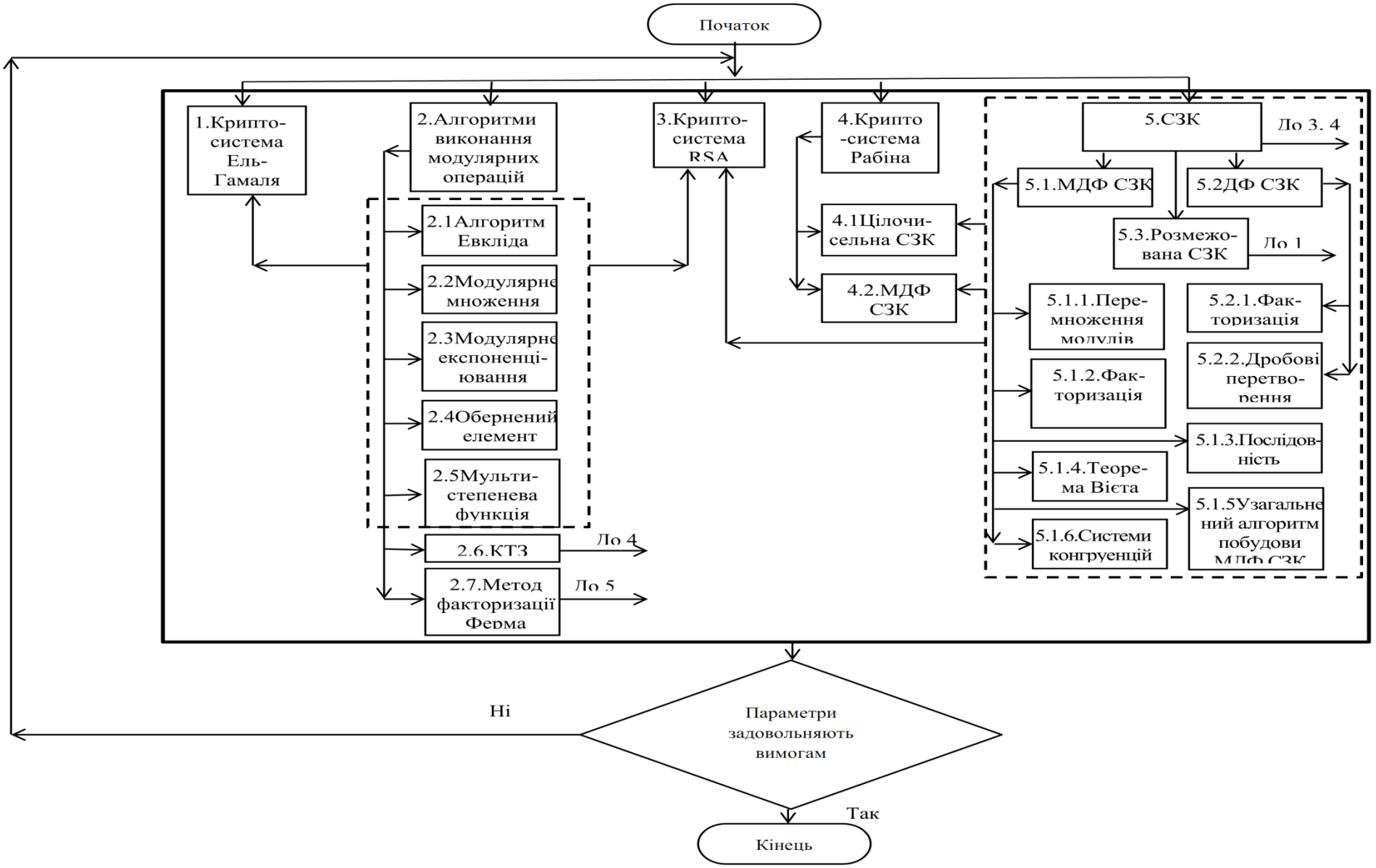


Рис. 2.1 – Схема адміністративного алгоритму використання розроблених методів в асиметричних криптосистемах

Модуль 5 включає підмодулі 5.1-5.3 (підбір модулів МДФ СЗК на основі перемноження модулів, факторизації, послідовності Фібоначчі, теореми Вієта, системи конгруенцій, узагальненого алгоритму побудови МДФ СЗК, підбір модулів ДФ СЗК на основі дробових перетворень та факторизації, а також використання розмежованої СЗК).

Усі ці методи знаходять своє застосування в асиметричних криптосистемах RSA, Рабіна та Ель-Гамаля, на що вказують відповідні напрями стрілок.

2.2 Метод пошуку залишку в розмежованій системі числення

Як було сказано вище, у всіх асиметричних алгоритмах шифрування інформаційних потоків найважливішими операціями є модулярне множення та модулярне експоненціювання багаторозрядних чисел.

Для пошуку залишку $a \bmod p$ на основі розмежованої [295] двійкової системи числення число n_0 – розрядне число a необхідно записати в двійковій системі числення $a=a_{n_0-1}2^{n_0-1}+ \dots +a_i2^i+ \dots +a_1 2+a_0$,. Тоді [296, 297]:

$$a \bmod p = \left(\sum_{i=0}^{n_0-1} \left(a_i 2^i \bmod p \right) \right) \bmod p = \left(\sum_{i=0}^{n_0-1} \left(a_i a_{1i} \right) \right) \bmod p . \qquad (2.1)$$

де a_i=0 або 1, $a_{1i}=2^i \bmod p$.

З (2.1) випливає, що шуканий залишок дорівнюватиме сумі тих степенів двійки (або a_{1i}), для яких відповідно a_i=1, що проілюстровано в табл. 2.1.

Слід зазначити також, що два послідовні значення a_{1i} та $a_{1\ i+1}$ пов’язані таким рекурентним співвідношенням:

Таблиця 2.1 - Таблиця знаходження залишку a mod p.

i	n_0-1	n_0-2		2	1	0
a_i	a_{n_0-1}	a_{n_0-2}	…	a_2	a_1	a_0
$2^i \bmod p$	$2^{n_0-1} \bmod p$	$2^{n_0-2} \bmod p$		$2^2 \bmod p$	$2^1 \bmod p$	$2^0 \bmod p$
a_{1i}	$a_{1\,n_0-1}$	$a_{1\,n_0-2}$	…	a_{12}	a_{11}	a_{10}

$$a_{1i+1} = \begin{cases} 2 \cdot a_{1\ i},\ 2 \cdot a_{1\ i} < p \\ 2 \cdot a_{1\ i} - p,\ 2 \cdot a_{1\ i} \geq p. \end{cases} \qquad (2.2)$$

Отже, для знаходження залишку за модулем в (2.2) не обов'язково виконувати обчислювально витратну операцію ділення з остачею, а можна обмежитися тільки відніманням. Крім того, множення на 2 дуже просто реалізується за допомогою дописування нуля в кінці двійкового запису числа.

В табл. 2.2 наведено приклад пошуку 171 mod 31.

Таблиця 2.2 – Пошук залишку 171 mod 31

i	7	6	5	4	3	2	1	0
a_i	1	0	1	0	1	0	1	1
$2^i \bmod 31$	$2^7 \bmod 31$	$2^6 \bmod 31$	$2^5 \bmod 31$	$2^4 \bmod 31$	$2^3 \bmod 31$	$2^2 \bmod 31$	$2^1 \bmod 31$	$2^0 \bmod 31$
a_{1i}	4	2	1	16	8	4	2	1

Отже, 171 mod 31=(4+1+8+2+1) mod 31=16. Слід зазначити, що результат отриманий на основі додавання мало розрядних залишків степенів двійки за відповідним модулем.

На рис. 2.2 представлена блок-схема методу пошуку залишку в розмежованій системі числення.

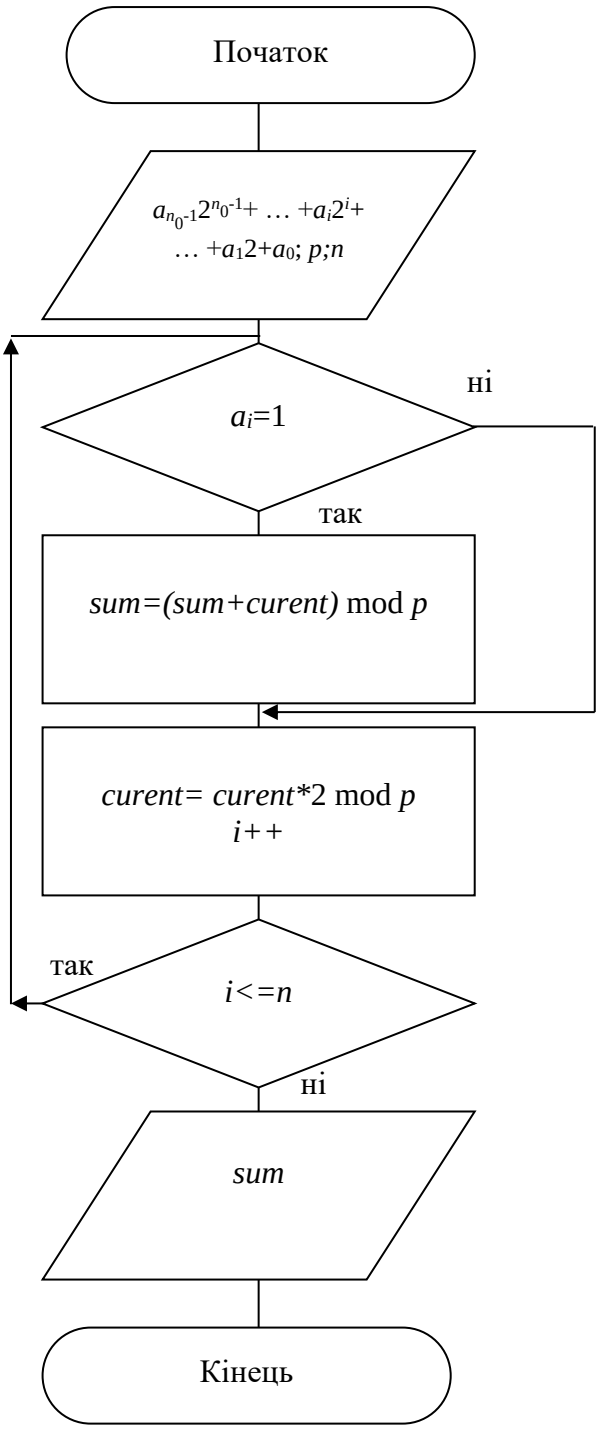


Рис. 2.2 - Блок-схема методу пошуку залишку в розмежованій системі числення

2.3 Методи модулярного множення

Нехай маємо два числа розрядністю n_0: $a=a_{n_0-1}2^{n_0-1}+\dots+a_i2^i+\dots+ a_12+a_0$ та $b=b_{n_0-1}2^{n_0-1}+\dots+b_i2^j+\dots+b_12+b_0$, де a_i, b_j=0 або 1. Для знаходження

результату їх множення за модулем p потрібно побудувати матрицю, представлену в табл. 2.3, де $c_{ij}=2^{i+j} \bmod p$ [37, 40-43, 295,298]. Тоді добуток чисел a та b за модулем p отримується згідно такої формули:

$$a \cdot b \bmod p = \left(\sum_{i=1}^{n_0-1} \left(\sum_{j=1}^{n_0-1} a_i b_j \right) c_{ij} \right) \bmod p = \left(\sum_{i=1}^{n_0-1} \left(\sum_{j=1}^{n_0-1} a_i b_j \right) 2^{i+j} \bmod p \right) \bmod p, \quad (2.3)$$

Це означає, що шуканий результат отримується у вигляді суми за модулем p тих c_{ij}, для яких відповідні a_i та b_j дорівнюють 1.

Таблиця 2.3 – Матриця для модулярного множення

	b_{n_0-1}	…	b_j	…	b_1	b_0
a_{n_0-1}	$c_{n_0-1\ n_0-1}$	…	$c_{n_0-1\ j}$	…	$c_{n_0-1\ 1}$	$c_{n_0-1\ 0}$
…	…	…	…	…	…	…
a_i	$c_{i\ n_0-1}$	…	c_{ij}	…	c_{i1}	c_{i0}
…	…	…	…	…	…	…
a_1	$c_{1\ n_0-1}$	…	c_{1j}	…	c_{11}	c_{10}
a_0	$c_{0\ n_0-1}$	…	c_{0j}	…	c_{01}	c_{00}

Слід зазначити, що $c_{ij} = c_{ji}$ і аналогічно до попереднього випадку кожне наступне значення c_{ij} визначається за допомогою рекурентних співвідношень:

$$c_{i\ j+1} = \begin{cases} 2 \cdot c_{ij}, & 2 \cdot c_{ij} < p \\ 2 \cdot c_{ij} - p, & 2 \cdot c_{ij} \geq p \end{cases}; \quad c_{i+1\ j} = \begin{cases} 2 \cdot c_{ij}, & 2 \cdot c_{ij} < p \\ 2 \cdot c_{ij} - p, & 2 \cdot c_{ij} \geq p. \end{cases} \quad (2.4)$$

На рис. 2.3 представлена блок-схема даного алгоритму, а в табл. 2.4 наведено приклад використання матричного методу для знаходження добутку 25·21 mod 29. Враховуючи, що $25_{10}=11001_2$ та $21_{10}=10101_2$ і

побудувавши відповідну матрицю, можна побачити, що 25·21 mod 29=(24+12+16+6+3+4+16+8+1) mod 29= 90 mod 29=3.

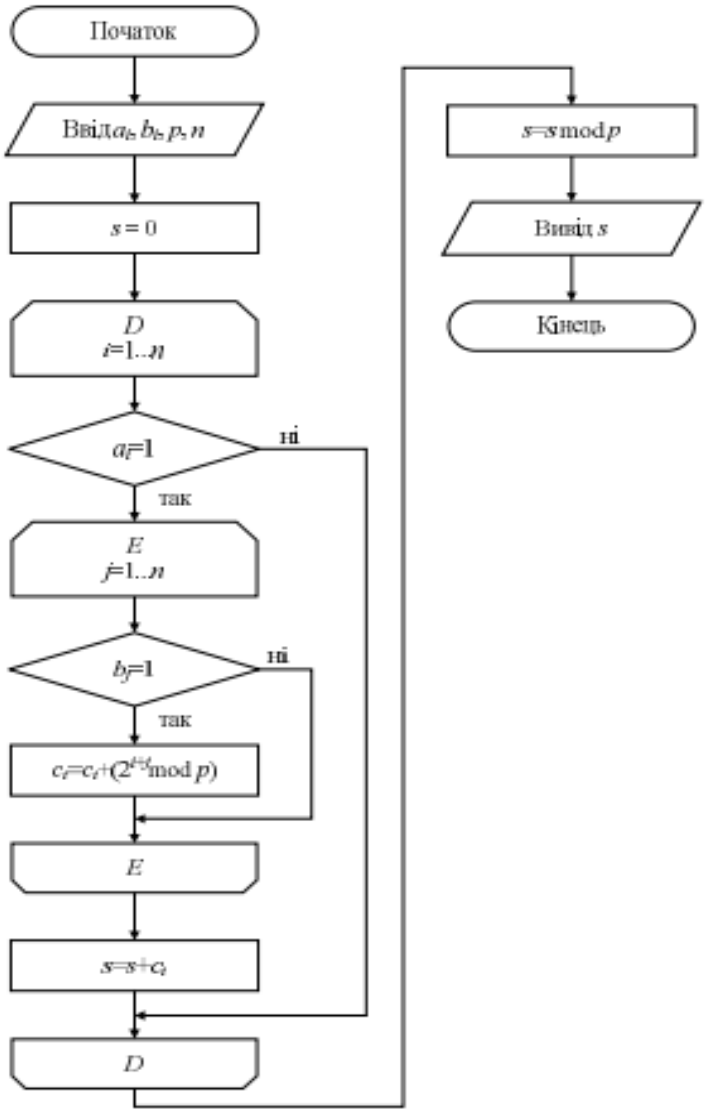


Рис. 2.3 - Блок схема матричного методу для модулярного множення

Даний метод дозволяє замінити операцію множення, яка має квадратичну обчислювальну складність $O1(n_0) = n_0^2$, матрично-модульною операцією сумування, яка характеризується лінійно-логарифмічною складністю $O2(n_0) = n_0 \log_2 n_0$.

Для удосконалення даного методу можна побудувати матрицю, представлену в табл. 2.5, де a_{1j}=2^j mod p.

Таблиця 2.4 – Матриця для модулярного множення 25·21 mod 29

25 / 21	1	1	0	0	1
1	24	12	6	3	16
0	12	6	3	16	8
1	6	3	16	8	4
0	3	16	8	4	2
1	16	8	4	2	1

Таблиця 2.5 – Матриця для удосконаленого матричного методу модулярного множення

	$a_{1\,2n_0-1}$	…	$a_{1\,n_0+1}$	$a_{1\,n_0}$	…	$a_{1\,i}$	…	a_{11}	a_{10}
b_{n_0-1}	a_{n_0}	…	a_1	a_0	…	…	…	0	0
…	…	…	…	…	…	…	…	0	0
b_j	…	…	…	…	…	…	…	0	0
…	0	…	…	…	…	…	…	0	0
b_1	0	…	a_{n_0}	a_{n_0-1}	…	a_{i-1}	…	a_0	0
b_0	0	…	0	a_{n_0}	…	a_i	…	a_1	a_0

Множення відбувається таким чином:

$$a\cdot b \bmod p=(a_0b_0a_{10}+(a_0b_1+ a_1b_0)a_{11}+(a_0b_2+ a_1b_1+a_2b_0)a_{12}+\ldots)\bmod p. \qquad (2.5)$$

Введемо позначення:

$$\begin{aligned} A_l &= a_{1l}\sum_{i=0}^{l} a_i b_{l-i}, (l=0,1,\ldots,n_0-1); \\ A_z &= a_{1z}\sum_{i=z}^{2n_0-2} a_{n_0-1-i+z}b_{i-n_0+1}, (z=n_0,n_0+1,\ldots,2n_0-2). \end{aligned} \qquad (2.6)$$

Тоді

$$a \cdot b \bmod p = \left(\sum_{l=0}^{n_0-1} A_l + \sum_{z=n_0}^{2n_0-2} A_z \right) \bmod p. \qquad (2.7)$$

З врахуванням (2.6) співвідношення (2.7) набуде такого вигляду:

$$a \cdot b \bmod p = \left(\sum_{l=0}^{n_0-1} \left(a_{1l} \sum_{i=0}^{l} a_i b_{l-i} \right) + \sum_{z=n_0}^{2n_0-2} \left(a_{1z} \sum_{i=z}^{2n_0-2} a_{n_0-1-i+z} b_{i-n_0+1} \right) \right) \bmod p. \qquad (2.8)$$

Причому, якщо b_j=1 та деякі a_s і a_q=1, то можна спростити обчислення за допомогою наступного співвідношення:

$$(a_s+a_q)a_{1i}= a_{1\,i+1}. \qquad (2.9)$$

В табл. 2.6 наведено приклад вдосконаленого матричного методу для пошуку значення виразу 21·25 mod 29. Знову числа 21 і 25 представляються в двійковій системі числення і на основі табл. 2.5 будується відповідна матриця.

Таблиця 2.6 – Приклад множення 21·25 mod 29

	19	24	12	6	3	16	8	4	2	1
1	0	1	0	1	0	1	0	0	0	0
1	0	0	1	0	1	0	1	0	0	0
0	0	0	0	1	0	1	0	1	0	0
0	0	0	0	0	1	0	1	0	1	0
1	0	0	0	0	0	1	0	1	0	1

З врахуванням співвідношень (2.8) та (2.9) можна одержати (24+12+6+3+16+16+8+4+1)mod29=(19+8+4+1) mod29=3.

Таким чином, отримано удосконалений новий метод заміни операції множення, яка має квадратичну обчислювальну складність $O1(n_0) = n_0^2$, матричномодульною операцією сумування з лінійно-логарифмічною складністю $O3(n_0) = \frac{1}{2} n_0 \log_2 n_0$. Графічні результати дослідження обчислювальних складностей наведені на рис. 2.4.

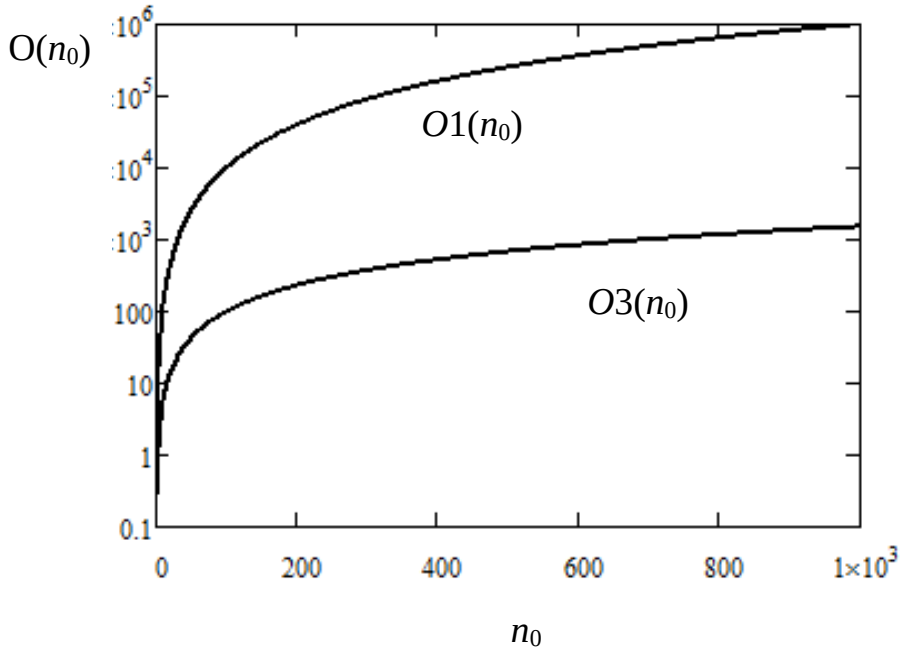


Рис. 2.4 – Порівняння обчислювальних складностей відомого та вдосконаленого методів

Отже, при використанні виразу (2.9) обчислювальна складність виконання операції модулярного множення суттєво зменшується, що дозволяє ефективно використовувати запропонований метод в алгоритмах захисту інформаційних потоків, побудованих на асиметричних криптосистемах.

Для зменшення об’єму пам’яті, в якій повинні зберігатися проміжні результати усіх матричних обчислень, можна використати векторно-

модульний метод модулярного множення двох чисел $a = \sum_{i=0}^{n_0-1} a_i \cdot 2^i$ та $b = \sum_{j=0}^{n_0-1} b_j \cdot 2^j$ [40-44]. В цьому випадку будується два вектор-рядки (c_i та a_i), перший з яких містить елементи $c_i = 2 \cdot c_{i-1} \bmod p$, $c_0 = 2^0 \cdot b \bmod p$, другий - a_i (табл. 2.7).

Таблиця 2.7– Матриця вектор-рядків для модульного множення

i	n_0-1	…	2	1	0
c_i	c_{n_0-1}	…	c_2	c_1	c_0
a_i	$a_{n_0\text{-}1}$	…	a_2	a_1	a_0

Слід зазначити, що кожне наступне значення c_i обчислюється за рекурентною формулою, аналогічною (2.2):

$$c_{i+1} = \begin{cases} 2 \cdot c_i, & 2 \cdot c_i < p \\ 2 \cdot c_i - p, & 2 \cdot c_i \geq p. \end{cases} \tag{2.10}$$

Результат модулярного множення для двох чисел отримується згідно такої формули:

$$a \cdot b \bmod p = \left(\sum_{i=0}^{n_0-1} a_i \cdot c_i \right) \bmod p, \tag{2.11}$$

тобто відбувається сумування тих c_i, для яких відповідні a_i дорівнюють 1.

На рис. 2.5 наведена блок-схема реалізації векторно-модульного методу модулярного множення.

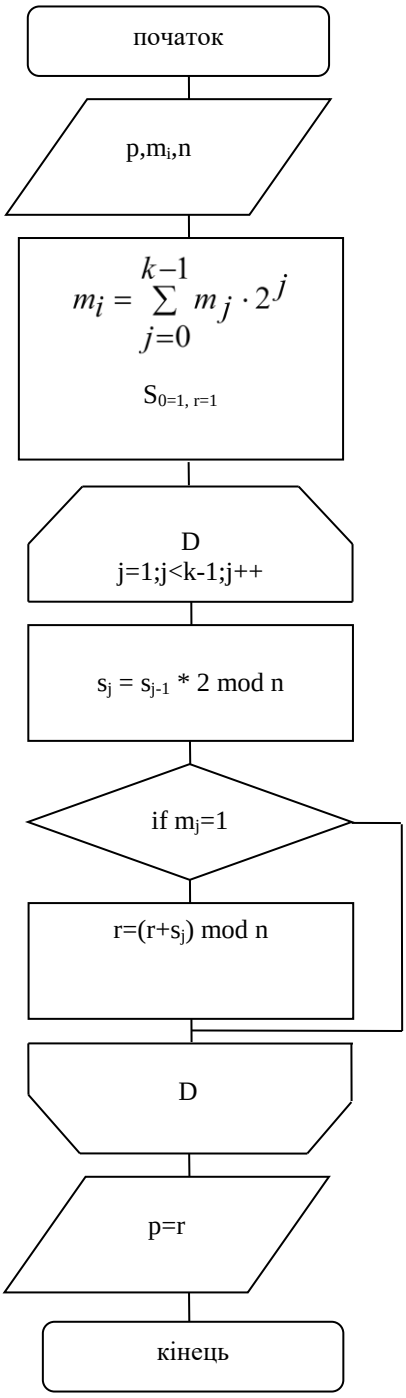


Рис. 2.5 - Блок-схема векторно-модульного методу модулярного множення

В табл. 2.8 наведено приклад векторно-модульного методу модулярного множення для пошуку значення виразу 21·25 mod 29. Число 21 представляється в двійковій системі числення: $21_{10}=10101_2$ і на основі табл. 2.7 будується відповідна матриця.

Отже, 21·25 mod 29=(23+13+25) mod 29=3.

Таблиця 2.8 - Приклад множення 21·25 mod 29 векторно-модульним методом

i	4	3	2	1	0
a_i	1	0	1	0	1
c_i	2·26mod29=23	2·13mod29=26	2·21mod29=13	2·25mod29=21	2^0·25mod29=25

Розроблений метод характеризується меншою часовою і апаратною складністями порівняно із матрично-модульним за рахунок зменшення кількості операцій додавання з $2\cdot\log_2 n_0$ до $\log_2 n_0$, тобто в два рази. При вирішенні задач криптографії збільшення швидкодії в два рази є суттєвим і значно розширює функціональні можливості апаратного забезпечення, а також спрощує реалізацію відповідних спецпроцесорів.

2.4 Методи модулярного експоненціювання

Для модулярного експоненіювання $a^x \bmod p$ (вважається, що $x \le \varphi(p)$, $\varphi(p)$ – значеня функції Ейлера від модуля p) потрібно використати проміжну матрицю, представлену в табл. 2.9 [38, 39, 43, 295, 298-301]. Її розмірність дорівнює розрядності n_0 модуля p. В стовбцях матриці записані величини $A_i = a^{2^i} \bmod p$ у двійковій системі числення, тобто a_{ij}=0, 1. Тоді будь–який степінь числа a записується за степенями двійки і шуканий результат можна отримати, перемноживши значення у стовбцях, для яких відповідні x_i у розкладі $x = \sum_{i=0}^{n_0-1} x_i \cdot 2^i$ дорівнюють одиниці, за допомогою такого виразу:

$$a^x \bmod p = \left(\prod_{i=0}^{n_0-1} a^{x_i 2^i} \right) \bmod p = \prod_{i=0}^{n_0-1} \left(\left(\sum_{j=0}^{n_0-1} a_{ij} 2^j \right)^{x_i} \right) \bmod p, \qquad (2.12)$$

де a_{ij} – біти двійкового запису числа $a^{2i} \bmod p = \sum_{j=0}^{n_0-1} a_{ij} 2^j$.

Таблиця 2.9–Матриця піднесення до степеня

$x_{n_0\text{-}1}$	…	x_i	…	x_1	x_0
$a_{n_0\text{-}1\ n_0\text{-}1}$	…	$a_{i\ n_0\text{-}1}$	…	$a_{1\ n_0\text{-}1}$	$a_{0\ n_0\text{-}1}$
…	…	…	…	…	…
$a_{n_0\text{-}1\ j}$	…	$a_{i\ j}$	…	$a_{1\ j}$	$a_{0\ j}$
…	…	…	…	…	…
$a_{n_0\text{-}1\ 1}$	…	$a_{i\ 1}$	…	$a_{1\ 1}$	$a_{0\ 1}$
$a_{n_0\text{-}1\ 0}$	…	$a_{i\ 0}$	…	$a_{1\ 0}$	$a_{0\ 0}$
$a^{2^{n_0-1}} \bmod p$	…	$a^{2^i} \bmod p$	…	$a^{2^1} \bmod p$	$a^{2^0} \bmod p$

Перевагами такого методу є здійснення операцій над залишками, а не над багаторозрядними числами, що дозволяє збільшити швидкодію алгоритму модулярного експоненціювання. Слід зазначити, що для заповнення матриці доцільно скористатися рекурентним співвідношенням:

$$a^{2^{i+1}} \bmod p = \left(a^{2^i} \bmod p \right)^2 \bmod p . \qquad (2.13)$$

В табл. 2.10 наведено приклад пошуку значення $23^{19} \bmod 29$ на основі матричного методу модулярного експоненціювання.

Отже, $23^{19} \bmod 29 = (7 \cdot 7 \cdot 23) \bmod 29 = 25$. Операцію множення можна виконувати методами, описаними в п. 2.2. Для зменшення матриці (табл. 2.10) доцільно використати векторно-модульний метод модулярного експоненціювання, у якому не визначаються біти двійкового запису чисел A_i, яке записується в десятковому вигляді (табл. 2.11). На рис. 2.6 наведена блок-схема векторно-модульного методу модулярного експоненціювання.

Таблиця 2.10 - Приклад пошуку значення $23^{19} \bmod 29$ на основі матричного методу модулярного експоненціювання

1	0	0	1	1
19				
0	1	1	0	1
0	0	0	0	0
1	1	1	1	1
1	1	0	1	1
1	1	0	1	1
7	23	20	7	23
$23^{2^4} \bmod 29$	$23^{2^3} \bmod 29$	$23^{2^2} \bmod 29$	$23^{2^1} \bmod 29$	$23^{2^0} \bmod 29$

Таблиця 2.11 – Матриця векторно-модульного методу модулярного експоненціювання

i	n_0-1	…	2	1	0
x_i	x_{n_0-1}	…	x_2	x_1	x_0
$a^{2^i} \bmod p$	$a^{2^{n_0-1}} \bmod p$	…	$a^{2^2} \bmod p$	$a^{2^1} \bmod p$	$a^{2^0} \bmod p$
A_i	A_{n_0-1}	…	A_2	A_1	A_0

Обчислення відбуваються за такою формулою:

$$a^x \bmod p = \left(\left(\prod_{i=0}^{n_0-1} a^{x_i 2^i} \right) \bmod p \right) \bmod p = \left(\prod_{i=0}^{n_0-1} \left(A_i^{x_i} \right) \bmod p \right) \bmod p. \quad (2.14)$$

В табл. 2.12 наведено приклад використання даного методу для пошуку $23^{19} \bmod 29$.

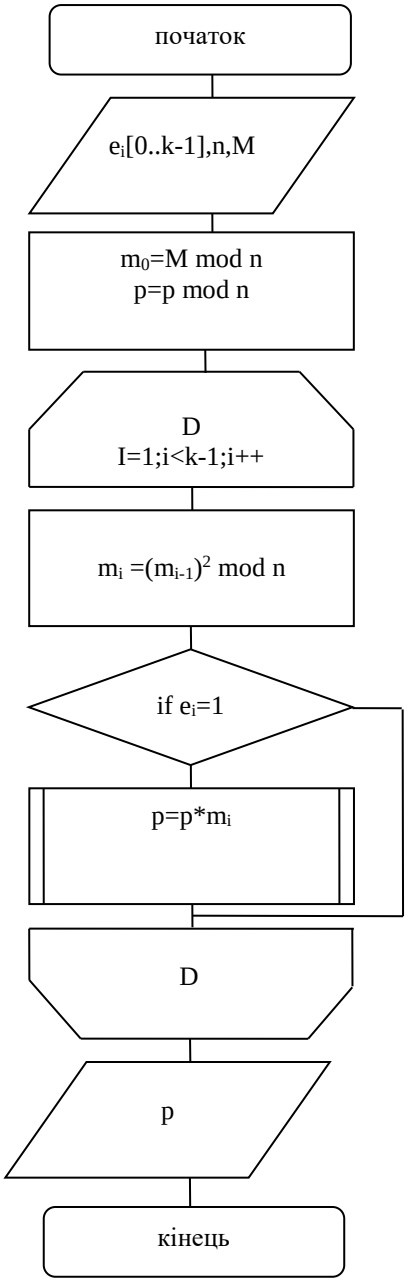


Рис. 2.6 - Блок-схема векторно-модульного методу модулярного експоненціювання

Таблиця 2.12 –Пошук $23^{19} \bmod 29$ векторно-модульним методом

i	4	3	2	1	0
x_i	1	0	0	1	1
$a^{2^i} \bmod p$	$a^{2^4} \bmod 29$	$23^{2^3} \bmod 29$	$23^{2^2} \bmod 29$	$23^{2^1} \bmod 29$	$23^{2^0} \bmod 29$
A_i	7	23	20	7	23

Звідси слідує, що 23^{19} mod 29= (7·7·23) mod 29=25.

Розрахунки показують, що запропонований алгоритм для піднесення до степеня числа за модулем *p* дає можливість зменшити складність з $O\left(n_0^3\right)$, або $O1(n_0^2 \log_2 n_0)$ (Монтгомері метод) до $O2(\frac{n_0^2 \log_2 n_0}{4})$, тобто ефективність зростає в 4 рази. Графічні залежності складностей представлені на рис. 2.7.

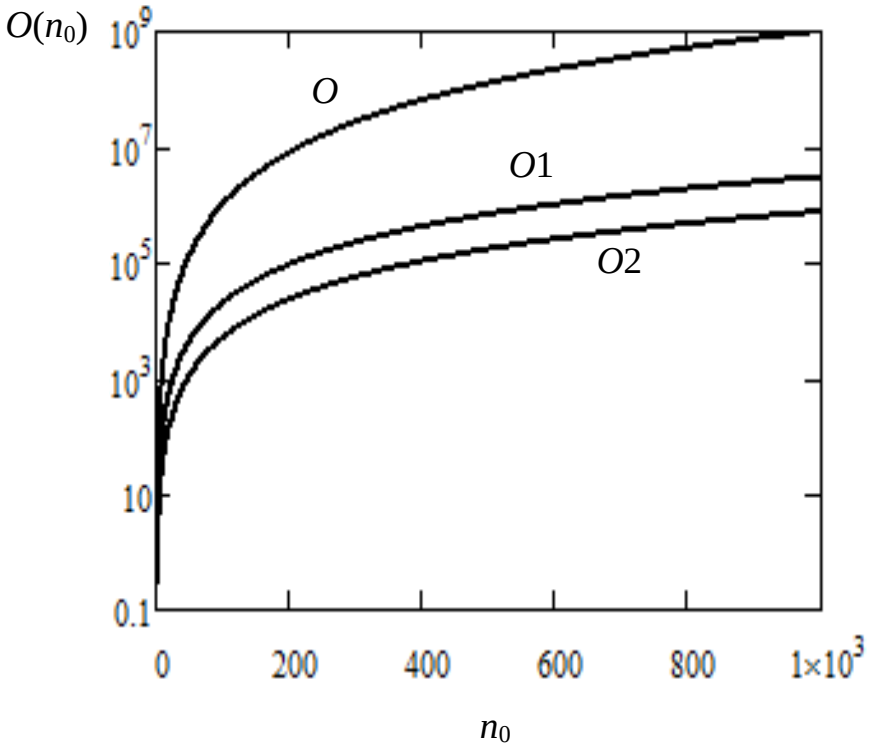


Рис. 2.7 – Часова складність операції модулярного піднесення до степеня

Експериментальні дослідження швидкодії стандартного і запропонованого алгоритмів для чисел різної розрядності (32, 64 і 96 біт) проводилися на комп'ютері із двох'ядерним процесором типу Intel (тактова частота 2,3 ГГц, оперативна пам'ять - 2 ГБайт). Результати цих досліджень представлені на рис. 2.8 [43].

Як видно з гістограми, запропоновані алгоритми потребують менше часу для виконання розглянутих операцій.

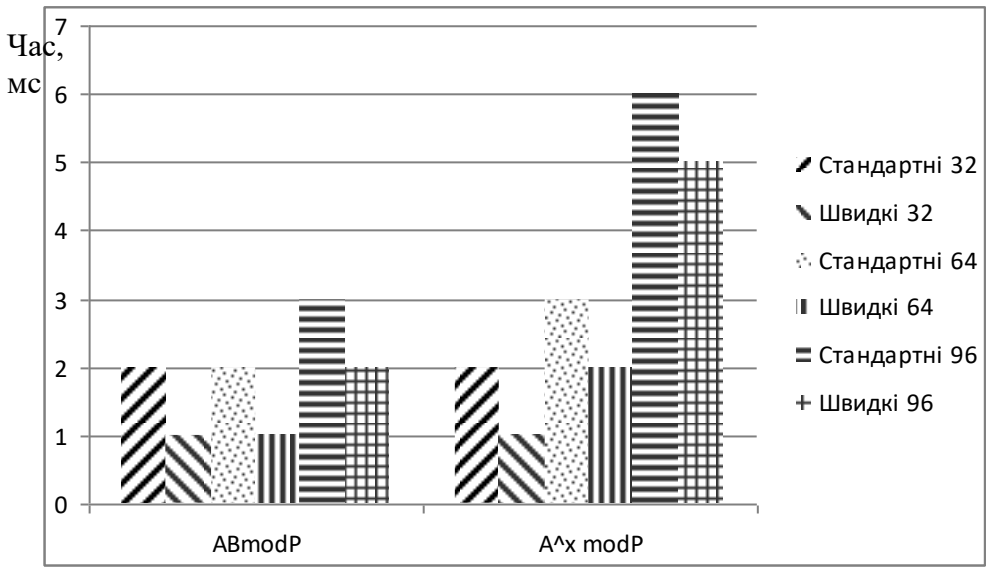


Рис. 2.8 – Експериментальне порівняння швидкодії стандартного і запропонованого алгоритмів для чисел різної розрядності

Отже, запропоновані методи модулярного множення та експоненціювання дозволяють зменшити часову складність за рахунок заміни операції множення додаванням.

Звідси випливає доцільність їх використання в асиметричних криптографічних алгоритмах захисту інформації для зменшення складності обчислень, генерування ключів, шифрування та розшифрування повідомлень тощо.

2.5 Методи пошуку оберненого елемента за модулем на основі додавання модуля та залишку

З теорії чисел відомо [9, 10], що вираз $a \cdot b \bmod p = 1$ можна переписати таким чином: $a \cdot b = \tilde{k} \cdot p + 1$, де $\tilde{k}$ – деяке ціле число. Звідси випливає, що для пошуку мультиплікативного оберненого елемента до модуля необхідно додати 1 і перевіряти, чи ділиться отримане число націло на a [302-304]. Якщо не ділиться, то тоді знову до отриманого числа додається послідовно

модуль до тих пір, поки результат ділення не буде цілим числом. Математично це записується так:

$$\begin{gathered} p_{01} = p+1;\ b_{01} = (p+1)/a; \\ p_{11} = 2\cdot p+1;\ b_{11} = (2\cdot p+1)/a; \\ \ldots. \\ p_{i1} = (i+1)\cdot p+1;\ b_{i1} = ((i+1)\cdot p+1)/a;\ b_{i1} \in Z. \end{gathered} \tag{2.15}$$

В табл. 2.13 наведено приклад застосування методу пошуку оберненого елемента на основі додавання модуля.

Таблиця 2.13 - Пошук оберненого елемента $41^{-1}\text{mod } 157$ на основі додавання модуля

i	0	1	2	3	4	5
p_{i1}	158	315	472	629	786	943
b_{i1}	3,85…	7,68…	11,51…	15,34…	19,17…	23

Отже, $41^{-1}\text{mod } 157=23$. Результат отримано без використання громіздких операцій ділення з остачею і множення.

Для зменшення чисел, що використовуються у даній процедурі, можна додавати не модуль, а залишок $p_{00}=p \bmod a$ до тих пір, поки остача від ділення отриманого результату на число a не буде дорівнювати 0. Математичний запис даного алгоритму має такий вигляд:

$$\begin{gathered} b_{02} = (p_{00}+1)\text{mod}a; \\ b_{12} = (b_{02}+p_{00})\text{mod}a; \\ b_{22} = (b_{12}+p_{00})\text{mod}a; \\ \ldots. \\ b_{i2} = (b_{i-1\ 2}+p_{00})\text{mod}a = 0. \end{gathered} \tag{2.16}$$

Шуканий обернений елемент обчислюється за такою формулою:

$$b = a^{-1} \bmod p = \frac{(i+1)p+1}{a}. \qquad (2.17)$$

В таблиці 2.14 наведено приклад застосування методу пошуку оберненого елемента $41^{-1} \bmod 157$ на основі додавання залишку, попередньо обчисливши, що $157 \bmod 41 = 34$.

Таблиця 2.14 - Пошук оберненого елемента $41^{-1} \bmod 157$ на основі додавання залишку

i	0	1	2	3	4	5
b_{i2}	35	28	21	14	7	0

За формулою (2.17) можна отримати, що $b = 41^{-1} \bmod 157 = \frac{6 \cdot 157 + 1}{41} = 23.$

На відміну від розширеного алгоритму Евкліда, запропоновані методи дають можливість розпаралелити процес пошуку оберненого елемента на декілька потоків. Початок обчислень у кожному потоці для методів додавання модуля і залишку визначається відповідно з таких формул:

$$N_0 = \left(\left[\frac{(j-1)a}{z} \right] + 1 \right) p + 1; \qquad (2.18)$$

$$N_1 = \left(\left(\left[\frac{(j-1)a}{z} \right] + 1 \right) p_{00} + 1 \right) \bmod a, \qquad (2.19)$$

де j - номер потоку, z - кількість потоків. Максимальна кількість ітерацій у кожному потоці становитиме $a/z+1$.

В табл. 2.15 представлена часова складність основних операцій запропонованого методу

Таблиця 2.15 – Часова складність базових операцій запропонованого методу

№	Основні операції	Часова складність запропонованих методів
1.	$i \cdot a_{10}+1$	$O(i \cdot n_0)$
2.	$(i \cdot a_{10}+1) \bmod a_3$	$O\left(i \cdot \left(\frac{n_0}{2} + \log_2 n_0/2\right)\right)$

Загальна часова складність пошуку оберненого елемента запропонованим методом, вважаючи max $i=a/z+1$, становить $O\left((a/z+1)\left(\frac{n_0}{2} + \log_2 n_0/2\right)\right)$. Класичний метод із використанням розширеного алгоритму Евкліда приводить до такого результату: $O(n_0^3)$ [305].

Отже, розроблений метод дозволяє уникати виконання складних операцій, зокрема, ділення з остачею, і виконувати обчислення над числами значно меншої розрядності у порівнянні із класичним методом пошуку оберненого елемента за модулем з використанням розширеного алгоритму Евкліда.

2.6 Алгоритмічне забезпечення криптосистеми RSA на основі векторно-модульного методу модулярного експоненціювання та множення

Піднесення до степеня у криптосистемі RSA доцільно виконувати на основі векторно-модульного методу модулярного експоненціювання згідно виразу $A^e \bmod n = \left(\prod_{i=0}^{n_0-1}\left(A_i^{e_i 2^i}\right)\bmod n\right)\bmod n = \prod_{i=0}^{n_0-1} f_i^{e_i} \bmod n$, де A – блок

відкритого тексту, e і n – відкритий ключ, n_0 – розрядність степеня e, e_i=0 або 1, $f_i = A^{2^i} \bmod n$, причому $f_i = (f_{i-1})^2 \bmod n$.

Розглянемо приклад шифрування/дешифрування за допомогою алгоритму RSA на основі векторно-модульного методу модулярного множення та експоненціювання. На першому етапі генеруються відкритий та таємний ключі [306, 307].

Нехай $p = 7$, $q = 11$, тоді $n = p \cdot q = 77$. Далі обчислюється значення функції Ейлера: $\varphi(n) = (p-1)(q-1) = 6 \cdot 10 = 60$. Натуральне число e, яке є відкритим ключем, має бути менше $\varphi(n)$ і взаємно просте з $\varphi(n)$. Для прикладу виберемо e=13. Тоді таємний ключ $d = e^{-1}(\bmod \varphi(n)) = 13^{-1}(\bmod 60)$. Далі, врахувавши, що $e_{10} = 60 \bmod 13 = 8 \neq 0$, на основі (2.16) та таблиці 2.14 будується табл. 2.16.

Згідно виразу (2.17) обчислюється таємний ключ: $d = e^{-1}(\bmod \varphi(n)) = 13^{-1}(\bmod 60) = \dfrac{8 \cdot 60 + 1}{13} = 37$.

Таблиця 2.16 – Пошук таємного ключа

i	0	1	2	3	4	5	6	7
e_{i1}	9	4	12	7	2	10	5	0

Для шифрування інформаційний потік подається у вигляді цілого числа A=17<n=77, над яким на основі співвідношення (1.9) потрібно виконати операцію модулярного експоненціювання, представивши e=13 в двійковій системі числення (13_{10}=1101_2):

$$A' = 17^{13} \bmod 77 = \left(17^{(1 \cdot 2^0 + 1 \cdot 2^2 + 1 \cdot 2^3}\right) \bmod 77. \qquad (2.20)$$

В табл. 2.17 повністю представлена процедура шифрування за допомогою криптоалгоритму RSA на основі векторно-модульного методу модулярного експоненціювання та множення згідно виразів (2.11), (2.14) та табл. 2.7, 2.11, а також методу пошуку залишку в розмежованій системі числення згідно формул (2.1), (2.2) та табл. 2.1.

Таблиця 2.17 - Шифрування за допомогою криптоалгоритму RSA на основі векторно-модульного методу

i	6	5	4	3	2	1	0
$13_{10}=1101_2$	0	0	0	1	1	0	1
$17^{2^i} \bmod 77$	53	58	60	37	53	58	17
$17^{13} \bmod 77$	$37 \cdot 53 \cdot 17 \bmod 77$						
$2^i \cdot 37 \bmod 77$	58	29	53	65	71	74	37
$53_{10}=110111_2$	0	1	1	0	1	0	1
$53 \cdot 37 \bmod 77$	$(29+53+71+37) \bmod 77=36$						
$2^i \cdot 36 \bmod 77$	71	74	37	57	67	72	36
$17_{10}=10001_2$	0	0	1	0	0	0	1
$36 \cdot 17 \bmod 77$	$(37+36) \bmod 77=73$						

Отже, $A'=17^{13} \bmod 77=73$.

Розшифрування виконується згідно виразу (1.10): $A=73^{37} \bmod 77$. Результати представлені в табл. 2.18.

Отже, використання векторно модульного методу модулярного множення та експоненціювання дозволяє зменшити часову складність процедур шифрування та розшифрування алгоритму RSA завдяки уникненню виконання громіздких операцій множення та ділення з остачею багаторозрядних чисел.

Таблиця 2.18 - Розшифрування за допомогою криптоалгоритму RSA на основі векторно-модульного методу модулярного експоненціювання та множення

i	6	5	4	3	2	1	0
$37_{10}=100101_2$	0	1	0	0	1	0	1
$73^{2^i} \bmod 77$	25	16	4	9	25	16	73
$73^{37} \bmod 77$	$16 \cdot 25 \cdot 73 \bmod 77$						
$2^i \cdot 25 \bmod 77$	60	30	15	46	23	50	25
$16_{10}=10000_2$	0	0	1	0	0	0	0
$16 \cdot 25 \bmod 77$	$15 \bmod 77=15$						
$2^i \cdot 15 \bmod 77$	36	18	9	43	60	30	15
$73_{10}=1001001_2$	1	0	0	1	0	0	1
$36 \cdot 17 \bmod 77$	$(36+43+15) \bmod 77=173$						

2.7 Алгоритмічне забезпечення криптосистеми Ель-Гамаля на основі векторно-модульного методу модулярного експоненціювання та множення

Розглянемо приклад шифрування/розшифрування алгоритму Ель-Гамаля. На першому етапі генеруються відкритий та закритий ключі [306].

Нехай p=29, g=19. Вибирається a=21 – випадкове ціле число для якого справджується нерівність: $1<x<p$. Далі обчислюється значення $h = g^a \bmod p = 19^{21} \bmod 29$ на основі використання векторно-модульного методу модулярного експоненціювання, тобто згідно формули:

$$19^{21} \bmod 29 = \left(19^{(1\cdot 2^0 + 1\cdot 2^2 + 1\cdot 2^4}\right) \bmod 29. \qquad (2.21)$$

Шуканий результат можна отримати на основі табл. 2.19:

Таблиця 2.19 - Пошук відкритого ключа

	i	4	3	2	1	0
	$21_{(10)}=10101_{(2)}$	1	0	1	0	1
	$19^{2^i} \bmod 29$	$19^{2^4} \bmod 29 = 16$	$19^{2^3} \bmod 29 = 25$	$19^{2^2} \bmod 29 = 24$	$19^{2^1} \bmod 29 = 13$	$19^{2^0} \bmod 29 = 19$
	$19^{21} \bmod 29$	$19 \cdot 24 \cdot 16 \bmod 29$				
	$2^i \cdot 24 \bmod 29$	7	18	9	19	24
	$19_{(10)}=10011_{(2)}$	1	0	0	1	1
	$19 \cdot 24 \bmod 29$	$(7+19+24) \bmod 29=21$				
	$2^i \cdot 21 \bmod 29$	17	23	26	13	21
	$16_{(10)}=10000_{(2)}$	1	0	0	0	0
	$16 \cdot 21 \bmod 29$	$19 \cdot 24 \cdot 16 \bmod 29=21 \cdot 16 \bmod 29=17$				

Отже, відкритим ключем буде трійка чисел (p, g, h)=(29, 19, 17), а закритим - a=21.

Вибирається випадкове ціле число r таке, що $1<r<p-1$. Нехай r=23.

Обчислюється значення $c_1 = g^r \bmod p = 19^{23} \bmod 29$ на основі такого виразу:

$$c_1 = 19^{23} \bmod 29 = \left(19^{(1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^4}\right) \bmod 29. \qquad (2.22)$$

Результати записані в табл. 2.20:

Для отримання шифротексту для відкритого тексту A=14 знаходиться значення $c_2 = h^r \cdot A \bmod p = 17^{23} \cdot 14 \bmod 29$.

Знайдемо спочатку значення $h^r \bmod p = 17^{23} \bmod 29$ на основі використання векторно-модульного алгоритму модулярного експоненціювання згідно формули (2.14):

Таблиця 2.20 - Пошук $c_1 = g^r \bmod p$

	i	4	3	2	1	0
	$23_{(10)}=10101_{(2)}$	1	0	1	1	1
	$19^{2^i} \bmod 29$	$19^{2^4} \bmod 29 = 16$	$19^{2^3} \bmod 29 = 25$	$19^{2^2} \bmod 29 = 24$	$19^{2^1} \bmod 29 = 13$	$19^{2^0} \bmod 29 = 19$
	$19^{23} \bmod 29$	$19 \cdot 13 \cdot 24 \cdot 16 \bmod 29 = 13 \cdot 17 \bmod 29$				
	$2^i \cdot 13 \bmod 29$	5	17	23	26	13
	$17_{(10)}=10001_{(2)}$	1	0	0	0	1
	$19 \cdot 24 \bmod 29$	$19 \cdot 13 \cdot 24 \cdot 16 \bmod 29 = (5+13) \bmod 29 = 18$				

$$17^{23} \bmod 29 = \left(17^{(1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^4}\right) \bmod 29. \qquad (2.23)$$

Результати шифрування представлені в табл. 2.21.

Таблиця 2.21 - Пошук шифротексту

	i	4	3	2	1	0
	$23_{(10)}=10001_{(2)}$	1	0	1	1	1
	$17^{2^i} \bmod 29$	$17^{2^4} \bmod 29 = 1$	$17^{2^3} \bmod 29 = 1$	$17^{2^2} \bmod 29 = 1$	$17^{2^1} \bmod 29 = 28$	$17^{2^0} \bmod 29 = 17$
	$14 \cdot 17^{23} \bmod 29$	$14 \cdot 17 \cdot 28 \cdot 1 \cdot 1 \bmod 29$				
	$2^i \cdot 14 \bmod 29$	21	25	27	28	14
	$17_{(10)}=10001_{(2)}$	1	0	0	0	1
	$14 \cdot 17 \bmod 29$	$(21+14) \bmod 29 = 6$				
	$2^i \cdot 28 \bmod 29$	5	17	23	26	28
	$6_{(10)}=00110_{(2)}$	0	0	1	1	0
	$2 \cdot 28 \bmod 29$	$14 \cdot 17 \cdot 28 \cdot 1 \cdot 1 \bmod 29 = (23+26) \bmod 29 = 23$				

Отримана пара (c_1, c_2)=(18, 23) є шифротекстом.

Розшифрування відбувається згідно формули $M = b \cdot (a^x)^{-1} \bmod p = c_2 c_1^{p-1-a} \bmod p = 23 \cdot 18^7 \bmod 29.$

Спочатку значення $18^{21} \bmod 29$ записується в розмежованій системі числення згідно формули (2.14):

$$18^7 \bmod 29 = \left(18^{(1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2}\right) \bmod 29. \qquad (2.24)$$

Згідно табл. 2.22 отримується шуканий результат. Відкритим текстом є повідомлення A=14.

Таблиця 2.22 - Розшифрування повідомлення

i	4	3	2	1	0
$7_{(10)}=00111_{(2)}$	0	0	1	1	1
$18^{2^i} \bmod 29$	$18^{2^4} \bmod 29 = 24$	$18^{2^3} \bmod 29 = 16$	$18^{2^2} \bmod 29 = 25$	$18^{2^1} \bmod 29 = 5$	$18^{2^0} \bmod 29 = 18$
$23 \cdot 18^{23} \bmod 29$	23·25·5·18 mod 29				
2^i·23 mod 29	20	10	5	17	23
$25_{(10)}=10001_{(2)}$	1	1	0	0	1
23·25 mod 29	(20+10+23) mod 29=24				
2^i·18 mod 29	27	28	14	7	18
$5_{(10)}=00110_{(2)}$	0	0	1	0	1
5·18 mod 29	(14+18) mod 29=3				
2^i·24 mod 29	7	18	9	19	24
$3_{(10)}=00011_{(2)}$	0	0	0	1	1
3·24 mod 29	23·25·5·18 mod 29=3·24 mod 29=(19+24) mod 29=14				

Даний підхід, який грунтується на використанні векторно-модульного методу модулярного експоненціювання в задачах

шифрування/розшифрування асиметричної криптосистеми Ель-Гамаля дозволяє зменшити часову складність за рахунок заміни операції медулярного експоненціювання операцією множення, а операції множення – додаванням.

2.8 Алгоритм Евкліда у розмежованій системі числення

Нехай потрібно знайти НСД чисел $a=a_{n_0-1}2^{n_0-1}+\ldots+a_i2^i+\ldots+a_12+a_0$ та $b=b_{n_0-1}2^{n_0-1}+\ldots+b_i2^i+\ldots+b_12+b_0$, причому $a>b=r_0$; a_i, b_i=0,1 [308].

Виходячи з стандартного алгоритму Евкліда (1.5), $r_1=a \bmod b=$ $=(a_{n_0-1}2^{n_0-1}+\ldots+a_i2^i+\ldots+a_12+a_0) \bmod b=\left(\sum_{i=0}^{n_0-1}\left(a_i 2^i \bmod b\right)\right)\bmod b=\left(\sum_{i=0}^{n_0-1}\left(a_i r_{1i}\right)\right)\bmod b$,

де $r_{1i}=2^i \bmod b$. Це означає, що шуканий залишок дорівнює сумі тих степенів двійки, для яких відповідні a_i=1. Два послідовні значення r_{1i} і $r_{1\,i+1}$ пов'язані рекурентним співвідношенням $r_{1\,i+1}=(2\cdot r_{1i})\bmod b$. Для пошуку залишку за модулем b не обов'язково виконувати ділення з остачею, а можна обмежитися відніманням аналогічно до (2.2): якщо $r_{1\,i+1}<b$, то воно залишається незмінним, в іншому випадку $r_{1i+1}=r_{1\,i+1}-b$. Найпростіше реалізувати описаний крок алгоритму Евкліда в розмежованій системі числення за допомогою табл. 2.23 відповідно табл. 2.1.

Таблиця 2.23 - Таблиця знаходження залишку $a \bmod b$.

i	n_0-1	n_0-2	…	2	1	0
a_i	a_{n_0-1}	a_{n_0-2}	…	a_2	a_1	a_0
r_{1i}	r_{1n_0-1}	r_{1n_0-2}	…	r_{12}	r_{11}	r_{10}

Згідно таблиці 2.23, r_1 шукається як сума r_{1i} за модулем b, над якими у верхньому рядку розміщено 1, тобто $r_1 = \left(\sum_{i=0}^{n_0-1} r_{1i} \right) \bmod b$ при умові, що a_i=1.

Аналогічно будується табл. 2.24.

Таблиця 2.24 - Таблиця знаходження залишку $b \bmod r_1$.

$b_i=r_{0\,i}$	$b_{n_0-1}=r_{0\,n_0-1}$	$b_{n_0-2}=r_{0\,n_0-2}$	…	$b_2=r_{0\,2}$	$b_1=r_{0\,1}$	$b_0=r_{0\,0}$
$r_{2\,i}$	$r_{2\,n_0-1}$	$r_{2\,n_0-2}$	…	r_{22}	r_{21}	r_{20}

Відповідно $r_2 = \left(\sum_{i=0}^{n_0-1} r_{2i} \right) \bmod r_1$ при умові b_i=1.

Звідси запишемо вираз для знаходження будь–якого залишку:

$$r_j = \left(\sum_{i=1}^{n_0-1} r_{j-2i} r_{ji} \right) \bmod r_{j-1}, \qquad (2.25)$$

де $r_{j-2\,i}$=0,1; $r_{ji}=2^i \bmod r_{i-1}$.

Відмітимо, що кількість кроків стандартного алгоритму Евкліда і алгоритму Евкліда у розмежованій системі числення однакові, крім того, він виключає можливість розпаралелення. Однак часова складність виконання кожного кроку істотно зменшується і становить $\log_2 n/2$.

Розглянемо приклад. Нехай потрібно обчислити НСД(3843, 1449).

Стандартний алгоритм Евкліда матиме такий вигляд [308]:

3843=1449·1+945

1449=945·1+504

945=504·1+441

504=441·1+63

441=63·7+0

Отже, НСД(3843, 1449)=63.

Алгоритм Евкліда в розмежованій системі числення зручно представити у вигляді табл. 2.25.

Таблиця 2.25 - Алгоритм Евкліда в розмежованій системі числення

	i	11	10	9	8	7	6	5	4	3	2	1	0
1	3843	1	1	1	1	0	0	0	0	0	0	1	1
2	2^imod1449	599	1024	512	256	128	64	32	16	8	4	2	1
3	(599+1024+512+256+2+1)mod 1449=945												
4	1449		1	0	1	1	0	1	0	1	0	0	1
5	2^imod945		79	512	256	128	64	32	16	8	4	2	1
6	(79+256+128+32+8+1)mod 945=504												
7	945			1	1	1	0	1	1	0	0	0	1
8	2^imod504			8	256	128	64	32	16	8	4	2	1
9	(8+256+128+32+16+1)mod 504=441												
10	504				1	1	1	1	1	1	0	0	0
11	2^imod441				256	128	64	32	16	8	4	2	1
12	(256+128+64+32+16+8)mod 441=63												
13	441				1	1	0	1	1	1	0	0	1
14	2^imod63				4	2	1	32	16	8	4	2	1
15	(4+2+32+16+8+1)mod 63=0												

Оскільки, як і в стандартному алгоритмі Евкліда, найбільшим спільним дільником двох чисел є остання відмінна від 0 остача, то НСД(3843, 1449)=63, причому результат отриманий безгроміздкої операції ділення.

2.9 Китайська теорема про залишки на основі додавання добутку модулів

Як зазначалося в п. 1.1, пошук оберненого елемента при реалізації КТЗ є досить громіздкою задачею. Крім того, використання багаторозрядних

чисел може привести до переповнення розрядної сітки при відновленні десяткового числа в (1.8).

З теорії чисел відомо, що перше рівняння (1.7) можна записати у вигляді $A=\gamma_1 p_1+r_1$, де $\gamma_1=0, 1, 2, \ldots$. Це означає, що для пошуку числа A до залишку потрібно додати модуль p_1 стільки разів, щоб виконувалось друге рівняння (1.7). Далі необхідно додавати добуток модулів p_1p_2, поки не буде виконуватися третє рівняння (1.7) [309, 310]. Процес продовжується до тих пір, поки не буде виконуватися останнє рівняння (1.7). Математично це записується таким чином:

$$
\begin{gathered}
A_1=r_1; \\
A_2= A_1+\gamma_1 p_1=r_1+\gamma_1 p_1;\ A_2 \bmod p_2=r_2; \\
A_3= A_2+\gamma_2 p_1 p_2=r_1+\gamma_1 p_1+\gamma_2 p_1 p_2;\ A_3 \bmod p_3=r_3; \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
A_i= A_{i-1}+\gamma_{i-1} p_1 p_2 \ldots p_{i-1};\ A_i \bmod p_i=r_i; \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
A_j=A=A_{j-1}+\gamma_{j-1} p_1 p_2 \ldots p_{j-1};\ A_j \bmod p_j=r_j.
\end{gathered}
\qquad (2.26)
$$

Розглянемо приклад. Нехай задана система порівнянь (не зменшуючи загальності вважаємо, $p_1>p_2>\ldots>p_j$, оскільки це дозволяє робити більші кроки при виконанні ітерацій) [309]:

$$
\begin{cases}
A \bmod 17=11 \\
A \bmod 13=10 \\
A \bmod 11=8 \\
A \bmod 7=4.
\end{cases}
\qquad (2.27)
$$

В табл. 2.26 наведено приклад КТЗ за допомогою додавання добутку модулів.

Таблиця 2.26 - Приклад КТЗ за допомогою додавання добутку модулів

i	1	2	3	4	5
$11+(i-1)\cdot 17$	11	28	45	62	
$(11+(i-1)\cdot 17)\bmod 13$	11	2	6	10	
$p_1\cdot p_2$	$17\cdot 13=221$				
$62+(i-1)\cdot 221$	62	283			
$(62+(i-1)\cdot 221)\bmod 11$	7	8			
$p_1\cdot p_2\ p_3$	$17\cdot 13\cdot 11=2431$				
$283+(i-1)\cdot 2431$	283	2714	5145	7576	10007
$(283+(i-1)\cdot 2431))\bmod 7$	3	5	0	2	4

Отже, розв’язком системи (2.27) є число 10007, яке отримане без використання громіздких операцій та необхідності контролю переповнення розрядної сітки при виконанні проміжних обчислень. На рис. 2.9 представлена блок-схема КТЗ, яка реалізується за допомогою додавання добутку модулів. Слід відмітити, що даний метод подібний до алгоритму Гарнера, однак у ньому уникається пошук оберненого елемента за модулем для отримання відповідних коефіцієнтів.

Для зменшення чисел, які використовуються у запропонованому методі, можна додавати не добуток модулів, а залишок цього добутку від ділення на відповідний модуль. Математичний запис даного методу виглядає таким чином [310]:

$$A_1=r_1;\ p_{11}=p_1 \bmod p_2;$$
$$(A_1+\gamma_1 p_{11}) \bmod p_2 = r_2;\ A_2= A_1+\gamma_1 p_1;\ p_{12} =(p_1p_2) \bmod p_3;$$
$$(A_2+\gamma_2 p_{12}) \bmod p_3 = r_3,\ A_3=A_2+\gamma_2 p_1p_2;\ p_{13} =(p_1p_2p_3) \bmod p_4;$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (2.28)$$
$$(A_{i-1}+\gamma_{i-1}p_{1\,i-1}) \bmod p_i=r_i;\ A_i= A_{i-1}+\gamma_{i-1}p_1p_2p_3\ldots p_{i-1};\ p_{1i} =(p_1p_2\ldots p_i) \bmod p_{i+1};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(A_{j-1}+\gamma_{j-1}p_{1\,j-1}) \bmod p_j= r_j;\ A=A_j=A_{j-1}+\gamma_{j-1}p_1p_2p_3\ldots p_{j-1}.$$

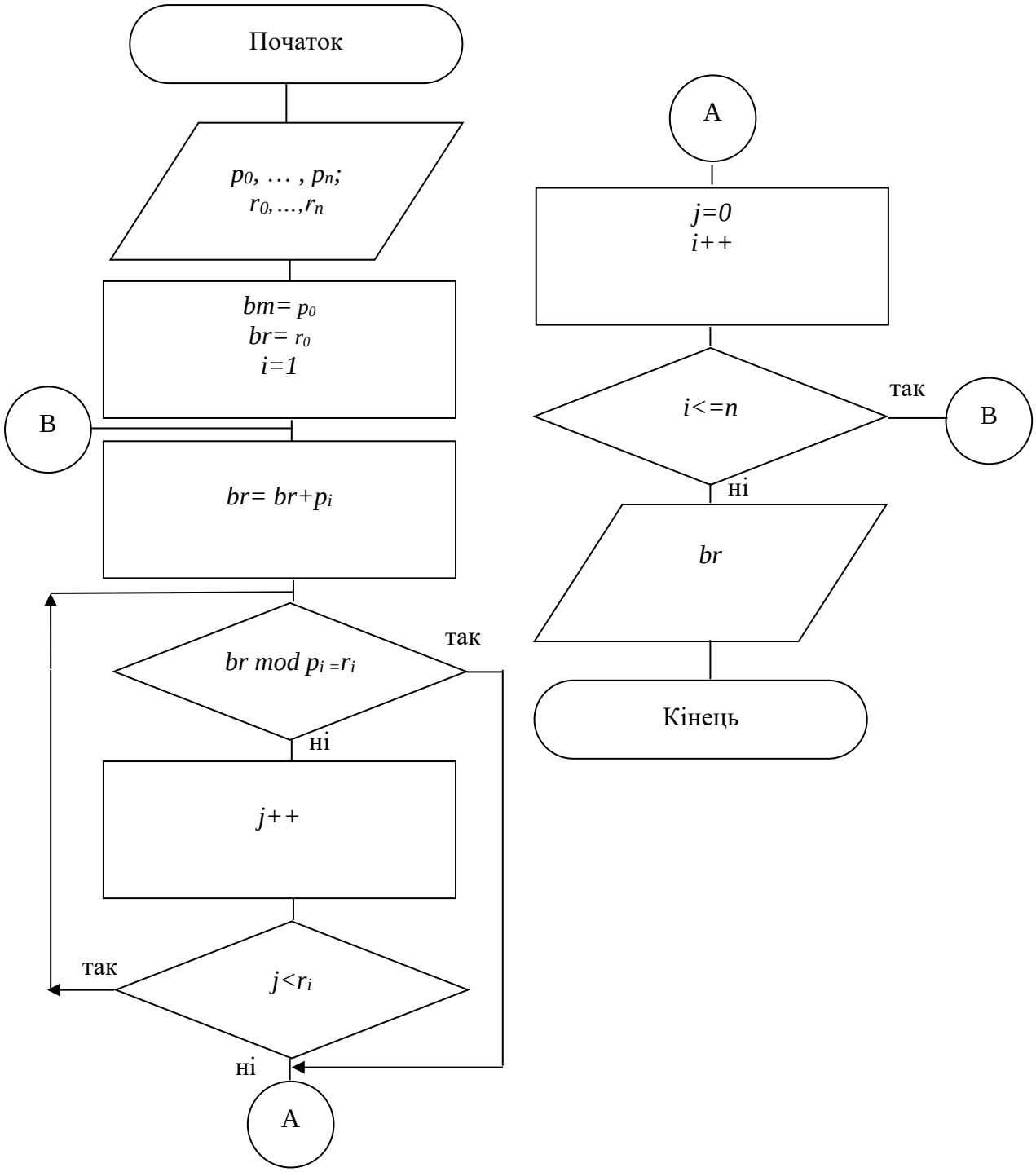


Рис. 2.9 – Блок схема КТЗ на основі додавання добутку модулів

В табл. 2.27 наведено приклад КТЗ за допомогою додавання залишку від добутку модулів.

Отже, розроблені методи дозволяють уникати виконання складних операцій, зокрема, ділення з остачею і пошуку оберненого елемента, та проводити обчислення над числами значно меншої розрядності у порівнянні з класичною КТЗ та алгоритмом Гарнера.

Таблиця 2.27 - Приклад КТЗ за допомогою додавання залишку від добутку модулів

i	1	2	3	4	5
p_{11}=17mod13=4					
(11+(i-1)·4)mod13	11	2	6	10	
A_2=11+(i-1)·17=11+3·17=62; 17·13=221; 221mod 11=1					
(62+(i-1)·1)mod11	7	8			
A_3=62+(i-1)·221=62+1·221=283; 17·13·11=2431; 2431mod 7=2					
(2431+(i-1)·2)mod7	3	5	0	2	4
A_4=A=283+(i-1)·2431=283+4·2431=10007; 10007mod 7=4.					

На відміну від них, запропоновані методи дозволяють розпаралелити процес виконання КТЗ. Початок обчислень в кожному потоці для методів додавання добутку модулів та залишку від добутку модулів, починаючи від найбільшого, відповідно визначається з такого виразу:

$$A_{1i} = \left[\frac{P}{j}\right](i-1)+1+\left(p_1-\left(\left[\frac{P}{j}\right](i-1)+1\right)\bmod p_1+r_1\right)\bmod p_1, \qquad (2.29)$$

де j-кількість потоків, i – номер потоку.

Подальші обчислення відбуваються аналогічно виразам (2.26)-(2.27) та таблицям 2.26-2.27.

2.10 Метод обчислення мультистепеневої функції

Даний метод зручно використати, коли операцію піднесення до степеня в асиметричних криптосистемах можна представити у вигляді мультистепеневої функції [311]. Отже, нехай потрібно обчислити значення такого виразу:

$$x = a^{b^c} \bmod \ p, \tag{2.30}$$

Для цього треба знайти розклад на прості множники найбільшого спільного дільника чисел a і p: d_0=НСД(a, p)=$2^{\alpha_1} \cdot 3^{\alpha_2} ... d_n^{\alpha_n}$. Тоді вираз (2.30) представляється у такому вигляді:

$$x = a^{r_1} \cdot a^{b^c - r_1} \bmod \ p, \tag{2.31}$$

де $r_1 = \max(\alpha_1, \alpha_2, ..., \alpha_n)$.

Поділивши обидві частини (2.31) на d_0, отримаємо:

$$x_1 = a_0 \cdot a^{b^c - r_1} \bmod \ p_0, \tag{2.32}$$

де $a_0 = \frac{a^{r_1}}{d_0}$, $p_0 = \frac{p}{d_0}$, $x_1 = \frac{x}{d_0}$.

Для того, щоб у рівності (2.32) зменшити степінь, необхідно, згідно теореми Ейлера, знайти залишок від ділення степеня на значення функції Ейлера від модуля: $y = \left(b^c - r_1\right) \bmod \varphi(p_0)$ або:

$$y + r_1 = b^c \bmod \varphi(p_0). \tag{2.33}$$

Аналогічно відбувається пошук НСД (b, $\varphi(p_0)$)= $2^{\beta_1} \cdot 3^{\beta_2} ... d_n^{\beta_n} = d_1$. Поділивши (2.33) на d_1, матимемо:

$$y_1 = b_0 \cdot b^{c - r_2} \bmod p_1 = \frac{y + r_1}{d_1}, \tag{2.34}$$

де $b_0 = \frac{a^{r_2}}{d_1}$, $p_1 = \frac{\varphi(p_0)}{d_1}$, $r_2 = \max(\beta_1, \beta_2, ..., \beta_n)$.

Таким чином відбувається перехід до лінійної конгруенції $z = (c - r_2) \bmod \varphi(p_1)$ з істотно меншим значенням модуля.

Для знаходження шуканого значення x з (2.34) можна отримати: $y_1 = b_0 \cdot b^z \bmod p_1$, відповідно $y = y_1 \cdot d_1 - r_1$, тоді з рівності (2.32) маємо: $x_1 = a_0 \cdot a^y \bmod p_1$ і $x = x_1 \cdot d_0$.

Розглянемо приклад [311]: $8^{8^8} \bmod 36 = x$.

Використовуючи послідовно вирази (2.31)–(2.34), матимемо:

$8 \cdot 8^{8^8 - 1} \bmod 36 = x$; $2 \cdot 8^{8^8 - 1} \bmod 9 = \frac{x}{4}$; $2 \cdot 8^y \bmod 9 = \frac{x}{4}$, де $y = (8^8 - 1) \bmod \varphi(9) = (8^8 - 1) \bmod 6$;

$8^8 \bmod 6 = 2^8 \bmod 6 = y + 1$; $2 \cdot 2^7 \bmod 6 = y + 1$; $2^7 \bmod 3 = \frac{y+1}{2}$;

$2^z \bmod 3 = \frac{y+1}{2}$, де $z = 7 \bmod \varphi(3) = 7 \bmod 2 = 1$.

Тоді $2^1 \bmod 3 = 2 = \frac{y+1}{2}$. Звідси y=3 і $2 \cdot 8^3 \bmod 9 = \frac{x}{4}$.

Отже $\frac{x}{4} = 7$ і x=28.

2.11 Вдосконалення методу факторизації Ферма

Як було сказано в п. 1.3, найбільш поширеним на даний час є метод факторизації Ферма, який ґрунтується на пошуку пар натуральних чисел A і B, для яких виконується рівність $n=A^2-B^2$, де $n=p \cdot q$ – відоме ціле число, що є добутком двох невідомих простих чисел p і q. Далі шукається $m = \lceil \sqrt{n} \rceil$ і обчислюється параметр $q(x)=(m+x)^2-n$, де x=1, 2, 3, … , до тих пір, поки деяке

значення $q(x)$ не буде рівним повному квадрату деякого числа. Тоді шуканий розклад $n=p\cdot q= A^2-B^2=(A-B)(A+B)$.

Найбільш обчислювально складними операціями в даному випадку є піднесення до квадрату та пошук квадратного кореня [312]. Для вдосконалення методу факторизації Ферма доцільно використати умову, що квадрати усіх цілих чисел представляються сумою непарних чисел, кількість яких рівна заданому числу [313-316]:

$$s^2 = \sum_{i=1}^{s}(2i-1). \qquad (2.35)$$

Тому, знайшовши m та $q_1(x)=q(x)$ при $x=1$, наступні кроки відбуваються згідно виразу $q_i=q_{i-1}+2(m+i)-1$, де $i=2, 3, 4, \ldots$ до тих пір, поки q_i не буде повним квадратом деякого числа. Розклад на множники визначатиметься таким виразом: $n=(m+i-q_i)(m+i+q_i)$.

В табл. 2.28 наведено приклад факторизації за допомогою класичного та вдосконаленого методу Ферма для $n=4717$ ($m_1=\left\lceil\sqrt{4717}\right\rceil=68$).

Таким чином отримано розклад числа 4717 на прості множники:

$$4717=71^2-18^2=(71+18)(71-18)=89\cdot 53.$$

Таблиця 2.28 - Приклад факторизації числа 4717 за допомогою класичного та вдосконаленого методів Ферма

x	$m+x$	$q(x)$, класичний метод	$q(x)$, вдосконалений метод
1	69	$69^2-4717=44$	$69^2-4717=44$
2	70	$70^2-4717=183$	$44+139=183$
3	71	$71^2-4717=324=18^2$	$183+141=324=18^2$

Слід зазначити, що кількість ітерацій в обох методах однакова. Однак у вдосконаленому методі Ферма виключається операція піднесення до

квадрату великих чисел. Крім того, арифметичні дії виконуються над числами набагато меншої розрядності, ніж у класичному. Але найбільш обчислювально трудомісткою операцією залишається добування квадратного кореня.

У запропонованому методі також використовується властивість (2.35), параметри $m_1=m$ та $q_{11}=q_1$ обчислюються аналогічно до попереднього випадку. Далі виконується така послідовність операцій $q_{1i}=q_{1\ i-1}-r_{1\ i-1}$, $r_{11}=1$, $r_{1\ i-1}=2i-3=r_{1\ i-2}+2$, $i=2, 3, \ldots$ до тих пір, поки для деякого i не буде виконуватись умова [317]:

$$q_{1i}-r_{1i}\le 0. \qquad (2.36)$$

При виконанні строгої нерівності (2.36) подальші обчислення відбуваються таким чином: $m_2=m_1+1$; $q_{21}=q_{1i}+2\ m_2+1-r_{1i}$, $r_{21}=r_{1i}+2$. Пошук наступних значень q_{2i}, r_{2i} здійснюється аналогічно до попереднього випадку. В загальному випадку усі зазначені розрахунки можна описати такими виразами:

$$q_{j\ i+1}= q_{ji}-r_{ji}\le 0,\ r_{j\ i+1}=r_{ji}+2, \text{ якщо } q_{ji}-r_{ji}>0; \qquad (2.37)$$

$$q_{j+1\ 1}= q_{ji}+2m_j+1-r_{ji}\le 0,\ r_{j\ i+1}=r_{ji}+2, \text{ якщо } q_{ji}-r_{ji}<0. \qquad (2.38)$$

При виконанні умови $q_{ji}-r_{ji}=0$ визначаються шукані величини $A=(m+j)$ і $B=\sqrt{(m+j)^2-n}$, яке буде натуральним числом. Слід зазначити, що в цьому випадку значення параметра j відповідає кількості кроків у класичному та удосконаленому методах Ферма. В табл. 2.29 наведено відповідний приклад для факторизації $n=53\cdot 89=4717$ ($m_1=\left\lfloor\sqrt{4717}\right\rfloor=68$) [317].

Таблиця 2.29 – Приклад факторизації числа 4717 запропонованим методом

i	j=1, m_1=68		j=2, m_2=69		j=3, m_3=70	
	q_{1i}	r_{1i}	q_{2i}	r_{2i}	q_{3i}	r_{3i}
1	69^2-4717=44	1	8+139-13=134	15	14+141-27=128	29
2	44-1=43	3	134-15=119	17	128-29=99	31
3	43-3=40	5	119-17=102	19	99-31=68	33
4	40-5=35	7	102-19=83	21	68-33=35	35
5	35-7=28	9	83-21=62	23	35-35=0	
6	28-9=19	11	62-23=39	25		
7	19-11=8<13	13	39-25=14<27	27		

Отже, розклад числа 4717 на прості множники здійснюється таким чином: $B=\sqrt{(68+3)^2-4717}=18$, $4717=71^2-18^2=(71+18)(71-18)=89\cdot 53$. Дана процедура виконана без використання обчислювально громіздкої операції добування квадратного кореня. Крім того, додавання та віднімання виконуються над числами меншої розрядності, ніж у двох попередніх методах, хоча кількість цих операцій є більшою.

Висновки до другого розділу

1. Розроблено методи пошуку залишку, модулярного множення, модулярного експоненціювання та пошуку найбільшого спільного дільника з використанням матричних та векторно-модульних перетворень у розмежованій системі числення залишкових класів, який дає можливість замінити операції піднесення до степеня та множення на, відповідно, множення та додавання малорозрядних залишків, що дозволяє зменшити обчислювальну складність при виконанні цих операцій.

2. Запропоновано методи пошуку оберненого елемента за модулем на основі додавання модуля та додавання залишку, який не містить громіздких операцій ділення з остачею та піднесення до степеня, дозволяє виконувати розпаралелення процесу обчислень та зменшити обчислювальну складність даної операції при застосуванні в асиметричних криптосистемах.

3. Здійснено реалізацію пошуку найбільшого спільного дільника, криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу модулярного множення та модулярного експоненціювання, уникнувши виконання обчислювально громіздких операцій модулярного піднесення до степеня, множення та ділення з остачею.

4. Запропоновано метод реалізації КТЗ на основі додавання добутку модулів, який дозволяє розпаралелити процес обчислень, уникнувши при цьому виконання операції ділення з остачею, виконувати дії над числами меншої розрядності в порівнянні з класичним методом та алгоритмом Гарнера, а також зменшити кількість переходів через модуль при переведенні чисел із СЗК в позиційну систему числення.

5. Розроблено метод пошуку мультистепеневої функції за модулем, який, базуючись на основі двократного використання функції Ейлера, заміни змінних та переходу до лінійної конгруенції, дозволяє зменшити обчислювальну складність за рахунок уникнення операції піднесення багаторозрядних чисел до степеня за модулем та виконання арифметичних дій над операндами, меншими від заданого модуля.

6. Удосконалено метод Ферма для факторизації багаторозрядних чисел на основі використання на кожній ітерації тільки операцій додавання та віднімання, що, в порівнянні з класичним методом Ферма, де потрібно виконувати громіздку операцію пошуку квадратного кореня, забезпечує зменшення розрядності операндів та спрощення алгоритму факторизації..

Основні результати другого розділу відображені у роботах [37-43, 296, 297, 299-304, 306-317].

3 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

У третьому розділі розроблені теоретичні основи для аналітичного пошуку коефіцієнтів базисних чисел та методи побудови ДФ СЗК на основі дробових перетворень та факторизації

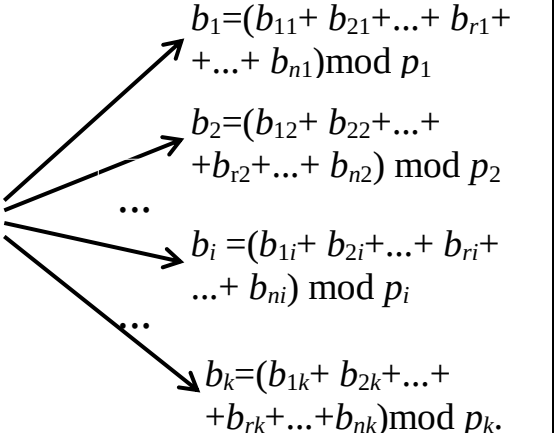
3.1 Перспективи використання різних форм СЗК в асиметричних криптосистемах

Використання різних форм СЗК також дозволяє підвищити продуктивність обчислювальних систем при зменшенні їх апаратної складності. У роботах [122, 123, 125, 149, 160, 295, 298] запропоновано та обґрунтовано використання чотирьох аналітичних моделей прямих та зворотніх перетворень СЗК (табл. 3.1), включаючи описану вище цілочисельну форму.

У табл. 3.1 використано такі позначення: K_k – число у позиційній (двійковій) системі числення; (b_1, b_2, …, b_i, …, b_k) – представлення числа в СЗК; (p_1, p_2, …, p_i, …, p_k) - набір натуральних взаємно простих модулів СЗК; b_i - найменший невід'ємний залишок; P – діапазон кодування чисел в СЗК; a_i – ранг; K – число модулів СЗК; B_i – базисні числа СЗК; res – символ операції пошуку найменшого невід'ємного залишку; int – символ операції виділення цілої частини дробового числа; mod - символ операції пошуку залишку по модулю; m_i – ранговий коефіцієнт СЗК; δp – дробова частина в нормалізованій формі СЗК; $[K_k]_0$, $[b_i]_0$ – відповідно позначення числа та залишку в нормалізованій формі СЗК.

Недоліком звичайної цілочисельної форми СЗК є виникнення значних труднощів при виконанні простої арифметичної операції порівняння двох чисел, що суттєво ускладнює програмну та апаратну реалізацію алгоритмів та відповідних процесів ділення.

Таблиця 3.1 – Аналітичні моделі прямих і зворотних перетворень залишкових класів

№	Пряме перетворення форми СЗК	Зворотнє перетворення форми СЗК
1.	Цілочисельна форма СЗК	
	$N_k = (b_1 b_2 ... b_i ... b_k)_{(p_1 p_2 ... p_i ... p_k)}$ $N_k = b_i (\mathrm{mod}\, p_i)$, $N_k = a_i p_i + b_i$, $P = \prod_{i=1}^{k} p_i$; 0≤K_k≤P.	$b_i = res N_k (\mathrm{mod}\, p_i)$; $N_k = res \sum_{i=1}^{k} b_i \cdot B_i (\mathrm{mod} P)$, $B_i = \frac{P}{p_i} \cdot m_i \equiv 1 (\mathrm{mod}\, P_i) \cdot$
2.	Нормалізована форма СЗК	
	$\frac{N_k}{P} = res \sum_{i=1}^{k} \frac{b_i \cdot B_i (\mathrm{mod} P)}{P}$, $[N_k]_0 = res \sum_{i=1}^{k} b_i \cdot \frac{B_i}{P} (\mathrm{mod}1)$, $0 \le [N_k]_0 \le P-1$; $\frac{B_i}{P} = \frac{1}{p_i}$, $\delta_p \le \frac{1}{P}$, $\frac{1}{p_i} = 0.\overbrace{gggggggg}^{n_i}\overbrace{}^{\delta_p}$,	$[N_k]_0 = res \sum_{i=1}^{k} b_i \cdot \frac{m_i}{p_i} (\mathrm{mod}1)$ $[N_k]_0 = res \sum_{i=1}^{k} [b_i]_0 \cdot m_i (\mathrm{mod}1)$, $[b_i]_0 = \frac{b_i}{p_i}$, $0 \le [b_i]_0 \le 1$. $N_k = \mathrm{int}[N_k]_0 \cdot P$.
3.	Розмежована форма СЗК	
	$N_k = N_{1k} + N_{2k} + ... + N_{ik} +$ $+ ... + N_{nk}$.	$N_k =$ 
4.	Досконала форма СЗК	
	$[N_k]_0 = res \sum_{i=1}^{k} [b_i]_0 (\mathrm{mod}1)$.	$b_i = \mathrm{int}\, res [N_k]_0 (\mathrm{mod}1) \cdot P_i$

Крім того, необхідність пошуку мультиплікативного оберненого елемента при переведенні чисел із СЗК в ПСЧ викликає труднощі для застосування цілочисельної СЗК в асиметричних криптосистемах.

В той же час, переваги однотактної матричної реалізації інших арифметичних операцій забезпечують широкі перспективи для застосування теоретичних основ цілочисельного перетворення СЗК для створення і широкомасштабного впровадження супершвидкісних процесорів у комп'ютерних системах [139, 140].

Перевагою нормалізованої СЗК є спрощення реалізації процесорів за рахунок виключення нелінійних операцій пошуку залишку по кожному із модулів, та заміни операції знаходження залишку mod P на операцію mod 1, яка виконується шляхом простого відкидання цілої частини результату згідно із операцією int. Однак використання нормалізованої СЗК пов'язане із заокругленням під час ділення на P, що ускладнює її використання в асиметричних криптосистемах.

Математичні операції над числами в розмежованій СЗК можуть бути розподілені по кожному з фрагментів процесора, що забезпечує глибший рівень розпаралелювання обробки інформації, і, відповідно, підвищення швидкодії процесора СЗК. Реалізований такий процесор може бути з суттєвим зменшенням апаратних засобів за кожним із модулів [166, 318]. Дана форма успішно може використовуватися в асиметричних криптосистемах, зокрема, при реалізації алгоритму Евкліда, модулярному множенні та експоненціюванні.

Очевидно, що наявність коефіцієнтів $m_i = P_i^{-1} \bmod p_i$ у цілочисельній формі СЗК (формула (1.36)) ускладнює реалізацію відповідного алгоритму. Дослідження різних наборів модулів p_i, яким відповідають коефіцієнти m_i у теоретико-числовому аспекті показали, що існують такі набори модулів p_1, p_2, ..., p_k, яким відповідають одиничні коефіцієнти m_i (m_1=m_2= ... =m_i = ... = =m_k=1) (наприклад, 2, 3, 5) або

$$P_i \bmod p_i = 1. \qquad (3.1)$$

Така форма СЗК була названа ДФ СЗК. На даний час вона є особливо перспективною для застосування в сучасних асиметричних криптосистемах. Пошук наборів модулів, що породжують ДФ СЗК, є окремою актуальною задачею.

В серії робіт [280-287] були розглянуті теоретичні основи побудови ДФ СЗК, яка дозволяє уникнути виконання громіздкої операції пошуку оберненого елемента за модулем та множення на нього згідно КТЗ. Однак її недоліком є те, що модулі ДФ СЗК дуже швидко зростають, що неприпустимо у випадку необхідності вибору модулів однакової розрядності, зокрема, в задачах завадостійкого кодування.

В роботах [288-293] була розроблена МДФ СЗК, у якій виконується умова

$$P_i \bmod p_i = 1,\ p_i\text{-}1 \ \text{ або } \ P_i \bmod p_i = \pm 1, \qquad (3.2)$$

тобто m_i=1 або p_i-1 (m_i=±1). Це дозволяє усунути недолік ДФ СЗК, а також зменшує ймовірність перевищення діапазону обчислень в (1.8).

Для прикладу розглянемо модулярне експоненціювання $a^x \bmod p$, розклавши основу степеня на залишки з використанням трьохмодулної цілочисельної та МДФ СЗК. Діапазон обчислень $P = p_1 \cdot p_2 \cdot p_3 > p$, вважаємо, що розрядність модулів p_1, p_2, p_3 приблизно рівна третині розрядності p, тобто $(n_0/3)$, де n_0 – розрядність модуля p. Після пошуку залишків $a \bmod p_1 = \alpha_1$, $a \bmod p_2 = \alpha_2$, $a \bmod p_3 = \alpha_3$ обчислюються значення $\alpha_1^x \bmod p_1$, $\alpha_2^x \bmod p_2$, $\alpha_3^x \bmod p_3$, з яких на основі КТЗ методом виділення квадратів визначається результат модулярного експоненціювання.

Основними операціями при використанні цілочисельної СЗК є знаходження залишків багаторозрядних чисел, пошук оберненого елемента

за модулем і піднесення до квадрату за модулем. Тому загальна часова складність набуває такого вигляду:

$$O\left(\left(\log_2 3\cdot(2\cdot\log_2{}^2\frac{n_0}{3}+\frac{n_0}{3}\right)+\frac{n_0^2\cdot 3}{2}+\left(\log_2 n_0/2\right)+\frac{3n_0^2}{2}\left(\log_2 n_0/6\right)\right).$$

Якщо вибрані модулі утворюють МДФ СЗК, то усувається операція пошуку оберненого елемента і часова складність буде така:

$$O1\left(\left(\log_2 3\cdot(2\cdot\log_2{}^2\frac{n_0}{3}+\frac{n_0}{3}\right)+\left(\log_2 n_0/2\right)+\frac{3n_0^2}{2}\left(\log_2 n_0/6\right)\right).$$

3.2 Теоретичні основи аналітичного пошуку коефіцієнтів базисних чисел СЗК

Як зазначалося в п. 1.5, найбільш поширені та зручні для апаратної реалізації набори спеціалізованих модулів мають вигляд 2^u, $2^u\pm1$ (див. таблицю 1.5). Однак при великій кількості модулів пошук оберненого елемента залишається найбільш обчислювально складною задачею при переведенні чисел із СЗК в позиційну систему числення. Для спрощення цієї операції розглянемо набір модулів у такому вигляді [319]:

$$\begin{cases} p_1=2^u-1; \\ p_2=2^u+1; \\ p_3=2^{2u}+1; \\ p_4=2^{4u}+1; \\ \quad\quad .... \\ p_i=2^{u\cdot 2^{i-2}}+1; \\ \quad\quad .... \\ p_{k-1}=2^{u\cdot 2^{k-3}}+1; \\ p_k=2^{u\cdot 2^{k-2}}+1, \end{cases} \tag{3.3}$$

де u – степінь двійки в модулі p_1, k – кількість модулів.

Із системи (3.3) неважко бачити, що кожен наступний модуль на дві одиниці більше від добутку всіх попередніх. Цим визначається взаємна простота модулів, оскільки всі вони є непарними. Крім того, діапазон розглянутих десяткових чисел для можливих розрахунків обмежується виразом $P = 2^{u\cdot 2^{k-1}} - 1$.

Для пошуку оберненого елемента $m_i = P_i^{-1} \bmod p_i$ запишемо систему рівнянь в такому вигляді:

$$\begin{cases} P_1 \bmod (2^u - 1) = (2^u + 1)(2^{2u} + 1)(2^{4u} + 1)\ldots\left(2^{u\cdot 2^{i-2}} + 1\right)\ldots\left(2^{u\cdot 2^{k-3}} + 1\right)\left(2^{u\cdot 2^{k-2}} + 1\right) \bmod (2^u - 1); \\ P_2 \bmod (2^u + 1) = (2^u - 1)(2^{2u} + 1)(2^{4u} + 1)\ldots\left(2^{u\cdot 2^{i-2}} + 1\right)\ldots\left(2^{u\cdot 2^{k-3}} + 1\right)\left(2^{u\cdot 2^{k-2}} + 1\right) \bmod (2^u + 1); \\ P_3 \bmod (2^{2u} + 1) = (2^{2u} - 1)(2^{4u} + 1)\ldots\left(2^{u\cdot 2^{i-2}} + 1\right)\ldots\left(2^{u\cdot 2^{k-3}} + 1\right)\left(2^{u\cdot 2^{k-2}} + 1\right) \bmod (2^{2u} + 1); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_i \bmod \left(2^{u\cdot 2^{i-2}} + 1\right) = \left(2^{u\cdot 2^{i-2}} - 1\right)\ldots\ldots\left(2^{u\cdot 2^{k-3}} + 1\right)\left(2^{u\cdot 2^{k-2}} + 1\right) \bmod \left(2^{u\cdot 2^{i-2}} + 1\right); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_{k-1} \bmod \left(2^{u\cdot 2^{k-3}} + 1\right) = \left(2^{u\cdot 2^{k-3}} - 1\right)\left(2^{u\cdot 2^{k-2}} + 1\right) \bmod \left(2^{u\cdot 2^{k-3}} + 1\right); \\ P_k \bmod \left(2^{u\cdot 2^{k-2}} + 1\right) = \left(2^{u\cdot 2^{k-2}} - 1\right) \bmod \left(2^{u\cdot 2^{k-2}} + 1\right). \end{cases} \quad (3.4)$$

В першому рівнянні (3.4) для кожного множника правої частини отримується залишок 2, тому $P_1 \bmod (2^u - 1) = 2^{k-1} \bmod (2^u - 1)$. В другому рівнянні (3.4) залишок від першого множника дорівнює -2, все інші 2, тому $P_2 \bmod (2^u + 1) = -2^{k-1} \bmod (2^u + 1)$.

У всіх інших рівняннях, аналогічно другому, перший залишок -2, інші 2, причому із збільшенням номера рівняння на 1 кількість множників (відповідно і двійок) зменшується також на 1. В результаті таких обчислень отримаємо систему:

$$\begin{cases} P_1 \bmod\left(2^u-1\right)=2^{k-1}\bmod\left(2^u-1\right); \\ P_2 \bmod\left(2^u+1\right)=-2^{k-1}\bmod\left(2^u+1\right); \\ P_3 \bmod\left(2^{2u}+1\right)=-2^{k-2}\bmod\left(2^{2u}+1\right); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_i \bmod\left(2^{u\cdot 2^{i-2}}+1\right)=-2^{k-(i-1)}\bmod\left(2^{u\cdot 2^{i-2}}+1\right); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_{k-1} \bmod\left(2^{u\cdot 2^{k-3}}+1\right)=-4; \\ P_k \bmod\left(2^{u\cdot 2^{k-2}}+1\right)=-2. \end{cases} \tag{3.5}$$

Тепер шукаємо величини $m_i = P_i^{-1} \bmod p_i$. В першому рівнянні степінь двійки в правій частині запишемо в такому вигляді: $k-1=a_1u+k_1$, де $0 \le k_1 < u$. Тоді $2^{k-1}\bmod\left(2^u-1\right)=\left(2^u\right)^a\cdot 2^{k_1}\bmod\left(2^u-1\right)=2^{k_1}\bmod\left(2^u-1\right)$. Звідси видно, що

$$m_1=\frac{2^u}{2^{k_1}}=2^{u-k_1}. \tag{3.6}$$

Аналогічно з другого рівняння: $k-1=a_2u+k_2$. Тоді:

$$-2^{k-1}\bmod\left(2^u+1\right)=-\left(2^u\right)^{a_2}\cdot 2^{k_2}\bmod\left(2^u+1\right)=-(-1)^{a_2}\cdot 2^{k_2}\bmod\left(2^u+1\right)=$$
$$=(-1)^{a_2+1}\cdot 2^{k_2}\bmod\left(2^u+1\right)=\begin{cases} 2^{k_2}\bmod\left(2^u+1\right), & a_2-\text{непарне} \\ -2^{k_2}\bmod\left(2^u+1\right), & a_2-\text{парне}. \end{cases}$$

Розглянемо два можливих випадки:

а) a_2 – непарне; до 2^u+2 потрібно додати модуль 2^u+1 стільки разів, щоб сума ділилась на 2^{k_2} (тобто другий доданок дорівнював 2^{k_2}). Отже,

$$2^u+2+\left(2^u+1\right)\left(2^{k_2}-2\right)=2^u+2+2^{u+k_2}+2^{k_2}-2\cdot 2^u-2=2^{u+k_2}-2^u+2^{k_2}.$$

Поділивши на 2^{k_2}, отримаємо обернений елемент: $m_2 = 2^u - 2^{u-k_2} + 1$;

б) a_2 - парне; отримане значення m_2 потрібно записати з протилежним знаком і додати модуль: $m_2 = -\left(2^u - 2^{u-k_2} + 1\right) \bmod \left(2^u + 1\right) = 2^{u-k_2}$.

Отже, остаточна формула матиме вигляд:

$$m_2 = \begin{cases} 2^u - 2^{u-k_2} + 1, \ a_2 - \text{непарне}; \\ 2^{u-k_2}, \ a_2 - \text{парне}. \end{cases} \tag{3.7}$$

Аналогічно з третього рівняння:

$$m_3 = \begin{cases} 2^{2u} - 2^{2u-k_3} + 1, \ a_3 - \text{непарне}; \\ 2^{2u-k_3}, \ a_3 - \text{парне}, \end{cases} \tag{3.8}$$

де k_3 і a_3 визначаються з рівності $k - 2 = 2ua_3 + k_3$.

З i-го рівняння:

$$m_i = \begin{cases} 2^{u \cdot 2^{i-2}} - 2^{u \cdot 2^{i-2} - k_i} + 1, \ a_i - \text{непарне}; \\ 2^{u \cdot 2^{i-2} - k_i}, \ a_i - \text{парне}, \end{cases} \tag{3.9}$$

де k_i та a_i визначаються з рівності $k - (i-1) = 2^{i-2}ua_i + k_i$.

Розглянемо (k-1) -е рівняння. Немає необхідності розписувати k_{i-1}, оскільки $2^2 < 2^{u \cdot 2^{k-3}}$. До $2^{u \cdot 2^{k-3}} + 2$ двічі потрібно додати модуль і поділити на -4. В результаті отримаємо:

$$m_{k-1} = 2^{u \cdot 2^{k-3} - 2}. \tag{3.10}$$

Аналогічно для останнього, k-го рівняння $\left(2^{u\cdot 2^{k-2}}+2\right)\Big/(-2)=-\left(2^{u\cdot 2^{k-2}-1}+1\right)$. Додавши модуль, отримаємо:

$$m_k = 2^{u\cdot 2^{k-2}-1}. \qquad (3.11)$$

В табл. 3.2 наведені значення p_i, P_i, m_i, а також діапазон можливих розрахунків для k=4 та різних значень u.

З таблиці видно, що величини P_1, m_1, P_2, m_2 при малих u можуть набувати різних значень, що залежить від парності або непарності коефіцієнта a_i. Інші значення P_i, m_i мають вигляд відповідного степеня двійки.

Таблиця 3.2 - Значення p_i, P_i, m_i, а також діапазон можливих розрахунків для k=4 та різних значень u

u	p_1	P_1	m_1	p_2	P_2	m_2	p_3	P_3	m_3	p_4	P_4	m_4	P
2	2^2-1	1	1	2^2+1	4	4	2^4+1	2^4-7	2	2^8+1	2^8-3	2^6	2^{32}-1
3	2^3-1	2	4	2^3+1	2	5	2^6+1	2^6-7	2^3	2^{12}+1	2^{12}-3	2^{10}	2^{48}-1
4	2^4-1	1	1	2^4+1	1	1	2^8+1	2^8-7	2^5	2^{16}+1	2^{16}-3	2^{14}	2^{64}-1
5	2^5-1	16	2	2^5+1	2^5-15	2	2^{10}+1	2^{10}-7	2^7	2^{20}+1	2^{20}-3	2^{18}	2^{80}-1
6	2^6-1	16	2^2	2^6+1	2^6-15	2^2	2^{12}+1	2^{12}-7	2^9	2^{24}+1	2^{24}-3	2^{22}	2^{96}-1
…	…	…	…	…	…	…	…	…	…	…	…	…	…
i	2^i-1	16	2^{i-4}	2^i+1	2^i-15	2^{i-4}	2^{2i}+1	2^{2i}-7	2^{2i-3}	2^{4i}+1	2^{4i}-3	2^{4i-2}	2^{16i}-1

На рис. 3.1 показано графік логарифмічної залежності степеня u для модуля p_1 від кількості модулів k для 512-розрядного процесора згідно виразу $u=2^{10-k}$.

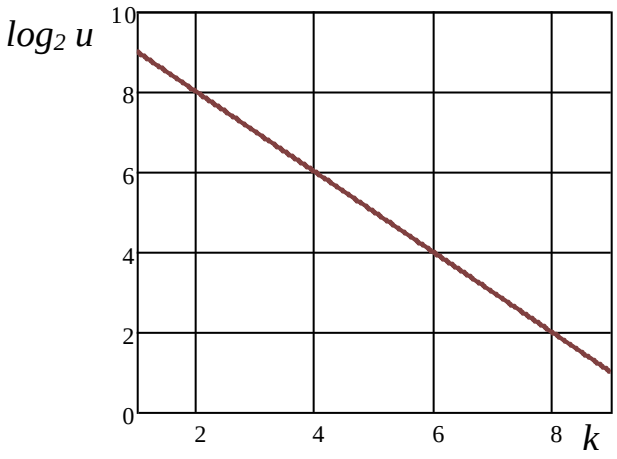


Рис. 3.1 - Графік логарифмічної залежності степеня u для модуля p_1 від кількості модулів k

З даного рисунка видно, что $\log_2 u$ лінійно спадає із збільшенням кількості модулів k.

3.3 Метод побудови ДФ СЗК на основі дробових перетворень

Для побудови набору модулів ДФ СЗК запишемо вираз (3.1) у вигляді такої системи [280-287]:

$$\begin{cases} P_1 \bmod p_1 = 1 \\ \ldots \\ P_k \bmod p_k = 1. \end{cases} \tag{3.12}$$

Домноживши кожне рівняння на відповідний модуль, отримаємо:

$$\begin{cases} P \bmod p_1^2 = p_1 \\ \ldots \\ P \bmod p_k^2 = p_k. \end{cases} \tag{3.13}$$

Розв’язуючи (3.13) стандартними методами теорії чисел згідно КТЗ, матимемо:

$$P = \left(\sum_{i=1}^{k} p_i P_i^2 m_i^2 \right) \bmod M , \qquad (3.14)$$

де $M = \prod_{i=1}^{k} p_i^2 = P^2$.

Врахувавши, що у ДФ СЗК коефіцінти m_i=1, та скоротивши модуль, ліву та праву частину (3.14) на їх спільний дільник $P = \prod_{i=1}^{k} p_i$, запишемо (3.14) таким чином:

$$\left(\sum_{i=1}^{k} P_i \right) \bmod P = 1. \qquad (3.15)$$

Вираз (3.15) еквівалентний рівності:

$$\sum_{i=1}^{k} P_i = \gamma P + 1, \qquad (3.16)$$

де γ=1, 2, 3, ...

Поділивши ліву та праву частини (3.16) на P, отримаємо остаточний вираз для пошуку набору модулів у ДФ СЗК:

$$\sum_{i=1}^{k} \frac{1}{p_i} = \gamma + \frac{1}{\prod_{i=1}^{k} p_i}. \qquad (3.17)$$

Розглянемо систему з трьох модулів. Представимо модулі p_2, p_3 у такому вигляді: $p_2=ap_1+b$, $p_3=cp_1p_2+d=cp_1(ap_1+b)+d$, де a і c – натуральні, b і d – цілі числа, причому $|b|<p_1$, $|d|<p_2$ [287]. Тоді з (3.12) отримуємо:

$$\begin{cases}(((ap_1+b)(cp_1(ap_1+b)+d))\bmod p_1=1\\(p_1(cp_1(ap_1+b)+d))\bmod(ap_1+b)=1\\(p_1(ap_1+b))\bmod(cp_1(ap_1+b)+d)=1.\end{cases} \quad (3.18)$$

З третього рівняння системи (3.18) випливає, що добуток $p_1(ap_1+b)$ повинен або дорівнювати 1, або бути на одиницю більший від значення модуля $cp_1(ap_1+b)+d$. Звідси слідує, що c=1, d=–1. З другого рівняння бачимо, що $(p_1(ap_1+b)-1)\bmod(ap_1+b)=-1$, тому повинна виконуватися умова $p_1\bmod(ap_1+b)=-1$. Це можливо тільки у випадку, коли a=1, b=1. Тоді система (3.18) набуде такого вигляду:

$$\begin{cases}(((p_1+1)(p_1(p_1+1)-1))\bmod p_1=1\\(p_1(p_1(p_1+1)-1))\bmod(p_1+1)=1\\(p_1(p_1+1))\bmod(p_1(p_1+1)-1)=1.\end{cases} \quad (3.19)$$

З першого рівняння системи (3.19) видно, що $(p_1(p_1+1)-1)\bmod p_1=-1$. Це приводить до умови (p_1+1) mod p_1= –1. Такий випадок можливий тільки у тому випадку, коли p_1=2. Дане число є унікальним для ДФ СЗК, оскільки –1 mod 2=1.

Це підтверджує, що єдино можливим набором для трьох модулів ДФ СЗК можуть бути тільки числа 2, 3, 5.

Дослідження рівняння (3.17) для великої кількості модулів, враховуючи, що сума ряду $\sum_{i=1}^{k}\frac{1}{p_i}$ розбіжна, тобто γ може бути як завгодно великим, є досить громіздкою задачею. Вона значно спрощується, коли

покласти p_1=2, p_2=3, оскільки в цьому випадку, згідно рівності (3.12), для взаємно простих модулів $P_1 \bmod 2 = 1,\ P_2 \bmod 3 = \pm 1$. Крім того, будь–який модуль можна представити у вигляді $p_i = 6t \pm 1$, де t – натуральне число. Поклавши γ=1, перепишемо (3.17) у такому вигляді [281]:

$$\sum_{i=3}^{k} \frac{1}{p_i} = \frac{1}{6} + \frac{1}{\prod_{i=3}^{k} p_i}. \tag{3.20}$$

Модуль p_3 виберемо так, щоб при відніманні $\frac{1}{p_3}$ в правій частині (3.20) в чисельнику отримати 1. Видно, що p_3=7.Тоді маємо:

$$\sum_{i=4}^{k} \frac{1}{p_i} = \frac{1}{42} + \frac{1}{\prod_{i=4}^{k} p_i}. \tag{3.21}$$

Аналогічно звідси випливає, що p_4=43:

$$\sum_{i=5}^{k} \frac{1}{p_i} = \frac{1}{1806} + \frac{1}{\prod_{i=5}^{k} p_i}. \tag{3.22}$$

Для останнього модуля p_k справедлива рівність:

$$\frac{1}{p_k} = \frac{1}{\prod_{i=1}^{k-1} p_i} + \frac{1}{p_k \cdot \prod_{i=1}^{k-1} p_i}. \tag{3.23}$$

Звідси отримуємо, що

$$p_k = \prod_{i=1}^{k-1} p_i - 1. \qquad (3.24)$$

На основі цього випливає закономірність побудови системи модулів ДФ СЗК:

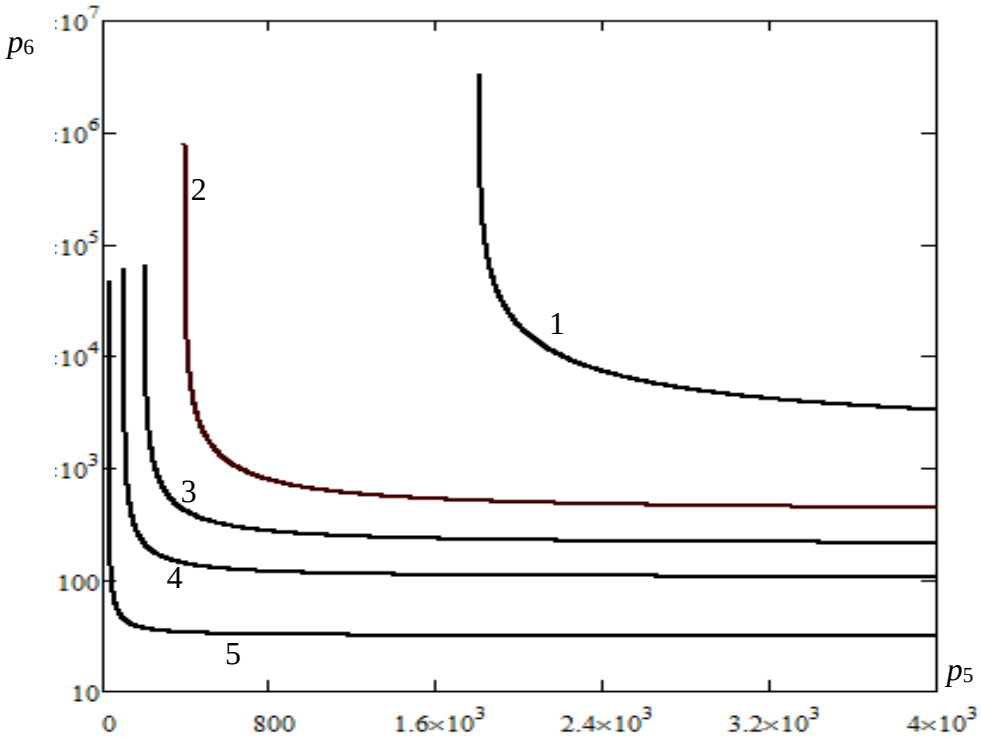
$$\begin{cases} p_1 = 2 \\ p_i = p_1 p_2 \ldots p_{i-1} + 1, \ 1 < i < k \\ p_k = p_1 p_2 \ldots p_{k-1} - 1. \end{cases} \qquad (3.25)$$

На рис. 3.2а) відповідно до (3.20) показано графіки залежності модуля p_6 від p_5, для яких потрібно вибрати цілочисельні значення, при різних p_3 та p_4. На рис. 3.2б) представлені відповідні їм діапазони, в межах яких можна виконувати арифметичні операції над десятковими числами.

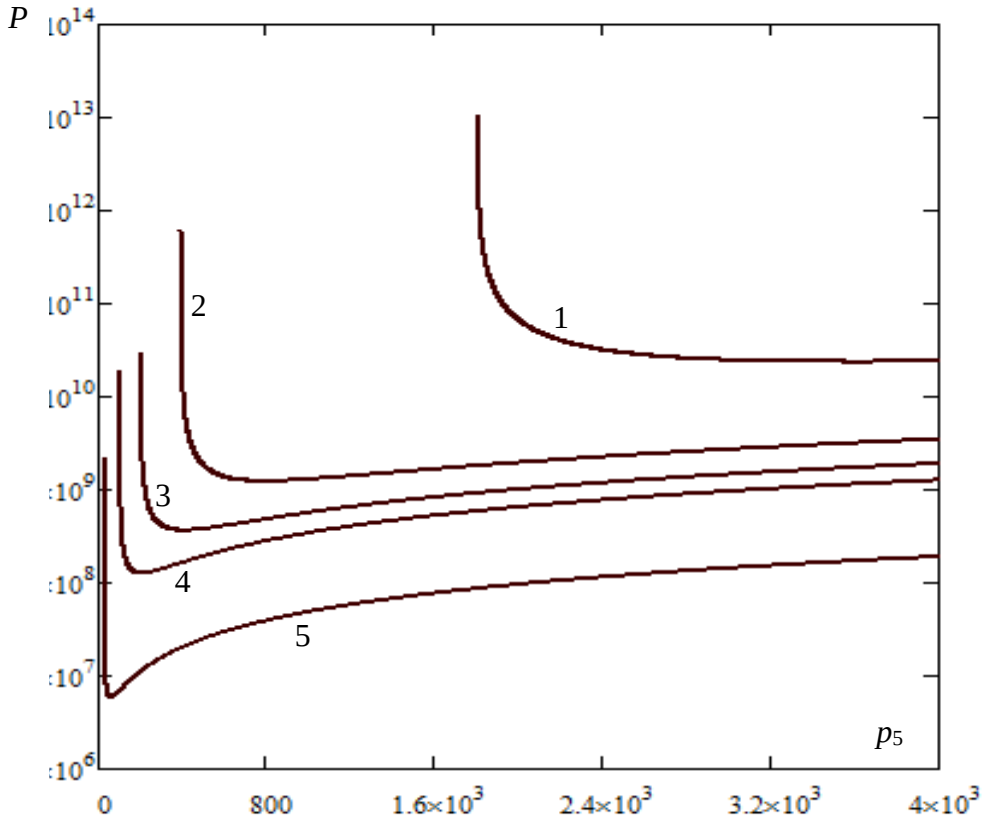
З графіків видно, що модуль p_6 різко зменшується, після чого набуває практично постійного значення. Діапазон обчислень при певному значенні p_5 має характерний мінімум, глибина якого збільшується із зменшенням самого діапазону обчислень.

3.4 Узагальнення методу дробових перетворень

Узагальнюючи вираз (3.25) і враховуючи, що для зменшення складності обчислень якомога більша кількість коефіцієнтів m_i має дорівнювати 1, представимо вибраний набір модулів у такому вигляді [284, 286]:



а)



б)

Рис. 3.2 - Графіки залежності p_6 від p_5 а) та відповідні діапазони десяткових чисел б): 1 – p_3=7, p_4=43; 2 – p_3=7, p_4=47; 3 – p_3=7, p_4=53; 4 – p_3=7, p_4=71; 5 – p_3=11, p_4=23

$$\begin{cases} p_1; \\ p_2 = p_1 + 1; \\ \ldots\ldots\ldots. \\ p_i = p_1 \cdot p_2 \ \ldots \ \ p_{i-1} + 1; \\ \ldots\ldots\ldots.. \\ p_k = p_1 \cdot p_2 \ \ldots \ \ p_{k-1} - 1. \end{cases} \tag{3.26}$$

Для знаходження m_i використаємо таку систему рівнянь:

$$\begin{cases} P_k \bmod \ \ p_k = \left(p_1 \cdot p_2 \ \ldots \ \ p_{k-1}\right) \bmod \left(p_1 \cdot p_2 \ \ldots \ \ p_{k-1} - 1\right) = 1 = m_k; \\ P_{k-1} \bmod \ \ p_{k-1} = \left(p_1 \cdot p_2 \ \ldots \ \ p_{k-2} \cdot p_k\right) \bmod \left(p_1 \cdot p_2 \ \ldots \ \ p_{k-2} + 1\right) = \\ = (-1) \cdot (-1) = 1 = m_{k-1}; \\ \ldots\ldots\ldots\ldots\ldots \\ P_i \bmod \ \ p_i = \left(p_1 \cdot p_2 \ \ldots \ \ p_{i-1} \cdot p_{i+1 \ \ldots} \cdot p_k\right) \bmod \left(p_1 \cdot p_2 \ \ldots \ \ p_{i-1} + 1\right) = \\ = (-1) \cdot 1 \cdot 1 \cdot \ldots \cdot (-1) = 1 = m_i; \\ \ldots\ldots\ldots\ldots\ldots \\ P_2 \bmod \ \ p_2 = \left(p_1 \cdot p_3 \ \ldots \ \ \cdot p_k\right) \bmod \left(p_1 + 1\right) = (-1) \cdot 1 \cdot 1 \cdot \ldots \cdot (-1) = 1 = m_2; \\ P_1 \bmod \ \ p_1 = \left(p_2 \cdot p_3 \ \ldots \ \ \cdot p_k\right) \bmod p_1 = 1 \cdot 1 \cdot \ldots \cdot (-1) = -1 = m_1. \end{cases} \tag{3.27}$$

З (3.27) видно, що всі m_i, крім m_1=–1, рівні 1. Якщо вибрати p_1=2, то система (3.27) переходить у (3.25), оскільки $1 \bmod 2 = -1 \bmod 2$.

Слід зазначити, що запропонований у (3.25) метод не вичерпує всіх можливих наборів для ДФ СЗК при заданих k. Наприклад, при k=5 набір модулів, отриманий за допомогою системи (3.25), буде такий: 2, 3, 7, 43, 1805. Однак відомі також набори 2, 3, 7, 83, 85 та 2, 3, 11, 17, 59.

Усі можливі набори модулів для ДФ СЗК при k=6, відповідні їм діапазони десяткових чисел і розрядності в двійковій системі числення (в дужках) наведені в табл. 3.3.

Таблиця 3.3 - Можливі набори модулів при k=6 для ДФ СЗК і відповідні їм діапазони десяткових чисел (в дужках – розрядності в двійковій системі числення)

№	p_1, p_2	p_3	p_4	p_5	p_6	P
1	2, 3 (2)	7 (3)	43 (6)	1807 (11)	3263441 (22)	$1{,}0650050423922\times10^{13}$ (44)
2				1811 (11)	654133 (20)	$2{,}139450562578\times10^{12}$ (41)
3				1819 (11)	252701 (18)	$8{,}30151592914\times10^{11}$ (41)
4				1825 (11)	173471 (18)	$5{,}7175174245\times10^{11}$ (40)
5				1871 (11)	51985 (16)	$1{,}7565866661\times10^{11}$ (38)
6				1901 (11)	36139 (16)	$1{,}24072631634\times10^{11}$ (37)
7				1945 (11)	25271 (15)	$8{,}876868357\times10^{10}$ (37)
8				2053 (12)	15011 (14)	$5{,}5656554898\times10^{10}$ (36)
9				2167 (12)	10841 (14)	$4{,}2427359282\times10^{10}$ (36)
10				2501 (12)	6499 (13)	$2{,}9354722194\times10^{10}$ (35)
11				3041 (12)	4447 (13)	$2{,}4423128562\times10^{10}$ (35)
12				3611 (12)	3613 (12)	$2{,}3562056658\times10^{10}$ (35)
13			47 (6)	395 (9)	779729 (20)	$6{,}0797809317\times10^{10}$ (36)
14				481 (9)	2203 (12)	$2{,}091735282\times10^{9}$ (31)
15			53 (6)	271 (9)	799 (10)	$4{,}81993554\times10^{8}$ (29)
16			71 (7)	103 (7)	61429 (16)	$1{,}8867671634\times10^{10}$ (35)
17		11(4)	23 (5)	31 (5)	47057 (16)	$2{,}214408306\times10^{9}$ (32)

Як видно з таблиці, розрядність чисел, над якими виконуються обчислення, зменшується приблизно в 2-3 рази. Величина P є максимальна при використанні набору модулів, отриманого за допомогою системи (3.25), що дозволяє розглядати найбільший діапазон десяткових чисел. Розрядність чисел при цьому зменшується вдвічі.

3.5 Побудова ДФ СЗК методом факторизації

В загальному випадку при γ=1 вираз (3.17) матиме вигляд:

$$\sum_{i=1}^{k}\frac{1}{p_i}=1+\frac{1}{\prod_{i=1}^{k}p_i}. \tag{3.28}$$

Вважаючи невідомими два останні модулі після відповідних математичних перетворень отримаємо таку формулу:

$$\left(p_{k-1}+p_k\right)\prod_{i=1}^{k-2}p_i+p_{k-1}p_k\left(\sum_{i=1}^{k-2}\frac{P}{p_i}-\prod_{i=1}^{k-2}p_i\right)=1. \tag{3.29}$$

Введемо заміну:

$$p_{k-1},p_k=\frac{a,b-\prod_{i=1}^{k-2}p_i}{\sum_{i=1}^{k-2}\frac{P}{p_i}-\prod_{i=1}^{k-2}p_i}. \tag{3.30}$$

Після підстановки (3.30) в (3.29) отримується умова, яка повинна виконуватися для визначення набору модулів ДФ СЗК:

$$ab=\sum_{i=1}^{k-2}\frac{P}{p_i}-\prod_{i=1}^{k-2}p_i+\left(\prod_{i=1}^{k-2}p_i\right)^2. \tag{3.31}$$

Це означає, що ліва частина (3.31) повинна бути факторизована, на основі чого визначаються параметри a та b. Крім того, як випливає з (3.30), модулі p_k та $p_{k\text{-}1}$ мають бути цілими числами, тобто

$$\left(a,b-\prod_{i=1}^{k-2}p_i\right)\mathrm{mod}\left(\sum_{i=1}^{k-2}\frac{P}{p_i}-\prod_{i=1}^{k-2}p_i\right)=0. \tag{3.32}$$

Отже, вирази (3.31) та (3.32) визначають умови для знаходження будь-якої кількості модулів ДФ СЗК, два з яких невідомі [280, 281].

Нехай k=6. На даний час відомі тільки такі набори модулів ДФ СЗК, в яких p_1=2, p_2=3. Тоді вираз (3.28) перепишеться так:

$$\sum_{i=3}^{6}\frac{1}{p_i}=\frac{1}{6}+\frac{1}{\prod_{i=3}^{6}p_i}. \tag{3.33}$$

Далі з (3.29) та (3.33) маємо:

$$6p_3p_4(p_5+p_6)=(p_4(p_3-6)-6p_3)p_5p_6+1. \tag{3.34}$$

Введемо позначення:

$$p_{5,6}=\frac{6p_3p_4+a,b}{p_4(p_3-6)-6p_3}. \tag{3.35}$$

Після підстановки (3.35) в (3.34), матимемо:

$$(6p_3p_4)^2-(p_4(p_3-6)-6p_3)=ab. \tag{3.36}$$

Ліва частина (3.36) має бути факторизована, на основі чого визначаються параметри a та b. Крім того, як випливає з (3.35), модулі p_5 та p_6 є цілочисельними, тобто:

$$(6p_3p_4 + a, b) \bmod (p_4(p_3 - 6) - 6p_3) = 0. \quad (3.37)$$

Отже, вирази (3.36) та (3.37) визначають умови знаходження будь-якого варіанту набору з шести модулів ДФ СЗК.

3.5.1 Часткові випадки

Перевіривши можливі значення p_3, можна побачити, що даний модуль може дорівнювати 7 або 11. Розглянемо ці випадки детальніше:

1) p_3=7. Вирази (3.36) та (3.37) відповідно трансформуються:

$$(42p_4)^2 - (p_4 - 42) = ab; \quad (3.38)$$

$$(42p_4 + a, b) \bmod (p_4 - 42) = 0. \quad (3.39)$$

Модуль p_4 має бути не менший від 43, оскільки набір 2, 3, 7, 41 утворює ДФ СЗК. Тоді умова (3.39) виконується завжди, а з першої можна отримати:

$$ab = (42 \cdot 43)^2 - 1 = 1805 \cdot 1807 = 5 \cdot 19 \cdot 19 \cdot 13 \cdot 139. \quad (3.40)$$

Використавши всі можливі перестановки множників у (3.40), можна отримати 12 варіантів наборів з 6 модулів ДФ СЗК при заданих модулях 2, 3, 7, 43, які представлені в табл. 3.4.

На рис. 3.3 представлений характер зміни значень модулів p_5 та p_6 в залежності від номера модуля, згідно табл. 3.4, у логарифмічній шкалі. Як видно з рисунка, модуль p_5 зростає повільно. В той же час, графік для p_6 істотно спадає із збільшенням номера модуля.

Таблиця 3.4 - Можливі варіанти наборів з 6 модулів ДФ СЗК при заданих модулях 2, 3, 7, 43

№	a	b	p_5	p_6
1	1	5·19·19·13·139	1807	3263441
2	5	19·19·13·139	1811	654133
3	13	5·19·19·139	1819	252701
4	19	5·19·13·139	1825	173471
5	5·13	19·19·139	1871	51985
6	5·19	19·13·139	1901	36139
7	139	5·19·19·13	1945	25271
8	19·13	5·19·139	2053	15011
9	19·19	5·13·139	2167	10841
10	5·139	19·19·13	2501	6499
11	5·13·19	19·139	3041	4447
12	5·19·19	13·139	3611	3613

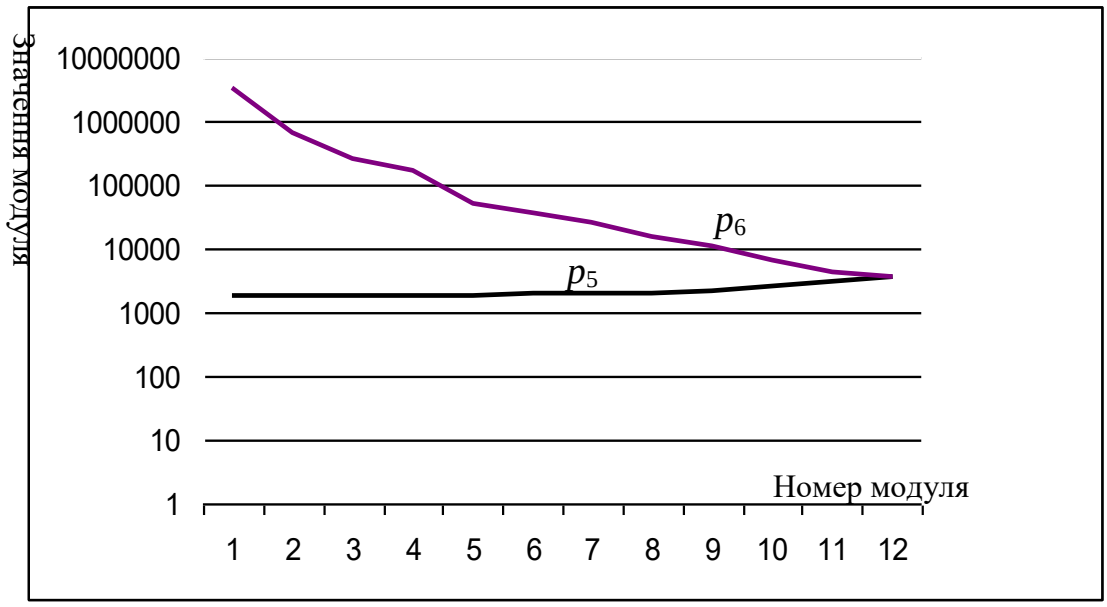


Рис. 3.3 - Характер зміни значень модулів p_5 та p_6 в залежності від номера модуля згідно табл. 3.4

При p_4=47 отримаємо: $p_{5,6} = \frac{1974 + a,b}{5}$, $ab = (42 \cdot 47)^2 - 5 = 9041 \cdot 431$.

Можливі два варіанти, які представлені у табл. 3.5.

Таблиця 3.5 - Можливі варіанти наборів з 6 модулів ДФ СЗК при заданих модулях 2, 3, 7, 47

№	a	b	p_5	p_6
1	1	9041·431	395	779729
2	431	9041	481	2203

При p_4=53 рівності (3.31) та (3.32) набудуть вигляду: $p_{5,6} = \frac{2226 + a,b}{11}$, $ab = (42 \cdot 53)^2 - 11 = 5 \cdot 151 \cdot 6563$. Оскільки 2226 mod 11=4, 5 mod 11=5, 151 mod 11=8, 6563 mod 11=7, то умову (3.32) задовольняє тільки значення a=5·151. Відповідно b=6563 і p_5=271, p_6=799.

Аналогічно можна знайти ще тільки один набір модулів: p_4=71, p_5=103, p_6=61429.

2) p_3=11. Вирази (3.31) та (3.32) набудуть відповідно такого вигляду:

$$(66p_4)^2 - (5p_4 - 66) = ab; \tag{3.41}$$

$$(66p_4 + a,b) \bmod (5p_4 - 66) = 0. \tag{3.42}$$

Умови (3.41) – (3.42) задовольняють такі значення: p_4=23, p_5=31, p_6=47057.

Отже, усі значення елементів табл. 3.3, отримані методом підбору, обчислені за допомогою аналітичних розрахунків.

3.6 Застосування ДФ СЗК у китайській теоремі про залишки

ДФ СЗК успішно може використовуватись в асиметричних криптосистемах, зокрема, у криптосистемі Рабіна, яка грунтується на застосуванні КТЗ, яка зводиться до розв'язання системи конгруенцій (1.7) [280, 281].

Розв'язок даної системи представлений в (1.8). Як уже зазначалося, пошук обернених елементів для коефіцієнтів $m_i = M_i^{-1} \bmod p_i$ становить значну обчислювальну складність. Однак якщо модулі p_1, p_2, … , p_k утворюють ДФ СЗК, тоді можна уникнути цієї громіздкої операції.

Нехай p_1=2, p_2=3, p_3=7, p_4=43, p_5=3611, p_6=3613 і потрібно розв'язати таку систему конгруенцій:

$$\begin{cases} x \bmod 2 = 1 \\ x \bmod 3 = 2 \\ x \bmod 7 = 5 \\ x \bmod 43 = 20 \\ x \bmod 3611 = 100 \\ x \bmod 3613 = 1000. \end{cases} \qquad (3.43)$$

В загальному випадку $x = \left(\sum_{i=1}^{6} r_i P_i m_i \right) \bmod P$, де $m_i = P_i^{-1} \bmod p_i$. В ДФ СЗК $m_i = 1$,. звідси: x=(3·7·43·3611·3613·1 + 2·7·43·3611·3613·2 + +2·3·43·3611·3613·5 +2·3·7·3611·3613·20 + 2·3·7·43·3613·100+ +2·3·7·43·3611·1000) mod 5225745 = (11781028329 + 15708037772 + +16830040470 + 10959096120 + 652507800+ +6521466000) mod 23562056658= =(62452176491) mod 23562056658 = 15328063175.

Отже, шукане значення x отримане за допомогою КТЗ без виконання громіздкої операції пошуку оберненого елемента за модулем, а

використовуючи додавання та множення, методи спрощення яких описані в п. 2.2.

При перетвореннях згідно КТЗ використовуються такі основні модульні операції: пошук оберненого елемента; пошук залишків; операції множення і додавання.

Тому при визначенні обчислювальних складностей відомого і запропонованого методів, які дозволяють виконувати перетворення КТЗ, потрібно врахувати складності вищезазначених операцій, які наведені в табл. 3.6 (f - кількість взаємно простих модулів).

Таблиця 3.6 - Обчислювальні складності основних операцій КТЗ

№	Основні операції	Обчислювальна складність операцій в запропонованому методі	Обчислювальна складність операцій в класичному методі
1.	Пошук оберненого елемента	відсутня	$O1(17{,}5f\cdot((n_0+1)^2+n_0^2+n_0)$
2.	Пошук залишків	$O\left(\log_2 n_0/2\right)$	$O1((n_0+1)^2+n_0)$
3.	Операція множення і додавання	$O\left(\log_2 f\cdot(2\cdot\log_2^2 n_0+n_0\right)$	$O1(f\cdot(2n_0^2+n_0))$

Враховуючи дані табл. 3.6, часова складність КТЗ із використанням запропонованого методу буде становити $O\left(\left(\log_2 f\cdot(2\cdot\log_2^2 n_0+n_0\right)+\left(\log_2 n_0/2\right)\right)\approx O(\log_2 f\cdot(2\cdot\log_2{}^2 n_0+n_0))$, а з використанням класичного методу – $O1(37f\cdot n_0^2+53{,}5f\cdot n_0+17{,}5f+n_0^2+3n_0+1))\approx O1(37f\cdot n_0^2)$.

На рис. 3.4 показано графіки залежності обчислювальних складностей від розрядності чисел n_0 у логарифмічній шкалі.

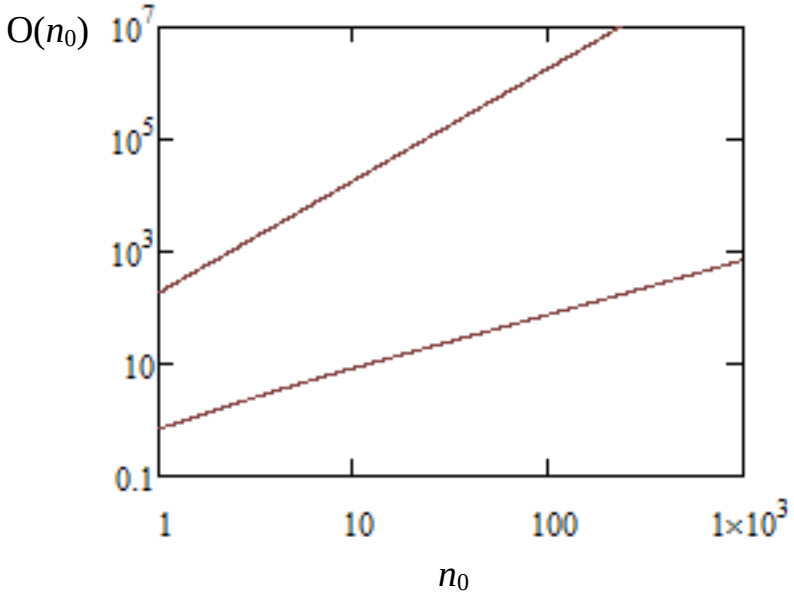


Рис. 3.4 - Графіки залежності обчислювальних складностей від розрядності чисел n_0 запропонованим методом $O(n_0)$ та класичним $O1(n_0)$

З рисунка видно, що використання запропонованого методу, який дозволяє аналітично обчислювати модулі ДФ СЗК та уникати операції пошуку оберненого елемента за модулем, істотно зменшує обчислювальну складність КТЗ відносно класичного.

Висновки до третього розділу

1. Проаналізовано сучасний стан та напрямки підвищення ефективності використання різних форм, зокрема, ДФ та МДФ, СЗК в асиметричних криптосистемах. Показано, що МДФ СЗК дозволяє уникнути операції пошуку оберненого елемента та множення на нього, а також будувати систему модулів однакової розрядності, що важливо для задач завадостійкого кодування.

2. Розроблено загальний метод пошуку модулів СЗК, який дає змогу зменшити обчислювальну складність при переведенні із СЗК у ПСЧ шляхом аналітичного пошуку коефіцієнтів базисних чисел, уникнувши громіздкої операції знаходження оберненого елемента за модулем.

3. Запропоновано метод побудови модулів ДФ СЗК на основі дробових перетворень та факторизації, що дозволяє зменшити обчислювальну складність за рахунок рівності одиниці коефіцієнтів базисних чисел при переведенні із СЗК у ПСЧ.

4. Продемонстровано використання ДФ СЗК у КТЗ, що дозволило зменшити обчислювальну складність КТЗ у порівнянні з класичним методом за рахунок зменшення кількості виконаних операцій.

Основні результати третього розділу відображені у роботах [280-287, 294, 319].

4 МЕТОДИ ПОБУДОВИ ТРЬОХМОДУЛЬНОЇ МОДИФІКОВАНОЇ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

У четвертому розділі розроблені теоретичні основи та методи побудови трьохмодульної МДФ СЗК на основі перемноження модулів, факторизації, використання послідовності Фібоначчі, узагальнення результатів яких дозволяє побудувати багатомодульну МДФ СЗК.

4.1 Метод перемноження модулів

У випадку обмеженої кількості модулів і необхідності розгляду великих чисел, які перевищують діапазон від 0 до $P=\prod_{i=1}^{k} p_i$, згідно (3.2), зручно підібрати такий набір модулів, щоб вони утворювали МДФ СЗК [294, 320]:

$$m_i = P_i^{-1} \bmod p_i = \pm 1. \tag{4.1}$$

Порівняно із ДФ СЗК, це збільшує обчислювальну складність, але вона все одно менша, ніж при пошуку оберненого елемента за модулем.

Запропонований метод дозволяє побудувати систему із двох модулів, що неможливо в ДФ СЗК. Для цього необхідно вибрати будь–які два послідовні числа p_1 і $p_2=p_1+1$, які будуть завжди взаємно простими, і для них завжди виконується умова:

$$\begin{cases} (p_1+1) \bmod p_1 = 1 \\ p_1 \bmod (p_1+1) = -1. \end{cases} \tag{4.2}$$

Для дослідження набору з трьох модулів запишемо систему, аналогічну (3.18) [294]:

$$\begin{cases}\left(\left(ap_1+b\right)\left(cp_1\left(ap_1+b\right)+d\right)\right)\bmod p_1=\pm 1\\ \left(p_1\left(cp_1\left(ap_1+b\right)+d\right)\right)\bmod\left(ap_1+b\right)=\pm 1\\ \left(p_1\left(ap_1+b\right)\right)\bmod\left(cp_1\left(ap_1+b\right)+d\right)=\pm 1.\end{cases} \quad (4.3)$$

Аналізуючи систему (4.3), подібно до дослідження (3.1), і прийнявши, що $|p_1|<|p_2|<|p_3|$, отримаємо: $a=c=b=1$, $d=\pm 1$. В цьому випадку ліва частина останньої системи спрощується, а права залежатиме від знаку біля коефіцієнта d. Легко бачити, що

$$\begin{cases}\left(\left(p_1+1\right)\left(p_1\left(p_1+1\right)\pm 1\right)\right)\bmod p_1=\mp 1\\ \left(p_1\left(p_1\left(p_1+1\right)\pm 1\right)\right)\bmod\left(p_1+1\right)=\mp 1\\ \left(p_1\left(p_1+1\right)\right)\bmod\left(p_1\left(p_1+1\right)\pm 1\right)=\pm 1.\end{cases} \quad (4.4)$$

Отже, якщо коефіцієнт $d=1$, то $m_1=m_2=-1$, $m_3=1$. При $d=-1$ маємо: $m_1=m_2=-1$, $m_3=1$. Знову наголосимо, що оскільки $-1 \bmod 2 = 1$, то отримаємо єдино можливий набір з трьох модулів для ДФ СЗК: $p_1=2$, $p_2=3$, $p_3=5$.

Розглянемо ряд прикладів:

1) нехай модулі $p_1=10$, $p_2=11$, $A=83<P=110$. Запишемо 83 у СЗК: $(83)_{10}=(3,\ 6)_{10,\ 11}$. Тоді згідно виразу (4.2), зворотнє перетворення із СЗК у десяткову систему числення матиме такий вигляд: $(-6\cdot 10 + 3\cdot 11) \bmod 110 = -27 \bmod 110 = 83$;

2) нехай $p_1=5$, $p_2=6$, $p_3=31$, $A = 802 < P = 930$. Запишемо 802 у СЗК: $(802)_{10}=(2,\ 4,\ 27)_{5,\ 6,\ 31}$. Для зворотного перетворення класичним методом потрібна така послідовність дій: $m_1=(6\cdot 31)^{-1} \bmod 5=1$; $m_2=(5\cdot 31)^{-1} \bmod 6=5$; $m_3=(5\cdot 6)^{-1} \bmod 31=30$; $A=(1\cdot 6\cdot 31\cdot 2+5\cdot 5\cdot 31\cdot 4+30\cdot 5\cdot 6\cdot 27) \bmod 930 = 27772 \bmod 930 = 802$. При використанні МДФ СЗК вказані обчислення

значно спрощуються: $m_1=(6\cdot31)^{-1}\text{mod } 5=1$; $m_2=(5\cdot31)^{-1}\text{mod } 6=-1 \text{ mod } 6$; $m_3=(5\cdot6)^{-1}\text{mod } 31=-1 \text{ mod } 31$; $A=(1\cdot6\cdot31\cdot2-1\cdot5\cdot31\cdot4-1\cdot5\cdot6\cdot27) \text{ mod } 930 = =-1058 \text{ mod } 930 =802$;

3) нехай $p_1=5$, $p_2=6$, $p_3=29$, $A = 802 < P = 870$. Запишемо 802 у СЗК: $(802)_{10}=(2, 4, 19)_{5, 6, 29}$. Для зворотного перетворення класичним методом потрібна така послідовність дій: $m_1=(6\cdot29)^{-1}\text{mod } 5=4$; $m_2=(5\cdot29)^{-1}\text{mod } 6=1$; $m_3=(5\cdot6)^{-1}\text{mod } 29=1$; $A=(4\cdot6\cdot29\cdot2+1\cdot5\cdot29\cdot4+1\cdot5\cdot6\cdot19) \text{ mod } 870 = 2542 \text{ mod } 870 = =802$. При використанні МДФ СЗК обчислення знову ж спрощуються: $m_1=(6\cdot29)^{-1}\text{mod } 5=-1 \text{ mod } 5$; $m_2=(5\cdot29)^{-1}\text{mod } 6=1$; $m_3=(5\cdot6)^{-1}\text{mod } 29=1$; $A=(-1\cdot6\cdot29\cdot2+1\cdot5\cdot29\cdot4+1\cdot5\cdot6\cdot19) \text{ mod } 930 = 802$.

Видно, що при використанні МДФ СЗК усувається необхідність пошуку оберненого елемента та множення на m_i при відновленні десяткового числа A.

Система (4.4) дозволяє записати загальну формулу для визначення різноманітних наборів будь–якої кількості модулів, у яких коефіцієнти $m_i=\pm1$. Вважаючи p_1 найменшим у наборі модулів, можна отримати [281]:

$$\begin{cases} p_2 = p_1 + 1 \\ p_i = p_1 p_2 \ldots p_{i-1} \pm 1, \end{cases} \tag{4.5}$$

де $i = 3, \ldots, k$.

Зауважимо, що умові $m_i=1$ відповідають додатні значення модулів p_i, а умові $m_i=-1$ – від’ємні.

4.2 Побудова трьохмодульної МДФ СЗК на основі факторизації

Для демонстрації запропонованого методу обмежимо наші розрахунки трьома модулями і запишемо вираз (4.1) у вигляді системи [291]:

$$\begin{cases} p_2 p_3 \bmod p_1 = \pm 1 \\ p_1 p_3 \bmod p_2 = \pm 1 \\ p_1 p_2 \bmod p_3 = \pm 1. \end{cases} \quad (4.6)$$

Обчислення, аналогічні розрахункам, проведеним в п. 3.3, приводять до такого рівняння:

$$\sum_{i=1}^{3} \frac{1}{p_i} = \gamma \pm \frac{1}{\prod_{i=1}^{3} p_i}, \quad (4.7)$$

де γ=±1, ±2, ±3, ...

На відміну від ДФ СЗК, коефіцієнт γ можна вибрати рівним 0, що при заданій кількості модулів відповідає найбільшому значенню P. Тоді остання рівність перепишеться у такому вигляді:

$$\frac{1}{p_1} + \frac{1}{p_2} + \frac{1}{p_3} = \pm \frac{1}{p_1 p_2 p_3}. \quad (4.8)$$

У ДФ СЗК, у якій найменші модулі набувають строго визначених значень (p_1=2, p_2=3), а у МДФ СЗК найменші модулі можуть бути будь-які. Домноживши (4.8) на P, отримаємо:

$$p_1 p_2 + p_2 p_3 + p_1 p_3 = \pm 1. \quad (4.9)$$

Представивши (4.9) у вигляді

$$p_2 p_3 + p_1 (p_2 + p_3) = \pm 1, \quad (4.10)$$

введемо позначення:

$$p_{2,3} = a,b - p_1. \qquad (4.11)$$

Після підстановки (4.11) в (4.10) та відповідних математичних перетворень будемо мати умову, яка повинна виконуватися для визначення набору з трьох модулів для МДФ СЗК:

$$\pm 1 + p_1^2 = ab. \qquad (4.12)$$

Це означає, що ліва частина (4.12) повинна бути факторизована, на основі чого визначаються параметри a та b. Отже, вираз (4.12) визначає умову для побудови МДФ СЗК з трьох модулів.

Нехай p_1=7. Тоді з (4.11) та (4.12) отримаємо: $p_{2,3} = a,b - 7$ і $ab = \pm 1 + 49 = \begin{cases} 50 = 2 \cdot 5 \cdot 5 \\ 48 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3. \end{cases}$ Усі можливі варіанти з трьох модулів для МДФ СЗК при p_1=7 представлені в табл. 4.1.

Для демонстрації характеру графіка залежності модулів їх потрібно перенумерувати в порядку зростання абсолютної величини p_3, що представлено у табл. 4.2.

На рис. 4.1 представлений характер зміни значень модулів p_3 та p_4 в залежності від номера модуля згідно табл. 4.2.

Як видно з рисунка, модуль p_2 відносно повільно зростає. В той же час, графік для значення модуля p_3 зростає інтенсивніше, доходить до максимуму приблизно посередині номерного діапазону модулів, а потім спадає майже до значення модуля p_2.

Слід зазначити, що найбільший діапазон обчислень буде в тому випадку, коли кожен наступний модуль є на одиницю більший від добутку абсолютних величин попередніх модулів. Крім того, з табл. 4.1 видно, що при застосуванні даних модулів МДФ СЗК розрядність чисел, над якими виконуються арифметичні операції, зменшується в 2-3 рази.

Таблиця 4.1 - Можливі варіанти систем з трьох модулів для МДФ СЗК при p_1=7 (в дужках – розрядність в двійковій системі числення).

№	p_1	ab	a	b	p_2	p_3	P
1	7 (3)	48	1	48	-6 (3)	41 (6)	1722 (11)
2			-1	-48	-8 (4)	-55 (6)	3080 (12)
3			2	24	-5 (3)	17 (5)	595 (10)
4			-2	-24	-9 (4)	-31 (5)	1953 (11)
5			3	16	-4 (3)	9 (4)	252 (9)
6			-3	-16	-10 (4)	-23 (5)	1610 (11)
7			4	12	-3 (2)	5 (3)	105 (7)
8			-4	-12	-11 (4)	-19 (5)	1463 (11)
9			6	8	-1 (1)	1 (1)	7 (3)
10			-6	-8	-13 (4)	-15 (4)	1365 (11)
11		50	1	50	-6 (3)	43 (6)	1806 (11)
12			-1	-50	-8 (4)	-57 (6)	3192 (12)
13			2	25	-5 (3)	18 (5)	630 (10)
14			-2	-25	-9 (4)	-32 (6)	2016 (11)
15			5	10	-2 (2)	3 (2)	42 (6)
16			-5	-10	-12 (4)	-17 (5)	1428 (11)

Таблиця 4.2 - Впорядкування модулів

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p_2	1	2	3	4	5	5	6	6	8	8	9	9	10	11	12	13
p_3	1	3	5	9	17	18	41	43	55	57	31	32	23	19	17	15

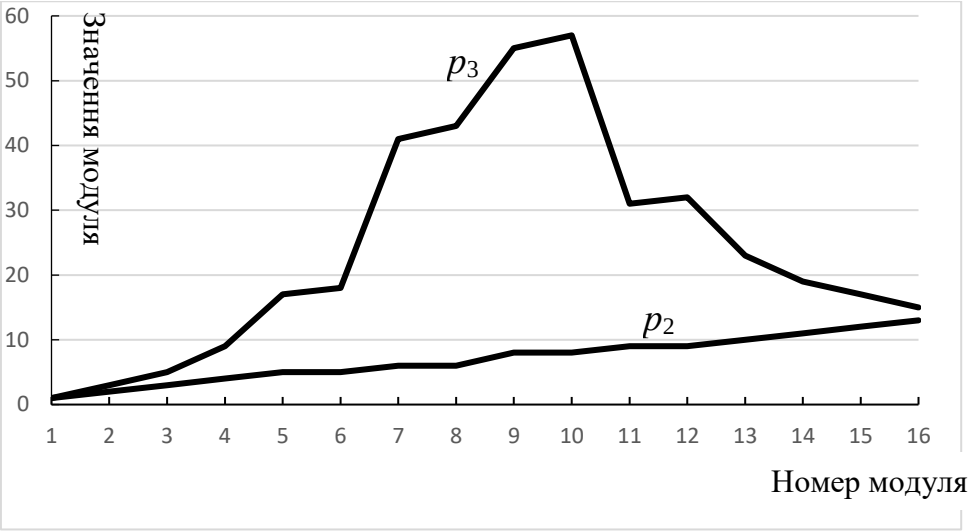


Рис. 4.1 - Характер зміни значень модулів p_2 та p_3 в залежності від номера модуля згідно табл. 4.2

4.3 Розробка трьохмодульної МДФ СЗК за допомогою теореми Вієта

У рівняння (4.10) входять добуток та сума невідомих модулів p_2 та p_3. Для їх пошуку введемо позначення $p_2 p_3 = \chi p_1 \pm 1$. Тоді відповідно $p_2 + p_3 = -\chi$. За допомогою теореми Вієта можна побудувати квадратне рівняння, цілочисельними коренями якого будуть значення шуканих модулів [293]:

$$x^2 + \chi x + \chi p_1 \pm 1 = 0. \tag{4.13}$$

Розв’язавши (4.13), невідомі модулі можна записати таким чином:

$$p_{2,3} = \frac{1}{2}\left(-\chi \pm \sqrt{\chi^2 - 4(\chi p_1 \pm 1)}\right). \tag{4.14}$$

З (4.14) видно, що розв’язки (4.13) будуть цілочисельні, коли дискримінант рівняння (4.13) являтиме собою повний квадрат деякого числа, яке зручно представити в такому вигляді:

$$\chi^2 - 4(\chi p_1 \pm 1) = (\chi - 2(p_1 + \rho))^2. \qquad (4.15)$$

Парність другого доданку в дужках в правій частині виразу випливає з вигляду дискримінанта у лівій частині (4.15). Після відповідних перетворень з (4.15) отримується:

$$\chi = 2p_1 + \rho + \frac{p_1^2 \pm 1}{\rho}. \qquad (4.16)$$

Отже, МДФ СЗК з трьох модулів існує, коли виконується умова $(p_1^2 \pm 1) \bmod \rho = 0$. Це означає, що параметр ρ обмежується інтервалом $\left[-p_1^2 - 1; p_1^2 + 1\right]$, ρ≠0. Дослідивши (4.16), можна побачити, що екстремуми χ визначаються з умови $\rho = \pm\sqrt{p_1^2 \pm 1}$. На рис. 4.2 представлено графік залежності величини χ від ρ, яке змінюється від -26 до 26, при p_1=5.

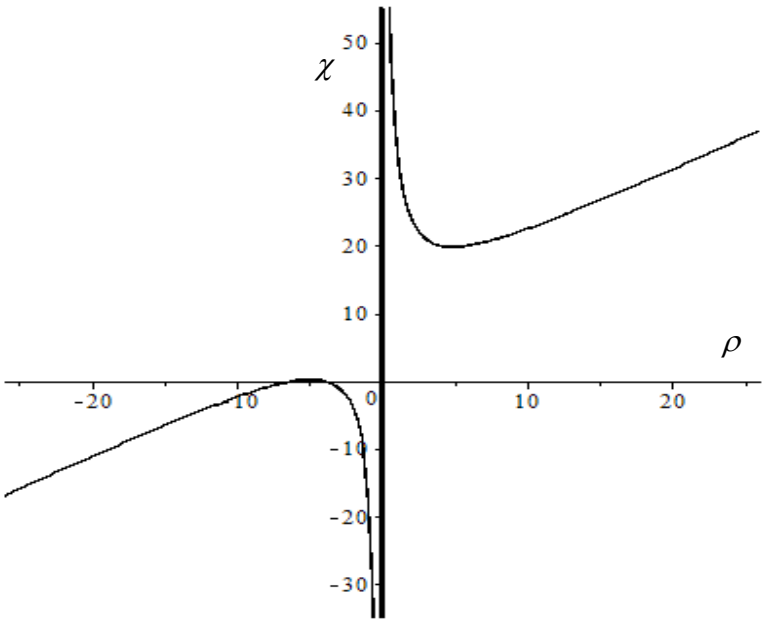


Рис. 4.2 - Графік залежності величини χ від параметра ρ

Графік та побудоване з (4.15) відносно ρ квадратне рівняння $\rho_2^2 + \rho_2(2p_1 - \chi) + p_1^2 \pm 1 = 0$ показують, що фіксованій величині χ відповідають два значення параметра ρ, причому, згідно теореми Вієта, якщо одне з них є цілочисельне, то і друге теж повинно бути цілим числом. Це дозволяє зменшити діапазон дослідження ρ до таких меж: $[-p_1 + 1; p_1 - 1]$, ρ≠0. На рис. 4.3 для прикладу показана поверхня, яка, згідно виразу $\chi = 2p_1 + \rho + \frac{p_1^2 - 1}{\rho}$, характеризує залежність параметра χ від значень p_1=2…10 та ρ=1… p_1-1.

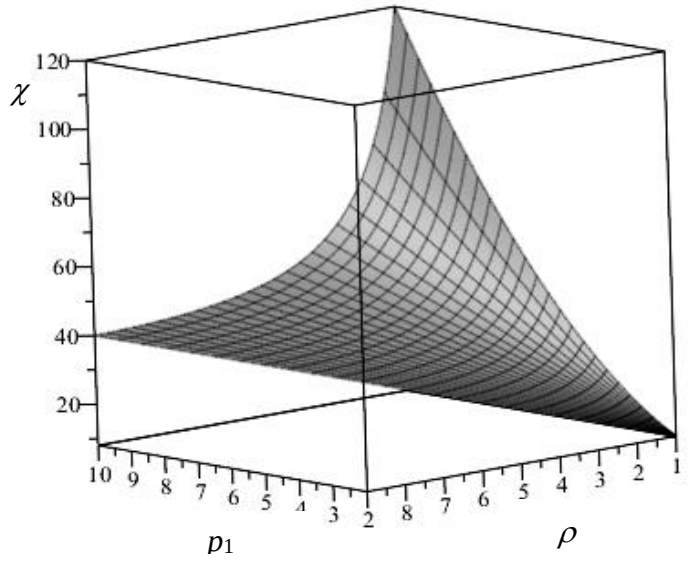


Рис. 4.3 - Графік залежності параметра s_1 від значень p_1=2…10 та ρ=1… p_1-1

Як видно з графіка, величина χ досягає максимуму при найбільшому значенні модуля p_1 та найменшому значенні параметра ρ. Мінімум χ знаходиться при мінімальних значеннях p_1 і ρ.

На рис. 4.4, 4.5 показано відповідно графіки залежності модулів $p_2 = \frac{1}{2}\left(-\rho + \sqrt{\rho^2 - 4(\rho p_1 - 1)}\right)$ та $p_3 = \frac{1}{2}\left(-\rho - \sqrt{\rho^2 - 4(\rho p_1 - 1)}\right)$ від значень модуля p_1=2…10 і параметра ρ=1…p_1-1.

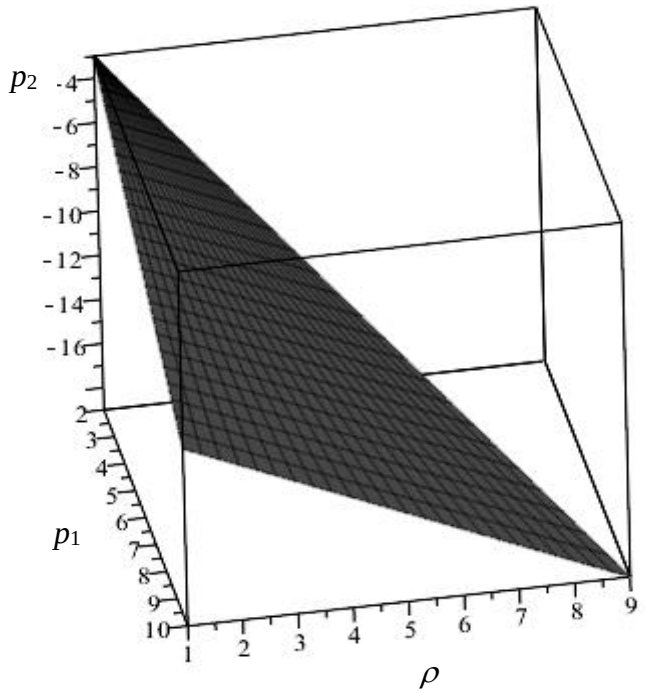


Рис. 4.4 - Графік залежності модуля p_2. від значень модуля p_1 і параметра ρ

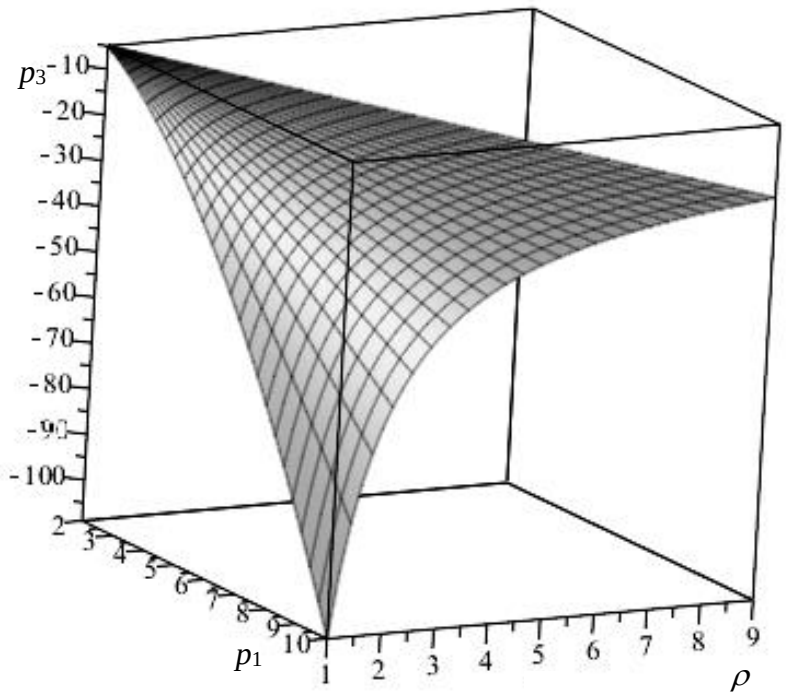


Рис. 4.5 - Графік залежності шуканого модуля p_3 від значень модуля p_1 та параметра ρ

З рис. 4.4, 4.5 видно, що графіком залежності модуля p_2 від значень p_1 і ρ є площина, а модуля p_3 - гіперболоїд. Мінімального абсолютного значення модулі p_2 та p_3 набувають при найменших величинах p_1 та ρ. Максимальне абсолютне значення p_2 буде у випадку, коли $p_1 \to \max$, $\rho \to \max$, відповідно $|p_3| \to \max$, якщо $p_1 \to \max$, $\rho \to \min$.

В табл. 4.3 представлено можливі значення p_2, p_3, відповідних їм параметрів ρ, χ для p_1=7.

Таблиця 4.3 - Можливі значення p_2, p_3, відповідних їм параметрів χ, ρ для p_1=7

№	p_1	ρ	χ	p_2	p_3
1	7	-6	0	1	-1
2		-5	-1	3	-2
3		-4	-2	5	-3
4		-3	-5	9	-4
5		-2	-12	18	-5
6		-2	-13	17	-5
7		-1	-35	41	-6
8		-1	-37	43	-6
9		1	65	-8	-57
10		1	63	-8	-55
11		2	41	-9	-32
12		2	40	-9	-31
13		3	33	-10	-23
14		4	30	-11	-19
15		5	29	-12	-17
16		6	28	-13	-15

З табл. 4.3 видно, що модуль p_3 набуває тільки від'ємних значень. При ρ<0 модуль p_2 додатній і навпаки, якщо ρ>0, то $p_2 < 0$.

4.4 Трьохмодульна МДФ СЗК на основі розв'язку систем конгруенцій

Нехай $p_1=a$, $p_2=a+b$ (причому a і b взаємно прості). Згідно умовам МДФ СЗК (3.2), повинна виконуватися така система [289, 290, 292]:

$$\begin{cases} ap_3 \bmod(a+b) = \pm 1, \\ (a+b)p_3 \bmod a = \pm 1, \\ a(a+b) \bmod p_3 = \pm 1. \end{cases} \tag{4.17}$$

Розглянемо спочатку перші два рівняння (4.17), які, з врахуванням перетворень $ap_3 \bmod(a+b) = -bp_3 \bmod(a+b)$ та $(a+b)p_3 \bmod a = bp_3 \bmod a$, представим таким чином:

$$\begin{cases} y \bmod(a+b) = \mp 1, \\ y \bmod a = \pm 1, \end{cases} \tag{4.18}$$

де $y = bp_3$.

Далі потрібно використати розширений алгоритм Евкліда та КТЗ для чисел a та (a+b). Обернені елементи для аналітичних розрахунків зручно шукати методом додавання модуля, запропонованим в п. 2.4, тобто до 1 додається модуль і перевіряється, чи ділиться націло отриманий результат на відповідне число. Якщо ні, то знову додається модуль і виконується вказана перевірка. Отже:

$$a^{-1} \bmod(a+b) = -b^{-1} \bmod(a+b) = -\frac{k_1(a+b)+1}{b}; \tag{4.19}$$

$$(a+b)^{-1} \bmod a = b^{-1} \bmod a = \frac{k_2 a + 1}{b}, \tag{4.20}$$

де k_1, k_2 дорівнюють кількості доданих модулів (a+b) та a відповідно.

Рівність коефіцієнтів $k_1 = k_2 = k$ в обох виразах підтверджується записом результату для розширеного алгоритма Евкліда:

$$\frac{(a+b)(ka+1)}{b} - \frac{a(k(a+b)+1)}{b} = 1. \tag{4.21}$$

Далі потрібно використати КТЗ для чотирьох випадків, записаних у (4.18): (1; 1), (1; -1), (-1; 1), (-1; -1). Отримаєм:

1) $y \bmod a(a+b) = 1;$

2) $y \bmod a(a+b) = \frac{2ka^2 + 2kab + 2a + b}{b} \bmod a(a+b) =$

$$= \frac{2ka(a+b) + 2a + b}{b} \bmod a(a+b) = \left(2ka + 1 + \frac{2a(ka+1)}{b}\right) \bmod a(a+b);$$

3) $y \bmod a(a+b) = -\frac{2ka^2 + 2kab + 2a + b}{b} \bmod a(a+b) =$

$$= -\frac{2ka(a+b) + 2a + b}{b} \bmod a(a+b) = -\left(2ka + 1 + \frac{2a(ka+1)}{b}\right) \bmod a(a+b);$$

4) $y \bmod a(a+b) = -1.$

Другий та третій випадки складні для аналізу і, як показує практика, для них отримується або $p_3 < p_2$, або взагалі третє рівняння системи (4.17) не виконується. По аналогії з (4.19), (4.20), перший та четвертий випадки приводять до таких результатів:

$$p_3 = b^{-1} \bmod a(a+b) = \frac{k_3 a(a+b)+1}{b}; \tag{4.22}$$

$$p_3 = -b^{-1} \bmod a(a+b) = -\frac{k_3 a(a+b)+1}{b} = \frac{(b-k_3)a(a+b)-1}{b}, \tag{4.23}$$

де k_3 дорівнює кількості доданих модулів $a(a+b)$.

З третього рівняння системи (4.17), враховуючи (4.22), (4.23), отримуємо:

$$a(a+b) \bmod \frac{k_3 a(a+b)+1}{b} = \pm 1; \tag{4.24}$$

$$a(a+b) \bmod \frac{(b-k_3)a(a+b)-1}{b} = \pm 1. \tag{4.25}$$

Вираз під функцією mod в (4.24), (4.25) потрібно помножити на b і цей добуток має бути на 1 більший або менший від $a(a+b)$. Це можливо при $k_3 = 1$ або $b - k_3 = 1$. Отже, отримуємо остаточний вираз для знаходження третього модуля:

$$p_3 = a + \frac{a^2 \pm 1}{b}, \tag{4.26}$$

який за абсолютною величиною на $\frac{a^2 \pm 1}{b}$ больший від першого.

Звідси випливає умова, при виконанні якої існує третій модуль для МДФ СЗК:

$$a^2 \bmod b = \pm 1. \tag{4.27}$$

Рівняння (4.26), (4.27) пояснюють, чому при заданому a не для всіх b можна підібрати третій модуль для МДФ СЗК. Крім того, з (4.26) видно, що $b<a$, інакше $p_3 < p_2$.

На рис. 4.6 показаний графік залежності p_3 від значень модулів p_1 та p_2 згідно (4.26), при умові $p_2 > p_1$.

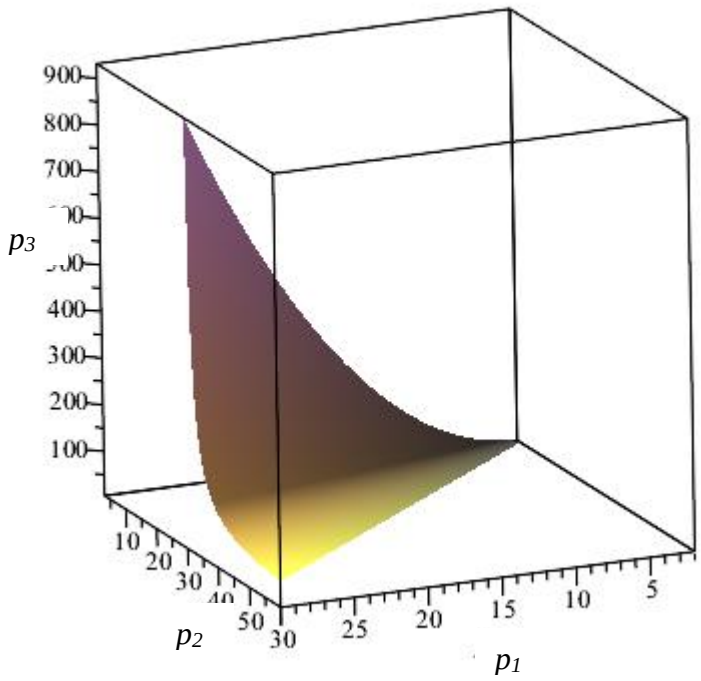


Рис. 4.6 - Поверхня, яка характеризує залежність p_3 від значень модулів p_1 та p_2

Як видно з графіка, p_3 досягає максимума, коли p_1 та p_2 приблизно однакові (p_1 при цьому є максимальним). Із збільшенням p_2 модуль p_3 зменшується і при максимальному значенні другого модуля p_2 та p_3 приймають приблизно однакові значення.

В табл. 4.4 представлені можливі набори для трьох модулів МДФ СЗК при p_1=11 та їх добуток (в дужках - розрядність цих чисел), який вказує на умову переповнення розрядної сітки.

Таблиця 4.4 – Набори для трьох модулів МДФ СЗК при p_1=11 та їх добуток (в дужках вказана розрядність чисел).

№	$p_1=a$	b	p_2	p_3	P
1	11 (4)	1	12 (4)	133 (8)	17556 (15)
2		1	12 (4)	131 (8)	17292 (15)
3		2	13 (4)	72 (7)	10296 (14)
4		2	13 (4)	71 (7)	10153 (14)
5		3	14 (4)	51 (6)	7854 (13)
6		4	15 (4)	41 (6)	6765 (13)
7		5	16 (5)	35 (6)	6160 (13)
8		6	17 (5)	31 (5)	5797 (13)
9		7	18 (5)	відсутній	-
10		8	19 (5)	26 (5)	5434 (13)
11		9	20 (5)	відсутній	-
12		10	21 (5)	23 (5)	5313 (13)

З аналізу таблиці можна зробити висновок, що при використанні трьох модулів МДФ СЗК розрядність чисел, над якими виконуються арифметичні операції, зменшуються в 2–2,5 рази. Крім того, значеню b=1 відповідає два значення p_3, які на 1 відрізняються від добутку двох попередніх модулів. Це випливає з (4.26), оскільки в цьому випадку $p_3 = a(a+1)\pm 1$. Також два значення p_3 отримується при b=2, оскільки p_1 непарне, тому обидва числа $p_1^2 \pm 1$ діляться націло на 2. При b=7 та 9 цілого значення p_3 не існує, оскільки $11^2 \bmod 7 \neq \pm 1$ і $11^2 \bmod 9 \neq \pm 1$.

Для подальшого дослідження модулів p_2 та p_3 табл. 4.4 необхідно трансформувати (табл. 4.5).

Таблиця 4.5 – Впорядкована система модулів

№	1	2	3	4	5	6	7	8	9	10
p_2	12	12	13	13	14	15	16	17	19	21
p_3	133	131	72	71	51	41	35	31	26	23

На рис. 4.7 показаний графік значень модулів p_2 та p_3 в залежності від їх номера згідно табл. 4.5.

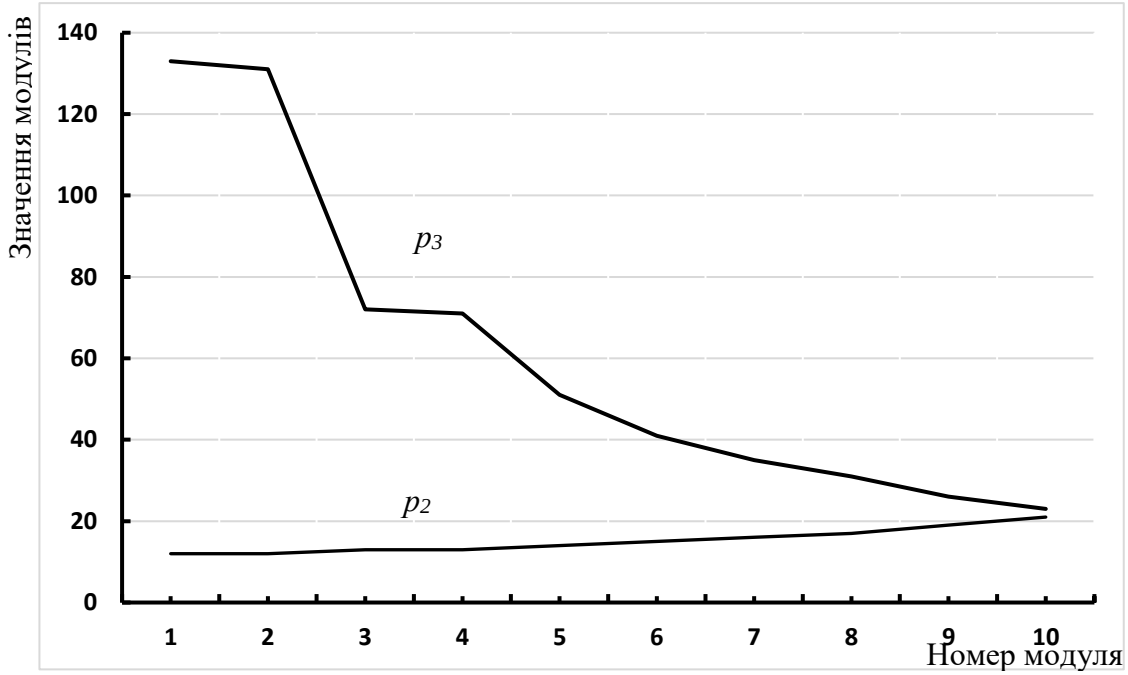


Рис. 4.7 - Графік значень модулів p_2 та p_3 в залежності від їх номера згідно табл. 4.5

Як видно з представленого графіка, модуль p_2 повільно збільшується із збільшенням його номера. В той же час, значення p_3 зменшується набагато інтенсивніше і в кінці розглянутого діапазону обидва модулі практично однакові.

4.5 Метод побудови системи модулів МДФ СЗК з використанням послідовності Фібоначчі

Цікавим є випадок, коли третій модуль дорівнює сумі абсолютних величин двох попередніх. Тоді з (4.26) можна отримати $a + \frac{a^2 \pm 1}{b} = 2a + b$ [289, 290, 292]. Переходячи до квадратного рівняння $b^2+ab-(a^2\pm1)=0$ відносно b і упускаючи його від'ємні корені, знаходимо:

$$b = \frac{-a + \sqrt{5a^2 \pm 4}}{2}. \qquad (4.28)$$

Це означає, що вираз $(5a^2\pm4)$ повинен бути повним квадратом деякого числа. Чисельні дослідження показують, що дану умову задовольняє послідовність Фібоначчі: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, … .

Цей ряд виникає не тільки в самих різноманітних математичних ситуаціях - комбінаторних, чисельних, геометричних, але і в біологічних системах. Використання даної послідовності для обчислювальних систем наведено, наприклад, в [321].

Аналітично n-ий елемент ряду представляється за допомогою формули Біне: $F_n = \frac{\left(1+\sqrt{5}\right)^n - \left(1-\sqrt{5}\right)^n}{2^n \cdot \sqrt{5}}$.

Тому доцільно в загальному випадку розглянути три послідовні елементи ряду Фібоначчі. Враховуючи, що кожен наступний елемент дорівнює сумі двох попередніх і квадрат кожного елемента на одиницю відрізняється від добутку попереднього і наступного членів, то отримаємо таку систему:

$$\begin{cases} F_{n+1}F_{n+2} \bmod F_n = F_{n-1}^2 \bmod F_n = \pm 1, \\ F_n F_{n+2} \bmod F_{n+1} = F_n^2 \bmod F_{n+1} = \pm 1, \\ F_n F_{n+1} \bmod F_{n+2} = F_{n+1}^2 \bmod F_{n+2} = \pm 1. \end{cases} \quad (4.29)$$

Отже, щоб отримати набір з трьох модулів для МДФ СЗК, в якому третій модуль дорівнює сумі абсолютних величин двох попередніх, необхідно вибрати будь-які три послідовні елемента з ряду Фібоначчі, починаючи з третього. На рис. 4.8 зображені графічні залежності F_n, F_{n+1} та F_{n+2} для перших 12 елементів послідовності.

З рисунка видно, що функції швидко зростають зі збільшенням числа n. Слід зауважити, що модулі МДФ СЗК знаходяться на перетині вертикальних ліній сітки з побудованими графіками, де значення функції приймають цілочисельні значення і задовольняють умови системи (4.29).

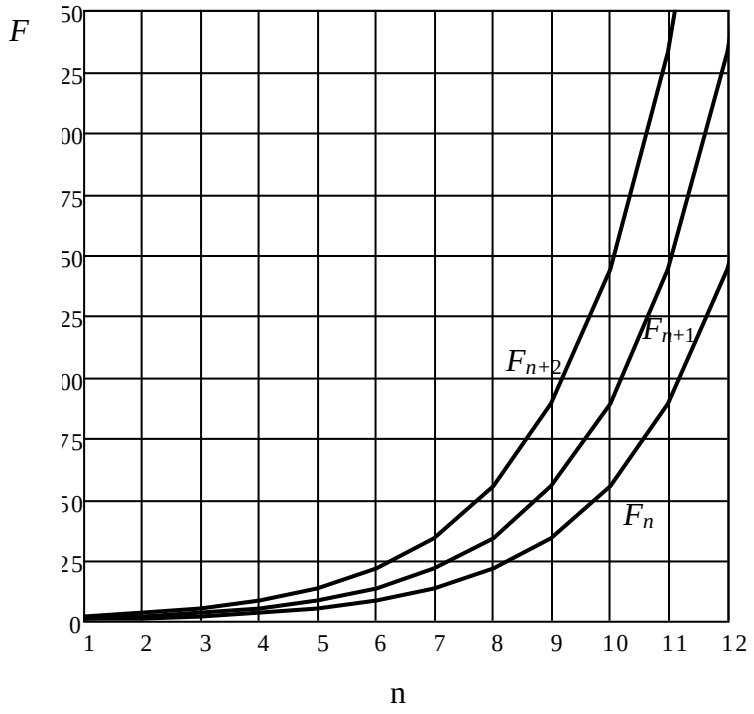


Рис. 4.8 – Графічні залежності F_n, F_{n+1} та F_{n+2} для перших 12 елементів послідовності Фібоначчі

4.6 Побудова трьохмодульної МДФ СЗК для багаторозрядних чисел

У випадку багаторозрядних чисел, коли задана різниця між другим і першим модулями (p_2-p_1=q<p_1, причому p_1 та q мають бути взаємно простими), представимо їх у вигляді p_1=qn-r, p_2=qn-r+q та побудуємо систему рівнянь для МДФ СЗК, яка відповідна (4.17) [289, 290, 292]:

$$\begin{cases}(qn-r)p_3 \bmod(qn-r+q)=qp_3 \bmod(qn-r+q)=\mp 1\\ (qn-r+q)p_3 \bmod(qn-r)=qp_3 \bmod(qn-r)=\pm 1\\ (qn-r)(qn-r+q)\bmod p_3=\pm 1.\end{cases} \tag{4.30}$$

Аналогічно до попереднього, розглянемо спочатку перші два рівняння (4.30), які з врахуванням відповідних математичних перетворень набудуть такого вигляду:

$$\begin{cases} z \bmod(qn-r+q)=\mp 1\\ z \bmod(qn-r)=\pm 1,\end{cases} \tag{4.31}$$

де z=qp_3.

Дана система розв’язується на основі КТЗ, яка передбачає пошук оберненого елемента за модулем. Оскільки вираз (4.31) розглядається у загальному вигляді, без числових значень, то у цьому випадку застосувати розширений алгоритм Евкліда неможливо і обернений елемент потрібно шукати за допомогою додавання модуля. Отже:

$$(qn-r)^{-1} \bmod(qn-r+q)=-q^{-1} \bmod(qn-r+q)=-\frac{k_1(qn-r+q)+1}{q}; \tag{4.32}$$

$$(qn-r+q)^{-1} \bmod(qn-r)=q^{-1} \bmod(qn-r)=\frac{k_2(qn-r)+1}{q}, \tag{4.33}$$

де k_1, k_2 дорівнюють кількості доданих модулів $(qn-r+q)$ та $(qn-r)$ відповідно.

Тоді запис виразу розширеного алгоритму Евкліда

$$\frac{k(qn-r)+1}{q}(qn-r+q)-\frac{k(qn-r+q)+1}{q}(qn-r)=1 \qquad (4.34)$$

показує рівність коефіцієнтів $k_1 = k_2 = k$ в обох рівняннях (4.32) і (4.33).

Далі розглядається КТЗ для чотирьох випадків, записаних в (4.31): (1; 1), (1; -1), (-1; 1), (-1; -1). Друга і третя пари залишків приводять до складних для аналізу виразів:

$$\begin{aligned} &z \bmod (qn-r)(qn-r+q) = \\ &= \pm \frac{2k(qn-r)(qn-r+q)+2(qn-r)+q}{q} \bmod (qn-r)(qn-r+q), \end{aligned} \qquad (4.35)$$

з яких, як показують чисельні розрахунки, отримується або $p_3 < p_2$, або не виконується взагалі третє рівняння системи (4.30).

З першої і четвертої пари залишків можна записати:

$$z \bmod (qn-r+q)(qn-r) = \pm 1. \qquad (4.36)$$

Тоді з (4.36), аналогічно до (4.32), (4.33), шукаються відповідні обернені елементи:

$$p_3 = q^{-1} \bmod (qn-r)(qn-r+q) = \frac{k_3(qn-r)(qn-r+q)+1}{q}; \qquad (4.37)$$

$$p_3 = -q^{-1} \bmod (qn-r)(qn-r+q) = -\frac{k_3(qn-r)(qn-r+q)+1}{q} = \\ = \frac{(b-k_3)(qn-r)(qn-r+q)-1}{q}, \tag{4.38}$$

де k_3 дорівнює кількості доданих модулів $(qn-r+q)(qn-r)$.

Далі, враховуючи (4.37), (4.38), з третього рівняння (4.30):

$$p_3 = qn(n+1) - r(2n+1) + \frac{r^2 \pm 1}{q}. \tag{4.39}$$

На рис. 4.9 показана залежність модуля p_3 від параметрів q і r при n=3, а на рис. 4.10 – залежність p_3 від n і r при сталій різниці $q=p_2-p_1=5$, згідно (4.39), де у чисельнику останнього доданка прийнято (r^2-1).

Як видно із рис. 4.9, при малих q та великих r модуль p_3 змінюється гіперболічно, в інших випадках – лінійно. На рис. 4.10 графіком є параболоїд, модуль p_3 зростає інтенсивніше при малих r та великих n.

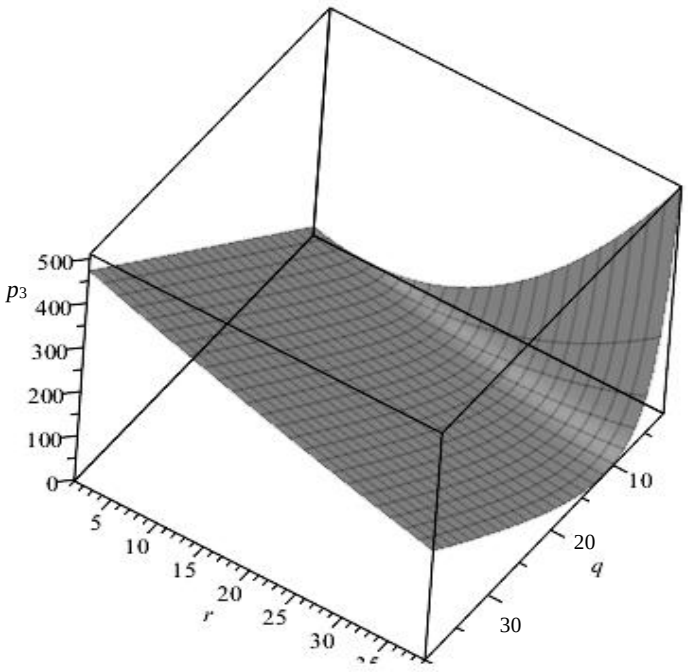


Рис. 4.9 - Залежність модуля p_3 від параметрів q і r при n=3

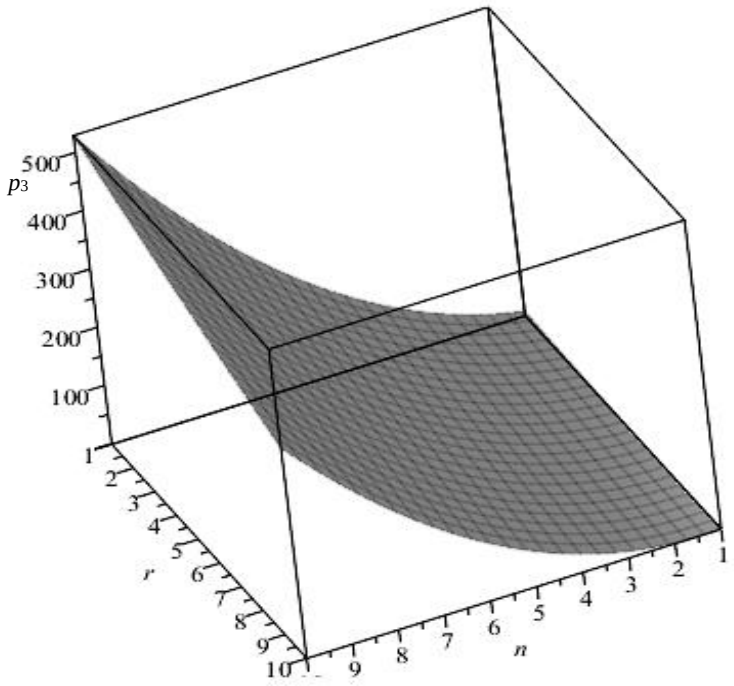


Рис. 4.10 - Залежність модуля p_3 від параметрів n і r при сталій різниці модулів $q=p_2-p_1=5$

З виразу (4.39) випливає умова існування третього модуля, яка має вигляд:

$$r^2 \bmod q = \pm 1. \qquad (4.40)$$

Це, наприклад, пояснює, що для q=7 третій модуль можна знайти тільки при r=1 і r=6. Крім того, коли p_2-p_1=1, тобто q=1, r=0, або p_2-p_1=2 (q=2, r=1), то тоді треті модулі p_3 можуть набирати відповідно по два значення: $p_3=n^2+n\pm 1$ та $p_3= 2n^2-(1\pm 1)/2$. У табл. 4.6 наведено аналітичні вирази для третього модуля p_3 і можливого діапазону обчислень P в залежності від числа n при зміні значення q від 1 до 7.

Знову ж розглянемо випадок, коли третій модуль дорівнює сумі абсолютних величин двох попередніх. Тоді з (4.39) можна отримати

$$qn(n+1) - r(2n+1) + \frac{r^2 \pm 1}{q} = 2(qn - r) + q.$$

Таблиця 4.6 - Аналітичні вирази для модуля p_3 та можливого діапазону обчислень P в залежності від числа n при зміні параметрів q та r

№	Q	r	p_1	p_2	p_3	P
1	1	0	n	n+1	n^2+n+1, n^2+n-1	n^4+2n^3+2n^2+n, n^4+2n^3-n
2	2	1	2n-1	2n+1	2n^2, 2n^2-1	8n^4-2n^2, 8n^4-6n^2+1
3	3	2	3n-2	3n+1	3n^2-n-1	27n^4-18n^3-12n^2+5n+2
4	3	1	3n-1	3n+2	3n^2+n-1	27n^4+18n^3-12n^2-5n+2
5	4	3	4n-3	4n+1	4n^2-2n-1	64n^4-64n^3-12n^2+14n+3
6	4	1	4n-1	4n+3	4n^2+2n-1	64n^4+64n^3-12n^2-14n+3
7	5	4	5n-4	5n+1	5n^2-3n-1	125n^4-150n^3+27n+4
8	5	3	5n-3	5n+2	5n^2-n-1	125n^4-50n^3-50n^2+11n+6
9	5	2	5n-2	5n+3	5n^2+n-1	125n^4+50n^3-50n^2-11n+6
10	5	1	5n-1	5n+4	5n^2+3n-1	125n^4+150n^3-27n+4
11	6	5	6n-5	6n+1	6n^2-4n-1	216n^4-288n^3+30n^2+44n+5
12	6	1	6n-1	6n+5	6n^2+4n-1	216n^4+288n^3+30n^2-44n+5
13	7	6	7n-6	7n+1	7n^2-5n-1	343n^4-490n^3+84n^2+65n+6
14	7	1	7n-1	7n+6	7n^2+5n-1	343n^4+490n^3+84n^2-65n+6

Перейшовши до квадратного рівняння $r^2 - qr(2n-1) + q^2\left(n^2 - n - 1\right) \pm 1 = 0$ відносно r, знайдемо:

$$r = \frac{q(2n-1) \pm \sqrt{5q^2 \pm 4}}{2}, \qquad (4.41)$$

тобто вираз ($5q^2$±4) має бути повним квадратом деякого числа. В табл. 4.7 представлено можливі набори модулів, отримані згідно (4.41), для різних q.

Таблиця 4.7 - Можливі набори модулів, отримані згідно (4.41), для різних q.

№	q	r	$p_1=qn-r$	$p_2= qn-r+q$	$p_3=p_1+p_2$
1	1	n-2	2	3	5
2	2	2n-3	3	5	8
3	3	3n-5	5	8	13
4	5	5n-8	8	13	21
5	8	8n-13	13	21	34
6	13	13n-21	21	34	55
7	21	21n-34	34	55	89

Як видно з табл. 4.7, значення отриманих модулів, параметра q, а також відповідних числових величин для r утворюють послідовність Фібоначчі, в якій кожен наступний елемент дорівнює сумі двох попередніх. Крім того, параметр r записується аналітично, однак модулі, які отримуються з його допомогою, набувають конкретних числових значень.

4.7 Приклад побудови трьохмодульної МДФ СЗК для багаторозрядних чисел

Нехай q=5, r=2.Тоді p_1=5n–2, p_2=5n+3. Перші два рівняння (4.30) утворять таку систему:

$$\begin{cases} 5p_3 \, mod(5n-2) = \pm 1 \\ 5p_3 \, mod(5n+3) = \pm 1. \end{cases} \qquad (4.42)$$

Обернені елементи $5^{-1} \, mod(5n-2)$ та $5^{-1} \, mod(5n+3)$ шукаємо з алгоритму Евкліда та його наслідку. Наведемо приклад розрахунку тільки для одного з них:

$5n-2=5(n-1)+3$ $1=3-1\cdot 2=3-1\cdot(5-1\cdot 3)=-1\cdot 5+2\cdot 3=-1\cdot 5+2\cdot((5n-2)-5(n-1))=$

$5=3\cdot 1+2$ $=2\cdot(5n-2)-5\cdot(2n-1)=1$

$3=2\cdot 1+1$

Отже, $5^{-1}\bmod(5n-2)=-(2n-1)\bmod(5n-2)=3n-1$. Аналогічно можна показати, що $5^{-1}\bmod(5n+3)=-(2n+1)\bmod(5n+3)=3n+2$.

Слід зазначити, що у даному випадку обернений елемент зручніше шукати методом додавання модуля, додаючи до 1 модуль стільки разів , щоб результат ділився на 5. Тобто, до числа $(5n-2)+1$ ще два рази додається модуль. Поділивши $(15n-6)+1$ на 5, отримаємо в результаті $(3n-1)$. Для другого модуля $(5n+3)+1+2\cdot(5n+3)= 15n+10$. Тоді обернене число буде дорівнювати $(3n+2)$.

Відповідно до цього, (4.42) трансформується у систему, зручну для застосування КТЗ, в якій необхідно розглянути чотири різних випадки, взявши по одному залишку з кожного рівняння:

$$\begin{cases} p_3 \bmod(5n-2)=3n-1; & 2n-1 \\ p_3 \bmod(5n+3)=3n+2; & 2n+1. \end{cases} \qquad (4.43)$$

Виходячи з (4.43), треба знайти число, яке при діленні на $(5n-2)$ дає остачу $(3n-1)$ або $(2n-1)$, а при діленні на $(5n+3)$ – остачу $(3n+2)$ або $(2n+1)$. Для цього знову потрібно використати алгоритм Евкліда:

$5n+3=(5n-2)+5$ $1=3-1\cdot 2=3-1\cdot(5-1\cdot 3)=-1\cdot 5+2\cdot 3=-1\cdot 5+2\cdot((5n-2)-5(n-1))=$

$5n-2=5\cdot(n-1)+3$ $=2\cdot(5n-2)-5\cdot(2n-1)=2\cdot(5n-2)-(2n-1)((5n+3)-(5n-2))=$

$5=3\cdot 1+2$ $=2\cdot(5n-2)-(2n-1)(5n+3)+(2n-1)(5n-2)$

$3=2\cdot 1+1.$ $=-(2n-1)(5n+3)+ +(2n+1)(5n-2).$

Згідно КТЗ для знаходження можливого значення p_3 потрібно розглянути окремо кожен з чотирьох випадків:

1) $(-(2n-1)(5n+3)(3n-1)+(2n+1)(5n-2)(3n+2))mod((5n+3)(5n-2))=$
$=(30n^2+6n-7) \bmod (25n^2+5n-6)= 5n^2+n-1;$

2) $(-(2n-1)^2(5n+3)+(2n+1)^2(5n-2))mod((5n+3)(5n-2))=$
$=(20n^2+4n-5) \bmod (25n^2+5n-6)= 20n^2+4n-5;$

3) $(-(2n-1)(5n+3)(2n-1)+(2n+1)(5n-2)(3n+2))mod((5n+3)(5n-2))=$
$=(10n^3+31n^2+3n-7) \bmod (25n^2+5n-6)= 10n^3+6n^2-2n-1;$

4) $(-(2n-1)(5n+3)(3n-1)+(2n+1)(5n-2)(2n+1))mod((5n+3)(5n-2))=$
$=(-10n^3+19n^2+7n-5) \bmod (25n^2+5n-6)= =(-10n^3+19n^2+7n-5);$

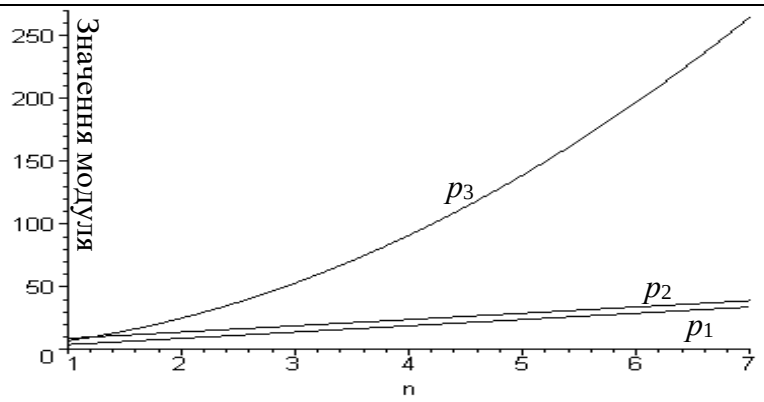
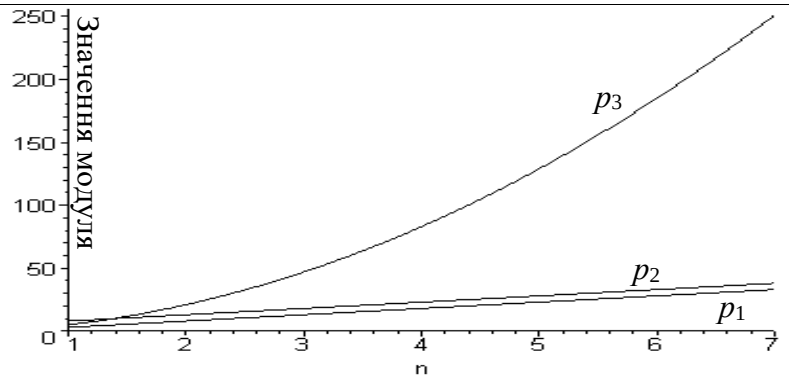
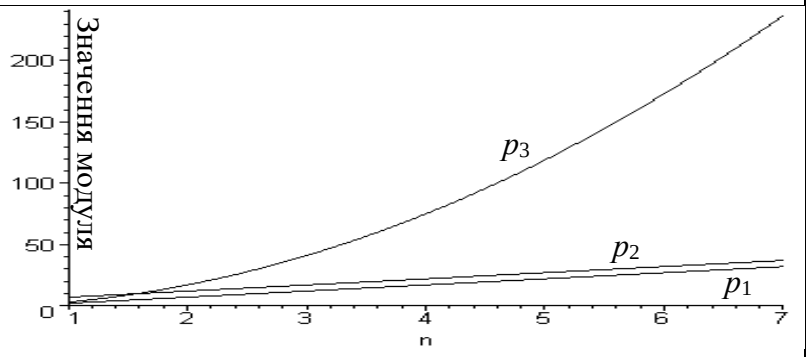
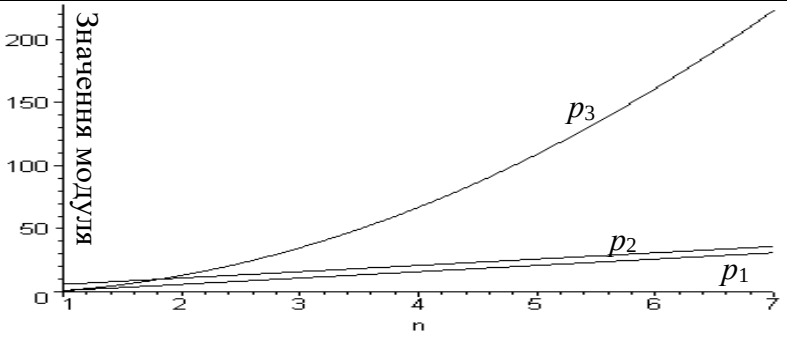
Далі необхідно перевірити виконання третього рівняння у системі (4.30). Безпосередньою підстановкою можна переконатися, що дану умову задовольняє результат, отриманий в першому рівнянні: $(25n^2+5n-6) \bmod (5n^2+n-1)= -1$.

Аналогічно можна знайти модуль p_3 для будь–якого іншого значення b. Відповідно в табл. 4.8 представлено аналітичні вирази для знаходження модулів та діапазону можливих обчислень, а також графічні залежності усіх модулів при різних b та n.

На рис. 4.11 показано графік залежності діапазона обчислень P від n при різних можливих значеннях параметра b.

Як видно з табл. 4.8 та рис. 4.11, модуль p_3 та діапазон обчислень P із збільшенням n зростають параболічно і тим інтенсивніше, чим менше значення параметра b. У табл. 4.9 наведено числові значення p_1, p_2, p_3 та P при a=5 для різних значень n та b.

Таблиця 4.8 - Аналітичні вирази для знаходження модулів та діапазону можливих обчислень, а також графічні залежності модулів при різних b та n

	p_1, p_2, p_3, P	Графічні залежності
1.	$p_1=5n-1$ $p_2=5n+4$ $p_3=5n^2+3n-1$ $P=125n^4+150n^3-27n+4$	
2.	$p_1=5n-2$ $p_2=5n+3$ $p_3=5n^2+n-1$ $P=125n^4+50n^3-50n^2-$ $-11n+6$	
3.	$p_1=5n-3$ $p_2=5n+2$ $p_3=5n^2-n-1$ $P=125n^4-50n^3-50n^2+$ $+11n+6$	
4.	$p_1=5n-4$ $p_2=5n+1$ $p_3=5n^2-3n-1$ $P=125n^4-150n^3+27n+4$	

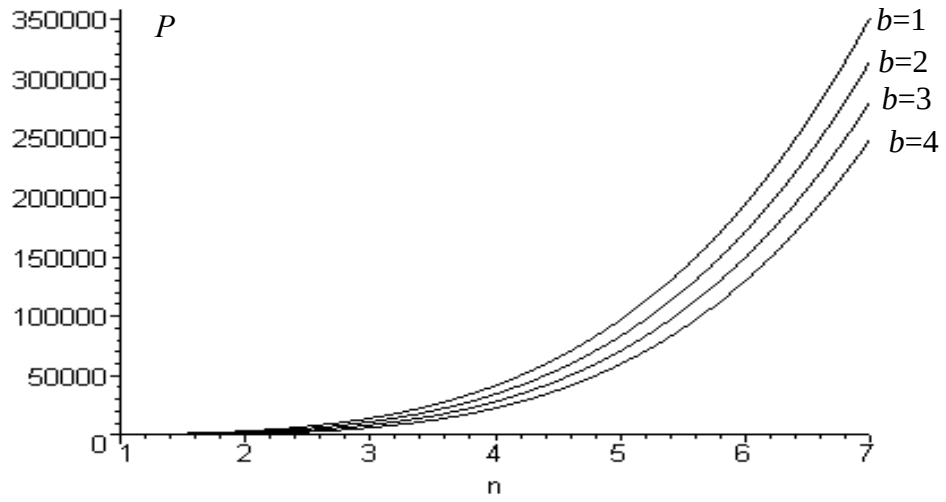


Рис. 4.11 - Графік залежності діапазона обчислень P від n при різних можливих значеннях параметра b

Таблиця 4.9 - Числові значення p_1, p_2, p_3 та P при a=5 для різних значень n та b

n	b	p_1	p_2	p_3	P
1	3	2	7	3	42
	2	3	8	5	120
	1	4	9	7	252
2	4	6	11	13	858
	3	7	12	17	1428
	2	8	13	21	2184
	1	9	14	25	3150
3	4	11	16	35	6160
	3	12	17	41	8364
	2	13	18	47	10998
	1	14	19	53	14098
4	4	16	21	67	22512
	3	17	22	75	28050
	2	18	23	83	34362
	1	19	24	91	41496

З табл. 4.9 видно, що при n=1 між модулями виконується таке співвідношення: $p_1 < p_3 < p_2$, в інших випадках модуль p_3 є найбільшим. Крім того, при сталому n і зміні b на 1 модуль p_3 змінюється на величину $2n$, що випливає із даних табл. 4.9.

Висновки до четвертого розділу

1. Розроблено методи побудови трьохмодульної МДФ СЗК на основі перемноження модулів, факторизації, теореми Вієта, розв'язку систем конгруенцій, використання послідовності Фібоначчі. Дана форма забезпечує зменшення обчислювальної складності при переведенні чисел із СЗК у ПСЧ за рахунок використання від'ємних модулів у КТЗ, рівності ±1 коефіцієнтів базисних чисел, що приводить до уникнення виконання операції пошуку оберненого елемента за модулем та множення на нього.

2. Розроблено метод побудови трьохмодульної МДФ СЗК на основі розв'язку систем конгруенцій для багаторозрядних чисел, коли відома різниця між двома першими модулями, що дало можливість зменшити розрядність чисел, над якими виконуються операції, під час пошуку модулів.

3. Наведено приклади застосування розроблених методів та обґрунтовано доцільність використання трьохмодульної МДФ СЗК в асиметричних криптосистемах.

Основні результати четвертого розділу відображені у роботах [288-294, 320].

5 МЕТОДИ ПОБУДОВИ БАГАТОМОДУЛЬНОЇ МОДИФІКОВАНОЇ ДОСКОНАЛОЇ ФОРМИ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ

У п’ятому розділі розроблено теоретичні основи побудови багатомодульної МДФ СЗК на основі факторизації, наведено приклади побудови чотирьох- та п’ятимодульної МДФ СЗК, а також їх застосування в технічних системах.

5.1 Метод побудови багатомодульної МДФ СЗК на основі факторизації

Для побудови МДФ СЗК із будь-якою кількістю модулів запишемо вираз (4.1) у вигляді системи [284, 291, 294, 320, 322-323]:

$$\begin{cases} P_1 \bmod p_1 = \pm 1 \\ \ldots \\ P_k \bmod p_k = \pm 1. \end{cases} \tag{5.1}$$

Розрахунки, аналогічні проведеним у пункті 3.3 приводять до такого виразу:

$$\sum_{i=1}^{k} \frac{1}{p_i} = \gamma \pm \frac{1}{\prod_{i=1}^{k} p_i}, \tag{5.2}$$

де γ=±1, ±2, ±3, ...

На відміну від ДФ СЗК, коефіцієнт γ можна вибрати рівним 0, що при заданій кількості модулів відповідає найбільшому значенню P. Тоді остання рівність перепишеться у такому вигляді:

$$\frac{1}{p_1}+\frac{1}{p_2}+\frac{1}{p_3}+\ldots+\frac{1}{p_{k-1}}+\frac{1}{p_k}=\pm\frac{1}{p_1p_2p_3\cdots p_{k-1}p_k}. \qquad (5.3)$$

У ДФ СЗК найменші модулі набувають строго визначених значень (p_1=2, p_2=3). У МДФ СЗК найменші модулі можуть бути будь-які. Врахувавши це та відповідно трансформувавши (5.3), отримаємо вираз, аналогічний (5.2) при γ=0:

$$\sum_{i=1}^{k}P_i=\pm 1. \qquad (5.4)$$

Нехай невідомими будуть два останні модулі p_{k-1} та p_k. Тоді (5.4) представимо у вигляді:

$$\begin{aligned}&p_{k-1}p_k\left(p_2p_3\ldots p_{k-2}+p_1p_3\ldots p_{k-2}+\ldots+p_1p_2\ldots p_{k-3}\right)+\\&+p_1p_2\ldots p_{k-2}\left(p_{k-1}+p_{k-2}\right)=\pm 1.\end{aligned} \qquad (5.5)$$

Введемо позначення:

$$p_{k-1,k}=\frac{a,b-p_1p_2\ldots p_{k-2}}{p_2p_3\ldots p_{k-2}+p_1p_3\ldots p_{k-2}+\ldots+p_1p_2\ldots p_{k-3}}. \qquad (5.6)$$

Підставивши (5.6) в (5.5), після відповідних математичних перетворень будемо мати умову, яка повинна виконуватися для визначення набору будь-якої кількості модулів МДФ СЗК:

$$\pm\left(p_2p_3\ldots p_{k-2}+p_1p_3\ldots p_{k-2}+\ldots+p_1p_2\ldots p_{k-3}\right)+\left(p_1p_2p_{k-2}\right)^2=ab. \qquad (5.7)$$

Відповідно ліва частина (5.7) повинна бути факторизована, на основі чого визначаються параметри a та b. Крім того, як випливає з (5.6), модулі p_k та p_{k-1} мають бути цілими числами, тобто

$$(a,b - p_1 p_2 \ldots p_{k-2}) \bmod \left(\left| p_2 p_3 \ldots p_{k-2} + p_1 p_3 \ldots p_{k-2} + \ldots + p_1 p_2 \ldots p_{k-3} \right| \right) = 0 . \quad (5.8)$$

Отже, вирази (5.7) та (5.8) визначають умови для знаходження будь-якої кількості модулів МДФ СЗК, два з яких невідомі.

5.2 Приклад побудови чотирьохмодульної МДФ СЗК

В якості прикладу запропонованого методу розглянемо МДФ СЗК, яка складається з чотирьох модулів. Умови (5.6) - (5.8) відповідно трансформуються [291]:

$$p_{3,4} = \frac{a,b - p_1 p_2}{p_1 + p_2}; \pm (p_1 + p_2) + (p_1 p_2)^2 = ab; (a,b - p_1 p_2) mod (p_1 + p_2) = 0 . \quad (5.9)$$

З (5.3) видно, що при k=4 модулі p_1 і p_2 повинні мати різні знаки. Очевидно, що, вважаючи модуль p_1 додатнім, найбільше варіантів буде, коли p_2=-(p_1+1), оскільки в цьому випадку третя умова (5.9) зникає. Перші дві матимуть такий вигляд:

$$p_{3,4} = -\left(a,b + p_1^2 + p_1\right); \ \pm 1 + \left(p_1 (p_1 + 1)\right)^2 = ab . \quad (5.10)$$

Нехай p_1=7, p_2=-8. Тоді з (5.9) отримаємо: $p_{3,4} = -(a,b + 56)$ і

$$ab = \pm 1 + 3136 = \begin{cases} 3135 = 3 \cdot 5 \cdot 11 \cdot 19 \\ 3137 \text{ - просте число.} \end{cases}$$

Усі можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=7, p_2=-8 представлені в табл. 5.1 (в дужках вказана розрядність модулів та діапазона обчислень у двійковій системі числення).

Для побудови та дослідження графіка залежності модулів їх потрібно перенумерувати в порядку зростання абсолютної величини p_3, що представлено у табл. 5.2.

Таблиця 5.1 - Можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=7, p_2=-8 (в дужках – розрядність в двійковій системі числення).

№	p_1, p_2	ab	a	b	p_3	p_4	P
1	7 (3), -8 (4)	3135	1	3135	-57 (6)	-3191 (12)	10185672 (24)
2			-1	-3135	-55 (6)	3079 (12)	9483320 (24)
3			3	1045	-59 (6)	-1101 (11)	3637704 (22)
4			-3	-1045	-53 (6)	989 (10)	2935352 (22)
5			5	627	-61 (6)	-683 (10)	2333128 (22)
6			-5	-627	-51 (6)	571 (10)	1630776 (21)
7			11	285	-67 (7)	-341 (9)	1279432 (21)
8			-11	-285	-45 (6)	229 (8)	577080 (20)
9			15	209	-71 (7)	-265 (9)	1053640 (21)
10			-15	-209	-41 (6)	153 (8)	351288 (19)
11			19	165	-75 (7)	-221 (8)	928200 (20)
12			-19	-165	-37 (6)	109 (7)	225848 (18)
13			33	95	-89 (7)	-151 (8)	752584 (20)
14			-33	-95	-23 (5)	39 (6)	50232 (16)
15			55	57	-111 (7)	-113 (7)	702408 (20)
16			-55	-57	-1 (1)	1 (1)	56 (6)
17		3137	1	3137	-57 (6)	-3193 (12)	10192056 (24)
18			-1	-3137	-55 (6)	3081 (12)	9489480 (24)

Таблиця 5.2 - Впорядкування модулів

№	1	2	3	4	5	6	7	8	9
p_3	1	23	37	41	45	51	53	55	55
p_4	1	39	109	153	229	571	989	3079	3081
№	10	11	12	13	14	15	16	17	18
p_3	57	57	59	61	67	71	75	89	111
p_4	3191	3193	1101	683	341	265	221	151	113

На рис. 5.1 представлений характер зміни значень модулів p_3 та p_4 в залежності від номера модуля згідно табл. 5.2 у логарифмічній шкалі з основою 2, на якій відразу ж видно розрядності отриманих модулів у двійковій системі числення.

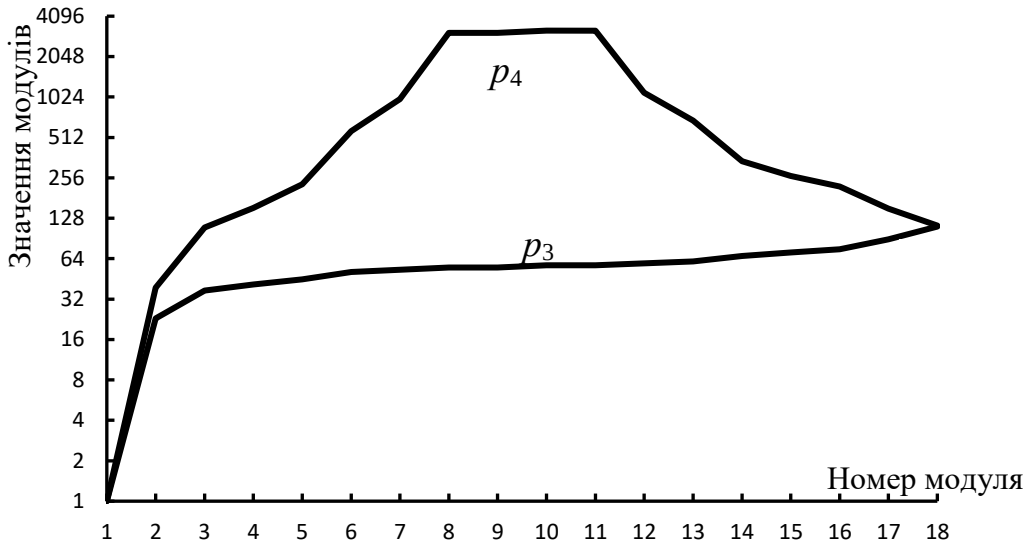


Рис. 5.1 - Характер зміни значень модулів p_3 та p_4 при p_1=7, p_2=-8 в залежності від номера модуля згідно табл. 5.2 у логарифмічній шкалі

Як видно з рисунка, модуль p_3 відносно повільно зростає. В той же час, графік для значення модуля p_4 зростає інтенсивніше, доходить до плоского максимуму приблизно посередині номерного діапазону модулів, а потім спадає до значення модуля p_3.

Слід зазначити, що найбільший діапазон обчислень буде в тому випадку, коли кожен наступний модуль є на одиницю більший від добутку абсолютних величин усіх попередніх модулів. Окрім того, із табл. 5.1 видно, що при застосуванні даних модулів, які утворюють МДФ СЗК розрядність чисел, над якими виконуються арифметичні операції, зменшується в 2-3 рази. Набір модулів 7, -8, -1, 1 у табл. 5.1 вказує, що числа 7 та -8 самі утворюють МДФ СЗК.

Чисельні розрахунки показують, що для p_1=7 в інших випадках, крім p_2=-8, найбільша кількість варіантів наборів модулів буде при p_2=-9 та p_2=-11. Тоді рівняння (5.9) набудуть відповідно такого вигляду:

$$p_{3,4} = -\frac{a,b+63}{2};\ \pm 2 + 63^2 = ab;\ (a,b-63)\bmod 2 = 0. \qquad (5.11)$$

$$p_{3,4} = -\frac{a,b+77}{4};\ \pm 4 + 77^2 = ab;\ (a,b-77)\bmod 4 = 0. \qquad (5.12)$$

Звідси випливає, що $ab = \pm 2 + 3969 = \begin{cases} 3967 \\ 3971 = 11 \cdot 19 \cdot 19 \end{cases}$ для p_2=-9 і $ab = \pm 4 + 5929 = \begin{cases} 5925 = 3 \cdot 5 \cdot 5 \cdot 79 \\ 5933 = 17 \cdot 349 \end{cases}$ для p_2=-11. В табл. 5.3 наведені впорядковані значення абсолютних величин модулів, отриманих з (5.11)-(5.12) аналогічно табл. 5.2 при p_2=-9 та p_2=-11, а також границя діапазону обчислень P (в дужках вказана розрядність представлених чисел).

Оскільки в розглянутих випадках параметри a та b є непарними числами, то третя умова з (5.11) для p_2=-9 виконується при всіх можливих значеннях a та b. Для p_2=-11 третя умова з (5.12) виконується тільки для половини можливих варіантів параметрів a та b.

На рис. 5.2 показані графіки залежності значень модулів p_3 та p_4 (суцільною лінією – для p_2=-9, пунктирною - для p_2=-11) від номера модуля згідно табл. 5.3 в логарифмічній шкалі з основою 2.

Як видно з рисунка, модуль p_3 відносно повільно збільшується. Графік для значення модуля p_4 збільшується інтенсивніше, приходить до плоского максимуму (причому для p_2=-9 максимум ширший) посередині номерного діапазону модулів, а потім спадає.

Таблиця 5.3 - Впорядковані значення абсолютних величин модулів p_3 та p_4 при p_2=-9 та p_2=-11 [291]

p_2	p_3,	Номер модуля			
	p_4	1	2	3	4
9 (4)	p_3	22(5)	26(5)	31(5)	31(5)
	p_4	73(7)	149(8)	1952(11)	1954(11)
	P	101178 (17)	244062 (18)	3812256 (22)	3816162 (22))
11 (4)	p_3	13(4)	15(4)	18(5)	19(5)
	p_4	40(6)	68(7)	277(9)	1462(11)
	P	40040 (16)	78540 (17)	383922 (19)	2138906 (22)
p_2	p_3,	Номер модуля			
	p_4	5	6	7	8
9 (4)	p_3	32(6)	32(6)	37(6)	41(6)
	p_4	2015(11)	2017(11)	212(8)	136(8)
	P	4062240 (22)	4066272 (22)	494172 (19)	351288 (19)
11 (4)	p_3	19(5)	20(5)	23(5)	38(6)
	p_4	1464(11)	513(10)	118(7)	39(6)
	P	2141832 (22)	790020 (20)	208978 (18)	114114 (17)

Слід відмітити, що не в усіх випадках множники, на які факторизується добуток ab, дають можливість отримати відповідні значення модулів. Це залежить від виконання третьої умови (5.9).

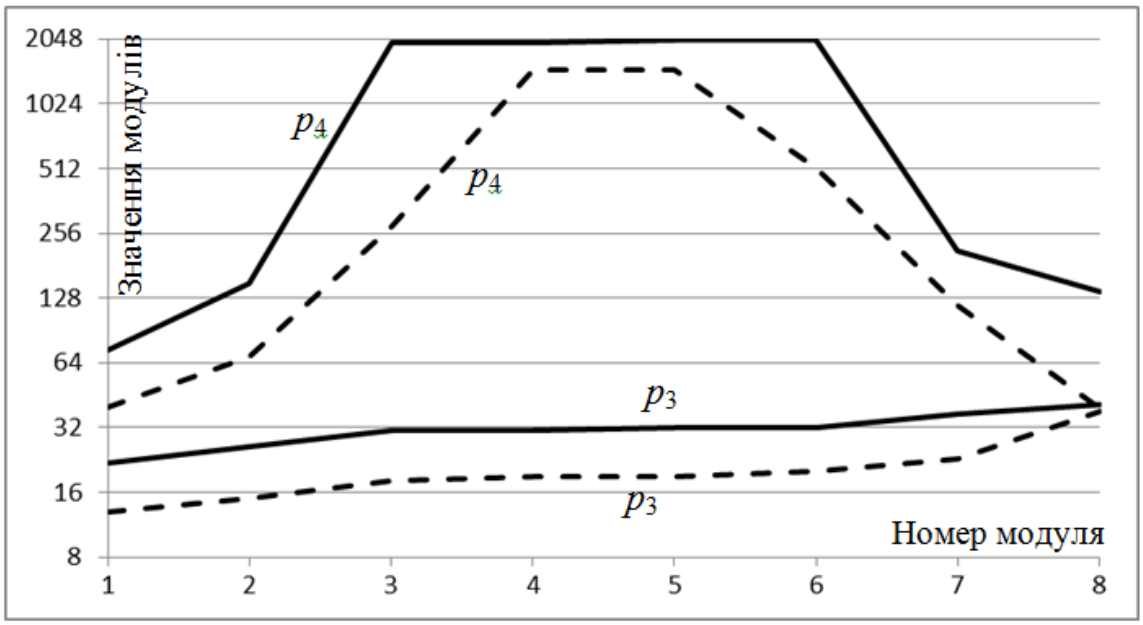


Рис. 5.2 – Характер змін значень модулів p_3 та p_4 при p_1=7, p_2=-9 (суцільна лінія) і p_2=-11 (пунктирна лінія) в залежності від номера модуля згідно табл. 5.3

Розглянемо випадок, коли p_1=7, p_2=-10. Тоді з (5.9) отримаємо: $p_{3,4} = -\frac{a,b+70}{3}$; $(a, b+70) \bmod 3 = 0$; $ab = \pm 3 + 4900 = \begin{cases} 4903 \\ 4897 = 59 \cdot 83. \end{cases}$ Усі можливі варіанти модулів представлені в табл. 5.4.

Таблиця 5.4 - Можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=7, p_2=-10 (в дужках – розрядність в двійковій системі числення).

№	p_1, p_2	ab	a	b	p_3	p_4	P
1	7 (3), -10 (4)	4903 (13)	1	4903	не існує	не існує	не існує
2			-1	-4903	-23 (5)	1611 (11)	2593710 (22)
3		4897 (13)	1	4897	не існує	не існує	не існує
4			-1	-4897	-23 (5)	1609 (11)	2590490 (22)
5			59	83	-43 (6)	-51 (6)	153510 (18)
6			-59	-83	не існує	не існує	не існує

Знову ж розрядність чисел, над якими виконуються операції, зменшується в 2-3 рази. Крім того, в половині з можливих випадків не існує цілочисельних значень p_3 і p_4.

У табл. 5.5 представлені інші можливі варіанти наборів модулів МДФ СЗК при p_1=7, отримані відповідно до умов (5.9).

Видно, що більшість варіантів отримані при a=±1, коли четвертий модуль на одиницю відрізняється від добутку трьох попередніх, що відповідає максимальній межі діапазону обчислень.

Таблиця 5.5 - Можливі варіанти систем з чотирьох модулів для МДФ СЗК при p_1=7, p_2= -12, -13, -15 (в дужках вказана розрядність модулів та діапазона обчислень у двійковій системі числення).

p_2	p_3	p_4	P
-12 (4)	-17 (5)	-1427 (11)	2037756 (22)
	-17 (5)	-1429 (11)	2040612 (22)
	-19 (5)	-145 (8)	231420 (18)
-13 (4)	-15 (4)	1364 (11)	1861860 (21)
	-15 (4)	1366 (11)	1864590 (21)
	-16 (5)	-291 (9)	423696 (19)
-15 (4)	-16 (5)	-73 (7)	122640 (17)

5.3 Побудова та дослідження п’ятимодульної МДФ СЗК

Для спрощення розрахунків обмежимося значенням першого модуля p_1=3, кількістю модулів k=5 і, не зменшуючи загальності рішення, вважатимемо [322], що

$$p_1=3<|p_2|<|p_3|<|p_4|<|p_5|. \qquad (5.13)$$

Чисельні розрахунки показують, що виконання цієї умови для цілочисельних розв'язків (5.3) можливе тільки тоді, коли p_2, p_3<0.

На рис. 5.3 для прикладу зображено вигляд поверхні, що характеризує залежність модуля p_5 від p_3 та p_4 згідно виразу:

$$p_5 = \frac{-1-3p_2p_3p_4}{p_2p_3p_4+3(p_3p_4+p_2p_4+p_2p_3)}, \tag{5.14}$$

який отримується з (5.3), при p_1=3 та p_2=-5.

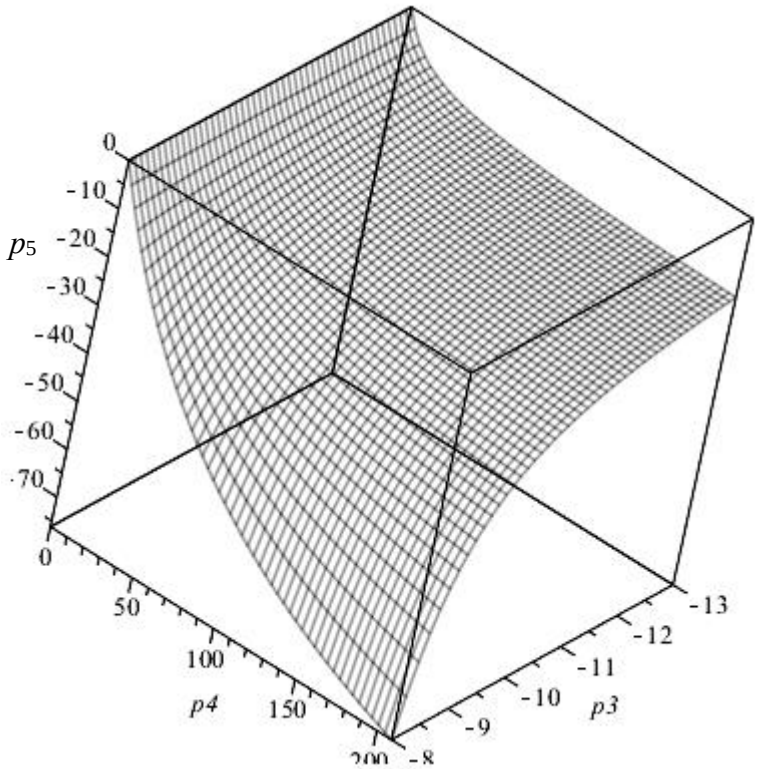


Рис. 5.3 - Вигляд поверхні, що характеризує залежність модуля p_5 від p_3 та p_4 при p_2=-5

Тоді, вважаючи невідомими два останні модулі p_4 та p_5, з (5.3) можна отримати діофантове рівняння другого порядку для їх пошуку:

$$p_4p_5(p_2p_3+3(p_2+p_3))+3p_2p_3(p_4+p_5)=\pm 1. \tag{5.15}$$

Введемо позначення:

$$p_{4,5} = \frac{a,b - 3p_2 p_3}{p_2 p_3 + 3(p_2 + p_3)}. \quad (5.16)$$

Після підстановки (5.16) в (5.15) та відповідних математичних перетворень можна отримати вираз для цілочисельного розв'язку (5.16):

$$\pm(p_2 p_3 + 3(p_2 + p_3)) + (3p_2 p_3)^2 = ab. \quad (5.17)$$

Ліву частину (5.17) потрібно факторизувати, на основі чого визначаються параметри a і b. Крім того, модулі p_4 та p_5 повинні бути цілими числами. Тому з (5.16) випливає:

$$(a,b - 3p_2 p_3) \bmod(|p_2 p_3 + 3(p_2 + p_3)|) = 0. \quad (5.18)$$

Вирази (5.17) і (5.18) визначають умови для знаходження п'яти модулів МДФ СЗК, два з яких невідомі.

При p_2=-4, p_3=-7 вирази (5.16)-(5.18) перетворюються таким чином: $p_{4,5} = \frac{a,b-84}{-5}$; $(a,b-84) \bmod 5 = 0$; $ab = \pm 5 + 7056 = \begin{cases} 7051 = 11 \cdot 641 \\ 7061 = 23 \cdot 307. \end{cases}$ В табл. 5.6 наведено всі можливі цілочисельні значення a і b, що визначаються факторизацією добутку $a \cdot b$, а також випадки, коли існують набори модулів МДФ СЗК і діапазон відповідних обчислень [322].

З табл. 5.6 видно, що модулі p_4 та p_5 набувають додатних значень, а розрядність чисел, над якими виконуються арифметичні операції, приблизно зменшується вдвічі. Крім того, у п'яти з восьми можливих випадків, що утворюються при факторизації, цілочисельних наборів модулів не існує.

Значенню p_4=17 відповідає два значення модуля p_5, кожне з яких на одиницю відрізняється від добутку чотирьох попередніх модулів.

Таблиця 5.6 - Можливі варіанти систем із п’яти модулів для МДФ СЗК при p_1=3, p_2=-4, p_3=-7 (в дужках – розрядність у двійковій системі числення)

№	p_1, p_2	ab	a	b	p_4	p_5	P
1		7051	1	7051	не існує		
2			-1	-7051	17(5)	1427(11)	2037756(21)
3	p_1=3(2)		11	641	не існує		
4	p_2=-4(3)		-11	-641	19(5)	145(8)	231420(18)
5	p_3=-7(3)	7061	1	7061	не існує		
6			-1	-7061	17(5)	1429(11)	2040612(21)
7			23	23	не існує		
8			-23	-23	не існує		

Зрозуміло, що найбільше шуканих наборів буде тоді, коли модулі p_1, p_2, p_3 самі утворюють МДФ СЗК, оскільки в цьому випадку $p_1p_2 + p_2p_3 + p_1p_3 = \pm 1$ і умова (5.18) виконується завжди. Кількість різних варіантів визначатиметься кількістю множників при факторизації лівої частини (5.17). Такі випадки доцільно розглянути детальніше.

При p_2=-4, p_3=-11 з виразів (5.16)-(5.17) можна отримати: $p_{4,5} = \dfrac{a,b-132}{-1}$; $ab = \pm 1 + 17424 = \begin{cases} 17423 = 7 \cdot 19 \cdot 131 \\ 17425 = 5 \cdot 5 \cdot 17 \cdot 41. \end{cases}$ Усі можливі варіанти модулів та величина діапазону обчислень представлені в табл. 5.7.

З аналізу табл. 5.7 випливає, що значення p_4 набувають тільки додатних значень, а знак p_5 протилежний до знаку a і b. Розрядність чисел, над якими виконуватимуться арифметичні операції, зменшується у 2-2,5 рази. Рядок 7, в якому p_4, p_5=±1, показує, що даний набір з трьох модулів p_1=3, p_2=-4, p_3=-11 утворює МДФ СЗК. Значенням p_4=131 та p_4=133 відповідає по два значення

модуля p_5, абсолютні величини яких на одиницю відрізняються від добутку чотирьох попередніх модулів [322].

Таблиця 5.7 - Можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-4, p_3=-11 (в дужках – розрядність у двійковій системі числення).

№	p_1, p_2, p_3	ab	a	b	p_4	p_5	P
1	3 (2), -4 (3), -11 (4)	17423	1	17423	131 (8)	-17291 (15)	298995972 (29)
2			-1	-17423	133 (8)	17555 (15)	308195580 (29)
3			7	2489	125 (7)	-2357 (12)	38890500 (26)
4			-7	-2489	139 (8)	2621 (12)	48090108 (26)
5			19	917	113 (7)	-785 (10)	11709060 (25)
6			-19	-917	151 (8)	1049 (11)	20908668 (25)
7			131	133	1 (1)	-1(1)	132 (8)
8			-131	-133	263 (9)	265 (9)	9199740 (24)
9		17425	1	17425	131 (8)	-17293 (15)	299030556 (29)
10			-1	-17425	133 (8)	17557 (15)	308230692 (29)
11			5	3485	127 (7)	-3353 (12)	56209692 (26)
12			-5	-3485	137 (8)	3617 (12)	65409828 (26)
13			17	1025	115 (7)	-893 (10)	13555740 (24)
14			-17	-1025	149 (8)	1157 (11)	22755876 (25)
15			25	697	107 (7)	-565 (10)	7980060 (23)
16			-25	-697	157 (8)	829 (10)	17180196 (25)
17			41	425	91 (7)	-293 (9)	3519516 (22)
18			-41	-425	173 (8)	557 (10)	12719652 (21)
19			85	205	47 (6)	-73 (7)	452892 (19)
20			-85	-205	217 (8)	337 (9)	9653028 (23)

Набір p_1=3, p_2=-4, p_3=-13 також утворює МДФ СЗК. Тому з (5.16)-(5.17) отримується: $p_{4,5} = \frac{a,b-156}{1}$; $ab = \pm 1 + 24336 = \begin{cases} 24335 = 5 \cdot 31 \cdot 157 \\ 24337 = \text{просте число.} \end{cases}$ Усі можливі варіанти модулів, діапазон можливих обчислень та відповідні розрядності представлені в табл. 5.8.

Таблиця 5.8 - Можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-4, p_3=-13 (в дужках – розрядність в двійковій системі числення).

№	p_1, p_2, p_3	ab	a	b	p_4	p_5	P
1	3 (2), -4 (3), -13 (4)	24335	1	24335	-155 (8)	24179 (15)	584648220 (30)
2			-1	-24335	-157 (8)	-24491 (15)	599833572 (30)
3			5	4867	-151 (8)	4711 (13)	110972316 (27)
4			-5	-4867	-161 (8)	-5023 (13)	126157668 (27)
5			31	785	-125 (7)	629 (10)	12265500 (24)
6			-31	-785	-187 (8)	-941 (10)	27450852 (25)
7			155	157	-1 (1)	1(1)	156 (8)
8			-155	-157	-311 (9)	-313 (9)	15185508 (24)
9		24337	1	24337	-155 (8)	24181 (15)	584696580 (30)
10			-1	-24337	-157 (8)	-24493 (15)	599882556 (30)

З табл. 5.8 слідує, що значення модуля p_4 набувають тільки від’ємних значень, а знак модуля p_5 збігається із знаком a та b. Розрядність чисел, над якими виконуватимуться відповідні арифметичні операції, зменшується в 2-2,5 рази. Рядок 7 табл. 5.8 показує, що даний набір з трьох модулів утворює МДФ СЗК. Значенням p_4=-155 та p_5=-157 відповідає по два значення модуля p_5, абсолютні величини яких на одиницю відрізняються від добутку абсолютних величин чотирьох попередніх модулів.

У двох останніх розглянутих випадках для проведення подальших досліджень розподілу абсолютних величин знайдених модулів їх потрібно перенумерувати в порядку зростання абсолютної величини $|p_4|$, що представлено у табл. 5.9, 5.10.

Таблиця 5.9 - Впорядкування модулів по зростанню $|p_4|$ при p_1=3, p_2=-4, p_3=-11

№	1	2	3	4	5	6	7	8	9	10
p_4	1	47	91	107	113	115	125	127	131	131
p_5	1	73	293	565	785	893	2357	3353	17291	17293
№	11	12	13	14	15	16	17	18	19	20
p_4	133	133	137	139	149	151	157	173	217	263
p_5	17555	17557	3617	2621	1157	1049	829	557	337	265

Таблиця 5.10 - Впорядкування модулів по зростанню $|p_4|$ при p_1=3, p_2=-4, p_3=-13

№	1	2	3	4	5	6	7	8	9	10
p_4	1	125	151	155	155	157	157	161	187	311
p_5	1	629	4711	24179	24181	24492	24493	5023	941	313

На рис. 5.4 представлений характер зміни значень модулів p_4 та p_5 в залежності від номера модуля згідно табл. 5.9, 5.10 у логарифмічній шкалі з основою 2, що вказує на розрядності отриманих модулів у двійковій системі числення.

Як видно з рис. 5.4, в обох випадках значення абсолютної величини $|p_4|$ відносно повільно зростає. В той же час, графік для модуля $|p_5|$ зростає набагато інтенсивніше, доходить до плоского максимуму посередині

номерного діапазону модулів, а потім спадає до значення абсолютної величини $|p_4|$.

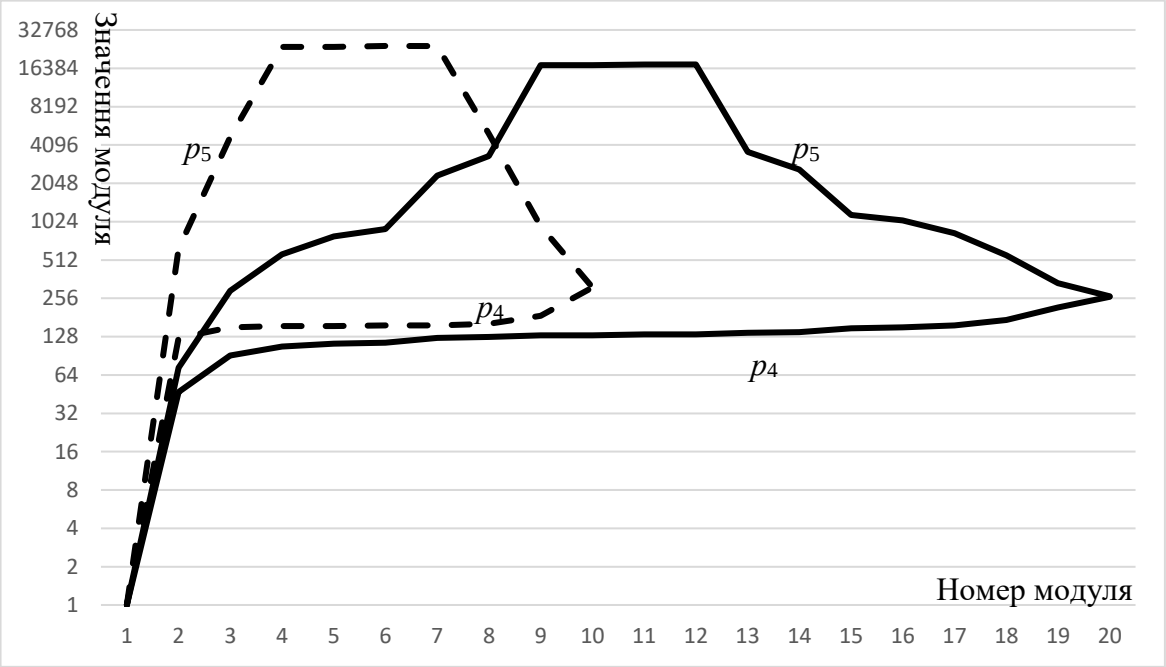


Рис. 5.4 - Характер зміни значень модулів p_4 та p_5 при p_1=3, p_2=-4, p_3=-11 (суцільна лінія) і p_3=-13 (пунктирна лінія) в залежності від номера модуля згідно табл. 5.9, 5.10

Набір модулів p_1=3, p_2=-5, p_3=-7 теж утворює МДФ СЗК, тому вирази (12)-(13) запишуться таким чином: $p_{4,5} = \dfrac{a,b-105}{-1} = 105 - a,b$ і

$$ab = \pm 1 + 11025 = \begin{cases} 11024 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 13 \cdot 53 \\ 11026 = 2 \cdot 37 \cdot 149. \end{cases}$$

Усі можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-5, p_3=-7 представлені в табл. 5.11. Результати чисельного експерименту (див. табл. 5.11) показують, що значення p_4 набувають додатних значень, а знак p_5 протилежний до знаку a і b. Розрядність чисел, над якими виконуватимуться арифметичні операції, зменшується у 2-3 рази. Рядок 17, в якому p_4, p_5=±1, показує, що даний набір з трьох модулів p_1=3, p_2=-5, p_3=-7 утворює МДФ СЗК.

Таблиця 5.11 - Можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-5, p_3=-7 (в дужках – розрядність в двійковій системі числення).

№	p_1, p_2, p_3	ab	a	b	p_4	p_5	P
1	3 (2), -5 (3), -7 (3)	11024 (14)	1	11024	104 (7)	-10919 (14)	119235480 (27)
2			-1	-11024	106 (7)	11129 (14)	123865770 (27)
3			2	5512	103 (7)	-5407 (13)	58476705 (26)
4			-2	-5512	107 (7)	5617 (13)	63106995 (26)
5			4	2756	101 (7)	-2651 (12)	28113855 (25)
6			-4	-2756	109 (7)	2861 (12)	32744145 (25)
7			8	1378	97 (7)	-1273(11)	12965505 (24)
8			-8	-1378	113 (7)	1483 (11)	17595795 (25)
9			13	848	92 (7)	-743 (10)	7177380 (23)
10			-13	-848	118 (7)	953 (10)	11807670 (24)
11			16	689	89 (7)	-584 (10)	5457480 (23)
12			-16	-689	121 (7)	794 (10)	10087770 (24)
13			26	424	79 (7)	-319 (9)	2646105 (22)
14			-26	-424	131 (8)	529 (10)	7276395 (23)
15			52	212	53 (6)	-107 (7)	595455 (20)
16			-52	-212	157 (8)	317 (9)	5225745 (23)
17			104	106	1 (1)	-1 (1)	105 (7)
18			-104	-106	209 (8)	211 (8)	4630395 (23)
19		11026 (14)	1	11026	104 (7)	-10921 (14)	119257320 (27)
20			-1	-11026	106 (7)	11131 (14)	123888030 (27)
21			2	5513	103 (7)	-5408 (13)	58487520 (26)
22			-2	-5513	107 (7)	5618 (13)	63118230 (26)
23			37	298	68 (7)	-193 (8)	1378020 (21)
24			-37	-298	142 (8)	403 (9)	6008730 (23)
25			74	149	31 (5)	-44 (6)	143220 (18)
26			-74	-149	179 (8)	254 (8)	4773930 (23)

Значенням p_4=103, 104, 106, 107 відповідає по два значення модуля p_5, що зумовлене наявністю спільних множників 1 і 2 в обох випадках розкладу добутку ab.

Модулі p_1=3, p_2=-5, p_3=-8 теж утворюють МДФ СЗК, тому вирази (5.16)-(5.17) приводять до таких результатів: $p_{4,5} = a, b - 120$ і $ab = \pm 1 + 14400 = \begin{cases} 14399 = 7 \cdot 11 \cdot 11 \cdot 17 \\ 14401 = \text{просте.} \end{cases}$ Усі можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-5, p_3=-8 представлені в табл. 5.12.

Таблиця 5.12 - Можливі варіанти систем з п’ятьох модулів для МДФ СЗК при p_1=3, p_2=-5, p_3=-8 (в дужках – розрядність в двійковій системі числення).

№	p_1, p_2, p_3	ab	a	b	p_4	p_5	P
1	3 (2), -5 (3), -8 (3)	14399 (14)	1	14399	-119 (7)	14279 (14)	203904120 (28)
2			-1	-14399	-121 (7)	-14519 (14)	210815880 (28)
3			7	2057	-113 (7)	1937 (11)	26265720 (25)
4			-7	-2057	-127 (7)	-2177 (12)	33177480 (25)
5			11	1309	-109 (7)	1189 (11)	15552120 (24)
6			-11	-1309	-131 (8)	-1429 (11)	22463880 (25)
7			17	847	-103 (7)	727(10)	8985720 (24)
8			-17	-847	-137 (8)	-967 (10)	15897480 (24)
9			77	187	-43 (6)	67 (7)	345720 (19)
10			-77	-187	-197 (8)	-307 (9)	7257480 (23)
11			119	121	-1 (1)	1 (1)	120 (7)
12			-119	-121	-239 (8)	-241 (8)	6911880 (23)
13		14401 (14)	1	14401	-119 (7)	14281 (14)	203932680 (28)
14			-1	-14401	-121 (7)	-14521 (14)	210844920 (28)

З табл. 5.12 видно, що значення p_4 набувають тільки від'ємних значень, а знак p_5 збігається із знаком a та b. Розрядність чисел, над якими виконуватимуться арифметичні операції, зменшується у 2-3 рази. Рядок 11 показує, що набір p_1=3, p_2=-5, p_3=-8 утворює МДФ СЗК. Значенням p_4=-119 та p_5=-121 відповідає по два значення модуля p_5, абсолютні величини яких на одиницю відрізняються від добутку абсолютних величин чотирьох попередніх модулів.

Аналогічно до випадку p_1=3, p_2=-4, в табл. 5.13, 5.14 представлено впорядкування модулів по зростанню абсолютної величини p_4.

Таблиця 5.13 – Впорядкування модулів по зростанню $|p_4|$ при p_1=3, p_2=-5, p_3=-7

№	1	2	3	4	5	6	7	8	9	10	11	12	13
p_4	1	31	53	68	79	89	92	97	101	103	103	104	104
p_5	1	44	107	193	319	584	743	1273	2651	5407	5408	10919	10921
№	14	15	16	17	18	19	20	21	22	23	24	25	26
p_4	106	106	107	107	109	113	118	121	131	142	157	179	209
p_5	11129	11131	5617	5618	2861	1483	953	794	529	403	317	254	211

Таблиця 5.14 - Впорядкування модулів по зростанню $|p_4|$ при p_1=3, p_2=-5, p_3=-8

№	1	2	3	4	5	6	7
p_4	1	43	103	109	113	119	119
p_5	1	67	727	1189	1937	14279	14281
№	8	9	10	11	12	13	14
p_4	121	121	127	131	137	197	239
p_5	14519	14521	2177	1429	967	307	241

На рис. 5.5 представлений характер зміни значень модулів p_4 та p_5 в залежності від номера модуля згідно табл. 5.13, 5.14 у логарифмічній шкалі з основою 2, що вказує на розрядності отриманих модулів у двійковій системі числення.

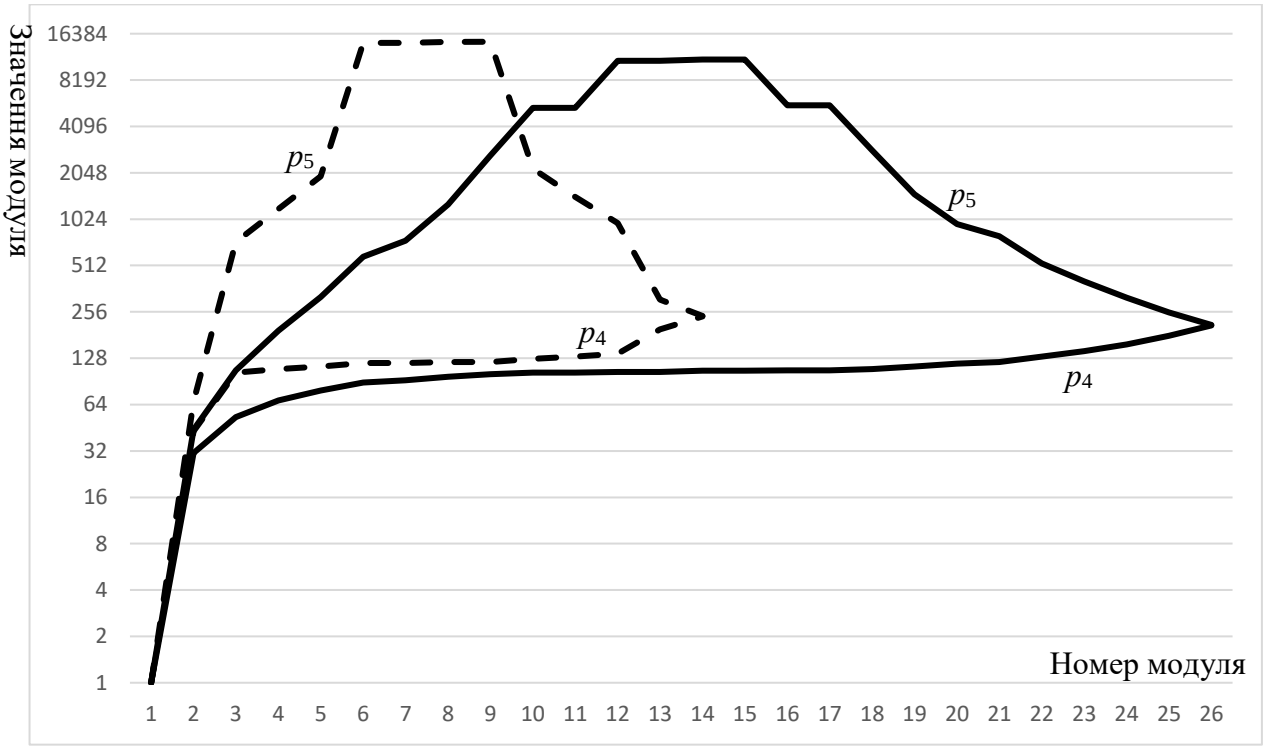


Рис. 5.5 - Характер зміни значень модулів p_4 та p_5 при p_1=3, p_2=-5, p_3=-7 (суцільна лінія) і p_3=-8 (пунктирна лінія) в залежності від номера модуля згідно табл. 5.13, 5.14

Характер відповідних графіків, представлених на рис. 5.4 і 5.5, є подібним, але плоский максимум на рис. 5.5 має більше значення.

В табл. 5.15 представлено впорядкування значення діапазона обчислень відповідно до табл. 5.10, 5.11, 5.13, 5.14 по зростанню абсолютної величини p_4.

На рис. 5.6 показано графік залежності діапазона обчислень згідно номера у табл. 5.15 для різних значень p_2, p_3.

Таблиця 5.15 - Впорядкування значення діапазона обчислень P по зростанню абсолютної величини p_4 для різних величин p_2, p_3 при p_1=3

№	P (p_2=-4, p_3=-11)	P (p_2=-4, p_3=-13)	P (p_2=-5, p_3=-7)	P (p_2=-5, p_3=-8)
1	132	156	105	120
2	452892	12265500	143220	345720
3	3519516	110972316	595455	8985720
4	7980060	584648220	1378020	15552120
5	11709060	584696580	2646105	26265720
6	13555740	599833572	5457480	203904120
7	38890500	599882556	7177380	203932680
8	56209692	126157668	12965505	210815880
9	298995972	27450852	28113855	210844920
10	299030556	15185508	58476705	33177480
11	308195580		58487520	22463880
12	308230692		119235480	15897480
13	65409828		119257320	7257480
14	48090108		123865770	6911880
15	22755876		123888030	
16	20908668		63106995	
17	17180196		63118230	
18	12719652		32744145	
19	9653028		17595795	
20	9199740		11807670	
21			10087770	
22			7276395	
23			6008730	
24			5225745	
25			4773930	
26			4630395	

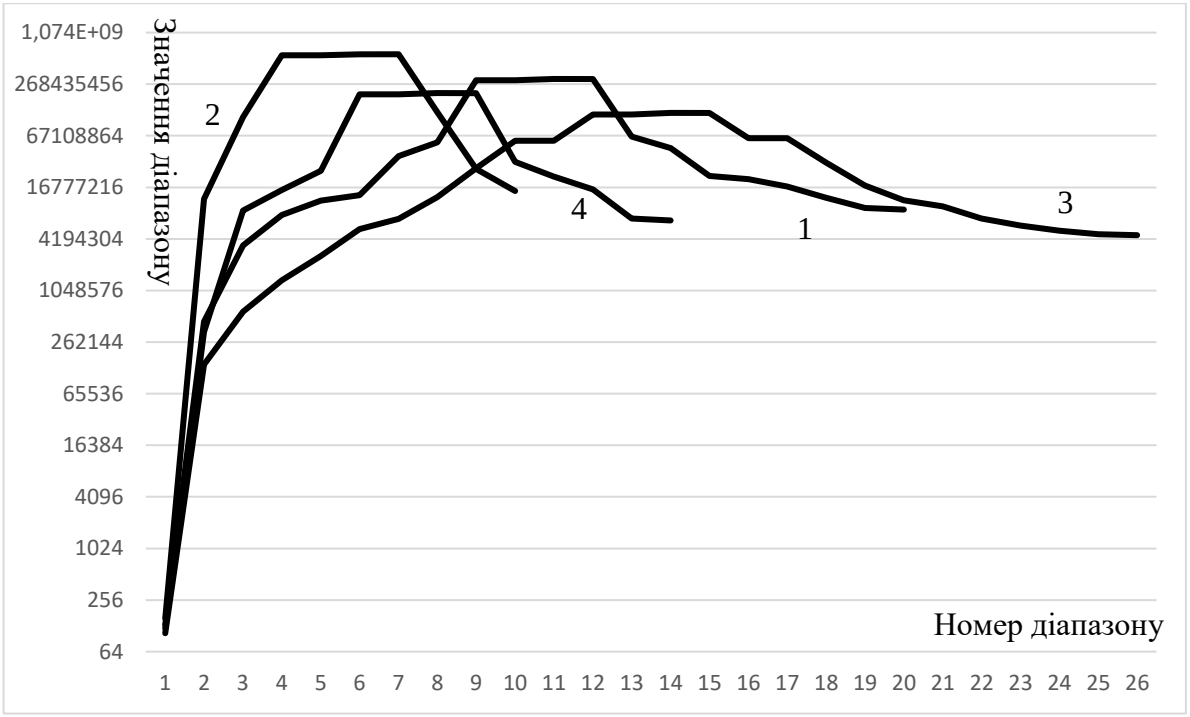


Рис. 5.6 - Графік залежності діапазона обчислень згідно номера у табл. 5.15 для різних значень p_2, p_3 (1 - p_2=-4, p_3=-11, 2- p_2=-4, p_3=-13, 3 - p_2=-5, p_3=-7, 4 - p_2=-5, p_3=-8)

Подібно до графіків для p_5 на рис. 5.4, 5.5, залежність діапазона обчислень P спочатку різко зростає, посередині номерного діапазона має плоский максимум, який для фіксованого p_2 розміщується вище при більшому $|p_3|$. При подальшому збільшенні номера графіки повільно спадають.

У табл. 5.16 для p_1=3 згідно аналогічних чисельних досліджень наведено інші можливі набори модулів при різних p_2, p_3, які утворюють МДФ СЗК.

У табл. 5.16 продемонстровано, що кількість наборів з п’яти модулів значно зменшується, якщо перші три з них не утворюють МДФ СЗК. Це пов’язано з необхідністю виконання умови (5.18). Розрядність чисел, над якими виконуватимуться арифметичні операції, зменшується у 2-3 рази. Найбільший діапазон обчислень при заданих першому модулю і їх кількості

буде тоді, коли абсолютні величини всіх наступних на одиницю більші від добутку абсолютних величин попередніх.

Таблиця 5.16 - Інші можливі набори модулів, які утворюють МДФ СЗК при p_1=3

№	p_2, p_3	p_4	p_5	P
1	-4 (3), -17 (5)	-41 (6)	-8363 (14)	69948132 (27)
2		-41 (6)	-8365 (14)	69964860 (27)
3	-5 (3), -11 (4)	-28 (5)	-149 (8)	688380 (20)
4		-23 (5)	949 (10)	3601455 (22)
5		-19 (5)	98 (7)	307230 (19)
6		-17 (5)	61 (6)	171105 (18)
7		-13 (4)	29 (5)	62205 (16)
8	-7 (3), -10 (4)	-11 (4)	2309 (12)	5333790 (23)
9		-11 (4)	2311 (12)	5338410 (23)

5.4 Метод розширення набору модулів МДФ СЗК

Для демонстрації методу та спрощення розрахунків обмежимо наші міркування п’ятьма модулями, перші три з яких утворюють ДФ СЗК [323]. Єдиним можливим варіантом є набір p_1=2, p_2=3, p_3=5. Тоді вираз (5.3) набуде такого вигляду:

$$\frac{31}{30}+\frac{1}{p_4}+\frac{1}{p_5}=1\pm\frac{1}{30p_4p_5}. \qquad (5.19)$$

Після відповідних математичних перетворень (5.19) трансформується в таку умову:

$$(31-30k)p_4 p_5 + 30(p_4 + p_5) = \pm 1. \qquad (5.20)$$

На рис. 5.7 представлений графік залежності p_5 від p_4 при s=1. З нього видно, що при p_4<-30 та p_4>1/30 модуль p_5 набуває від'ємних значень, в інших випадках модуль p_5 додатній.

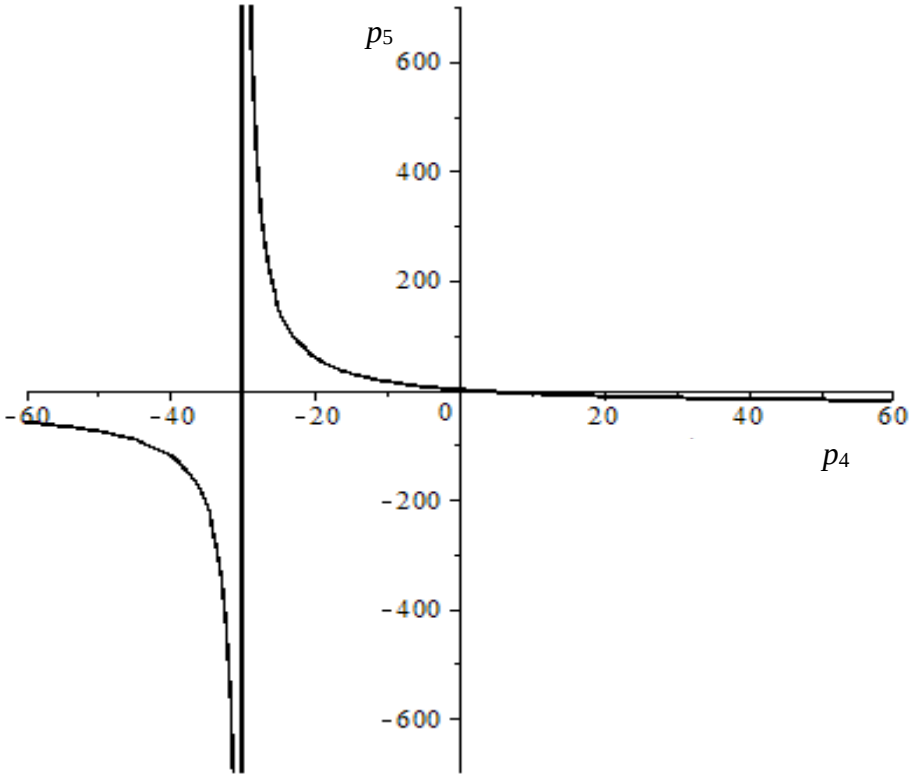


Рис. 5.7 – Графік залежності p_5 від p_4 при k=1

Ввівши заміну $p_{4,5} = \dfrac{a,b-30}{31-30s}$, отримаємо вираз для цілочисельного розв'язку (5.20): $ab = \pm(31-30s)+30^2$.

Розглянемо різні значення параметра s.

1). s=0. Тоді $ab = \pm 31 + 30^2 = \begin{cases} 931 = 7 \cdot 7 \cdot 19 \\ 869 = 11 \cdot 79. \end{cases}$

В табл. 5.17 наведено всі можливі цілочисельні значення a і b, що визначаються факторизацією, а також випадки, коли набори модулів МДФ СЗК існують та відповідний їм діапазон обчислень.

Таблиця 5.17 - Можливі варіанти систем із п’яти модулів для МДФ СЗК при p_1=2, p_2=3, p_3=5 і s=0 (в дужках – розрядність в двійковій системі числення)

№	ab	a	b	p_4	p_5	P
1	869	1	869	не існує		
2		-1	-869	-1	-29 (5)	870 (10)
3		11	79	не існує		
4		-11	-79	не існує		
5	931	1	931	не існує		
6		-1	-931	-1	-31 (5)	930 (10)
7		7	133	не існує		
8		-7	-133	не існує		
9		19	49	не існує		
10		-19	-49	не існує		

З табл. 5.17 видно, що розрядність чисел, над якими виконуються арифметичні операції, приблизно зменшується вдвічі. Значення p_4=-1 вказує, що набори із чотирьох модулів 2, 3, 5, 29 і 2, 3, 5, 31 утворюють МДФ СЗК, в яких кожен наступний модуль відрізняється на одиницю від добутку попередніх. Крім того, в восьми із десяти можливих випадків, які утворюються при факторизації, відповідних цілочисельних наборів модулів не існує.

2). s=1. Тоді $ab = \pm 1 + 30^2 = \begin{cases} 901 = 17 \cdot 53 \\ 899 = 29 \cdot 31. \end{cases}$ При даній умові будь-яким значенням a і b відповідатимуть цілочисельні модулі p_4 та p_5. Результати наведено в табл. 5.18.

Таблиця 5.18 - Можливі варіанти систем із п’яти модулів для МДФ СЗК при p_1=2, p_2=3, p_3=5 і s=1 (в дужках – розрядність в двійковій системі числення)

№	ab	a	b	p_4	p_5	P
1	899	1	899	-29 (5)	869 (10)	756030 (20)
2		-1	-899	-31 (5)	-929 (10)	863970 (20)
3		29	31	-1 (1)	1 (1)	30 (5)
4		-29	-31	-59 (6)	-61 (6)	107970 (17)
5	901	1	901	-29 (5)	871 (10)	757770 (20)
6		-1	-901	-31 (5)	-931 (10)	865830 (20)
7		17	53	-13 (4)	23 (5)	8970 (14)
8		-17	-53	-47 (6)	-83 (7)	117030 (17)

Табл. 5.18 показує, що розрядність чисел, над якими виконуються арифметичні операції, зменшується приблизно у 2-3 рази. Третій рядок таблиці вказує, що модулі 2, 3, 5 утворюють ДФ СЗК. Модуль p_4 завжди від’ємний, знак модуля p_5 збігається із знаком параметрів a і b, причому у цьому випадку p_4>-30, що узгоджується з графіком на рис. 5.7. Найбільший діапазон обчислень буде в тому випадку, коли абсолютна величина кожного наступного модуля більша на одиницю від добутку абсолютних величин попередніх модулів.

Чисельні розрахунки показують, що при інших значеннях параметра s отримані модулі відрізнятимуться від знайдених при s=0, 1 лише знаком. Для проведення подальших досліджень розподілу абсолютних величин усіх отриманих наборів модулів їх потрібно перенумерувати в порядку зростання $|p_4|$, що представлено у табл. 5.19.

Таблиця 5.19 - Впорядкування модулів по зростанню $|p_4|$ при p_1=2, p_2=3, p_3=5

№	1	2	3	4	5	6	7	8	9	10
p_4	1	1	1	13	29	29	31	31	47	59
p_5	1	29	31	23	869	871	929	931	83	61

На рис. 5.8 представлений характер зміни значень модулів p_4 та p_5 в залежності від номера модуля згідно табл. 5.19 у логарифмічній шкалі, не враховуючи значення p_4=1.

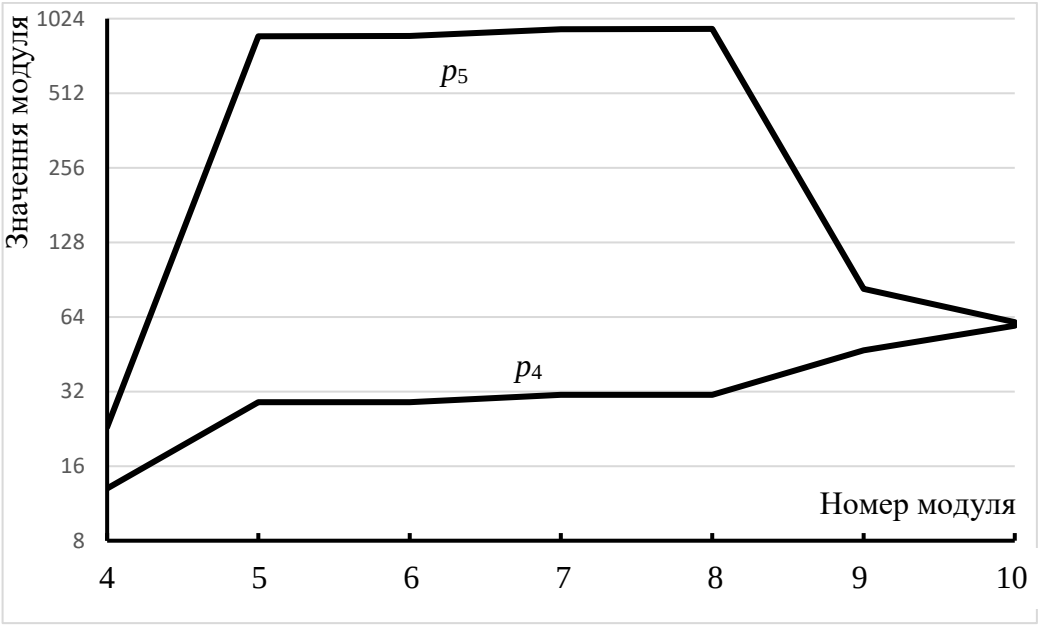


Рис. 5.8 - Характер зміни значень модулів p_4 та p_5 при p_1=2, p_2=3, p_3=5 в залежності від номера модуля згідно табл. 5.19

Як видно з рис. 5.8, значення $|p_4|$ відносно повільно зростає. В той же час, графік для $|p_5|$ зростає набагато інтенсивніше, доходить до плоского максимуму, а потім спадає до значення $|p_4|$.

5.5 Застосування МДФ СЗК в технічних системах

З курсу фізики відомо, що загальний опір послідовно з'єднаних резисторів дорівнює сумі опорів кожного з них [324]. Зворотна операція при класичному підході є практично нездійсненна. Її вирішення було б важливим кроком для розробки засобів автоматизованого управління спеціалізованими комп'ютерними системами, технологічні об'єкти для яких характеризуються специфічними особливостями, що виключають можливість безпосереднього доступу людини. Однією з головних задач є визначення тиску та розподілених температурних полів у різних точках та на різних рівнях досліджуваного середовища (наприклад, для контролю умов зберігання та обліку руху нафто – та газопродуктів). Такі задачі особливо актуальні в геофізиці, нафтогазовидобувній, вугільній, металургійній, космічній та інших галузях промисловості, а також в метеорології.

Важливим аспектом розвитку досліджень в цій галузі є розробка підходів, методів, алгоритмів та комп'ютерних засобів побудови розподілених сенсорів на основі використання СЗК.

Сенсори для вимірювання розподілених значень тиску або температури, як правило, реалізуються двома способами: або багатоточковою структурою з паралельним інформаційним каналом, або на основі позиційних систем числення з моноканальною лінією передачі інформації [325]. Недоліком першого способу є наявність великої кількості автономних ліній зв'язку від сенсорів до мікроконтролерів.

В основі другого методу побудови багатопараметричного сенсора (БПС) лежить послідовне з'єднання термо- або п'єзорезисторів R_i, опір кожного з яких може змінюватися стрибкоподібно на величину ΔR_i, що визначає точність вимірювання і відповідає основі системи числення. Тоді загальний опір БПС визначається аналітичним виразом $R_x = \sum_{i=1}^{n} R_i$.

Недоліком даного методу вимірювання є значна різниця між ΔR_i, яка відповідає $\frac{\Delta R_{i+1}}{\Delta R_i} = F$, де F – основа позиційної системи числення. Наприклад, при F=2 та F=10 відповідно ΔR_i в кожному каналі повинно змінюватися в 2 та 10 разів. Крім того, кожний наступний сенсор має в F разів більший діапазон вимірювання порівняно з попереднім.

Нехай маємо t однакових послідовно з'єднаних термо- або п'єзорезисторів R_0 (рис. 5.9), опір яких може змінюватися стрибкоподібно з кроком $\Delta R_i = \frac{R_0}{p_i}$, де p_i–взаємнопрості числа або модулі, які визначають точність вимірювання [326, 327]. При цьому максимальне значення вимірюваної величини для кожного резистора $D_i = R_0\left(1 - \frac{1}{p_i}\right)$, а, відповідно, повне максимальне значення:

$$D = \sum_{i=1}^{t} D_i = R_0 \cdot \sum_{i=1}^{t}\left(1 - \frac{1}{p_i}\right). \qquad (5.21)$$

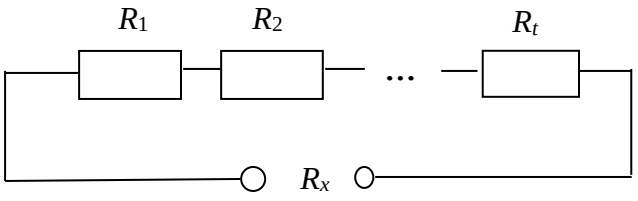


Рис. 5.9 - Схема вимірювальної системи

Загальний опір $R_x = \sum_{i=1}^{n} R_i$, який визначається безпосереднім вимірюванням, для системи, показаної на рис. 5.9, потрібно представити у вигляді суми шуканих опорів R_i. Знайдемо кількість можливих комбінацій $P = \prod_{i=1}^{n} p_i$ та базисні числа m_i. Далі виконується така послідовність дій:

$$X=R_x \bmod R_0,\ Y=\frac{XP}{R_0},\ c_i=(m_i \cdot Y) \bmod p_i. \qquad (5.22)$$

Шукані опори кожного резистора визначаються з виразу $R_i=c_i \cdot \Delta R_i$.

Для прикладу візьмемо три резистори опором R_0=10 Ом, p_1=3, p_2=4, p_3=5. Тоді P=60; ΔR_1=3,33 Ом; ΔR_2=2,5 Ом; ΔR_3=2 Ом; m_1=2, m_2=3, m_3=3. Згідно (5.21), діапазон вимірювання лежить в межах від 0 до 22,17 Ом.

Нехай прилад зафіксував величину R_x=12,83 Ом. Як наслідок отримуємо $X=12{,}83 \bmod 10=2{,}83$; $Y=\frac{2{,}83 \cdot 60}{10} \approx 17$; $c_1=(2 \cdot 17) \bmod 3=1$; $c_2=(3 \cdot 17) \bmod 4=3$; $c_3=(3 \cdot 17) \bmod 5=1$. Тоді шукані величини R_1 =1·3,33 Ом=3,33 Ом; R_2 =3·2,5 Ом=7,5 Ом; R_3 =1·2 Ом=2 Ом.

Шукані опори R_i можна шукати за допомогою таблиць. Це пришвидшує отримання результатів, однак вимагає застосування більшого об'єму пам'яті. Відповідно до попереднього прикладу, будуємо табл. 5.20 [326]. У першому стовпчику знаходяться усі можливі значення параметра X, розміщені в порядку зростання. Величина Y у другому стовпчику, отримана за допомогою (5.22), вказує на порядковий номер відповідного параметра X. В останньому, дев'ятому стовпчику вказані можливі значення повного опору, які може зафіксувати прилад.

Якщо вимірювальний прилад зафіксував R_x=12,83 Ом, то $X=12{,}83 \bmod 10 = 2{,}83$. Знаходимо дану величину в табл. 6.1. Їй відповідають величиниc_1=1, c_2=3, c_3=1. Отже, шукані значення R_1 =3,33 Ом; R_2 =7,5 Ом; R_3 =2 Ом.

На рис. 5.10 представлений графік залежності зміни загального опору R_x від вказаного параметра X, який змінюється з кроком 0,1667 при p_1=3, p_2=4, p_3=5. З рис. 5.10 видно, що графік для опору R_x носить стрибкоподібний характер, причому максимуми (крім двох значень) та мінімуми змінюються лінійно.

Таблиця 5.20 - Побудова БПС табличним методом для p_1=3, p_2=4, p_3=5

X	Y	c_1	R_1	c_2	R_2	c_3	R_3	R_x
1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0
0,17	1	2	6,67	3	7,5	3	6	20,17
0,33	2	1	3,33	2	5	1	2	10,33
0,5	3	0	0	1	2,5	4	8	10,5
0,67	4	2	6,67	0	0	2	4	10,67
0,83	5	1	3,33	3	7,5	0	0	10,83
1	6	0	0	2	5	3	6	11
1,17	7	2	6,67	1	2,5	1	2	11,17
1,33	8	1	3,33	0	0	4	8	11,33
1,5	9	0	0	3	7,5	2	4	11,5
1,67	10	2	6,67	2	5	0	0	11,67
1,83	11	1	3,33	1	2,5	3	6	11,83
2	12	0	0	0	0	1	2	2
2,17	13	2	6,67	3	7,5	4	8	22,17
2,33	14	1	3,33	2	5	2	4	12,33
2,5	15	0	0	1	2,5	0	0	2,5
2,67	16	2	6,67	0	0	3	6	12,67
2,83	17	1	3,33	3	7,5	1	2	12,83
3	18	0	0	2	5	4	8	13
3,17	19	2	6,67	1	2,5	2	4	13,17
3,33	20	1	3,33	0	0	0	0	13,33
3,5	21	0	0	3	7,5	3	6	13,5
3,67	22	2	6,67	2	5	1	2	13,67
3,83	23	1	3,33	1	2,5	4	8	13,83
4	24	0	0	0	0	2	4	4
4,17	25	2	6,67	3	7,5	0	0	14,17
4,33	26	1	3,33	2	5	3	6	14,33
4,5	27	0	0	1	2,5	1	2	4,5
4,67	28	2	6,67	0	0	4	8	14,67

Продовження таблиці 5.20

1	2	3	4	5	6	7	8	9
4,83	29	1	3,33	3	7,5	2	4	14,83
5	30	0	0	2	5	0	0	5
5,17	31	2	6,67	1	2,5	3	6	15,17
5,33	32	1	3,33	0	0	1	2	5,33
5,5	33	0	0	3	7,5	4	8	15,5
5,67	34	2	6,67	2	5	2	4	15,67
5,83	35	1	3,33	1	2,5	0	0	5,83
6	36	0	0	0	0	3	6	6
6,17	37	2	6,67	3	7,5	1	2	16,17
6,33	38	1	3,33	2	5	4	8	16,33
6,5	39	0	0	1	2,5	2	4	6,5
6,67	40	2	6,67	0	0	0	0	6,67
6,83	41	1	3,33	3	7,5	3	6	16,83
7	42	0	0	2	5	1	2	7
7,17	43	2	6,67	1	2,5	4	8	17,17
7,33	44	1	3,33	0	0	2	4	7,33
7,5	45	0	0	3	7,5	0	0	7,5
7,67	46	2	6,67	2	5	3	6	17,67
7,83	47	1	3,33	1	2,5	1	2	7,83
8	48	0	0	0	0	4	8	8
8,17	49	2	6,67	3	7,5	2	4	18,17
8,33	50	1	3,33	2	5	0	0	8,33
8,5	51	0	0	1	2,5	3	6	8,5
8,67	52	2	6,67	0	0	1	2	8,67
8,83	53	1	3,33	3	7,5	4	8	18,83
9	54	0	0	2	5	2	4	9
9,17	55	2	6,67	1	2,5	0	0	9,17
9,33	56	1	3,33	0	0	3	6	9,33
9,5	57	0	0	3	7,5	1	2	9,5
9,67	58	2	6,67	2	5	4	8	19,67
9,83	59	1	3,33	1	2,5	2	4	9,83

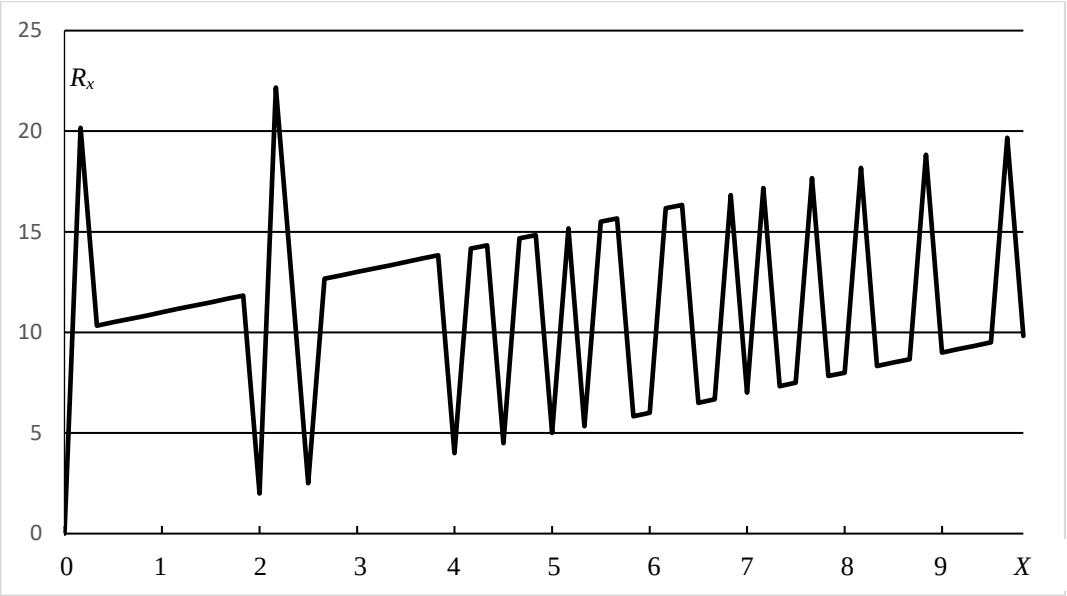


Рис. 5.10 - Графік залежності зміни загального опору R_x від параметра X при p_1=3, p_2=4, p_3=5

Для досягнення приблизно однакової точності вимірювання опору кожного резистора вибрані взаємно прості модулі не повинні відрізнятися дуже сильно. Прикладом може бути набір з трьох послідовних чисел, зокрема, p_1=99, p_2=100, p_3=101 для R_0=100 Ом. За умови більшої кількості модулів їх можна вибрати таким чином: p_1=101, p_2=102, p_3=103, p_4=107, p_5=109, … .

З точки зору теорії чисел для істотного зменшення кількості обчислень модулі бажано вибрати так, щоб вони утворювали ДФ СЗК, для яких m_i=1. Це дозволяє уникнути виконання досить громіздкої операції знаходження оберненого елемента за модулем $m_i = M_i^{-1} \bmod p_i$ та множення Y на базисні числа m_i, оскільки

$$c_i = Y \bmod p_i. \qquad (5.23)$$

У табл. 5.21 представлено усі можливі значення відповідних параметрів при модулях, які утворюють ДФ СЗК [326]: p_1=2, p_2=3, p_3=5, опір R_0=10 Ом. Звідси ΔR_1=5 Ом; ΔR_2=3,33 Ом; ΔR_3=2 Ом.

Таблиця 5.21 - Побудова БПС табличним методом для p_1=2, p_2=3, p_3=5

X	Y	c_1	R_1	c_2	R_2	c_3	R_3	R_x
0	0	0	0	0	0	0	0	0
0,33	1	1	5	1	3,33	1	2	10,33
0,67	2	0	0	2	6,67	2	4	10,67
1	3	1	5	0	0	3	6	11
1,33	4	0	0	1	3,33	4	8	11,33
1,67	5	1	5	2	6,67	0	0	11,67
2	6	0	0	0	0	1	2	2
2,33	7	1	5	1	3,33	2	4	12,33
2,67	8	0	0	2	6,67	3	6	12,67
3	9	1	5	0	0	4	8	13
3,33	10	0	0	1	3,33	0	0	3,33
3,67	11	1	5	2	6,67	1	2	13,67
4	12	0	0	0	0	2	4	4
4,33	13	1	5	1	3,33	3	6	14,33
4,67	14	0	0	2	6,67	4	8	14,67
5	15	1	5	0	0	0	0	5
5,33	16	0	0	1	3,33	1	2	5,33
5,67	17	1	5	2	6,67	2	4	15,67
6	18	0	0	0	0	3	6	6
6,33	19	1	5	1	3,33	4	8	16,33
6,67	20	0	0	2	6,67	0	0	6,67
7	21	1	5	0	0	1	2	7
7,33	22	0	0	1	3,33	2	4	7,33
7,67	23	1	5	2	6,67	3	6	17,67
8	24	0	0	0	0	4	8	8
8,33	25	1	5	1	3,33	0	0	8,33
8,67	26	0	0	2	6,67	1	2	8,67
9	27	1	5	0	0	2	4	9
9,33	28	0	0	1	3,33	3	6	9,33
9,67	29	1	5	2	6,67	4	8	19,67

Відповідно, максимальна величина вимірювання D=19,67 Ом. Нехай прилад зафіксував R_x=13,67 Ом. Тоді X=13,67 mod 10=3,67; $Y = \frac{3{,}67 \cdot 30}{10} = 11$; c_1=11*mod* 2=1; c_2=11*mod* 3=2; c_3=11*mod* 5=1. Шукані значення R_1 =1·5=5 Ом; R_2=2·3,33=6,67 Ом; R_3=1·2=2 Ом. Такі ж величини можна отримати за допомогою табл. 5.21.

Недоліком даного методу є те, що у ДФ СЗК модулі дуже швидко зростають і, відповідно, істотно відрізняються точності вимірювання у різних точках.

На рис. 5.11 представлений графік залежності для зміни загального опору R_x від параметра X, який змінюється з кроком 0,33 при значеннях модулів p_1=2, p_2=3, p_3=5.

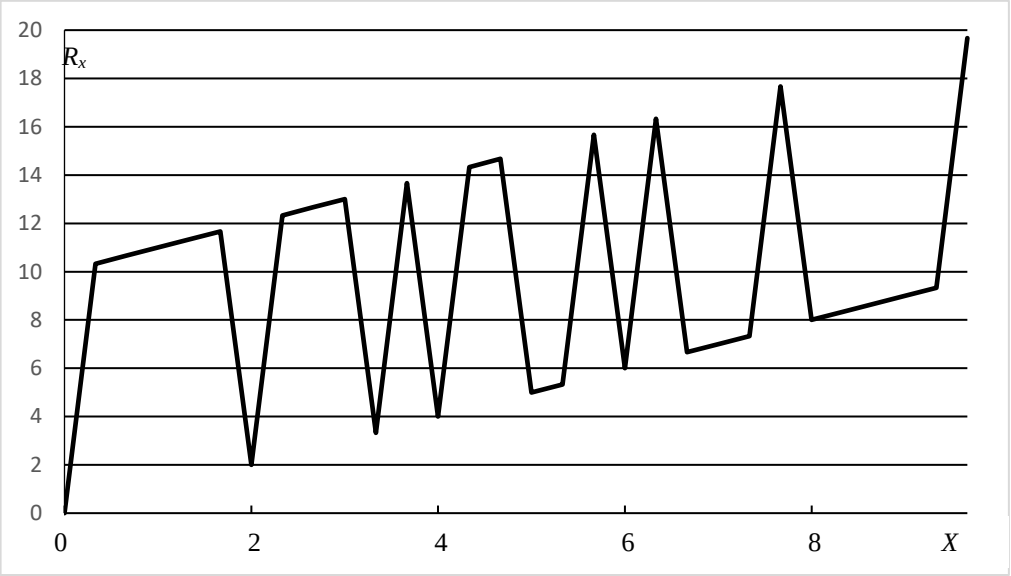


Рис. 5.11 - Графік залежності зміни загального опору R_x від параметра X при p_1=2, p_2=3, p_3=5

З рис. 5.11 видно, що графік для R_x теж носить стрибкоподібний характер, однак, на відміну від попереднього, усі максимуми та мінімуми змінюються лінійно. Крім того, графік є обернено симетричним відносно середини діапазону осі абсцис. При малих значеннях X спостерігаються ширші максимуми, при великих (більше середини) – ширші мінімуми.

Якщо вимірювання необхідно виконати лише у двох точках, то набір модулів зручно вибрати у вигляді двох великих послідовних чисел, що забезпечує приблизно однакову високу точність. Такі модулі утворюють МДФ СЗК, для якої виконується умова $m_i = M_i^{-1} \bmod p_i = \pm 1$.

Це також дозволяє уникнути пошуку оберненого елемента за модулем та множення Y на базисні числа, оскільки $m_1=p_2 mod\ p_1=(p_1+1)\ mod\ p_1=1$, $m_2=p_1\ mod\ (p_1+1)= =-1\ mod\ (p_1+1)=p_1$. Звідси:

$$C_1=Y\ mod\ p_1,\ c_2=(p_2-Y\ mod\ p_2). \qquad (5.24)$$

У табл. 5.22 для прикладу представлено можливі значення відповідних параметрів при $p_1=5$, $p_2=6$, $R_0=10$ Ом, $\Delta R_1=2$ Ом; $\Delta R_2=1{,}67$ Ом.

Максимальна величина, яку може вказати вимірювальний прилад, $D=16{,}33$ Ом. Тоді $m_1=1$, $m_2=5 \bmod 6=-1 \bmod 6$.

Нехай, наприклад, прилад зафіксував величину $R_x=12{,}67$ Ом. Аналітичним методом будемо мати: $X=12{,}67 \bmod 10=2{,}67$; $Y=\dfrac{2{,}67\cdot 30}{10}=8$; $c_1=8 \bmod 5=3$; $c_2=6-(8 \bmod 6)=4$. Шукані значення $R_1=3\cdot 2=6$ Ом; $R_2=4\cdot 1{,}67=6{,}67$ Ом. За допомогою таблиці 5.22 отримується той самий результат.

На рис. 5.12 представлений графік залежності зміни загального опору R_x від параметра X, який змінюється з кроком 0,33 при $p_1=5$, $p_2=6$.

З рис. 5.12 видно, що кількість максимумів зменшилася, а при збільшенні X мінімуми стають ширшими.

Якщо вимірювання необхідно провести у більш, ніж двох точках, то модулі слід вибрати так, щоб вони утворювали МДФ СЗК і неістотно відрізнялися один від одного.

Таблиця 5.22 - Побудова БПС табличним методом для p_1=5, p_2=6

X	Y	c_1	R_1	c_2	R_2	R_x
0	0	0	0	0	0	0
0,33	1	1	2	5	8,33	10,33
0,67	2	2	4	4	6,67	10,67
1	3	3	6	3	5	1
1,33	4	4	8	2	3,33	11,33
1,67	5	0	0	1	1,67	1,67
2	6	1	2	0	0	2
2,33	7	2	4	5	8,33	12,33
2,67	8	3	6	4	6,67	12,67
3	9	4	8	3	5	13
3,33	10	0	0	2	3,33	3,33
3,67	11	1	2	1	1,67	3,67
4	12	2	4	0	0	4
4,33	13	3	6	5	8,33	14,33
4,67	14	4	8	4	6,67	14,67
5	15	0	0	3	5	5
5,33	16	1	2	2	3,33	5,33
5,67	17	2	4	1	1,67	5,67
6	18	3	6	0	0	6
6,33	19	4	8	5	8,33	16,33
6,67	20	0	0	4	6,67	6,67
7	21	1	2	3	5	7
7,33	22	2	4	2	3,33	7,33
7,67	23	3	6	1	1,67	7,67
8	24	4	8	0	0	8
8,33	25	0	0	5	8,33	8,33
8,67	26	1	2	4	6,67	8,67
9	27	2	4	3	5	9
9,33	28	3	6	2	3,33	9,33
9,67	29	4	8	1	1,67	9,67

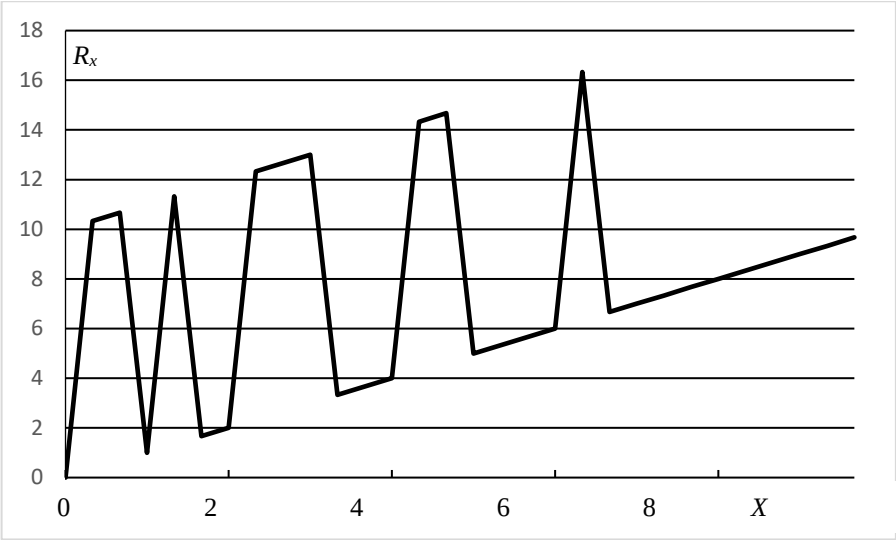


Рис. 5.12 - Графік залежності зміни загального опору R_x від параметра X при p_1=5, p_2=6

Для прикладу візьмемо чотири резистори, вибравши модулі таким чином: p_1=8, p_2=9, p_3=11, p_4=13 і P=10296. Легко перевірити, що m_1=7mod 8=–1 mod 8, m_2=1, m_3=1, m_4=12mod 13=–1 mod 13, тобто вибрані модулі задовольняють умову МДФ СЗК. Для R_0=10 Ом будемо мати: ΔR_1=1,25 Ом; ΔR_2=1,11 Ом; ΔR_3=0,91 Ом; ΔR_4=0,77 Ом. Нехай вимірювальний прилад зафіксував величину R_x=11,55 Ом. Тоді аналогічно до формул (5.24) можна отримати шукані значення резисторів: X=11,55mod10=1,55; $Y = \frac{1{,}55 \cdot 10296}{10} \approx 1596$; c_1=8-1596mod 8=8-4=4; c_2=1596mod 9=3; c_3=1596mod 11=1; c_4=13-1596mod 13=13-10=3; R_1 =4·1,25 Ом=5 Ом; R_2 = 3·1,11 Ом=3,33 Ом; R_3 =1·0,91 Ом=0,91 Ом; R_4=3·0,77 Ом=2,31 Ом. З отриманих результатів видно, що сума знайдених опорів дорівнює значенню, яке показав прилад.

Висновки до п’ятого розділу

1. Розроблено узагальнений алгоритм для побудови для побудови багатомодульної МДФ СЗК на основі аналітичних виразів, що забезпечує

зменшення обчислювальної складності в порівнянні із звичайною цілочисельною СЗК за рахунок уникнення виконання відповідних операцій при переведенні чисел із СЗК у позиційну систему числення.

2. Наведено приклад та на основі графічних залежностей досліджено поведінку модулів і діапазону обчислень для чотирьох- та п’ятимодульної МДФ СЗК, що дозволяє обгрунтувати вибір системи модулів в залежності від класу задач, які необхідно розв’язати.

3. Розроблено метод розширення набору модулів МДФ СЗК, що дозволяє за необхідності збільшити діапазон обчислень за рахунок введення нових модулів, які утворюють МДФ СЗК.

4. Розроблено приклад використання МДФ СЗК в технічних системах на основі побудови розподіленого термо- або п’єзоелектричного сенсора за допомогою звичайної цілочисельної та МДФ СЗК, який дозволяє визначити опір кожного резистора при відомому значенні загального опору їх послідовного з’єднання, розроблено рекомендації щодо вибору наборів резисторів з точки зору теорії чисел.

Основні результати п’ятого розділу відображені у роботах [284, 291, 294, 320, 322-323, 326-327].

6 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПОШУКУ МОДУЛІВ РІЗНИХ ФОРМ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ ТА ЇХ ЗАСТОСУВАННЯ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ

У шостому розділі наведена програмна реалізація методів підбору модулів для ДФ та МДФ СЗК, а також розроблено трьохмодульну криптосистему Рабіна на основі звичайної цілочисельної та МДФ СЗК та побудовано її HDL-модель.

6.1 Програмна модель побудови ДФ СЗК

Програма для підбору модулів ДФ СЗК написана на мові Java. Обрана мова програмування має наступні переваги: простий синтаксис; повна підтримка об’єктно орієнтованого підходу; відносно висока продуктивність; архітектурна незалежність і портативність.

Для факторизації чисел можна використати одну зі сторонніх бібліотек, наприклад: GMP-ECM. Функція primeFactor виконує факторизацію чисел.

Роботу програми складається з таких етапів:

1) знаходження добутку модулів згідно формули;
2) факторизація отриманого добутку;
3) перебір всіх можливих комбінацій отриманих чисел;
4) перевірка комбінацій на цілочисельнісь модулів;
5) обчислення модулів для чисел, що відповідають умові.

Клас для пошуку модулів містить наступні функції:

– calcAB – функція для обчислення добутку a*b. Повертає число, що буде факторизватися. Параметри функції: m – масив відомих модулів.

– calcP45 – функція для обчислення невідомих модулів P4 і P5. Параметри функції: ab – один з добутків; m – масив відомих модулів;

– test – функція для перевірки модулів. Дана функція повертає true, якщо модуль – ціле число. Параметри функції: ab – добуток факторизованих чисел; m – масив відомих модулів;

– generateModules – виконує перебір усіх комбінацій множників;

– calculate – головна функція, результатом роботи якої є список модулів.

Функції calcP45, calcAB, test працюють відповідно до формул (3.35)-(3.37). Змінна list містить список множників факторизованого числа. У масиві modules зберігаються відомі модулі P0-P3. Алгоритм роботи даної програми представлений у вигляді блок–схеми на рис. 6.1.

Нижче наведено фрагмент програми для генерування модулів досконалої форми:

```
private static TreeSet<Pair> generateModules(LinkedList<Integer> list,
        int[] modules) {
    TreeSet<Pair> data = new TreeSet<>(new Comparator<Pair>() {
        public int compare(Pair p1, Pair p2) {
            return p1.a - p2.a + p1.b - p2.b;
    // додаємо до множників одиницю
    if (!list.contains(1))
        list.add(1);
    // перебір можливих комбінацій множників
    for (int count = 0; count < list.size(); count++) {
        for (int i = 1; i < list.size(); i++) {
            int dobA = 1, dobB = 1;
            int j = 0;
            // порахувати добутки
            for (Integer el : list) {
                if (j < i) {
                    dobA *= el;
                else {
                    dobB *= el;
                j++;
  // перевірити чи задовільняють числа задану умову і додати в список
            if (test(dobA, modules) && test(dobB, modules)) {
                int p4 = calcP45(dobA, modules);
                int p5 = calcP45(dobB, modules);
                data.add(new Pair(p4, p5));
        // перемістити останній елемент на перше місце
        int last = list.getLast();
        list.removeLast();
        list.addFirst(last);
    return data;
```

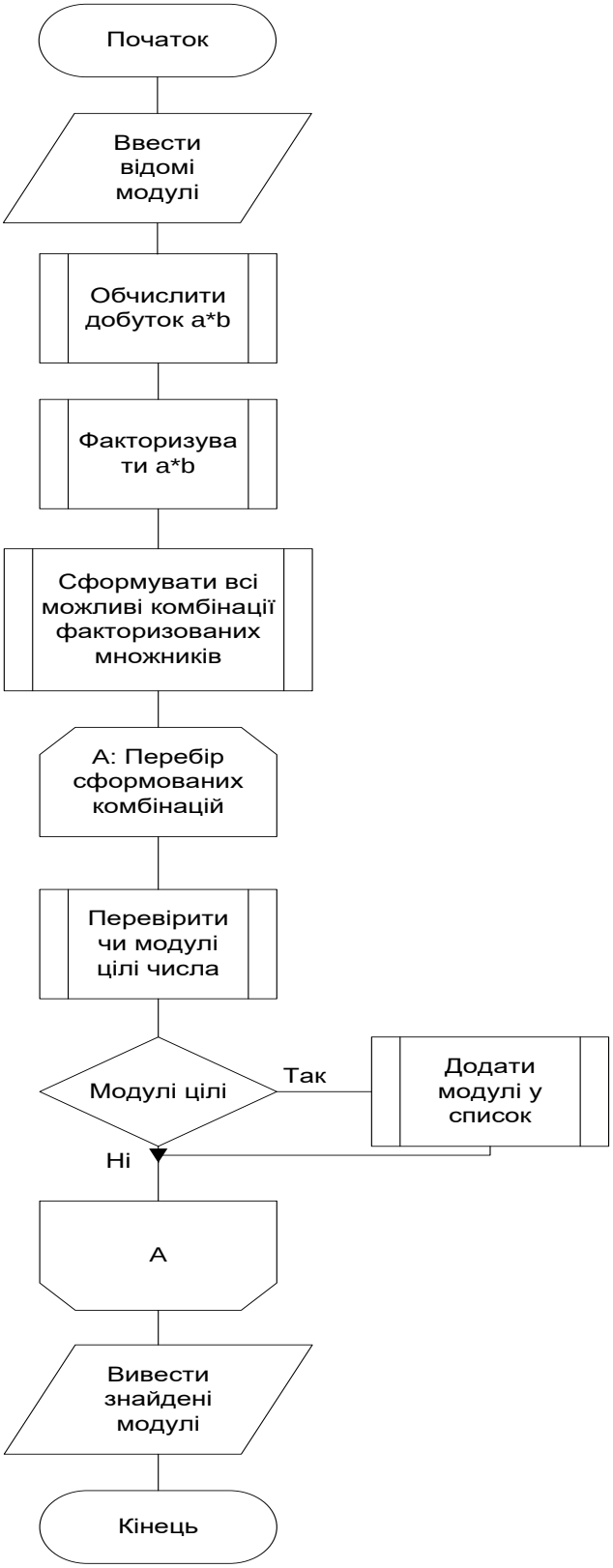


Рис. 6.1 - Блок схема алгоритму програми

На рис. 6.2 показане головне вікно програми, що призначене для вводу модулів.

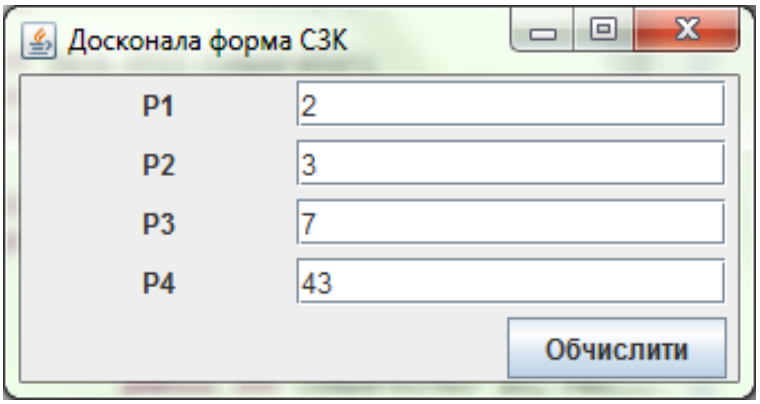


Рис. 6.2 - Головне вікно програми

Після натискання на кнопку «Обчислити» програма буде виводити діалогове вікно із знайденими модулями (рис. 6.3), які задовольняють умову ДФ СЗК.

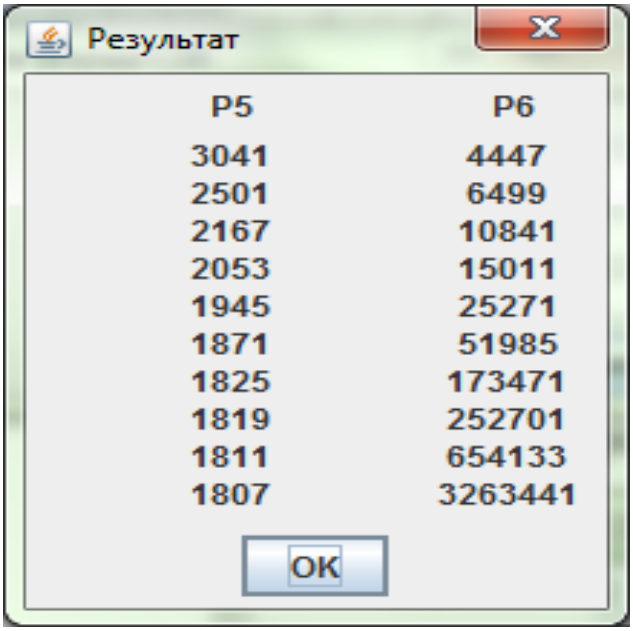


Рис. 6.3 - Вивід результатів роботи програми

Для зберігання модулів використовується модуль TreeSet, що являє собою структуру даних для зберігання упорядкованих елементів множини. У

даній структурі даних будуть зберігатися об’єкти класу Pair. Щоб уникнути повторюваних елементів в конструктор передається його власний компаратор. Компаратор знаходить сумарну різницю елементів. Якщо елементи рівні, результатом роботи компаратора буде 0. Також необхідно додати до множників 1, оскільки це число також буде частиною результату під час факторизації.

Тепер заповнюється модуль TreeSet можливими комбінаціями множників. При цьому множники у кожному випадку повинні бути розділені на дві множини A та B, де $B=S/A$, S – це множники факторизованого добутку ab.

Щоб виконати перебір, необхідно в циклі збільшувати кількість елементів одної з множин і паралельно перелічувати всі можливі варіанти комбінацій множників для даної кількості елементів у множинах. Для цього на кожній ітерації виконується переставляння останнього множника у множині на перше місце шляхом послідовного виклику методів getLast, removeLast, addFirst.

Для кожної сформованої комбінації обчислюється добуток елементів для кожної множини. Після цього перевіряється цей добуток на його відповідність умові (3.37). Якщо обидва числа задовольняють умову, то обчислюються модулі за формулою (3.35) і зберігаються у список з результатами.

В множині TreeSet зберігаються об’єкти класу Pair. Даний об’єкт містить два цілочисельних поля a і b. Конструктор даного класу написаний так, щоб числа a і b завжди ініціалізувались згідно умови $a<b$.

Функція calculate спочатку обчислює добуток ab за допомогою виклику функції calcAB. Далі отриманий добуток факторизується. Після цього перевіряється кількість елементів отриманої множини множників. Якщо кількість елементів більша одиниці (число не просте), то викликається функція для генерування модулів generateModules. В іншому випадку повертається null.

6.2 Програмна реалізація методу побудови МДФ СЗК

Програма також написана на мові Java. Для факторизації чисел можна використати одну зі сторонніх бібліотек, наприклад: GMP-ECM. Функція primeFactor виконує факторизацію чисел.

Робота програми складається з таких етапів: знаходження добутку модулів; факторизація отриманого добутку; перебір всіх можливих комбінацій отриманих чисел; обчислення модулів для чисел, що відповідають умові МДФ СЗК.

При реалізації програми потрібно вирішити наступні задачі:

1) написати ефективний алгоритм перебору всіх можливих добутків ab;

2) розробити універсальну архітектуру, яка буде дозволяти підтримувати окремі реалізації обчислення невідомих модулів при заданій їх кількості. Логіку, яка є спільною для всіх реалізацій, необхідно винести в окремий клас. Для забезпечення універсальності потрібно розробити відповідні інтерфейси;

3) розробити клас, що забезпечить зберігання і сортування модулів, та клас, що буде зберігати значення модулів (Java Bean).

Алгоритм роботи даної програми представлений у вигляді блок-схеми на рис. 6.4.

Як видно з блок-схеми, обчислення добутку ab і обчислення невідомих x, y представлені у вигляді наперед визначених процесів. Блок-схема описує загальний алгоритм пошуку для будь-якої кількості модулів. На рис. 6.5 наведені блок-схеми, що описують зазначені процеси обчислення для 4-ох модулів.

Для зручної заміни реалізації потрібно оголосити наступні інтерфейси:

- Analizer містить метод analize для перевірки факторизованого добутку ab. У якості парамерів приймає два цілих числа: a і b відповідно;

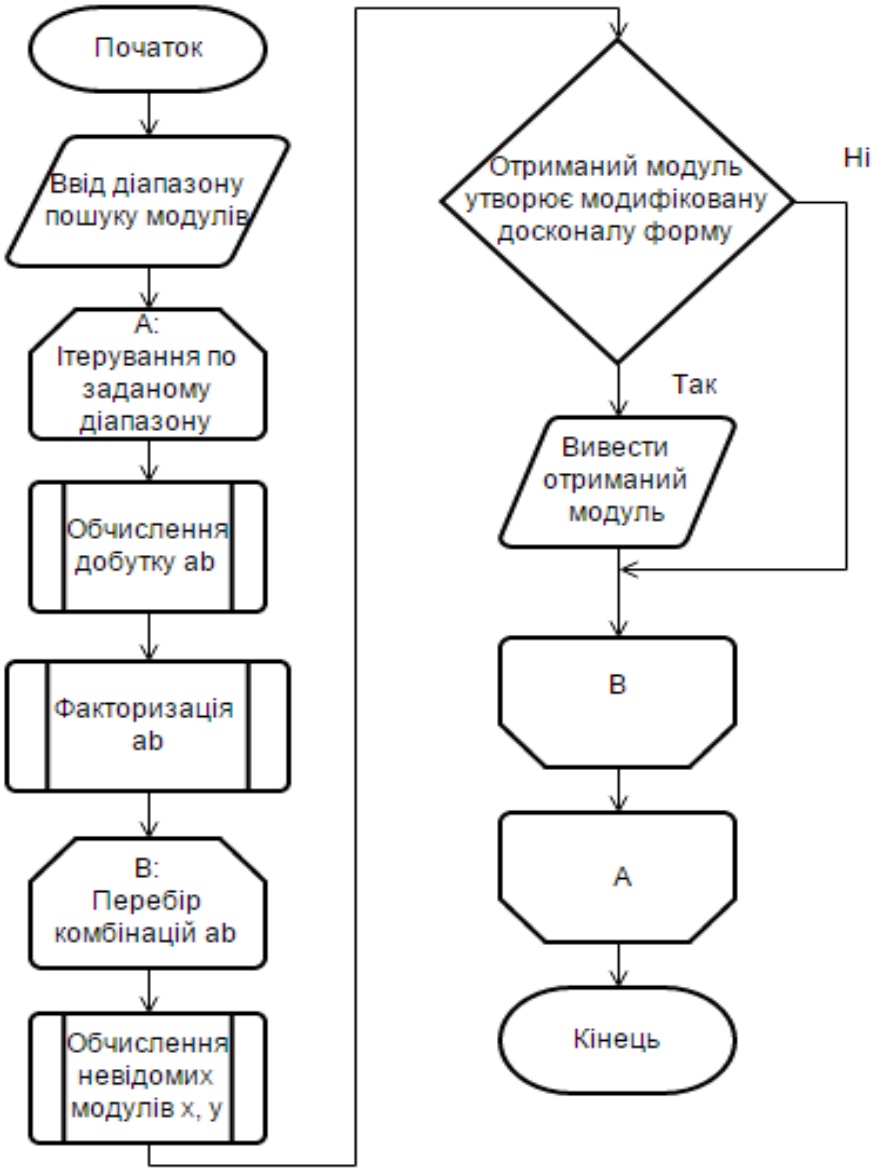


Рис. 6.4 - Загальна блок-схема алгоритму програми

- Permutation є абстракцією класу для генерування всеможливих добутків числа *ab*, яке потрібно факторизувати. Цей інтерфейс має єдиний метод, що у якості параметрів приймає вектор чисел (факторизований добуток *ab*) і об’єкт, що реалізує інтерфейс Analizer. Метод даного об’єкту буде викликаний для кожного обчисленого добутку.

На основі блок-схеми можна виділити наступні класи:

- HpfModules - Java Bean для представлення об’єктів модулів модифікованої досконалої форми СЗК. Даний об’єкт містить поле для

зберігання масиву чисел і метод isHalfPerfect. Цей метод повертає true, якщо модулі, що містяться в об’єкті класу, утворюють МДФ СЗК. Також клас містить статичний метод для перевірки коректності модулів validate, що приймає екземпляр класу HpfModules, і повертає true, якщо модулі не дублюються і не дорівнюють 0;

- ModulesStore - виконує зберігання і сортування знайдених модулів. Найважливіший метод getSortedModules() - повертає множину Set<HpfModules>;

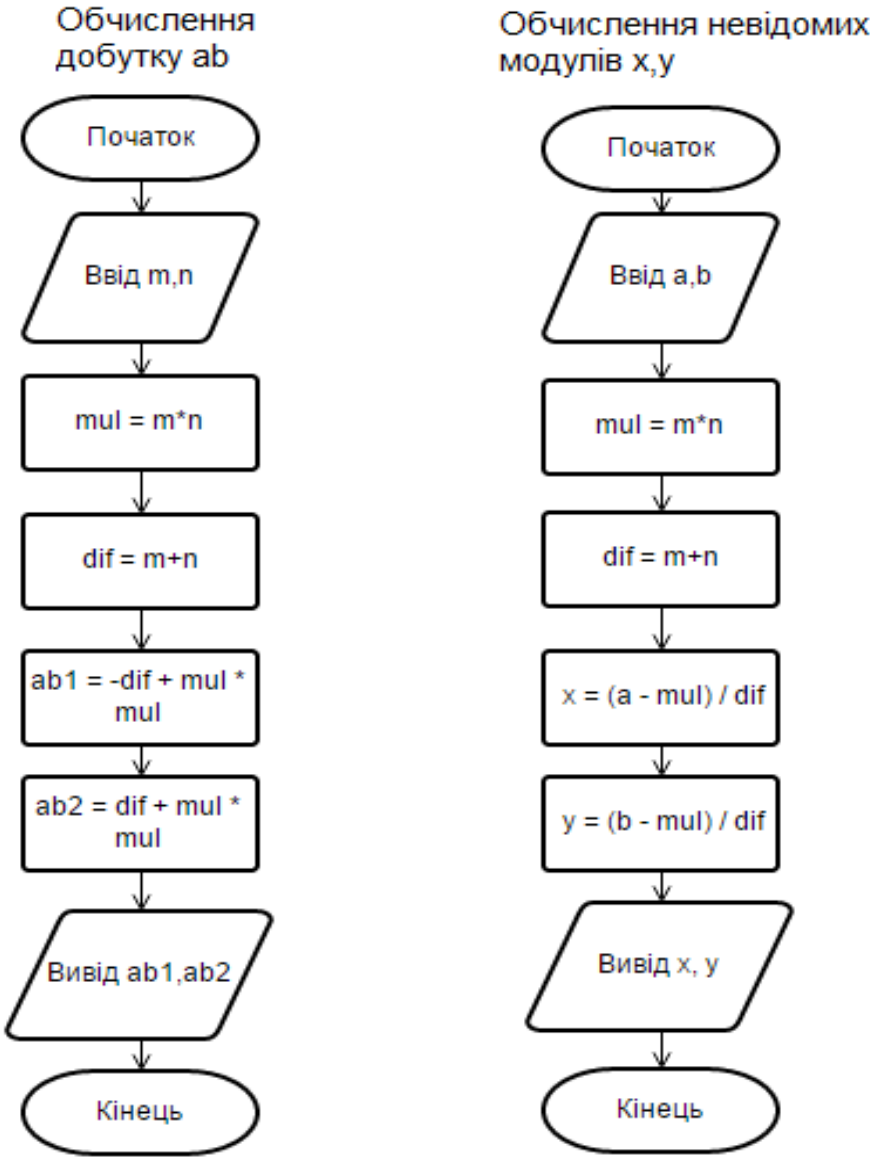


Рис. 6.5 - Блок-схеми алгоритму для обчислення *ab* і невідомих *x, y* для 4-х модулів

- PrimeFactor - містить метод calculate(). Даний метод приймає у якості параметра число, яке буде факторизоване. Результатом роботи методу є масив цілих чисел, на які було розкладено аргумент;

- RecursivePermutation - реалізує інтерфейс Permutation. Виконує перебір можливих комбінацій множників ab за допомогою оптимізованого алгоритму рекурсивним шляхом;

- ThreeModulesCalculator, FourModulesCalculator, FiveModulesCalculator - резалізації інтерфейсу ModulesCalculator. Дані класи містять логіку обчислення для 3, 4 і 5 модулів МДФ СЗК відповідно. Дані класи також реалізують інтерфейс Analize. Такий підхід забезпечує універсальність при використанні класу RecursivePermutation із різними реалізація ModulesCalculator. З іншої сторони реалізація ModulesCalculator не прив’язана до частини, за яку відповідає клас Analize, тому метод analize при зростанні складності може бути перенесений в окремий клас;

- Main, ResultDialog - відповідають за графічний інтерфейс налаштування пошуку і відображення результатів.

Нижче наведено фрагмент програми для генерування модулів МДФ СЗК:

```
public void calculate(int n, int m) {
    this.m = m;
    this.n = n;
    dif = n - m;
    mul = n * m;

    int ab = (-dif + mul * mul);
    processAb(ab);
    ab = (dif + mul * mul);
    processAb(ab);
}

private void processAb(int ab) {
    int[] primes = PrimeFactor.calculate(ab);

    permutation.permutate(primes, this);
}
@Override
public void analyze(int a, int b) {
```

```
        simpleAnalize(a, b);
        //we should count -1 also
        //but multiply should be positive so we have 2 minuses
        simpleAnalize(-a, -b);
    }
    private void simpleAnalize(int a, int b) {
        //maybe here should be a minus before a & b
        int x = (a - mul) / dif;
        int y = (b - mul) / dif;
        Integer[] mods = new Integer[]{n, m, x, y};
        if (HpfModules.validate(mods)) {
            HpfModules module = new HpfModules(mods);
            if (module.isHalfPerfect()) {
                store.addModule(module);
            }
```

Наведений фрагмент програми ілюструє пошук для 4-х модулів МДФ СЗК. Описані методи належать класу FourModulesCalculator, який містить логіку обчислення добутку *ab* і невідомих x,y. Даний клас містить наступні методи:

- calculate - метод, що приймає відомі модулі m,n. Даний метод ініціює пошук невідомих x,y для заданих m,n;
- processAb - призначений для факторизації добутку a,b і перебору можливих комбінацій його множників. У якості параметру отримує добуток ab. Для перебору множинків ab використовується допоміжний клас RecursivePermutation, що реалізує інтерфейс Permutation. Інтерфейс Permutation є абстракцією класу що дозволяє перебирати можливі множинки заданого вектора. При цьому передбачається, що всі отримані множники будуть передані екземпляру класу Analizer, який повинен провести аналіз отриманих даних;
- analyze - фактично є методом, що проводить аналіз всіх отриманих добутків факторизованого ab, тобто викликає simpleAnalize для двох можливих випадків [*a, b*] і [*-a, -b*];
- simpleAnalize - обчислює невідомі x, y і перевіряє чи вектор [*m, n, x, y*] утворює МДФ СЗК.

На рис. 6.6 показане головне вікно програми, що призначене для налаштування пошуку модулів. Воно містить список вибору кількості модулів і поля для вводу діапазону пошуку для відомих модулів m, n.

Оскільки заміна дозволяє знайти тільки 2 невідомих модулі, то кількість діапазонів їх пошуку, які потрібно задати, буде змінюватися залежно від обраної кількості модулів. Для трьох модулів потрібно задати один діапазон, для чотирьох - два і т.д.

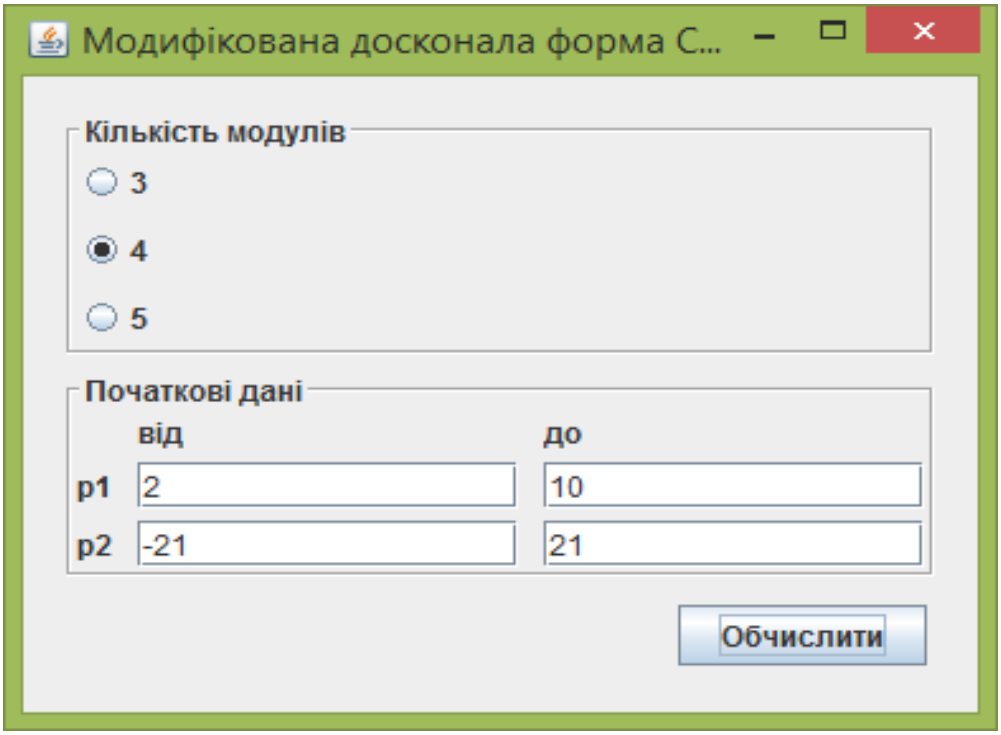


Рис. 6.6 - Головне вікно програми

Після натискання на кнопку «Обчислити» програма виводить діалогове вікно з знайденими модулями (рис. 6.7). Далі програма почне проходить заданий діапазон чисел і шукає модулі для кожного випадку заданого p_i. При цьому, якщо діапазон значний, то обчислення будуть виконуватися значний період часу. Щоб графічний інтерфейс при цьому відповідав на запити користувача, необхідно всі обчислення винести в окремий потік:

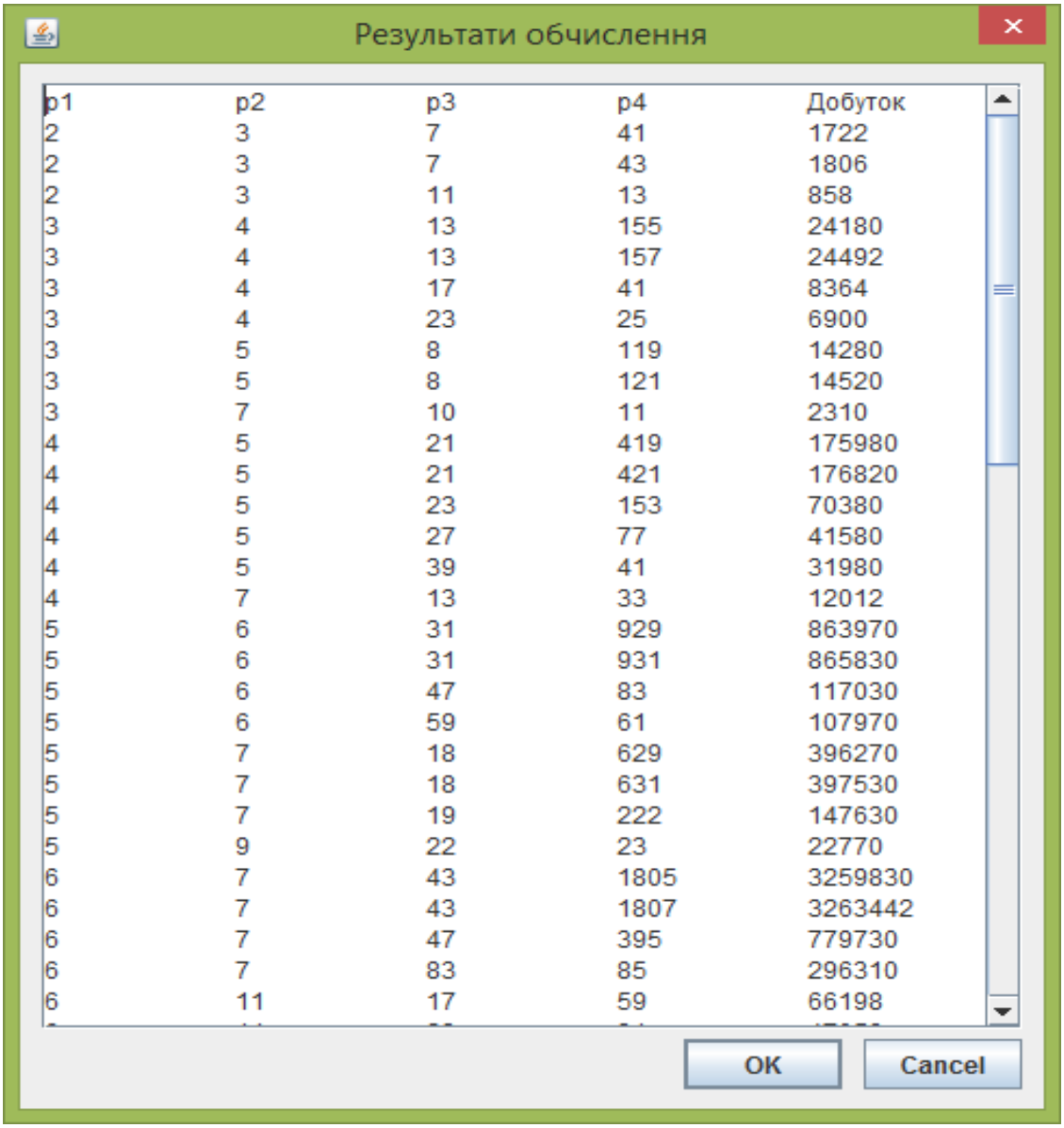


Рис. 6.7 - Вивід результатів роботи програми

```
calculateBtn.addActionListener(event -> {
        Thread calculateThread = new Thread(() -> {
            //обчислення модулів
        }).start();
```

Для зберігання модулів використовується модуль TreeSet, що являє собою структуру даних для зберігання упорядкованих елементів множини. У даній структурі даних будуть зберігатися об’єкти класу Pair. Щоб уникнути повторюваних елементів в конструктор передається його власний компаратор. Компаратор знаходить сумарну різницю елементів. Якщо

елементи рівні, результатом роботи компаратора буде 0. Також необхідно додати до множників 1, оскільки 1 буде частиною результату під час факторизації.

Щоб виконати перебір можливих множників факторизованого *ab*, необхідно в циклі збільшувати кількість елементів однієї із множин і паралельно перелічувати всі можливі варіанти комбінацій множників для даної кількості елементів у множинах. Для цього на кожній ітерації виконується переставляння останнього множника у множині на перше місце шляхом послідовного виклику методів getLast, removeLast, addFirst.

Для кожної сформованої комбінації обчислюється добуток елементів для кожної множини. Після цього перевіряється цей добуток на відповідність умові (5.8). Якщо два числа задовольняють умову, то обчислюються модулі за формулою (5.5) і зберігаються у список з результатами.

6.3 Побудова трьохмодульної криптосистеми Рабіна на основі різних форм СЗК

Для шифрування інформаційних потоків з використання трьохмодульної криптосистеми Рабіна вибирається три великих простих числа *p, q i r* [294, 328-330]. Обчислюється значення $n=p\cdot q\cdot r$, де число n — відкритий ключ, а p, q і r – закритий.

Шифрування повідомлення A (текст) відбувається за допомогою відкритого ключа n за формулою (1.16).

При дешифруванні криптограми A' вводяться додаткові допоміжні величини k, l *i* d:

$$k = A' \bmod p;\ l = A' \bmod q;\ d = A' \bmod r \qquad (6.1)$$

Значення x, y і z шукаються з порівнянь:

$$x^2 \equiv k(\mathrm{mod}\, p), \tag{6.2}$$

$$y^2 \equiv l(\mathrm{mod} q), \tag{6.3}$$

$$z^2 \equiv d(\mathrm{mod} r). \tag{6.4}$$

Для знаходження x, y і z необхідно обчислити значення кореня квадратного за модулем. Класичні підходи з використання символів Якобі або Лежандра є трудомісткими [312]. Тому пропонується метод, аналогічний до методу пошуку оберненого елемента на основі додавання модуля, який вимагає тільки операції додавання та перевірки, чи є число повним квадратом, що суттєво зменшує обчислювальну складову методу Рабіна. Отже, для того, щоб знайти значення $\sqrt{k} \bmod p$, необхідно виконати наступну послідовність дій: $k + p$, $k + 2p$, …, $k + i \cdot p$, де i - значення, при якому $k + i \cdot p$ буде повним квадратом. Аналогічним чином шукається $y^2 \equiv l(\mathrm{mod} q)$, $z^2 \equiv d(\mathrm{mod} r)$. Оскільки розв'язками порівнянь (6.2)-(6.4) буде 6 значень, то для дешифрування потрібно розв'язати вісім систем порівнянь, що утворюються як комбінації можливих варіантів пошуку відкритого повідомлення [329]:

$$\begin{cases} A_1 \equiv x(\mathrm{mod}\, p); \\ A_1 \equiv y(\mathrm{mod} q); \\ A_1 \equiv z(\mathrm{mod} r); \end{cases} \begin{cases} A_2 \equiv -x(\mathrm{mod}\, p); \\ A_2 \equiv y(\mathrm{mod} q); \\ A_2 \equiv z(\mathrm{mod} r); \end{cases} \begin{cases} A_3 \equiv x(\mathrm{mod}\, p); \\ A_3 \equiv -y(\mathrm{mod} q); \\ A_3 \equiv z(\mathrm{mod} r); \end{cases} \begin{cases} A_4 \equiv x(\mathrm{mod}\, p); \\ A_4 \equiv y(\mathrm{mod} q); \\ A_4 \equiv -z(\mathrm{mod} r); \end{cases} \tag{6.5}$$

$$\begin{cases} A_5 \equiv -x(\mathrm{mod}\, p); \\ A_5 \equiv -y(\mathrm{mod} q); \\ A_5 \equiv z(\mathrm{mod} r); \end{cases} \begin{cases} A_6 \equiv -x(\mathrm{mod}\, p); \\ A_6 \equiv y(\mathrm{mod} q); \\ A_6 \equiv -z(\mathrm{mod} r); \end{cases} \begin{cases} A_7 \equiv x(\mathrm{mod}\, p); \\ A_7 \equiv -y(\mathrm{mod} q); \\ A_7 \equiv -z(\mathrm{mod} r); \end{cases} \begin{cases} A_8 \equiv -x(\mathrm{mod}\, p); \\ A_8 \equiv -y(\mathrm{mod} q); \\ A_8 \equiv -z(\mathrm{mod} r). \end{cases}$$

Одне з рішень систем порівнянь (6.5) буде шуканим повідомленням A.

Для вирішення задачі знаходження розв'язків (6.5) потрібно для кожної системи використати КТЗ, тобто для першої системи шукане число A

представлене у вигляді набору $(x, y, z)_{p, q, r}$ найменших невід'ємних залишків від ділення цього числа на фіксовані натуральні попарно взаємно прості числа p, q, r ($x = A_1 \bmod p$, $y = A_1 \bmod q$, $z = A_1 \bmod r$), які є відкритими ключами. При цьому повинна виконуватись умова $0 \le A_1 < n-1$.

Знаходження A_1 згідно КТЗ є досить складним та громіздким процесом:

$$A_1 = (x \cdot B_1 + y \cdot B_2 + z \cdot B_3) \bmod n, \qquad (6.6)$$

де $B_i = S_i m_i$, $i = 1, 2, 3$, $S_1 = \frac{n}{p}$, $S_2 = \frac{n}{q}$, $S_3 = \frac{n}{r}$, m_i шукається з виразів $(S_1 m_1) \bmod p = 1$, $(S_2 m_2) \bmod q = 1$, $(S_3 m_3) \bmod r = 1$.

Задача суттєво спрощується, коли модулі p, q, r підібрано таким чином, що вони утворюють МДФ СЗК, тобто $m_i = \pm 1$. Це дозволяє уникнути виконання громіздкої процедури пошуку оберненого елемента за модулем та множення в (6.6) на базисні числа B_i.

Наведемо приклад. Нехай $p=11$, $q=13$, $r=17$, $n= 2431$, відкритий текст $A=1031$. Зашифроване повідомлення: $A' = 1031^2 \bmod 2431 = 614$. Розшифровування буде здійснюватися в такій послідовності: $k = 614 \bmod 11 = 9$, $l = 614 \bmod 13 = 3$, $d = 614 \bmod 17 = 2$. Далі необхідно обчислити квадратичні лишки $x^2 \equiv 9 (\bmod 11)$, $y^2 \equiv 3 (\bmod 13)$, $z^2 \equiv 2 (\bmod 17)$. Для даного прикладу $\sqrt{9} \bmod 11 = 3$ і 11-3=8, $\sqrt{3+13} \bmod 13 = 4$ та 9, $\sqrt{2 + 2 \cdot 17} \bmod 17 = 6$ та 11. З отриманих коренів можна утворити 8 різних комбінацій: (3, 4, 6), (3, 4, 11), (3, 9, 6), (3, 9, 11), (8, 4, 6), (8, 4, 11), (8, 9, 6), (8, 9, 11).

Для застосування КТЗ потрібно знайти значення m_i. Стосовно вибраного прикладу на основі методу додавання модуля можна знайти, що $S_1 = 13 \cdot 17 = 221$, $(S_1 m_1) \bmod p = 221 \cdot m_1 \bmod 11 = 1 \cdot m_1 \bmod 11 = 1$, звідси $m_1 = 1$.

Аналогічно S_2=11·17=187, $(S_2 m_2) \bmod q = 187 \cdot m_2 \bmod 13 = 5 \cdot m_2 \bmod 13 = 1$,

тоді $m_2 = \frac{(1+3\cdot 13)}{5} = 8$ і S_3=11·13=143,

$(S_3 m_3) \bmod r = 143 \cdot m_3 \bmod 17 = 7 \cdot m_3 \bmod 17 = 1$, $m_3 = \frac{(1+2\cdot 17)}{7} = 5$.

Згідно (6.5) отримуються такі результати:

1). A_1 = (3·1·221+4·8·187+6·5·143) mod2431 =(663+5984+4290) mod2431= = 10937 mod2431 = 1213;

2). A_2=(3·1·221+4·8·187+11·5·143) mod2431= (663+5984+7865) mod2431= = 14512 mod2431 = 2357;

3). A_3=(3·1·221+9·8·187+6·5·143) mod2431=(663+13464+4290) mod2431= = 18417 mod2431 = 1400;

4). A_4=(3·1·221+9·8·187+11·5·143) mod2431=(663+13464+7865) mod2431= = 21992 mod2431 = 113;

5). A_5=(8·1·221+4·8·187+6·5·143) mod2431= (1768+5984+4290) mod2431= = 12042 mod2431 = 2318;

6). A_6=(8·1·221+4·8·187+11·5·143) mod2431=(1768+5984+7865) mod2431= = 15617 mod2431 = 1031;

7). A_7=(8·1·221+9·8·187+6·5·143) mod2431=(1768+13464+4290) mod2431= = 19522 mod2431 = 74;

8). A_8=(8·1·221+9·8·187+11·5·143) mod2431=(1768+13464+7865) mod2431= = 23097 mod2431=1218.

Звідси видно, що зашифрованому повідомленню відповідає A_6.

Задача суттєво спрощується, коли модулі *p, q, r* підібрано таким чином, що вони утворюють МДФ СЗК, тобто m_i=±1. Це дозволяє уникнути виконання громіздкої процедури пошуку оберненого елемента за модулем та множення в (6.5) на m_i. Нехай *p*=11, *q*=17, *r*=31, *n*= 5797. Тоді *A*'=1031^2mod 5797=2110, $k = 2110 \bmod 11 = 9$, $l = 2110 \bmod 17 = 2$, $d = 2110 \bmod 31 = 2$; $\sqrt{9} \bmod 11 = 3$ і 8, $\sqrt{2+2\cdot 17} \bmod 17 = 6$ та 11,

$\sqrt{2+2\cdot 31}\bmod 31=8$ та 23. З отриманих коренів утворюється 8 різних комбінацій: (3, 6, 8), (3, 6, 23), (3, 11, 8), (3, 11, 23), (8, 6, 8), (8, 6, 23), (8, 11, 6), (8, 11, 23).

Далі S_1=17·31=527, $(S_1 m_1)\bmod p = 527\cdot m_1 \bmod 11 = 10\cdot m_1 \bmod 11 = 1$, звідси m_1=10. Аналогічно S_2=11·31=341, $(S_2 m_2)\bmod q = 341\cdot m_2 \bmod 17 = 1\cdot m_2 \bmod 13 = 1$, m_2=1; і S_3=11·17=187, $(S_3 m_3)\bmod r = 187\cdot m_3 \bmod 31 = 1\cdot m_3 \bmod 17 = 1$, m_3=1.

Тоді згідно (6.5) можна отримати:

1). A_1=(3·10·527+6·1·341+8·1·187) mod5797=(15810+2046+1496) mod5797= = 19325 mod5797 = 1961;

2). A_2=(3·10·527+6·1·341+23·1·187) mod5797=(15810+2046+4301) mod5797= = 22157 mod5797 = 4766;

3). A_3=(3·10·527+11·1·341+8·1·187) mod5797=(15810+3751+1496) mod5797= = 21057 mod5797 = 3666;

4). A_4=(3·10·527+11·1·341+23·1·187) mod5797=(15810+3751+4301) mod5797= = 22157 mod5797 = 674;

5). A_5=(8·10·527+6·1·341+8·1·187) mod5797=(42160+2046+1496) mod5797= = 45702 mod5797 = 5123;

6). A_6=(8·10·527+6·1·341+23·1·187) mod5797=(42160+2046+4301) mod5797= = 48507 mod5797 = 2131;

7). A_7=(8·10·527+11·1·341+8·1·187) mod5797=(42160+3751+1496) mod5797= = 47407 mod5797 = 1031;

8). A_8=(8·10·527+11·1·341+23·1·187) mod5797=(42160+3751+4301) mod5797= = 22157 mod5797 = 3836.

Зашифрованому повідомленню відповідає значення A_7. Оскільки відкритий ключ більший, ніж в попередньому випадку, то відповідно і проміжні результати набувають більших значень. Однак якщо врахувати, що m_1=10 mod11=-1 mod11, то отримаємо такі розрахунки:

1). A_1=(-3·1·527+6·1·341+8·1·187) mod5797=(-1581+2046+1496) mod5797= = 1961 mod5797 = 1961;

2). A_2=(-3·1·527+6·1·341+23·1·187) mod5797=(-1581+2046+4301) mod5797= = 4766 mod5797 = 4766;

3). A_3=(-3·1·527+11·1·341+8·1·187) mod5797=(-1581+3751+1496) mod5797= = 3666 mod5797 = 3666;

4). A_4=(-3·1·527+11·1·341+23·1·187) mod5797=(-1581+3751+4301) mod5797= = 6471 mod5797 = 674;

5). A_5=(-8·1·527+6·1·341+8·1·187) mod5797=(-4216+2046+1496) mod5797= = -674 mod5797 = 5123;

6). A_6=(-8·1·527+6·1·341+23·1·187) mod 5797=(-4216+2046+4301) mod5797= = 2131 mod5797 = 2131;

7). A_7=(-8·1·527+11·1·341+8·1·187) mod5797=(-4216+3751+1496) mod5797= = 1031 mod5797 = 1031;

8). A_8=(-8·1·527+11·1·341+23·1·187) mod5797=(-4216+3751+4301) mod5797= = 3836 mod5797 = 3836.

Видно, що проміжні результати набувають менших значень, ніж в попередньому прикладі [328]. Крім того, тільки у двох випадках з восьми потрібно додавати або віднімати *n*, щоб отримати невід'ємне число, менше за *n*.

6.4 HDL-модель трьохмодульної криптосистеми Рабіна

Для дослідження часових характеристик модифікованого криптоалгоритму Рабіна та алгоритму, в якому модулі утворюють МДФ СЗК було обрано Active-HDL - середовище розробки, моделювання і верифікації проектів для програмованих логічних інтегральних схем чи програмованих логічних матриць [329].

Пропонований програмний засіб складається з трьох основних блоків, які в загальному формують криптопроцесор, що передбачає роботу із програмованою логічною інтегральною схемою.

Функціональні модулі у запропонованій програмі наступні:

- модуль підключення необхідних бібліотек;
- модуль оголошення внутрішніх та зовнішніх портів (інтерфейсу проекту);
- модуль опису поведінки розробленої моделі.

Оголошення бібліотек включає декілька основних компонент, до яких програма буде звертатись у подальшому. В даному випадку опис доступу до необхідних бібліотек в середовищі Active-HDL має вигляд:

LIBRARY IEEE;

IEEE.STD_LOGIC_1164.ALL;

IEEE.NUMERIC_STD.ALL;

IEEE.MATH_REAL.ALL;

Бібліотека LIBRARY IEEE - це бібліотека вищого рівня, розроблена всесвітньою організацією IEEE, яка постійно оновлюється. Тобто за її допомогою виконується доступ і використання бібліотек нижнього рівня.

Бібліотека IEEE.STD_LOGIC_1164.ALL - це стандартизований пакет даних, який використовується при моделюванні і описі бітових типів даних. Логічна система цього пакету передбачає в коді програми роботу над логічними елементами. Проте виконання такого примітивного завдання, як моделювання роботи вентильного транзистора, виходить за межі функціоналу цієї бібліотеки.

Бібліотека IEEE.NUMERIC_STD.ALL – пакет, що передбачає оголошення цифрових і деяких математичних функцій, які використовуються у проектах, що будуть синтезуватись. У цьому пакеті оголошується принцип роботи програмного продукту із числовими масивами і виконання з ними елементарних операцій (+,-,*,/ зсуву ліворуч і праворуч).

Бібліотека IEEE.MATH_REAL.ALL використовується як доповнення до пакету IEEE.NUMERIC_STD.ALL, щоб можна було виконувати із числами більш складні математичні операції - піднесення до степеня, добування коренів різного степеня, знаходження модулів і багато інших.

Модуль оголошення внутрішніх і зовнішніх портів являє собою виділений блок, який приймає значення і по суті керує всім програмним продуктом, адже саме в цьому блоці описується, звідки будуть зчитуватись дані і куди їх у подальшому буде записано після обробки. Цей розділ складається із двох блоків:

– generic, в який записуються усі константні значення:

```
generic (
        p1 : integer := 11;
        p2 : integer := 13;
        p3 : integer := 17
            );
```

- port, в якому описується вхідні та вихідні порти програми:

```
port (
        clk : in std_logic;
        v1,v2,v3,v4,v5,v6,v7,v8 : out integer
         );
```

У блоці generic описано три основні параметри p1, p2, p3, які відповідають за вхідні числові змінні, що будуть використовуватись як закритий ключ при шифруванні.

У блоці port оголошено два основних елементи - це вхід та вихід в програмі. За вхід використовується тактовий сигнал clk, а вихідними сигналами є значення v1,…,v8. Блок опису коду програми організований таким чином, що на його початку описано усі внутрішні змінні, які будуть використовуватись в програмному продукті. Ці змінні оголошуються за допомогою зарезервованого слова signal і описуються як дійсні значення у відповідних діапазонах:

```
signal n: real range 1.0 to 9999999.0 :=1.0;
signal w: real range 0.0 to 9999999999999.0 :=1.0;
signal word: real range 1.0 to 9999999.0;
signal g111: real range 0.0 to 5000000.0 :=1.0;
signal g222: real range 0.0 to 5000000.0 :=1.0;
signal g333: real range 0.0 to 5000000.0 :=1.0;
signal sqrt1: real range 0.0 to 5000000.0 :=1.0;
signal sqrt2: real range 0.0 to 5000000.0 :=1.0;
signal sqrt3: real range 0.0 to 5000000.0 :=1.0;
signal g1: real range 0.0 to 5000000.0 :=1.0;
signal g2: real range 0.0 to 5000000.0 :=1.0;
signal g3: real range 0.0 to 5000000.0 :=1.0;
signal p1_g1: real range 0.0 to 5000000.0 :=1.0;
signal p2_g2: real range 0.0 to 5000000.0 :=1.0;
signal p3_g3: real range 0.0 to 5000000.0 :=1.0;
signal k1: real range 0.0 to 5000000.0 :=1.0;
signal k2: real range 0.0 to 5000000.0 :=1.0;
signal k3: real range 0.0 to 5000000.0 :=1.0;
signal m1: real range 0.0 to 5000000.0 :=1.0;
signal m2: real range 0.0 to 5000000.0 :=1.0;
signal m3: real range 0.0 to 5000000.0 :=1.0;
signal m11: real range 0.0 to 5000000.0 :=1.0;
signal m22: real range 0.0 to 5000000.0 :=1.0;
signal m33: real range 0.0 to 5000000.0 :=1.0;
```

В основному усі ці змінні застосовуються для проміжних обчислень. Змінна word позначає блок інформації, який буде шифруватися.

Поведінку модифікованого алгоритму Рабіна описано як процес від clk:

```
read: process(clk)
    begin
if clk'event and clk='1' then
```

```
n<=p1*p2*p3;
w<=word**2.0 mod n ;
g111<=w mod p1;
g222<=w mod p2;
g333<=w mod p3;
sqrt1<=g111;
sqrt2<=g222;
sqrt3<=g333;
end process;
```

В процесі обчислюються відповідно до запропонованого алгоритму змінні v1,…,v8. Функціональна схема розробленого пристрою має вигляд, представлений на рис. 6.8.

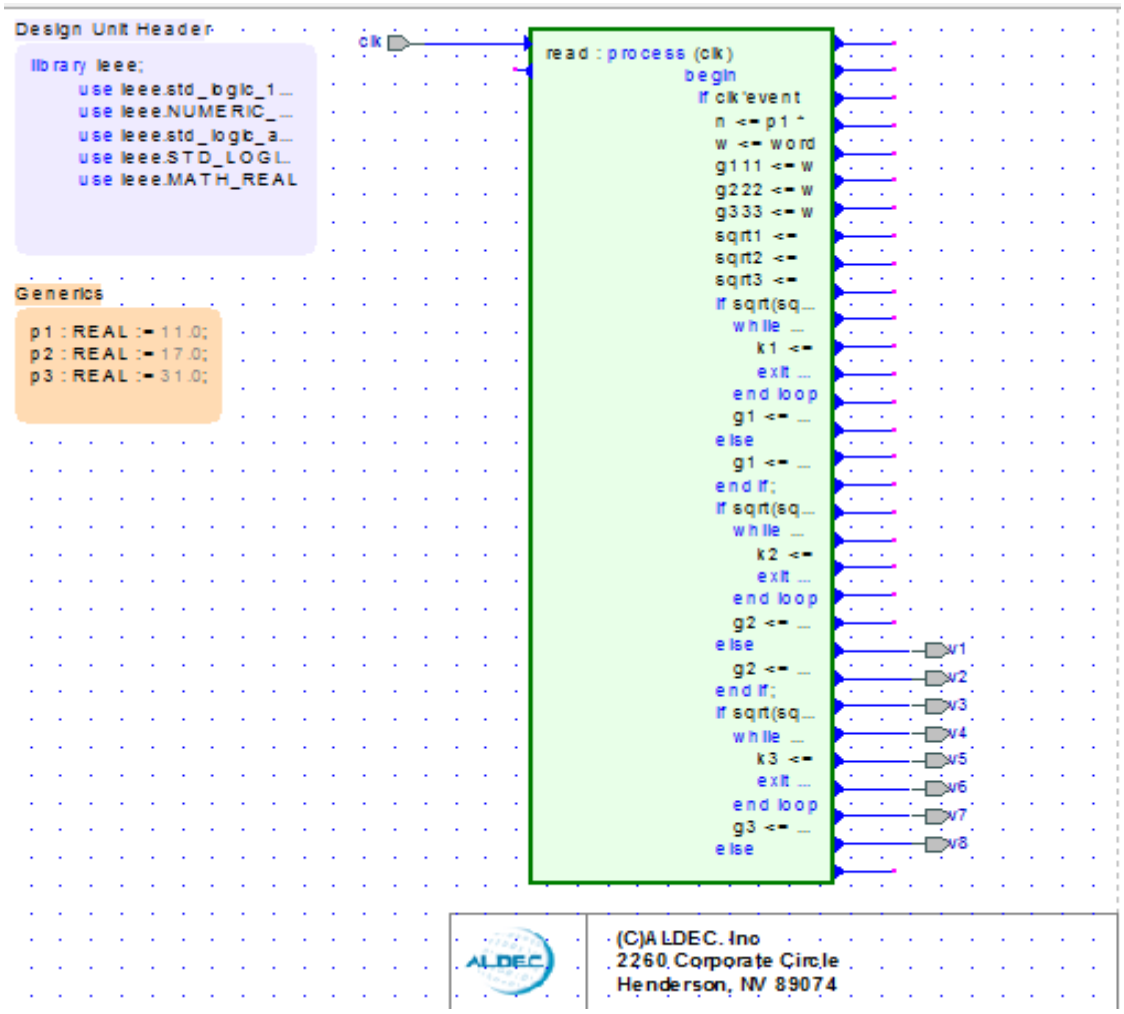


Рис. 6.8 – Функціональна схема розробленого пристрою

На рис. 6.9 представлена блок-схема HDL-моделі трьохмодульного криптоалгоритму Рабіна.

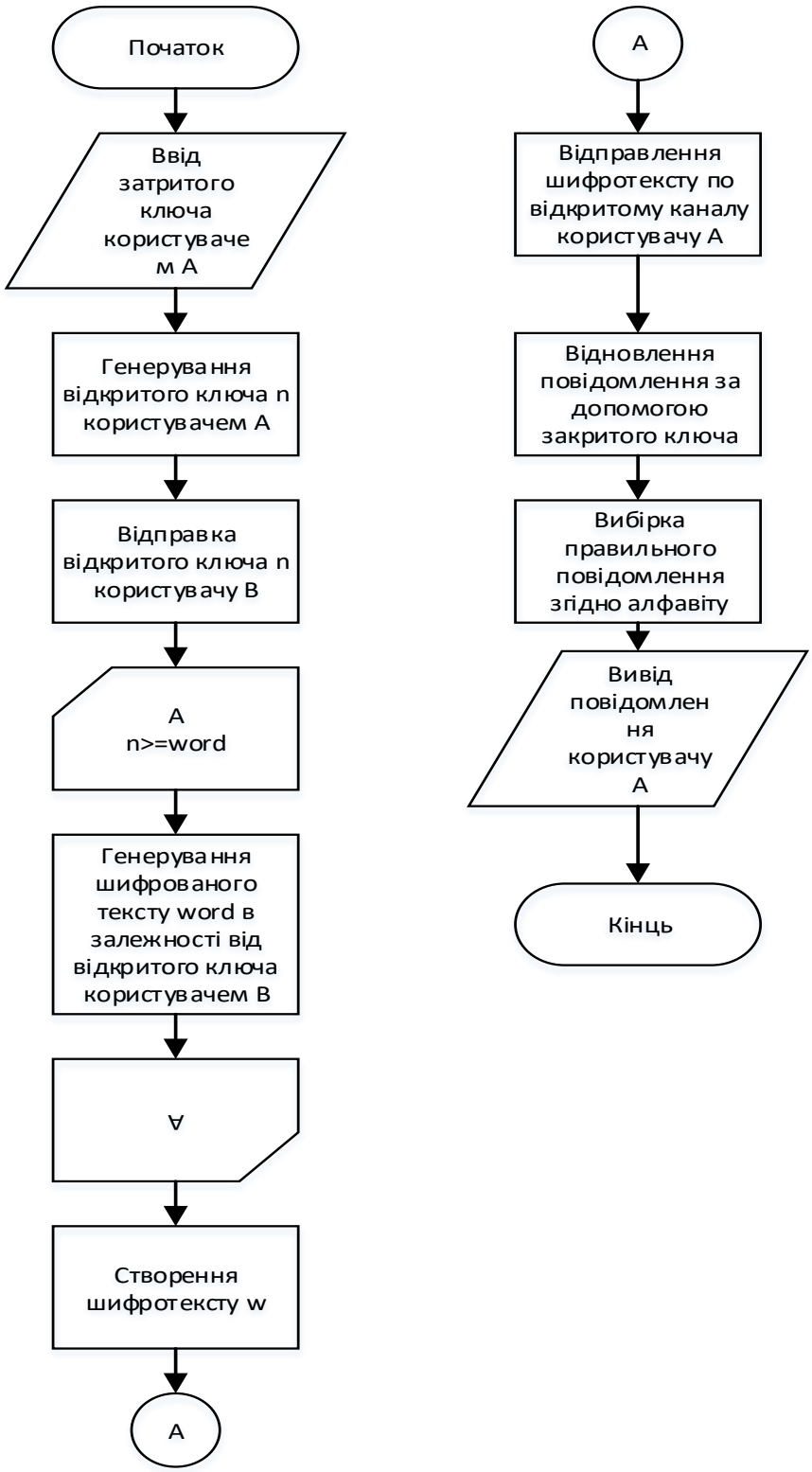


Рис. 6.9 - Блок-схема HDL-моделі модифікованого криптоалгоритму Рабіна

Для побудови часової діаграми симуляції роботи трьохмодульного криптоалгоритму Рабіна без використання МДФ СЗК був вибраний такий ряд вхідних параметрів: p=11, q=13, r=17 – вхідні таємні ключі; Word = 1024 – блок для шифрування.

На часовій діаграмі (рис. 6.10) показані результати моделювання роботи проектованого пристрою. З її аналізу випливає, що при проходженні 1,4 мікросекунди (1400 нс) і встановлення усіх проміжних параметрів у статичний стан система виводить результат моделювання на один із вихідних портів.

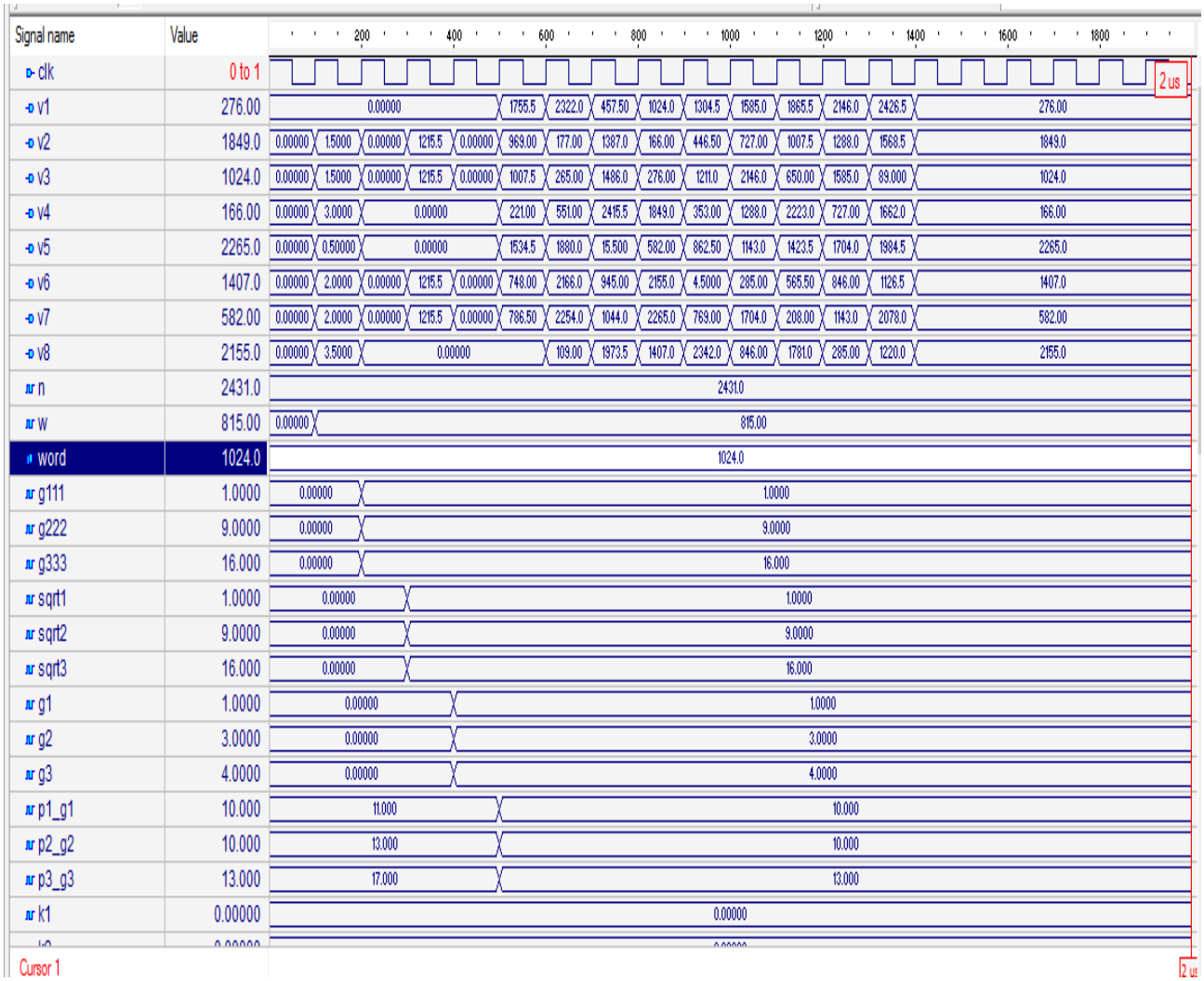


Рис. 6.10 – Часова діаграма симуляції роботи модифікованого алгоритму Рабіна

Часова діаграма під час симуляції процесу шифрування трьохмодульного криптоалгоритму Рабіна з використанням МДФ СЗК подана на рис. 6.11. Шифрувався такий же самий блок відкритого тексту Word = 1024. Були вибрані прості модулі p=11, q=17, r=31, які утворюють МДФ СЗК. Їх добуток P=5797 більший, ніж в попередньому випадку (P=2431), що в загальному повинно спричинити збільшення часу роботи алгоритму.

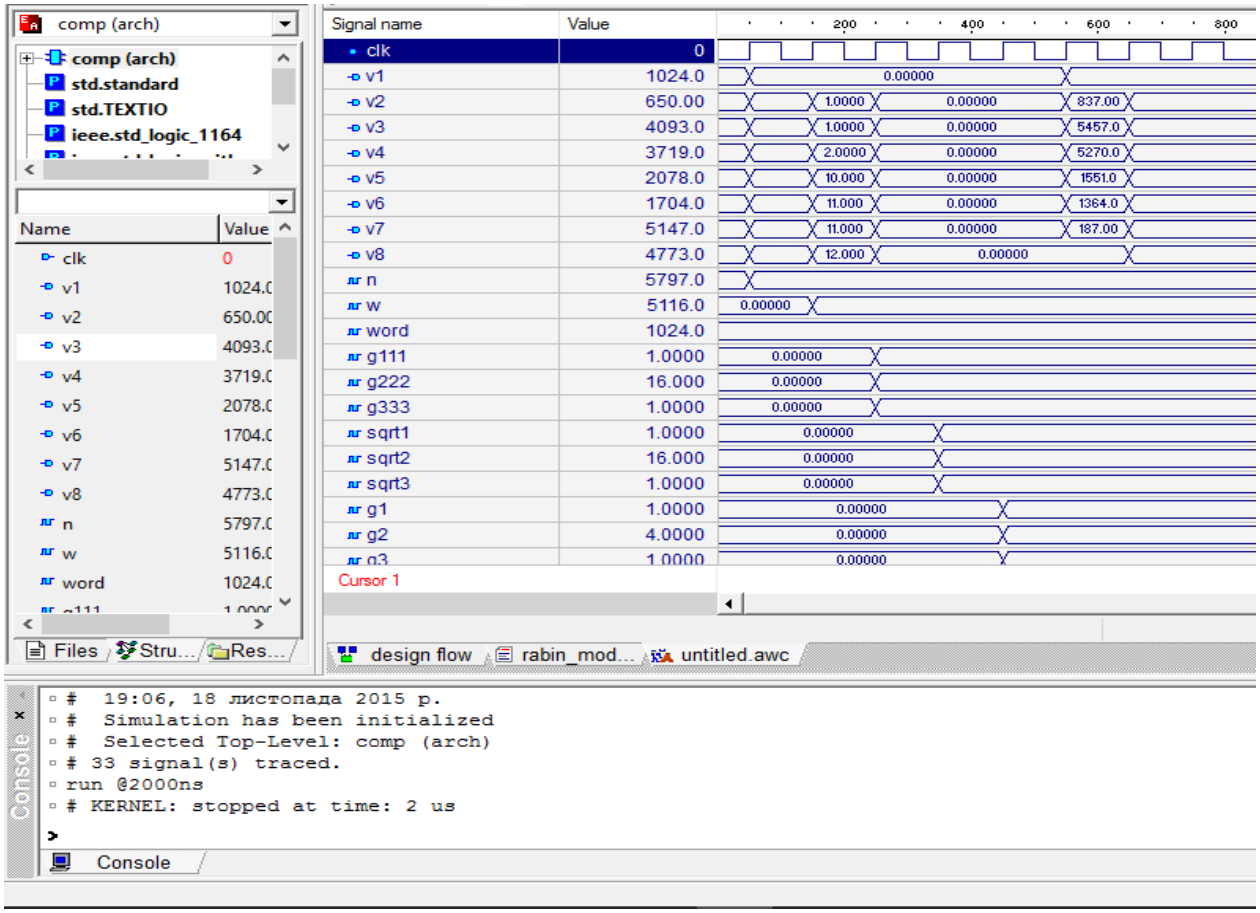


Рис. 6.11 – Часова діаграма трьохмодульного криптоалгоритму Рабіна з використанням МДФ СЗК

Але з рис. 6.11 видно, що час роботи трьохмодульного криптоалгоритму Рабіна з використанням МДФ СЗК, який дозволяє уникнути виконання процедури пошуку оберненого елемента за модулем,

становить 650 нс, тобто зменшився приблизно вдвічі в порівнянні з попереднім [329].

Крім того, даний метод при виборі модулів одного порядку має перевагу перед класичним у стійкості за рахунок збільшення блоку відкритого тексту для шифрування.

Висновки до шостого розділу

1. За допомогою мови програмування Java здійснена програмна реалізація методів підбору модулів ДФ та МДФ СЗК, що дозволило обґрунтувати вибір системи модулів для вирішення відповідних задач.

2. Розроблено трьохмодульний криптоалгоритм Рабіна на основі звичайної цілочисельної та МДФ СЗК, який дозволяє розширити блок шифрування без зменшення криптостійкості алгоритму. Показано, що застосування МДФ СЗК забезпечує зменшення значень операндів та кількість арифметичних операцій, необхідних при розшифруванні.

Основні результати шостого розділу відображені у роботах [294, 328-334].

7 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ АСИМЕТРИЧНИХ КРИПТОСИСТЕМ. МЕТОДОЛОГІЯ ОПРАЦЮВАННЯ БАГАТОРОЗРЯДНИХ ЧИСЕЛ В АСИМЕТРИЧНИХ КРИПТОСИСТЕМАХ

У сьомому розділі проведені експериментальні дослідження та методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах з використанням різних форм СЗК.

7.1 Програмний застосунок для реалізації методів модулярного експоненціювання

Для експериментальних досліджень методів модулярного експоненціювання було обрано комп’ютер HP ProBook 4540S, з процесором Intel i5, тактова частота 2,3 GHz, оперативна пам’ять 6 GB, операційна система Linux Mint 17.1, середовище програмування Python 3.4.0. Вибір останнього зумовлений його модульністю та можливістю повторного використання коду [335]. Це є інтерпретована об'єктно-орієнтована мова програмування високого рівня із динамічною семантикою. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формах на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована. Безсумнівною її перевагою є можливість роботи з великорозрядними числами.

Вибір методів зумовлювався часом модулярного експоненціювання, який для різних методів мав бути приблизно одного порядку, та можливістю виконання операцій над великорозрядними числами. В зв’язку з цим не

досліджувалися методи прямого піднесення до степеня, при якому вже для 16-бітних чисел відбувалося переповнення розрядної сітки, та множення справа наліво або зліва направо, що вимагало виконання x-1 операцій множення (x – показник степеня) і, відповідно, збільшувався час обчислень. З цієї ж причини не був використаний метод СЗК, коли модуль P є добутком декількох простих співмножників ($P=p_1p_2\ldots p_k$), і його потрібно факторизувати, що приводило до істотного збільшення часової складності. Отже, для дослідження були вибрані чотири методи [336]:

1) бінарний або метод пониження степеня за допомогою квадратів [337], який описується таким виразом:

$$N = a^x \bmod p = \left(a^{x-2x_1} a_1^{x_1-2x_2} \ldots a_i^{x_i-2x_{i+1}} \ldots\right) \bmod p = \left(\prod_{i=0}^{[\log_2 x]} a_i^{x_i-2x_{i+1}}\right) \bmod p, \quad (7.1)$$

де $a_0=a$, $x_0=x$, $a_i = a_{i-1}^2 \bmod p$; $x_i = \left[\frac{x_{i-1}}{2}\right]$;

2) метод пониження степеня за допомогою кубів (3-арний) [337], який можна записати аналогічно (7.1):

$$N = a^x \bmod p = \left(a^{x-3x_1} a_1^{x_1-3x_2} \ldots a_i^{x_i-3x_{i+1}} \ldots\right) \bmod p = \left(\prod_{i=0}^{[\log_3 x]} a_i^{x_i-3x_{i+1}}\right) \bmod p, \quad (7.2)$$

де $a_0=a$, $x_0=x$, $a_i = a_{i-1}^3 \bmod p$; $x_i = \left[\frac{x_{i-1}}{3}\right]$;

3) використання СЗК, коли основа степеня розбивається на залишки від ділення на попарно взаємно прості модулі, далі відбувається пониження степеня бінарним методом і відновлення десяткового запису числа. Всю процедуру можна описати такими виразами:

$$N = a^x \bmod p = \left(\sum_{i=1}^{k} m_i M_i a_i \right) \bmod p, \qquad (7.3)$$

де $a_i = (a \bmod p_i)^x \bmod p_i$, $p = \prod_{i=1}^{k} p_i$, $P_i = \frac{p}{p_i}$, $m_i = P_i^{-1} \bmod p_i$, k – кількість модулів. Пошук оберненого елемента відбувається за допомогою алгоритму Евкліда та його наслідку;

4) використання МДФ СЗК, згідно якої модулі підібрані таким чином, що $m_i = P_i \bmod p_i = \pm 1$. Це дозволяє уникнути виконання громіздкої операції пошуку оберненого елемента за модулем та множення в (7.3) на базисні числа m_i, що, відповідно, приводить до зменшення часу обчислень.

Для модулярного піднесення до степеня $a^x \ mod \ p$ в залежності від розрядності n_0 та кількості одиничок у двійковому записі числа (ваги Хемінга) параметри a, x та p визначалися таким чином: $a=r(n_0\text{-}3, \Delta)$, $x=r(n_0, \Delta)$, де цілочисельна функція

$$r(n_0, \Delta) = 2^{n_0} - 1 - \left[RND \cdot \left[\frac{\Delta \cdot n_0}{100} \right] \right], \qquad (7.4)$$

де $0<RND<1$ – випадкова величина, заданий параметр Δ набуває дискретних значень 0, 10, 30, 50 та 80.

Останній вказує, що при $\Delta=0$ вага Хемінга чисел $a=2^{n_0}\text{-}1$ та $x=2^{n_0\text{-}3}\text{-}1$ дорівнює їх розрядності. Збільшення цього параметра означає, що Δ% молодших розрядів у двійковому записі a та x може змінитися випадковим чином під дією функції *RND*.

Для кожного методу піднесення до степеня використовувався один і той самий модуль, який залежить від розрядності числа x і є добутком трьох попарно взаємно простих співмножників $p=p_1 \cdot p_2 \cdot p_3$, що утворюють МДФ

СЗК: $p_1 = 2^{\left\lceil \frac{n_0}{3} \right\rceil + 1}$, p_2=2p_1-1, p_2=2p_1+1. Отримані результати наведені в табл. 7.1 (у першому стовпчику 1 – пониження степеня за допомогою квадратів, 2 – за допомогою кубів, 3 – СЗК, 4 – МДФ СЗК) [336].

Слід відмітити, що модуль p перевищує діапазон обчислень, який визначається відповідною розрядністю n_0, приблизно у 8-16 разів. Усі розрахунки для кожного випадку проводилися 10 разів і визначався середній час виконання модулярного експоненціювання (в мікросекундах).

Як слідує з табл. 7.1, швидкість піднесення до степеня за модулем істотно залежить від методу, яким воно виконується. Так, при малих розрядностях (до 256 включно) методи СЗК та МДФ СЗК істотно поступаються в часі двом іншим (приблизно в 2-4 рази), а найшвидше модулярне експоненціювання виконується методом пониження степеня за допомогою кубів. Це ж стосується випадку, коли n_0=512 і Δ=0. Для інших Δ і тієї ж розрядності мінімальний час буде при використанні бінарного методу, а СЗК та МДФ СЗК вже відстають приблизно в 1,5 раза і, починаючи з n_0=1024 вони дають мінімальний час.

На рис. 7.1 в логарифмічній шкалі представлено графіки залежності часу виконання операції модулярного експоненціювання різними методами від розрядності чисел для Δ=0, тобто для чисел з максимальним значенням ваги Хемінга.

Слід зазначити, що при інших значеннях Δ залежності мають приблизно такий самий характер, тому вони не наведені. З рисунка видно, що графіки для СЗК та МДФ СЗК практично збігаються. Це пов’язано з тим, що при переведенні чисел із СЗК в десяткову систему числення для модулів, які утворюють МДФ СЗК і використовуються в дослідженні, потрібна мала кількість ітерацій при застосуванні алгоритму Евкліда, його наслідку та КТЗ. На рис. 7.2 наведено графіки залежності часу виконання модулярного піднесення до степеня різними методами від значення ваги Хемінга Δ при найбільш поширеній розрядності n_0=1024.

Таблиця 7.1 - Час виконання операції модулярного експоненціювання різними способами

Розрядність	8	16	32	64	128	256	512	1024	2048	4096
Δ=0										
1	14	24	55	108	254	746	2251	10727	64628	405196
2	12	23	44	91	200	655	2162	11987	75411	466016
3	58	83	159	299	628	1430	3430	10366	37564	216596
4	62	72	150	266	610	1340	3195	9956	36861	208496
Δ=10										
1	10	18	39	81	179	487	1955	9452	61290	404806
2	9	17	34	66	148	445	1997	10055	68190	458023
3	43	58	114	211	456	997	2858	9266	34831	199364
4	40	53	105	193	451	969	2563	8417	34413	198145
Δ=30										
1	10	18	38	78	177	487	1919	9353	60907	403917
2	9	17	35	68	146	441	2038	10097	68163	457405
3	43	61	115	211	458	998	2848	9009	34768	199534
4	40	53	103	193	443	974	2546	8409	34378	197433
Δ=50										
1	11	18	37	79	177	494	1954	9465	61187	404641
2	8	17	33	67	146	442	1978	10224	68165	457931
3	44	61	113	211	459	1004	2847	9108	34826	199136
4	40	53	103	193	442	965	2553	8443	34361	197780
Δ=80										
1	10	17	37	78	181	492	1918	9419	60949	404217
2	9	18	33	67	146	444	2015	10074	68148	458093
3	42	61	112	209	458	1001	2856	9086	34738	199704
4	40	51	101	194	440	974	2560	8394	34355	197588

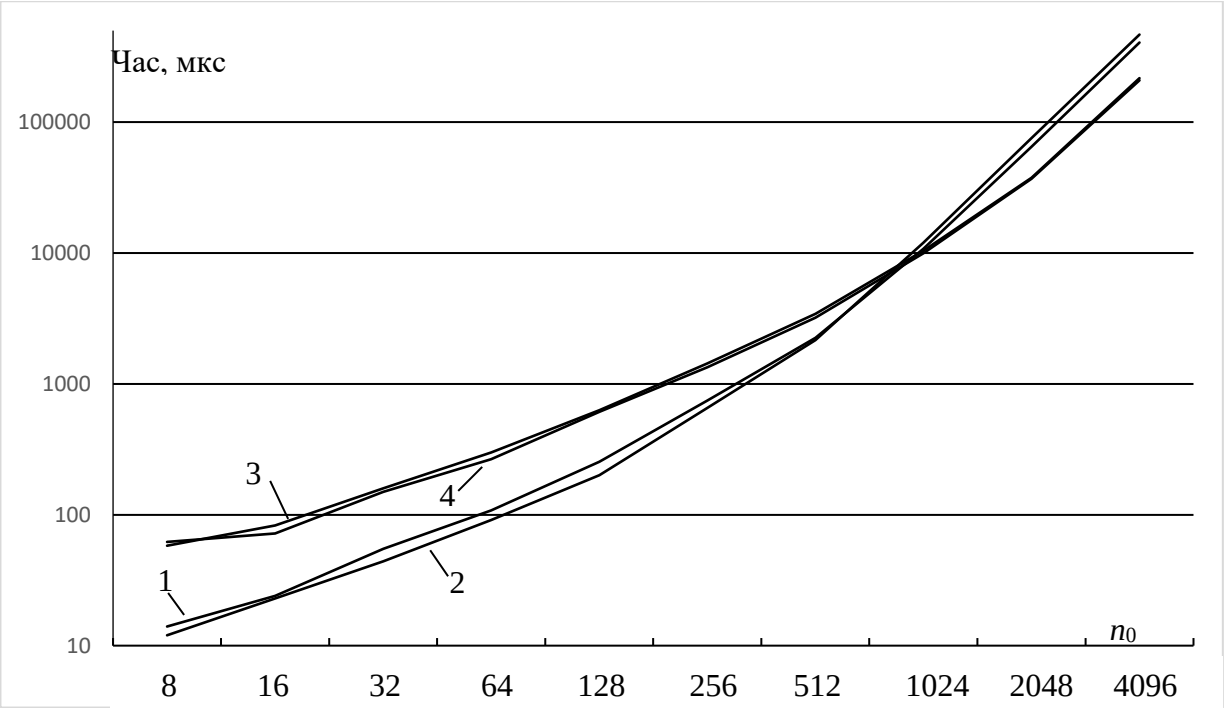


Рис. 7.1 - Графіки залежності часу виконання операції модулярного експоненціювання різними методами від розрядності чисел для Δ=0 (1 – пониження степеня за допомогою квадратів, 2 – за допомогою кубів, 3 – використання СЗК, 4 – використання МДФ СЗК)

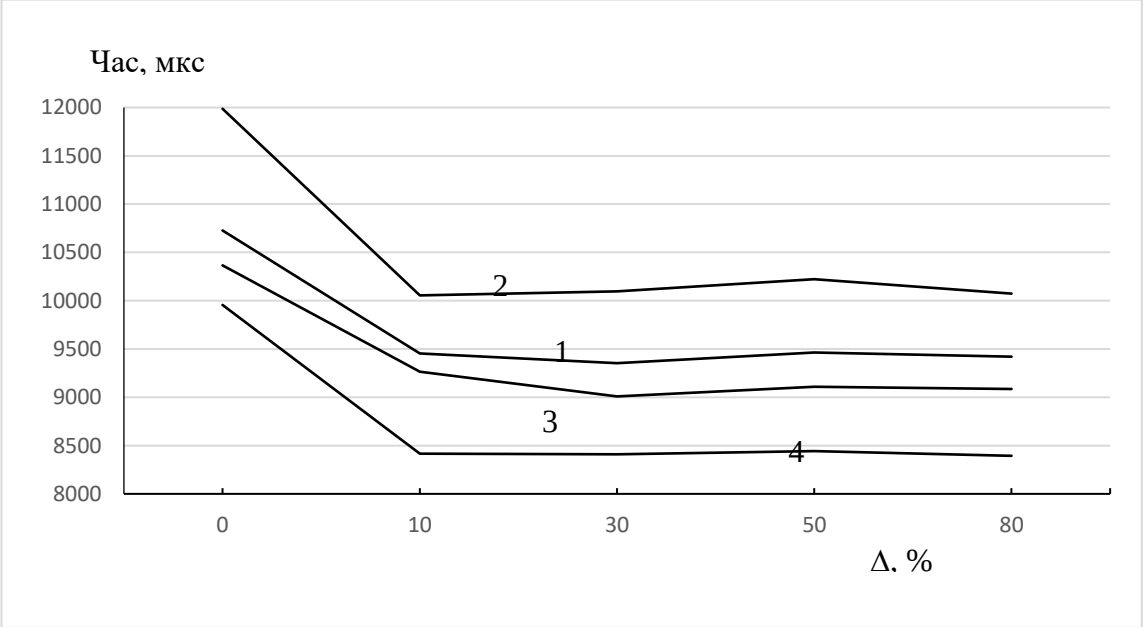


Рис. 7.2 - Графіки залежності часу виконання модулярного піднесення до степеня різними методами від значення ваги Хемінга Δ при n=1024 (1 – пониження степеня за допомогою квадратів, 2 – за допомогою кубів, 3 – використання СЗК, 4 – використання МДФ СЗК)

Як видно з рис. 7.2, всі графіки мають приблизно однаковий характер. Найбільший час затрачається при максимальному значенні ваги Хемінга (Δ=0). При зменшенні останньої час різко зменшується і надалі залишається майже постійним, здійснюючи невеликі коливання відносно деякого значення.

7.2 Дослідження сумісного виконання алгоритму Евкліда та множення у криптосистемі Рабіна

Сумісне виконання алгоритму Евкліда та множення вимагає строго послідовної реалізації, що істотно зменшує швидкодію обчислювальних систем. Прикладом може бути криптосистема Рабіна, у якій для застосування КТЗ використовується алгоритм Евкліда, а для формування відкритого ключа необхідно перемножити ті ж самі числа. Тому досить гостро постає питання про можливість розпаралелення виконання подібних операцій.

Як було відмічено в п. 1.1, сучасний математичний запис алгоритму Евкліда має вигляд (1.5). НСД двох чисел a та b, обчислений за допомогою алгоритму Евкліда, дорівнює останньому ненульовому члену послідовності r_i. Далі відбувається перемноження чисел a і b.

Слід зазначити, що теоретичні дослідження складностей вказаних операцій не завжди показують реальну картину для швидкодії обчислень, так як не враховують всіх факторів, що впливають на роботу обчислювальної системи, зокрема, звертань до пам’яті, особливостей завантаженості процесора, його розрядності і розпаралелення обчислень тощо. Тому доцільно розглянути експериментальні дослідження.

В криптосистемі Рабіна задані числа a та b є простими, тому у такому випадку НСД $(a, b)=r_n=1$. На відміну від виконання алгоритму Евкліда із наступним перемноженням цих самих чисел пропонується, згідно (1.5), у множенні використати проміжні і кінцеві результати алгоритму Евкліда таким чином [338]:

$$a \cdot b = r_0^2 q_1 + r_1^2 q_2 + r_2^2 q_3 + ... + r_{n-1}^2 q_n + r_n . \qquad (7.5)$$

Слід відмітити, що кількість доданків у (7.5) відповідає кількості кроків в алгоритмі Евкліда. Хоча даний метод передбачає виконання більшої кількості арифметичних операцій, але вони виконуватимуться над числами меншої розрядності. Суттєвим кроком для підвищення швидкодії є наявність таблиці квадратів у пам'яті комп'ютера, хоча це приводить до збільшення використання ресурсів обчислювальної системи. Блок-схема алгоритму наведена на рис. 7.3.

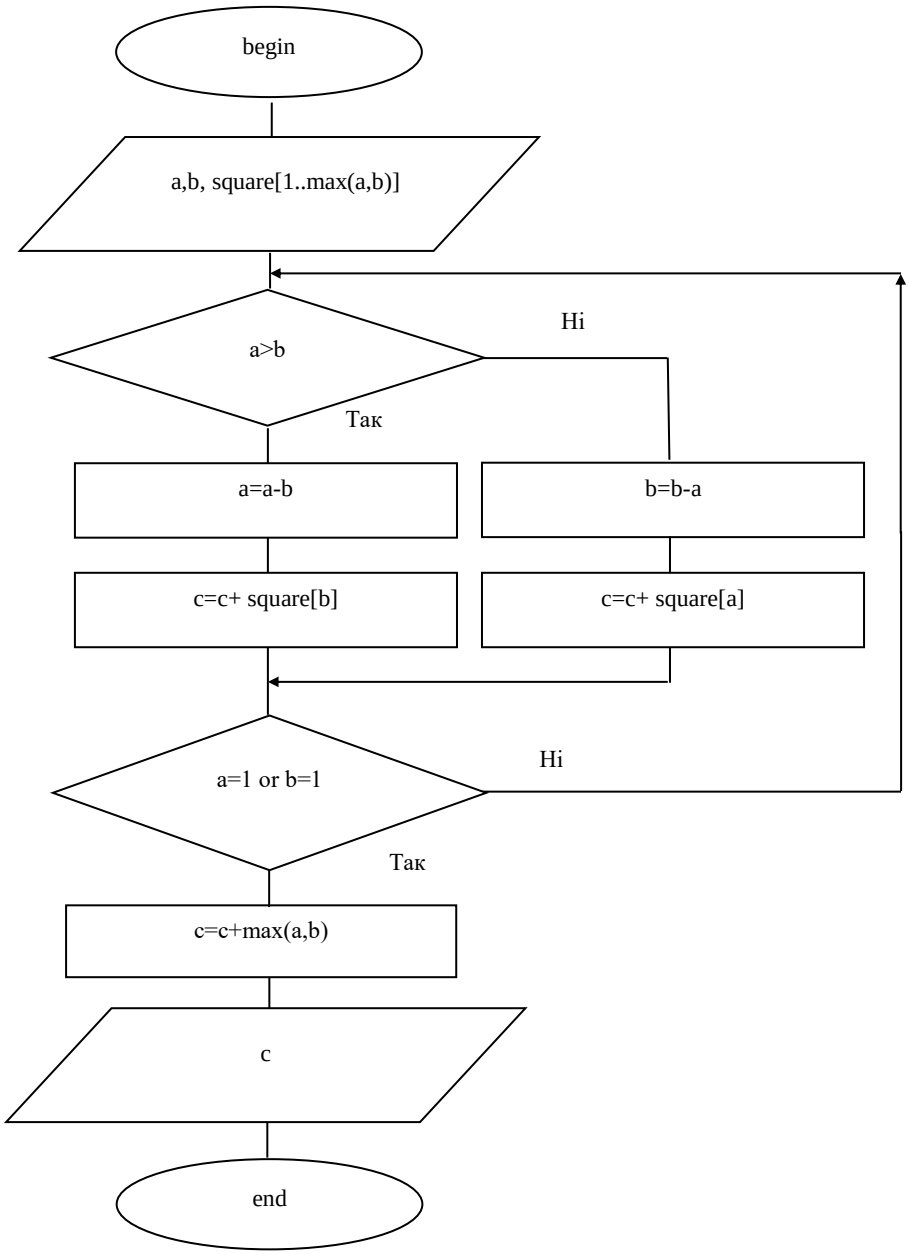


Рис. 7.3 – Блок-схема запропонованого алгоритму

Для експериментальних досліджень було обрано портативний комп’ютер Lenovo B50-70 з процесором Intel Pentium 3558U (1.7 ГГц). Об’єм оперативної пам’яті у пристрої становив 4 GB. При проектуванні програмного комплексу, який забезпечував обчислення, було обрано мову програмування високого рівня C++.

Для роботи із багаторозрядними числами використовувалась бібліотека спеціального призначення, написана А. Ленстрою. Це дозволило забезпечити гнучкі можливості роботи з багаторозрядними числами, розмір яких залежить лише від доступних системних ресурсів. Обрана мова програмування і кросплатформенність спец-бібліотеки дозволяють трансформувати коди під різні архітектури і операційні системи.

Найбільш розповсюдженим методом для програмної реалізації алгоритму Евкліда є послідовне віднімання меншого числа від більшого, поки різниця не стане меншою від від’ємника. Тоді цю ж саму процедуру потрібно виконати з від’ємником і різницею. Процес віднімання буде тривати до тих пір, поки від’ємник і різниця не стануть однакові.

В табл. 7.2 представлено час сумісного виконання алгоритму Евкліда і перемноження чисел класичним (t_1) та запропонованим (t_2) методами для b=2, 3, …, 70 при a=71, а на рис. 7.4 – відповідна графічна залежність (t_1 та $t_{1сер}$ – пунктирна лінія, t_2 та $t_{2сер}$ – суцільна) [338].

Графіки носять осцилюючий характер, що пояснюється різною кількістю кроків алгоритму Евкліда для різних чисел. В 44 випадках з 69, що становить 64%, обчислення запропонованим методом виконуються швидше, у 10 (14%) – повільніше, у 15 випадках (22%) час виконання обома методами однаковий. Середні значення часу становлять відповідно $t_{1сер}$=0,037783 с та $t_{2сер}$=0,029754 с, що також представлено на графіку. Отже, швидкодія збільшилася в середньому в 1,27 рази.

Таблиця 7.2 - Час сумісного виконання алгоритму Евкліда та перемноження чисел класичним (t_1) та запропонованим (t_2) методами

b	2	3	4	5	6	7	8	9	10	11	12
t_1, мс	16	31	31	31	31	15	32	31	31	32	31
t_2, мс	31	31	32	16	31	32	31	31	16	31	31
b	13	14	15	16	17	18	19	20	21	22	23
t_1, мс	32	31	47	47	31	31	46	32	63	31	31
t_2, мс	31	15	32	16	31	31	32	47	31	31	16
b	24	25	26	27	28	29	30	31	32	33	34
t_1, мс	31	32	47	47	47	32	46	32	47	47	31
t_2, мс	15	31	46	47	31	31	32	31	31	31	16
b	35	36	37	38	39	40	41	42	43	44	45
t_1, мс	31	32	31	47	62	46	47	31	47	47	62
t_2, мс	16	15	32	47	31	32	47	47	31	47	32
b	46	47	48	49	50	51	52	53	54	55	56
t_1, мс	47	46	31	31	47	47	47	46	47	32	46
t_2, мс	31	32	32	47	31	32	31	16	32	31	32
b	57	58	59	60	61	62	63	64	65	66	67
t_1, мс	31	47	32	47	31	32	31	31	31	31	31
t_2, с	16	15	31	16	31	31	32	31	32	16	31
b	68	69	70								
t_1, мс	47	31	31	Середній час: $t_{1сер}$=0,037783 с.							
t_2, с	32	31	16	Середній час: $t_{2сер}$=0,029754 с.							

У табл. 7.3 представлено середній час сумісного виконання алгоритму Евкліда і множення класичним (t_3) та запропонованим (t_4) методами у випадку, коли значення простих чисел a перебувають в межах однієї розрядності від 67 до 127, а на рис. 7.5 – відповідні графіки залежності середнього часу виконання операції від номера числа. При цьому b змінюється від 2 до a-1.

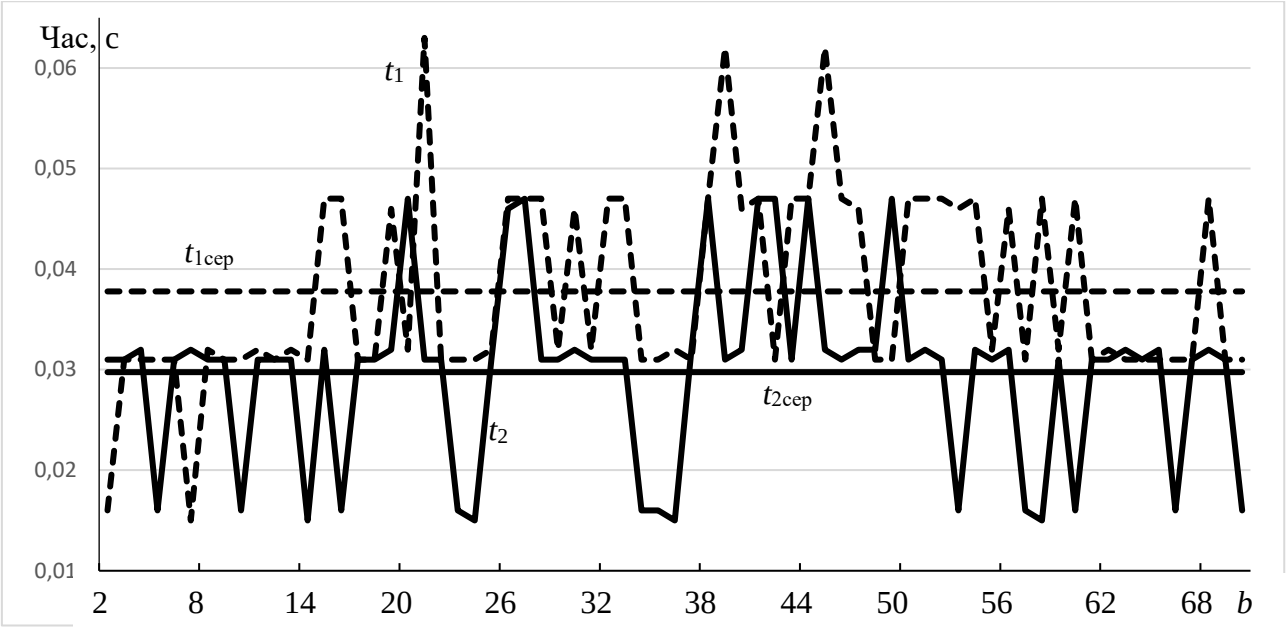


Рис. 7.4 - Графічна залежність часу сумісного виконання алгоритму Евкліда та перемноження чисел класичним (t_1) і запропонованим (t_2) методами

З рис. 7.5 видно, що середній час роботи запропонованим методом в усіх випадках менший, ніж класичним. Загальний тренд вказує на зростання часу при збільшенні заданих чисел, причому графік для класичного методу зростає інтенсивніше.

Таблиця 7.3 - Середній час сумісного виконання алгоритму Евкліда та перемноження класичним (t_3) та запропонованим (t_4) методами

№	1	2	3	4	5	6	7
a	67	71	73	79	83	89	97
t_3, с	0,03637	0,03778	0,03727	0,03842	0,03852	0,03830	0,03714
t_4, с	0,03	0,02975	0,03010	0,02999	0,03093	0,03020	0,03095
№	8	9	10	11	12	13	
a	101	103	107	109	113	127	
t_3, с	0,0388	0,03849	0,04018	0,03924	0,03954	0,03954	
t_4, с	0,03222	0,03125	0,03043	0,03082	0,03057	0,03144	

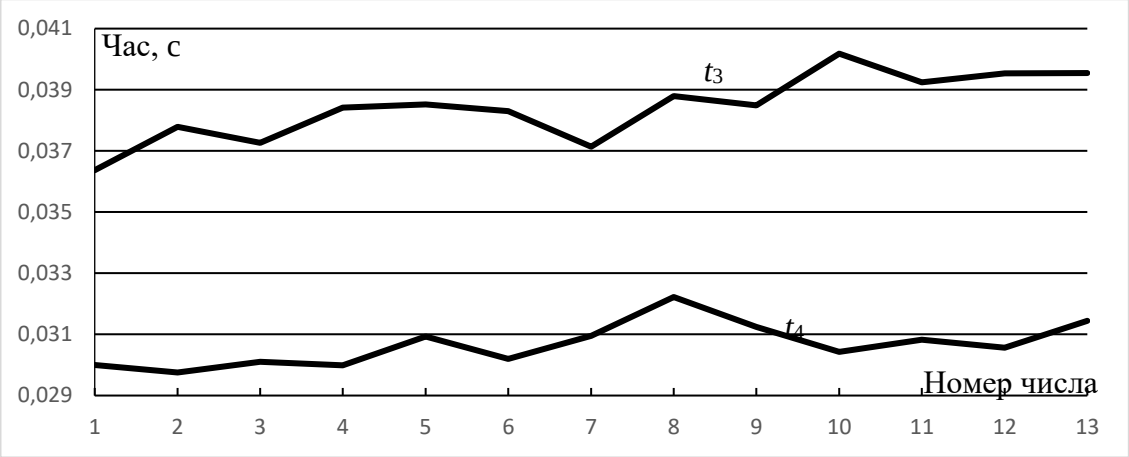


Рис. 7.5 - Графічна залежність середнього часу сумісного виконання алгоритму Евкліда і перемноження класичним (t_3) і запропонованим (t_4) методами від номера числа згідно таблиці 2

В табл. 7.4 представлено середній час сумісного виконання алгоритму Евкліда і перемноження класичним (t_5) і запропонованим (t_6) методами у випадку, коли розрядність n простих чисел a перебуває в межах від 7 до 16 біт, а на рис. 7.6 – відповідні графічні залежності в логарифмічній шкалі. Число b змінюється аналогічно до попереднього випадку [338].

Таблиця 7.4 - Середній час сумісного виконання алгоритму Евкліда і перемноження класичним (t_5) і запропонованим (t_6) методами

a	131	181	257	359	521	727
$\log_2 a$	7	7,5	8	8,5	9	9,5
t_5, с	0,040612	0,039525	0,041859	0,043527	0,044844	0,046127
t_6, с	0,031426	0,032223	0,032906	0,034521	0,034865	0,035659
a	1031	1447	2053	2897	4099	5791
$\log_2 a$	10	10,5	11	11,5	12	12,5
t_5, с	0,048333	0,049314	0,051011	0,051922	0,05336	0,054791
t_6, с	0,037083	0,037722	0,039205	0,039943	0,040651	0,041505
a	8209	11587	16411	23173	32771	46337
$\log_2 a$	13	13,5	14	14,5	15	15,5
t_5, с	0,056475	0,05788	0,059334	0,060874	0,062263	0,063511
t_6,с	0,042528	0,043577	0,044608	0,045555	0,04639	0,047953

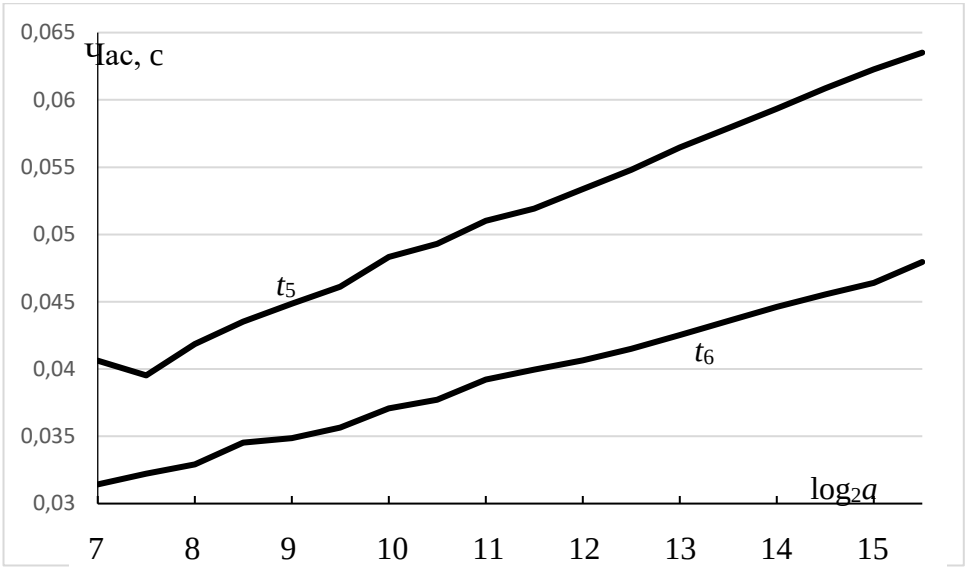


Рис. 7.6 - Графічна залежність середнього часу сумісного виконання алгоритму Евкліда і перемноження класичним (t_5) і запропонованим (t_6) методами від розрядності числа a

Видно, що усереднений час роботи збільшується майже лінійно із збільшенням розрядності числа a, причому графік для класичного методу зростає інтенсивніше.

В табл. 7.5 представлено середній час сумісного виконання алгоритму Евкліда та перемноження класичним (t_7) та запропонованим (t_8) методами у випадку, коли розрядність n_0 простих чисел a (параметру a присвоювалося значення найменшого простого числа, яке перевищувало 2^{n_0}) перебуває в межах від 16 до 44 біт, а на рис. 7.7 – відповідні графічні залежності у логарифмічній шкалі. Число b набувало 10000 різних значень.

Таблиця 7.5 - Середній час сумісного виконання алгоритму Евкліда та перемноження класичним (t_7) та запропонованим (t_8) методами

$\log_2 a$	16	20	24	28
t_7, с	0,06722	0,07799	0,08828	0,09779
t_8, с	0,05012	0,05857	0,06734	0,07577
$\log_2 a$	32	36	40	44
t_7, с	0,10631	0,11857	0,12602	0,13206
t_8, с	0,08644	0,09782	0,10525	0,10915

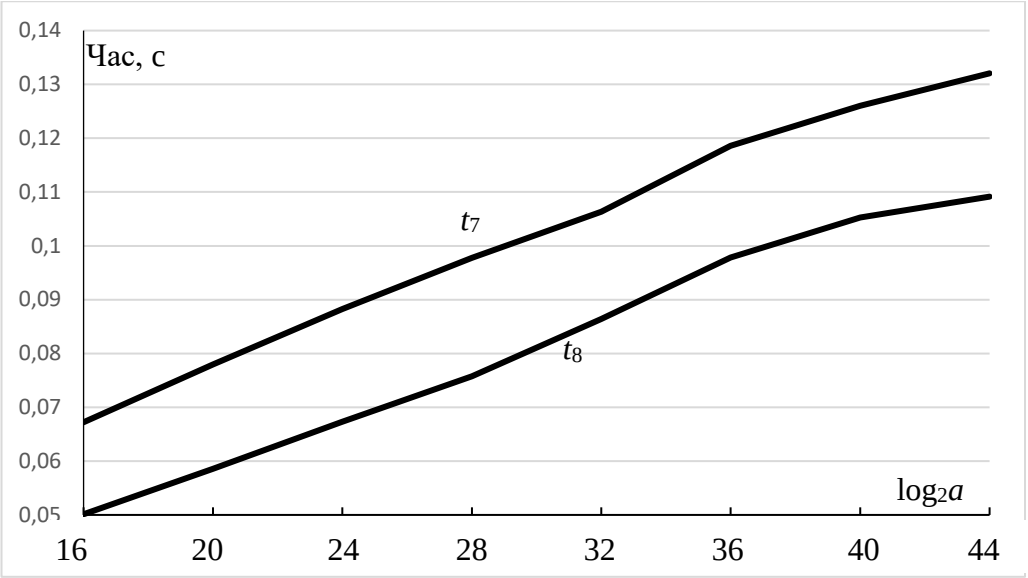


Рис. 7.7 - Графічна залежність середнього часу сумісного виконання алгоритму Евкліда і перемноження класичним (t_7) і запропонованим (t_8) методами від розрядності числа a

Як видно з рисунка, графіки розміщені практично паралельно. При великих розрядностях ($n_0 \geq 40$) інтенсивність зростання t_8 зменшується.

Для нівелювання випадкових впливів на час роботи усі обчислення повторювалися 5000 разів.

7.3 Експериментальне дослідження часових характеристик методів факторизації

Для експериментального дослідження факторизації числа $n=p \cdot q$ множник p вибирався фіксованим і дорівнював найбільшому простому числу певної розрядності. Далі визначався ряд простих чисел тієї ж розрядності, що і p, з якого рівномірно в порядку зростання вибиралися 1000 значень для числа q. Після перемноження p на q таймер фіксував час факторизації кожного з 1000 отриманих добутків [315, 317].

Усі обчислення проводилися на портативному комп'ютері Lenovo B50-70 з процесором Intel Pentium 3558U (1.7 ГГц). Об'єм оперативної пам'яті у пристрої становив 4 GB. При проектуванні програмного комплексу, який

забезпечував обчислення, було обрано мову програмування високого рівня C++, що дозволяє трансформувати коди під різні архітектури та операційні системи.

Для зменшення випадкових впливів на роботу комп'ютера усі обчислення повторювалися 10 разів. Результати записувалися в файл з розширенням .csv. На рис. 7.8 показано головне вікно програми, а на рис. 7.9 – зразок отриманого файла.

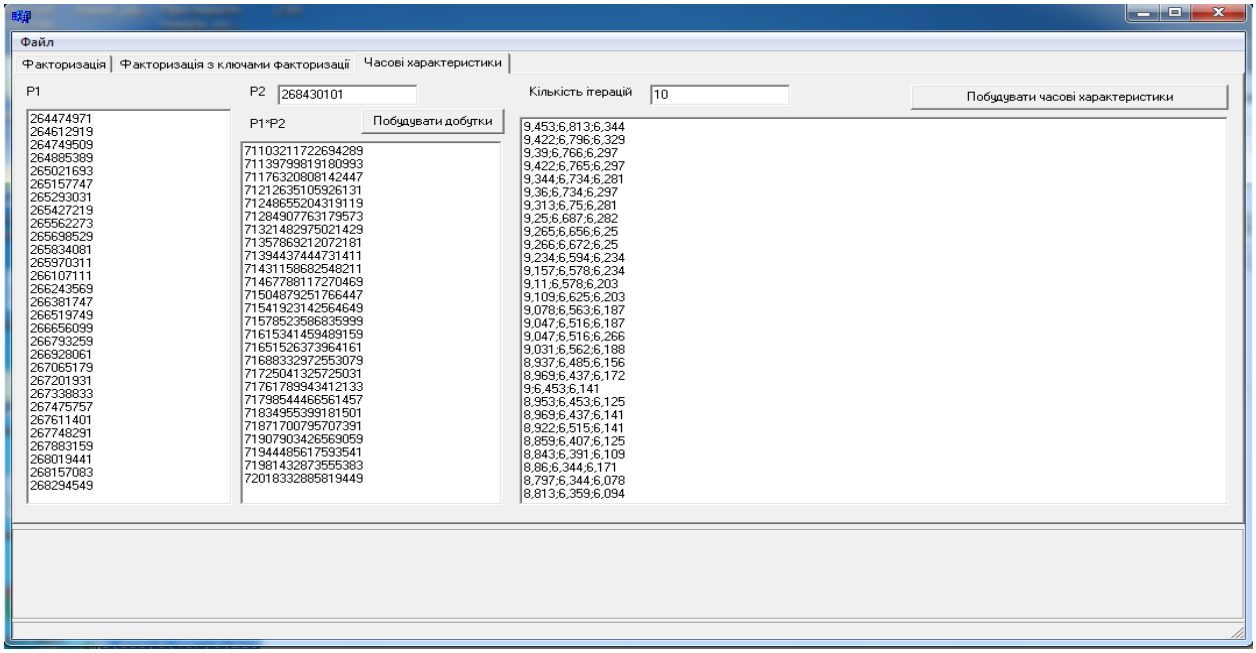


Рис. 7.8 – Головне вікно програми

Часові характеристики програмної реалізації класичного (крива 1) і вдосконаленого (крива 2) методів Ферма, а також запропонованого, згідно (2.36)-(2.37), методу (крива 3) представлено на рис. 7.10 і 7.11 для розрядностей 30 (p=1073736121) та 32 (p=4294952719) біти відповідно. Методи перевірки та зберігання простих чисел описані в [339-343].

Усі графіки носять спадаючий характер, що свідчить про зменшення часу факторизації із зменшенням різниці між множниками, добуток яких факторизується. Запропонований алгоритм, зменшення часу для якого відбувається лінійно, має перевагу перед іншими при малих значеннях q.

	A	B	C	D
1	18,265	13,11	8,828	
2	18,047	12,937	8,813	
3	17,89	12,891	8,812	
4	17,938	12,812	8,766	
5	17,844	12,781	8,813	
6	17,765	12,781	8,766	
7	17,766	12,812	8,719	
8	17,719	12,718	8,75	
9	17,688	12,781	8,719	
10	17,672	12,656	8,766	
11	17,64	12,641	8,766	
12	17,562	12,594	8,734	
13	17,563	12,547	8,687	
14	17,484	12,563	8,672	
15	17,5	12,515	8,641	
16	17,375	12,5	8,703	
17	17,36	12,437	8,641	
18	17,297	12,468	8,672	
19	17,422	12,453	8,61	
20	17,312	12,453	8,61	
21	17,218	12,328	8,594	
22	17,188	12,375	8,593	
23	17,141	12,297	8,625	

Рис. 7.9 – Зразок отриманого файла з розширенням .csv

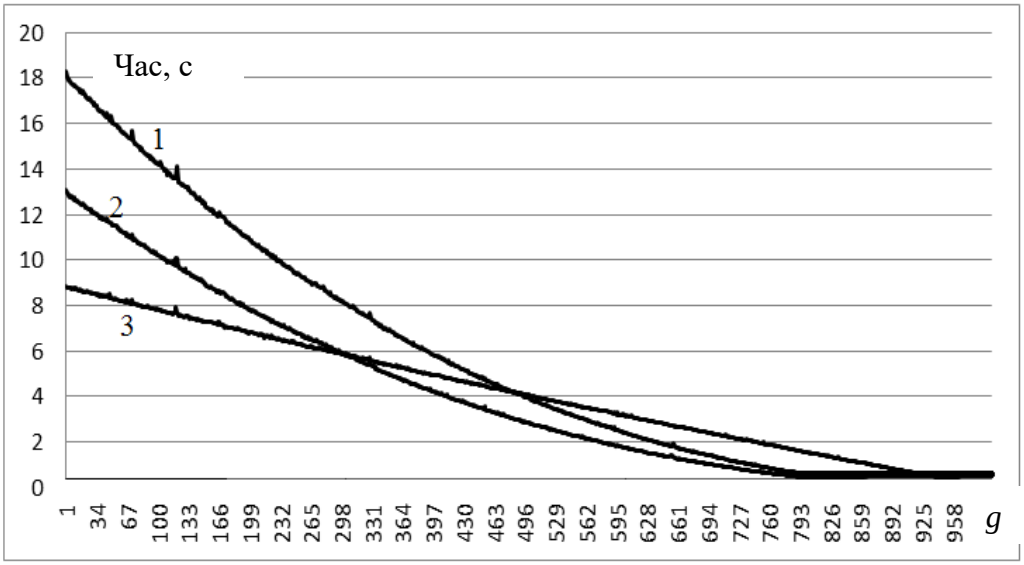


Рис. 7.10 - Часові характеристики програмної реалізації методів факторизації для множників розрядністю 30 біт (g – номер числа)

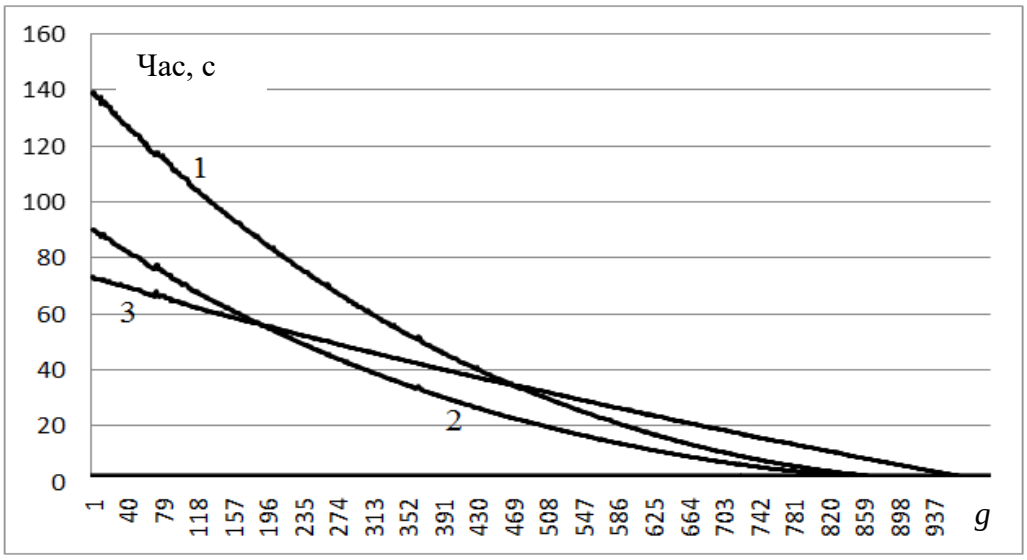


Рис. 7.11 - Часові характеристики програмної реалізації методів факторизації для множників розрядністю 32 біт (g – номер числа)

При збільшенні q найменшим часом характеризується вдосконалений алгоритм Ферма. Подальше зростання q приводить до того, що запропонований алгоритм використовує найбільше часу в порівнянні з двома іншими. Слід зазначити, що при збільшенні розрядності чисел точки перетину прямої лінії із кривими, отриманими при використанні методу Ферма, на графіках зсуваються ліворуч.

На рис. 7.12 зображено графіки залежності середнього значення часу факторизації трьома методами від розрядності чисел для 1000 значень, вибраними вищеописаним способом. Видно, що усі графіки зростають за параболічним законом із збільшенням розрядності. Найменшим середнім часом факторизації характеризується вдосконалений алгоритм Ферма, а найбільшим - класичний.

Отже, запропонований метод доцільно застосовувати тоді, коли велика різниця між числами, добуток яких потрібно факторизувати. В табл. 7.6 наведено результати факторизації добутку двох простих чисел при фіксованому p=3571 та змінному q (t_1 – класичний метод Ферма, t_2 – вдосконалений метод Ферма, t_3 – запропонований алгоритм).

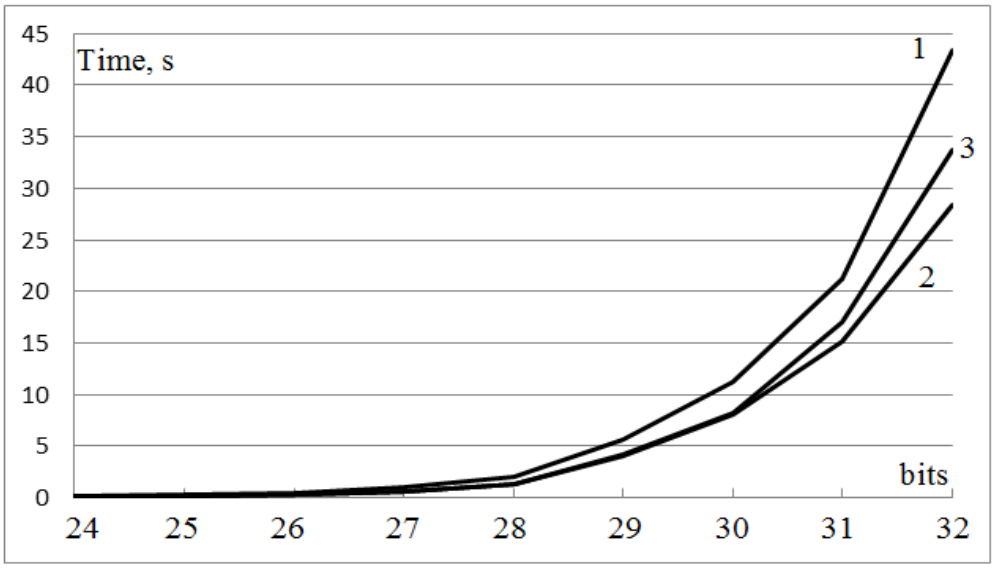


Рис. 7.12 - Графіки залежності середнього значення часу факторизації від розрядності чисел: 1 – класичний метод Ферма; 2 – вдосконалений метод Ферма; 3 – запропонований метод

Таблиця 7.6 – Часові характеристики факторизації

p	q	$p \cdot q$	t_1, с	t_2, с	t_3, с
3571	99991	357067861	0,031	0,031	0,016
3571	323789	1156250519	0,172	0,157	0,032
3571	523771	1870386241	0,297	0,281	0,047
3571	723791	2584657661	0,438	0,406	0,063
3571	1913803	6834190513	1,156	1,109	0,203
3571	2913803	10405190513	1,781	1,703	0,313
3571	7913809	28260211939	4,86	4,656	0,875
3571	9913807	35402204797	6,265	5,828	1,094
3571	19191383	68532428693	12,843	12,156	2,407
3571	39192331	139955814001	26,875	26,235	4,969
3571	49392341	176380049711	32,859	34,406	6,172
3571	139392347	497770071137	103,078	86,391	18,031
3571	337392373	1204828163983	214,141	220,375	42,483
3571	931392317	3326001964007	646	571,828	116,453
3571	1931392319	6897001971149	1112,657	1106,469	269,156
3571	2971215073	10610209025683	2775,718	2765,922	437,531
3571	6712170737	23969161701827	11159,56	9080,454	937,67

З табл. 7.6 видно, що удосконалений метод Ферма збільшує швидкодію факторизації приблизно в 1,1-1,2 рази, а запропонований – в 5-11 разів порівняно з класичним методом Ферма.

7.4 Часові характеристики програмної реалізації методів пошуку оберненого елемента за модулем

Для експериментальних досліджень було обрано портативний комп'ютер Lenovo B50-70 з процесором Intel Pentium 3558U (1.7 ГГц). Об'єм оперативної пам'яті в пристрої становив 4 GB. При проектуванні програмного комплексу, який забезпечував обчислення, було обрано мову програмування високого рівня C++, що дозволяє трансформувати коди під різні архітектури та операційні системи [303].

Даний комп'ютер не дозволяє розпаралелювати обчислення на будь-яку кількість потоків, тому після визначення меж усі обчислення виконувалися послідовно, а час пошуку оберненого елемента за модулем вибирався в тому потоці, де знаходився шуканий результат.

На рис. 7.13 показано графічну залежність середнього часу пошуку оберненого елемента на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) від кількості паралельних потоків z при n=1021 (рис. 7.13а) та n=65521 (рис. 7.13б). Останні є найбільшими простими десяти- та шістнадцятирозрядними числами. Число a пробігає значення від 2 до n-1 з кроком 1 [303].

З рис. 7.13 видно, що в обох випадках метод додавання модуля володіє більшою швидкодією, ніж додавання залишку. Графіки мають приблизно однаковий характер. При розпаралелюванні на два та три потоки час обчислень різко зменшується порівняно з послідовним виконанням усіх операцій (z=1). При подальшому збільшенні z інтенсивність зменшення часу спадає. Незначне збільшення часу спостерігається, коли z=7, 8, і надалі графік повільно спадає. Тому для подальших досліджень було вибрано z=6.

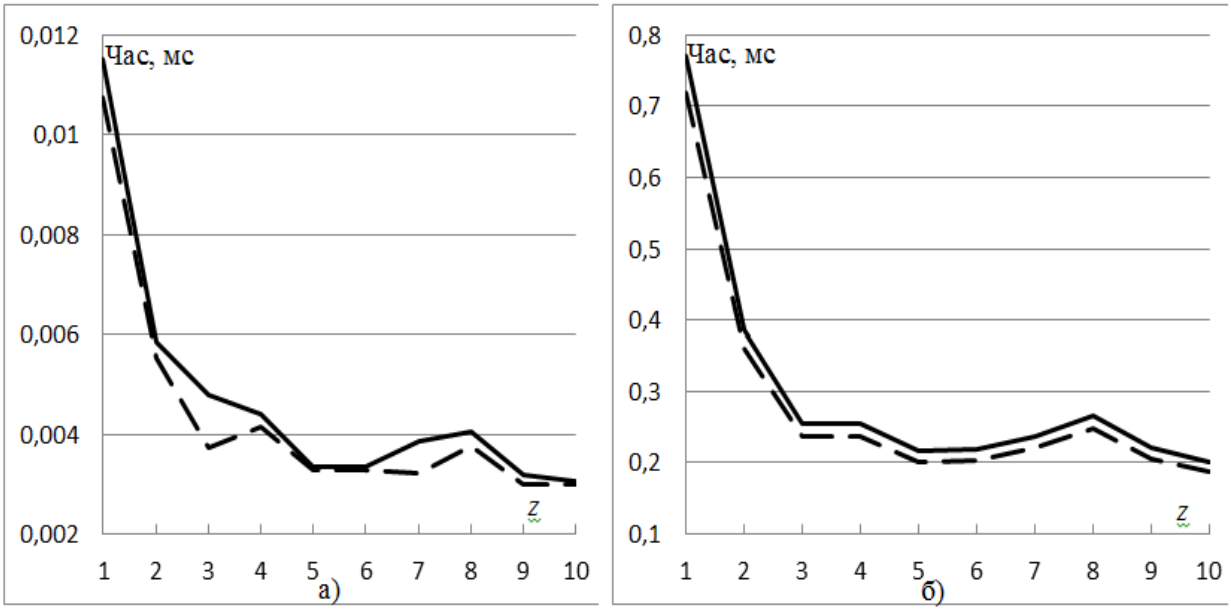


Рис. 7.13 - Графічна залежність середнього часу пошуку оберненого елемента на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) від кількості паралельних потоків при n=1021 (рис. 7.13а) та n=65521 (рис. 7.13б)

На рис. 7.14 показано графічну залежність часу пошуку оберненого елемента на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) при z=6 та n=65521, а в табл. 7.7 представлений фрагмент з частиною отриманих результатів [303].

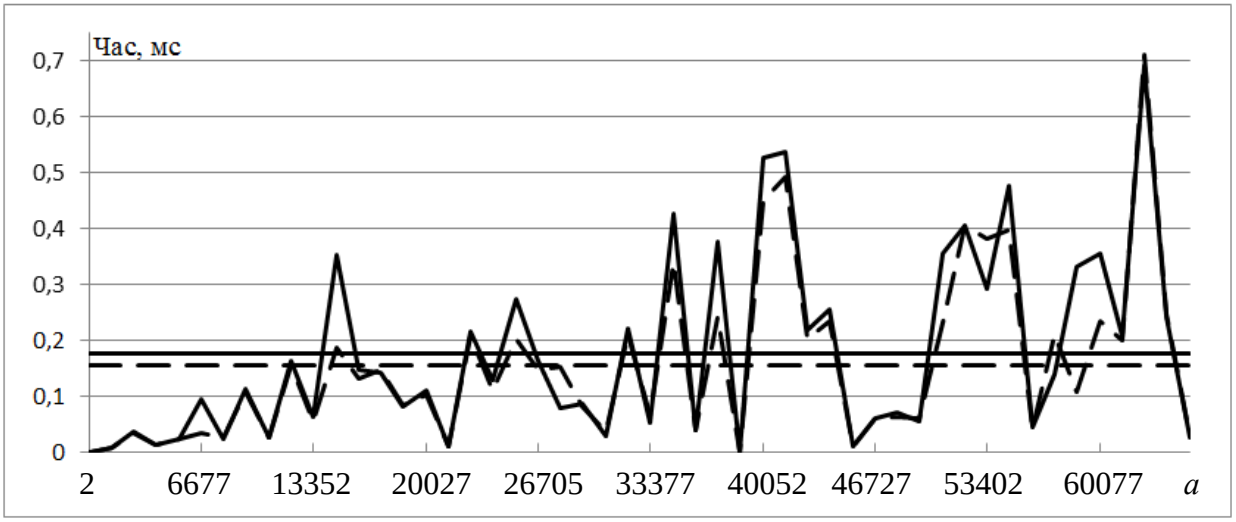


Рис. 7.14 - Графічна залежність часу пошуку обернених елементів на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) при z=6 та n=65521 та середні часи обчислень (штрихова та суцільна прямі відповідно)

Число *a* змінюється від 2 до 65417 з кроком 1335, який був вибраний з міркувань того, щоб розрахунки оберненого елемента за модулем здійснювалися для 50-ти значень. Крім того, наведено середній час обчислень обома методами (штрихова пряма – додавання модуля, суцільна пряма – додавання залишку). Графіки часу для пошуку оберненого елемента носять осцилюючий характер, однак загальний тренд вказує на збільшення часу із ростом числа *a*. В переважній більшості випадків метод додавання модуля є швидший (усереднений час обчислень розглянутих прикладів становить 0,15499 мс), ніж метод додавання залишку (середній час – 0,176144 мс), в 1,136 рази [303].

Таблиця 7.7 – Фрагмент з частиною отриманих результатів

		Час пошуку, мс	
a	*b*	Додавання модуля	Додавання залишку
17357	55744	0,146275	0,142446
18692	49968	0,08318	0,08077
20027	29906	0,10303	0,111585
21362	22510	0,009311	0,009736
22697	35106	0,207195	0,216458
24032	50643	0,109552	0,129591
25367	33547	0,202232	0,272227
26702	30557	0,146747	0,164092
28037	25648	0,152182	0,078502

На рис. 7.15 показано графічну залежність середнього часу пошуку оберненого елемента на основі розширеного алгоритму Евкліда (крива 1) та запропонованих методів додавання модуля (штрихові лінії 3, 5) та залишку (суцільні лінії 2, 4) при z=1 (криві 2, 3) і z=6 (криві 3,5) від розрядності

модуля n_0. Модуль n вибирався найбільшим простим, але меншим за 2^{n_0}, де n_0 змінювалося від 10 до 19,5 з кроком 0,5. Число a набувало 1000 різних значень. Крок його зміни вибирався таким, щоб охопити весь діапазон від 1 до n.

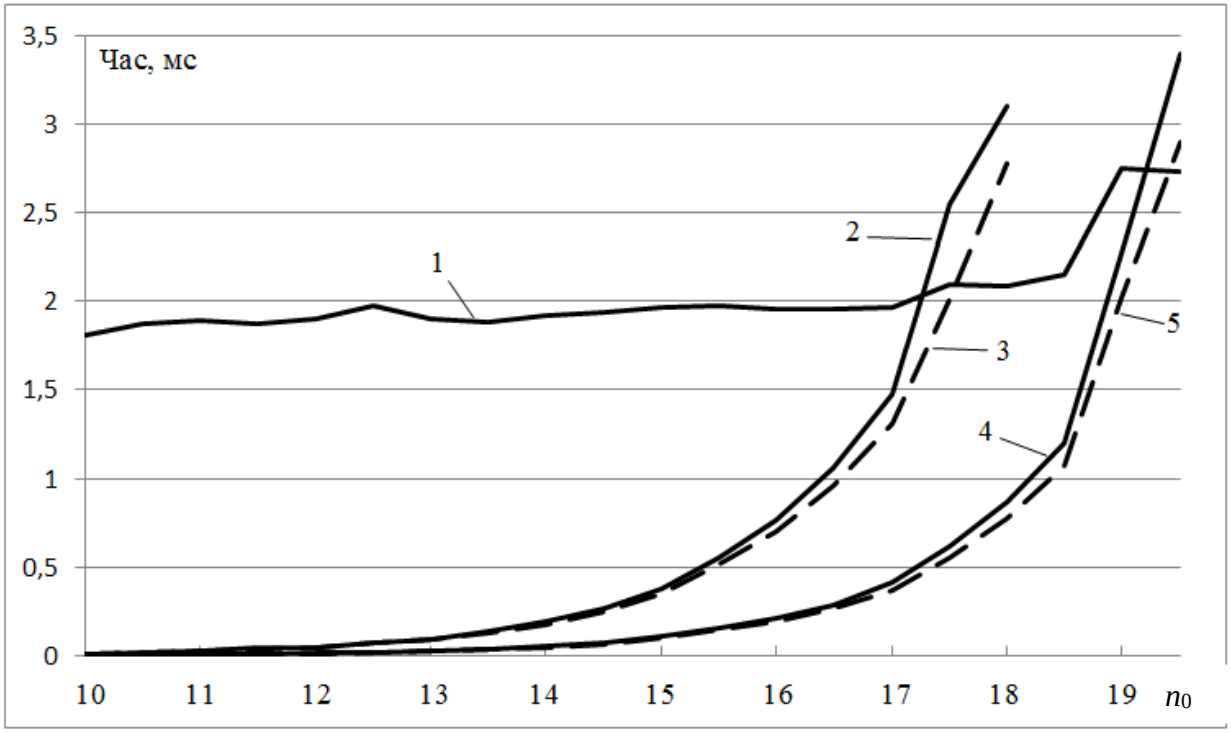


Рис. 7.15 – Графічна залежність часу пошуку оберненого елемента на основі розширеного алгоритму Евкліда (крива 1) та запропонованих методів додавання модуля (криві 3, 5) та залишку (криві 2, 4) при z=1 (криві 2, 3) z=6 (криві 4,5) від розрядності модуля

Як видно з рис. 7.15, при використанні алгоритму Евкліда із збільшенням розрядності чисел час збільшується дуже повільно. При n_0=19 час різко зростає і далі знову наступає практично горизонтальна ділянка графіка. В обох запропонованих методах середній час збільшується приблизно параболічно, причому метод додавання модуля має більшу швидкодію. При z=1, тобто без розпаралелення, алгоритм Евкліда випереджає запропоновані алгоритми при n_0=17,5, а для z=6 – при n_0=19,5. Це

вказує на необхідність розпаралелення процесу обчислень для зменшення часу пошуку оберненого елемента за модулем.

Для нівелювання випадкових впливів на час роботи усі експерименти повторювалися 100 разів.

7.5 Дослідження апаратної реалізації методів пошуку оберненого елемента за модулем

Дослідження часових характеристик апаратної реалізації методу пошуку оберненого елемента на основі додаванням модуля здійснювалося у порівнянні із розширеним алгоритмом Евкліда [302].

Для реалізації розширеного алгоритму Евкліда використовується декілька видів позначень, зокрема power_1 та modulo, які у відповідності позначають значення a та p у виразі $a^{-1} \bmod p$ для знаходження оберненого елементу за модулем.

Пропонований програмний засіб складається з трьох основних блоків, які в загальному формують криптопроцесор, що передбачає роботу із програмованою логічною інтегральною схемою.

Функціональні модулі у запропонованій програмі наступні:

- модуль підключення необхідних бібліотек;
- модуль оголошення внутрішніх та зовнішніх портів (інтерфейсу проекту);
- модуль опису поведінки розробленої моделі.

Оголошення бібліотек включає декілька основних компонент, до яких програма буде звертатись у подальшому. В даному випадку опис доступу до необхідних бібліотек в середовищі Active-HDL має вигляд:

```
LIBRARY IEEE;
IEEE.STD_LOGIC_1164.ALL;
IEEE.NUMERIC_STD.ALL;
IEEE.MATH_REAL.ALL;
```

Бібліотека LIBRARY IEEE - це бібліотека вищого рівня, розроблена всесвітньою організацією IEEE, яка постійно оновлюється. Тобто за її допомогою виконується доступ і використання бібліотек нижнього рівня.

Бібліотека IEEE.STD_LOGIC_1164.ALL - це стандартизований пакет даних, який використовується при моделюванні і описі бітових типів даних. Логічна система цього пакету передбачає в коді програми роботу над логічними елементами. Проте виконання такого примітивного завдання, як моделювання роботи вентильного транзистора, виходить за межі функціоналу цієї бібліотеки.

Бібліотека IEEE.NUMERIC_STD.ALL – пакет, що передбачає оголошення цифрових і деяких математичних функцій, які використовуються у проектах, що будуть синтезуватись. У цьому пакеті оголошується принцип роботи програмного продукту із числовими масивами і виконання з ними елементарних операцій (+,-,*,/ зсуву ліворуч і праворуч).

Бібліотека IEEE.MATH_REAL.ALL використовується як доповнення до пакету IEEE.NUMERIC_STD.ALL, щоб можна було виконувати із числами більш складні математичні операції - піднесення до степеня, добування коренів різного степеня, знаходження модулів і багато інших.

Модуль оголошення внутрішніх і зовнішніх портів являє собою виділений блок, який приймає значення і по суті керує всім програмним продуктом, адже саме в цьому блоці описується, звідки будуть зчитуватись дані і куди їх у подальшому буде записано після обробки. Цей модуль містить такий блок:

- port, в якому описується вхідні та вихідні порти програми:

```
port (
clk : in std_logic
        );
```

У ньому оголошується внутрішній індикатор цифрових імпульсів, тактових сигналів.

Блок опису коду програми організований таким чином, що на його початку описано усі внутрішні змінні, які будуть використовуватись в програмному продукті. Ці змінні оголошуються за допомогою зарезервованого слова signal і описуються як дійсні значення у відповідних діапазонах:

```
signal modulo: INTEGER ;
signal power_1: INTEGER  ;
signal exite: INTEGER :=0 ;
```

В основному усі ці змінні застосовуються для проміжних обчислень.

Пошук оберненого елемента за модулем за допомогою розширеного алгоритму Евкліда описано як процес від clk:

```
read: process(clk)
        begin
            if clk'event and clk='1' then
                if (power_1-1) mod modulo /= 0 then
                    while (exite*power_1 - 1) mod modulo /=0
                        loop
                        exite<=exite+1;
                        exit when exite+1>exite;
                    end loop;
                else exite<= 1;
                end if;
            end if;
        end process;
```

Функціональна схема розробленого пристрою представлена на рис. 7.16, а на рис. 7.17 – блок-схема його роботи [302].

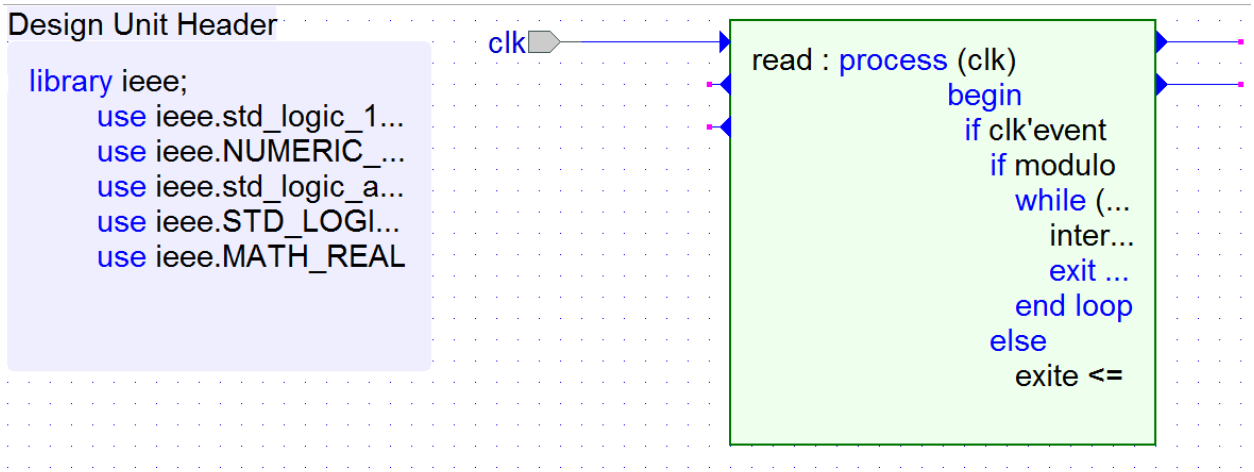


Рис. 7.16 - Функціональна схема розробленого пристрою

Для знаходження оберненого елементу за модулем при додаванні модуля поведінку алгоритму описується також як процес від clk:

```
read: process(clk)
                begin
                    if clk'event and clk='1' then
                        while internal<1 loop
                            d<=modulo mod power_1;
                            c<=d+1;
                            internal<=1;
                            exit when internal<1;
                        end loop;

                            while c>0 loop
                                    c<=(d+c) mod power_1;

    exite<=((internal*modulo)+1)/power_1;
                                    internal<=internal+1;
                                    exit when c>0;
                            end loop;
                    end if;
    end process;
```

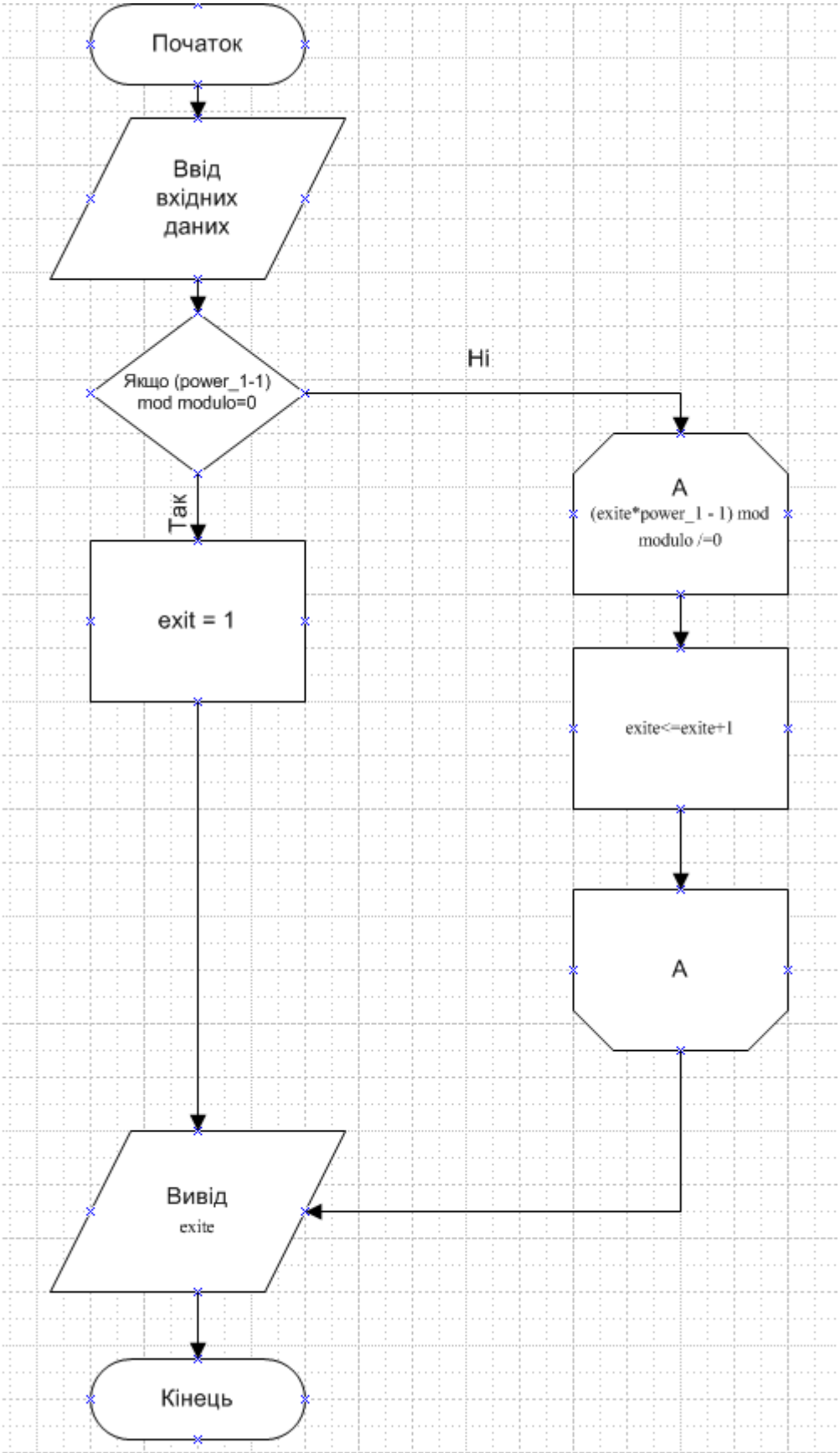


Рис. 7.17 – Блок-схема алгоритму роботи розробленого пристрою

На рис. 7.18 представлена блок-схема роботи пристрою пошуку оберненого елемента на основі додавання модуля, а на рис. 7.19 – його функціональна схема.

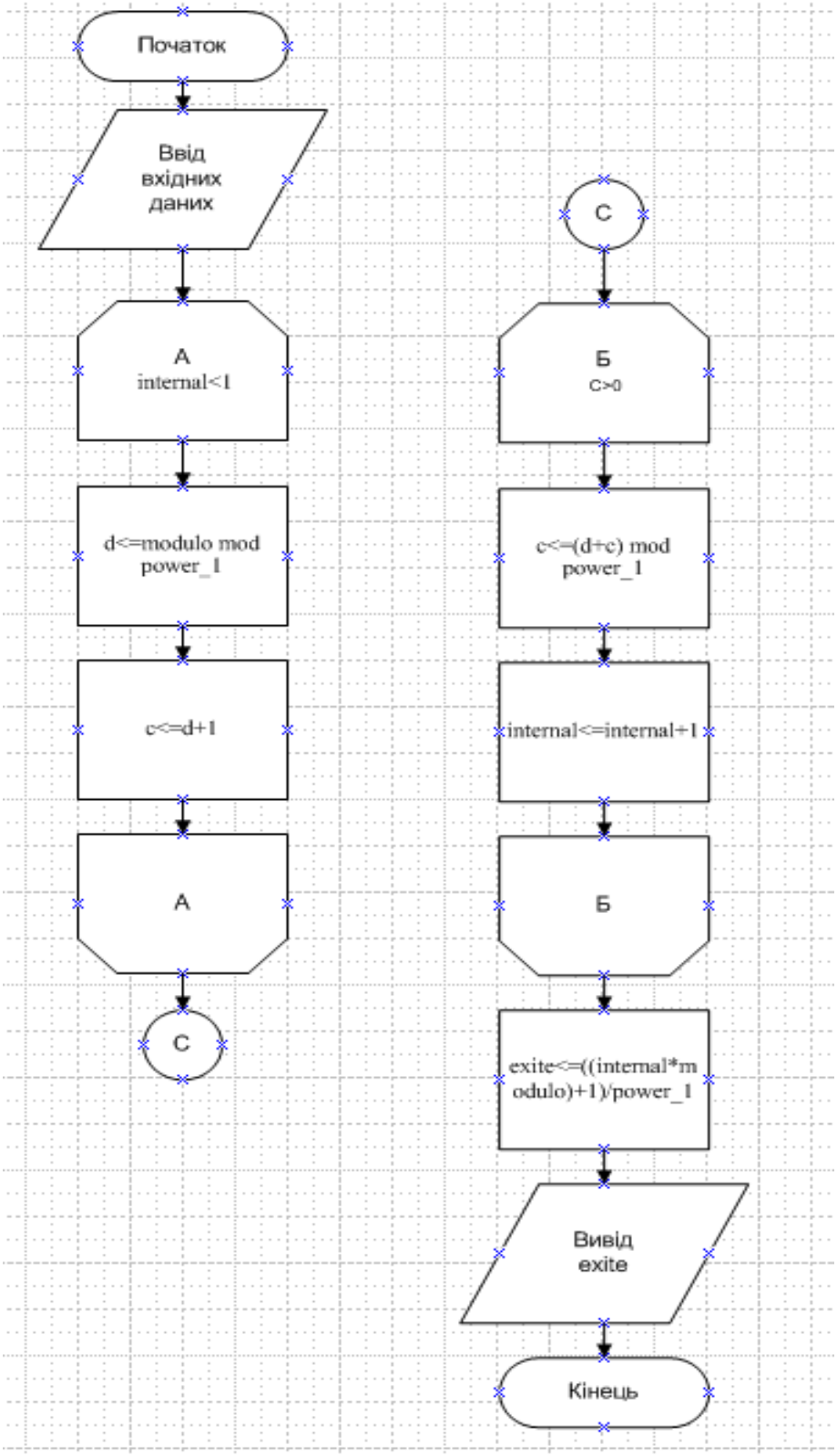


Рис. 7.18 – Блок-схема алгоритму роботи пристрою пошуку оберненого елемента на основі додавання модуля

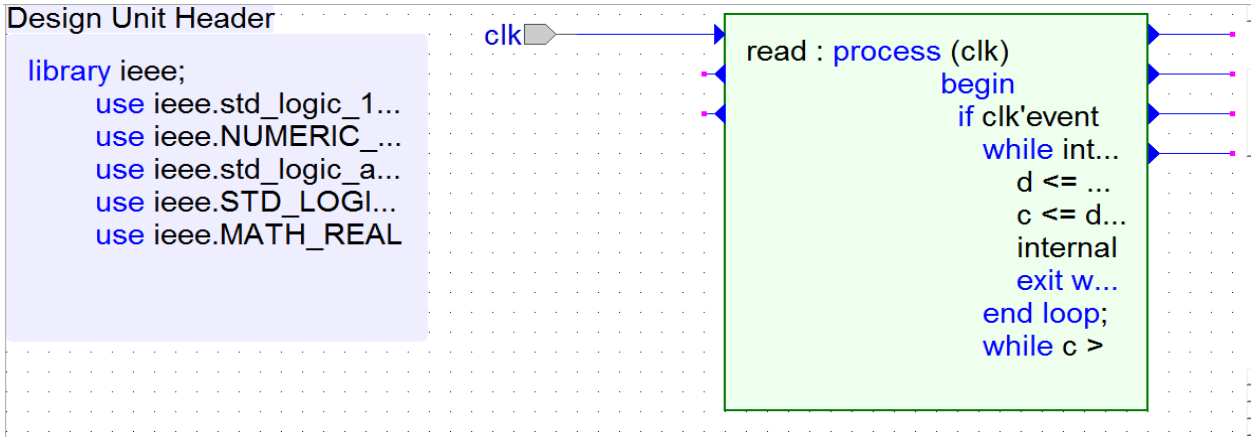


Рис. 7.19 - Функціональна схема пристрою пошуку оберненого елемента на основі додавання модуля

Програмно/апаратна реалізація пошуку оберненого елементу за модулем на основі додавання залишку представлена у варіанті знаходження значення від тактового процесу:

```
read: process(clk)
                begin
                    if clk'event and clk='1' then

                    if modulo mod power_1 /= 0 then
                    while (modulo*internal + 1) mod power_1 /=0
                        loop

                            internal<=internal+1;
                            exit when internal+1>internal;

                        end loop;
                    else exite <= 1;
                    exite<=(modulo*internal+1)/power_1;
                    end if;
    end process;
```

На рис. 7.20 представлена блок-схема роботи пристрою пошуку оберненого елемента на основі додавання залишку, а на рис. 7.21 – його функціональна схема.

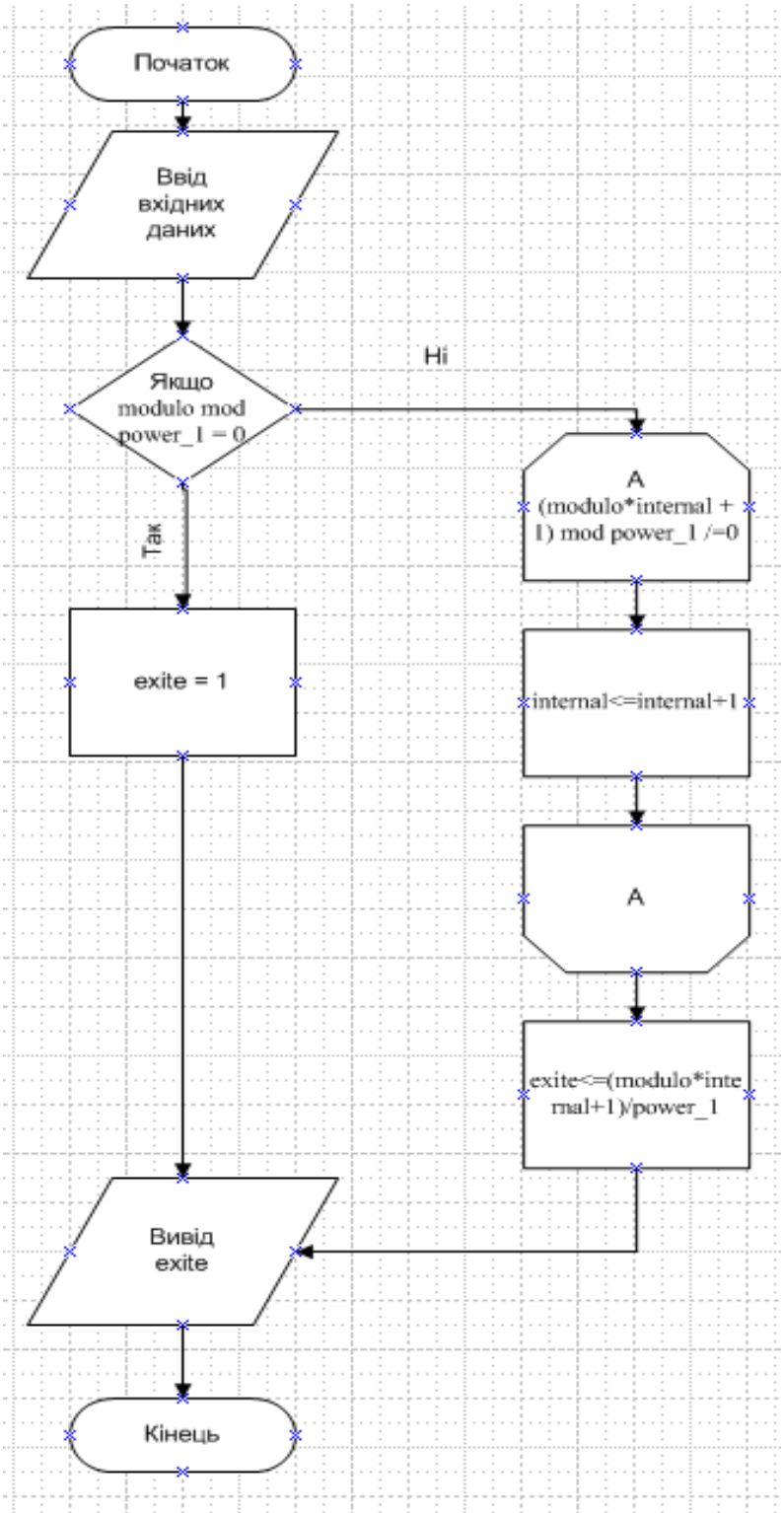


Рис. 7.20 – Блок-схема алгоритму роботи пристрою пошуку оберненого елемента на основі додавання залишку

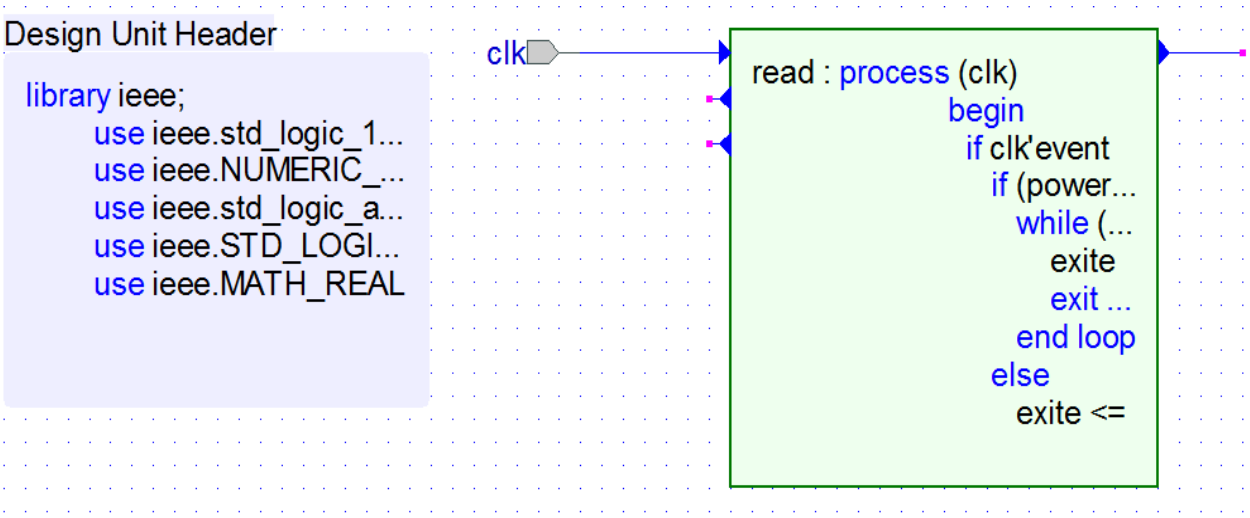


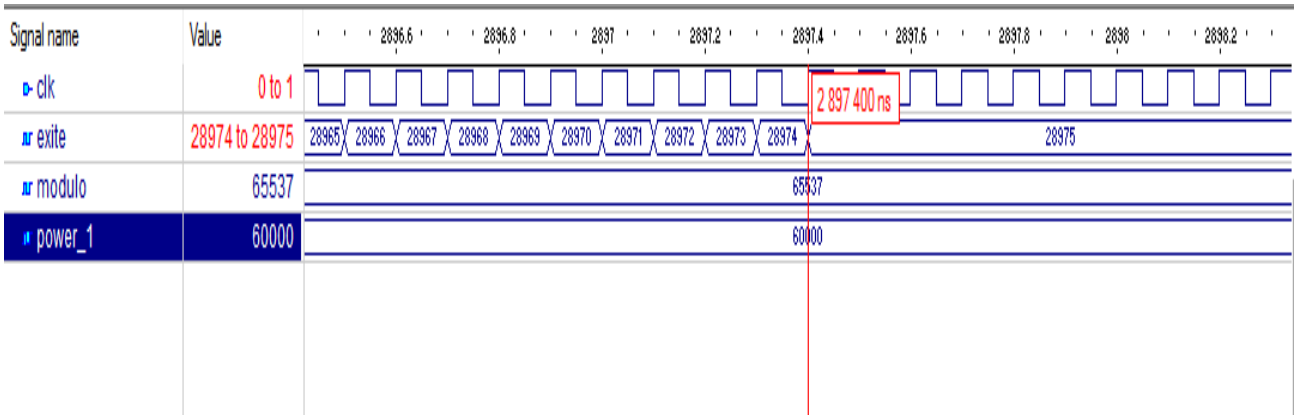
Рис. 7.21 - Функціональна схема пристрою пошуку оберненого елемента на основі додавання залишку

Для експериментального дослідження часових затрат обчислення оберненого елемента за модулем із використанням розширеного алгоритму Евкліда і запропонованого методу для кожного з них було реалізовано програмно-апаратний модуль на базі середовища розробки Aldec Active-HDL 9.1.

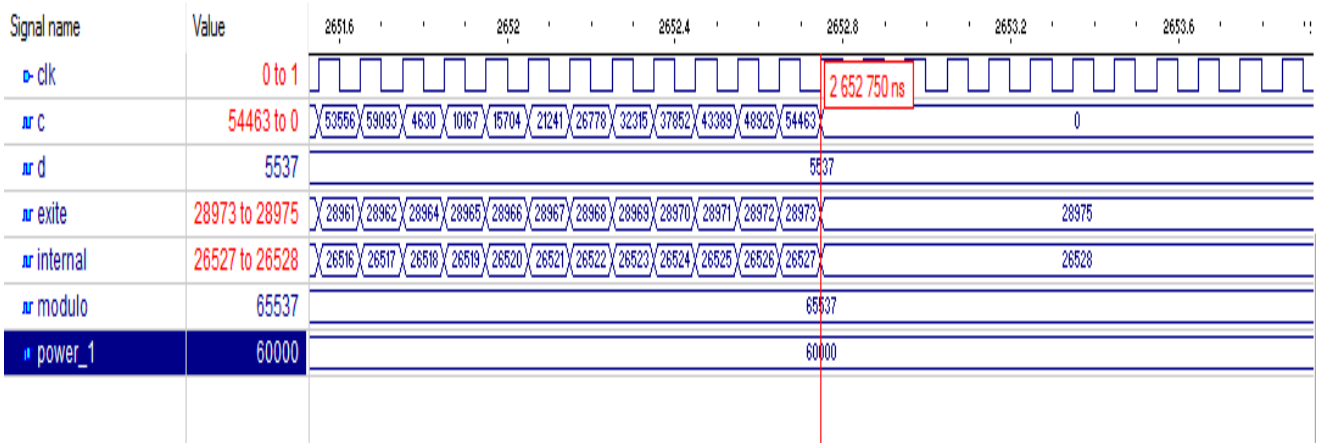
На рис. 7.22 представлені часові діаграми роботи спроектованих пристроїв пошуку оберненого елемента за модулем (для прикладу було вибрано $60000^{-1}\text{mod}65537$) за допомогою розширеного алгоритму Евкліда та додавання модуля.

Наведений приклад показує, що час роботи системи зменшився з 2,8974 мс (за допомогою розширеного алгоритму Евкліда) до 2,65275 мс (за допомогою запропонованого методу додавання модуля), тобто приблизно в 1,09 разів.

При дослідженні часових параметрів програмно-апаратної реалізації вищеназваними методами число p=65537 вибиралося простим та фіксованим, щоб для будь-якого елемента $a<p$ існував обернений елемент. Число a змінювалося від 5000 до 65000 із кроком 5000. Результати чисельного експерименту наведено у табл. 7.8 [302].



а)



б)

Рис. 7.22 - Часова діаграма пошуку оберненого елемента за модулем: а) на основі алгоритму Евкліда; б) на основі додавання модуля

На рис. 7.23 представлено графічні залежності часових затрат пошуку оберненого елемента за допомогою розширеного алгоритму Евкліда (графік 1) і на основі додавання модуля (крива 2) згідно із даними табл. 7.8, та усереднені значення отриманих результатів (прямі 3 і 4 відповідно).

Результати проведених досліджень вказують, що тільки в двох з тринадцяти випадків при a=40000 і a=45000 запропонований метод поступається класичному, що пояснюється необхідністю виконання великої кількості операцій додавання. Середній час пошуку оберненого елемента за модулем на основі алгоритму Евкліда (3343600 ps) в 1,47 разів більший від аналогічного параметра із використанням методу додавання залишків (2280619 ps).

Таблиця 7.8 – Порівняльні характеристики часових затрат обчислення оберненого елемента за модулем

№	a	$a^{-1}\mathrm{mod}p$	Час роботи, ps	
			Алгоритм Евкліда	Додавання модуля
1	5000	20015	201300	113540
2	10000	42776	4277500	652750
3	15000	50363	5036200	1152750
4	20000	21388	2138700	652750
5	25000	4003	400200	152750
6	30000	57950	5794900	2652750
7	35000	40309	4030800	2152750
8	40000	10694	1069300	3552750
9	45000	60479	5542000	6553550
10	50000	34770	3476900	2652750
11	55000	37567	3756600	3152750
12	60000	28975	2897400	2652750
13	65000	56994	4845000	3553450
С. ч	-	-	3343600	2280619

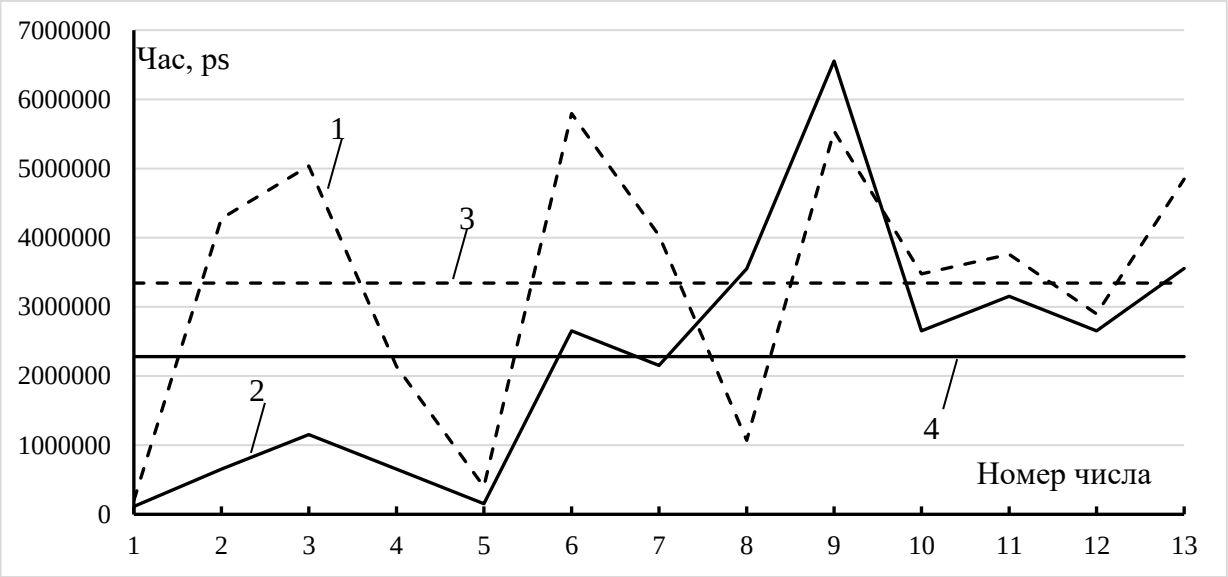


Рис. 7.23 – Часові характеристики VHDL – реалізації методів пошуку оберненого алгоритму за модулем

7.6 Часові характеристики трьохмодульної криптосистеми Рабіна

У табл. 7.9 (результати розміщені в порядку зростання діапазону обчислень P) наведено час t_{1i} та t_{2i} (i=1, 2) виконання трьохмодульного криптоалгоритму Рабіна на основі звичайної цілочисельної СЗК та для деяких значень простих модулів, які утворюють МДФ СЗК [328-330].

Таблиця 7.9 – Час виконання трьохмодульної криптосистеми Рабіна

№	p	q	r	P	Word=131071		Word=229376	
					t_{11}	t_{12}	t_{21}	t_{22}
1	41	71	97	282367	4100	1000	4100	1300
2	43	71	109	332777	4200	3000	4200	1900
3	41	53	181	393313	4100	900	4100	1800
4	29	31	449	403651	2900	800	2900	1300
5	37	41	379	574943	3700	1500	3700	2200
6	53	89	131	617927	5300	3200	5300	1000
7	59	79	233	1086013	5900	5500	5900	5100
8	67	101	199	1346633	6700	2900	6700	3000
9	59	71	349	1461961	7300	7200	5900	2400
10	41	43	881	1553203	5000	4900	11900	11800
11	67	89	271	1615973	6700	3600	6700	4900
12	53	59	521	1629167	9900	9800	5300	4000
13	71	101	239	1713869	7100	4600	7100	2200
14	61	71	433	1875323	6100	4200	6100	3500
15	79	131	199	2059451	7900	3200	7900	3000
16	97	109	881	9314813	9700	4900	11900	11800

Блоки відкритого тексту вибиралися таким чином: Word1=131071, вага Хемінга якого є максимальною для даної розрядності (131071=2^{17}-1), та

Word2=229376, у двійковому записі якого в трьох старших розрядах знаходяться одиниці, а у всіх решти розрядах - нулі.

Час для різних чисел позначений відповідно t_{j1} та t_{j2} (j=1, 2). Розшифровування у звичайній цілочисельній СЗК (i=1) відбувається за формулою (6.6), обернені елементи ($m_1=p$-1, $m_2=m_3$=1) шукаються на основі додавання модуля. При використанні МДФ СЗК розшифровування відбувалося за формулою:

$$M_1 = \left(-x \cdot S_1 + y \cdot S_2 + z \cdot S_3\right) \bmod n . \qquad (7.6)$$

З табл. 7.9 видно, що для модулів, які утворюють МДФ СЗК, використання виразу (7.6) зменшує час криптоперетворень. При шифруванні Word1 для модулів p=41, q=53, r=181 досягається максимальна перевага приблизно в 4,6 разів.

В цей же час існують модулі криптоперетворень p=53, q=59, r=521 для яких спостерігається мінімальна перевага, тобто прискорення досягається 1,01 разів. Це пояснюється різною кількістю ітерацій при пошуку оберненого елемента [329].

У криптоперетворенні блоку Word2 максимальна і мінімальна переваги у 5,3 та 1,008 разів відповідно спостерігаються при використанні модулів p=53, q=89, r=131 та p=97, q=109, r=881.

В середньому час виконання операцій при використанні МДФ СЗК для Word1 та Word2 зменшився відповідно у 1,58 і 1,63 рази.

На рис. 7.24 представлені графіки часу виконання криптоперетворень від номера наборів модулів згідно табл. 7.9.

З рис. 7.24 видно, що всі графіки носять осцилюючий характер, однак загальний тренд спрямований на збільшення часу виконання криптоперетворень при збільшенні можливого діапазону виконання операцій. Запропоновану HDL-модель трьохмодульного криптоалгоритму Рабіна з використанням МДФ СЗК можна успішно реалізувати на сучасних

програмованих логічних інтегральних схемах або програмованих логічних матрицях.

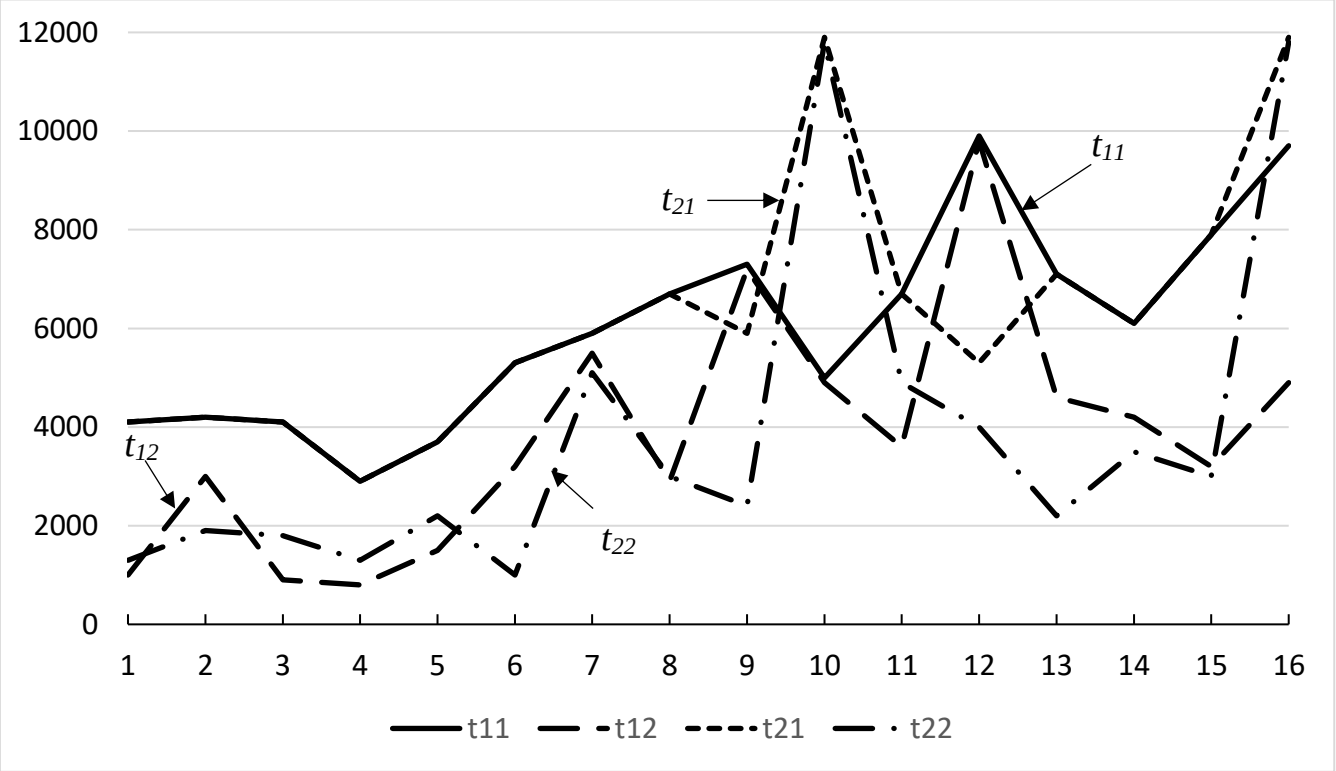


Рис. 7.24 - Графіки залежності часу шифрування від номера наборів модулів

7.7 Дослідження програмної реалізації множення у СЗК та МДФ СЗК

Для програмної реалізації операції множення в СЗК і МДФ СДК була обрана високорівнева мова програмування загального призначення Python. Приклад введення вхідних параметрів у головному вікні програми представлений на рис. 7.25 [344].

Результати розміщуються у файл з розширенням .csv, ім’я якого записане в останньому рядку головного вікна і включає в себя усі вхідні параметри. Приклад отриманого файла з часом та результатами множення двох чисел наведений на рисунку 7.26.

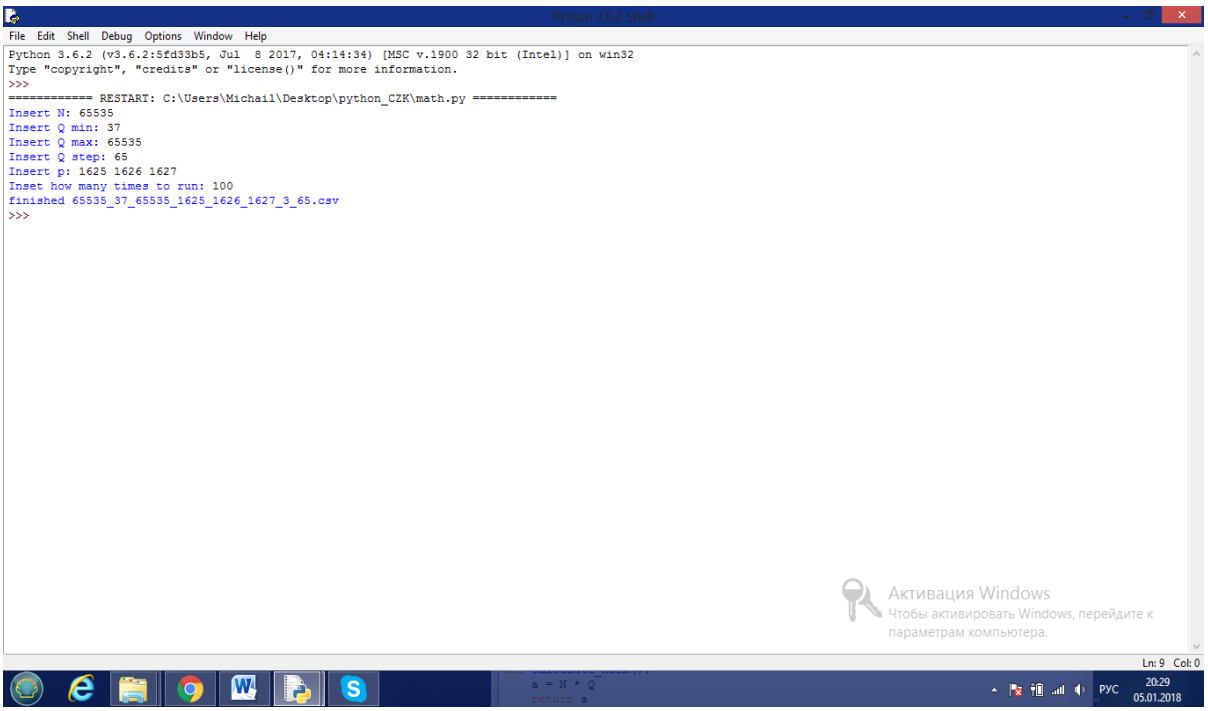


Рисунок 7.25 – Головне вікно програми

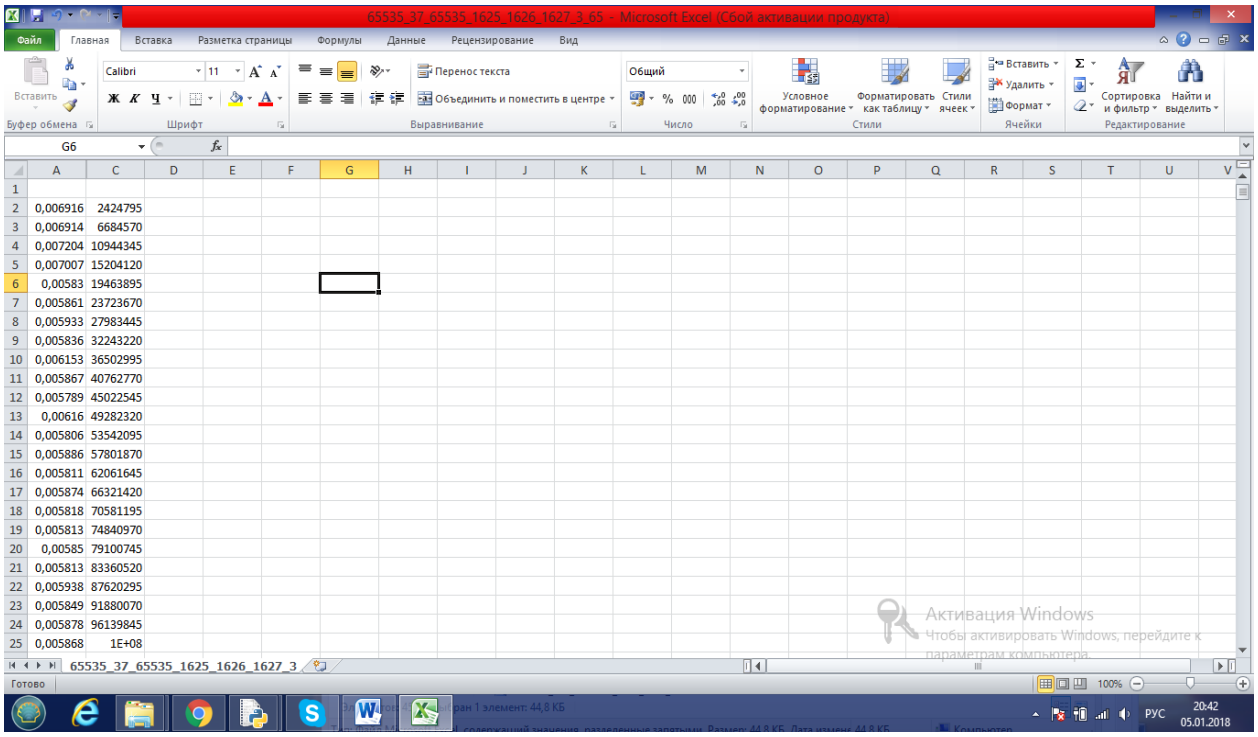


Рис. 7.26 – Приклад отриманого файла

На рис. 7.27 представлені часові характеристики виконання операції множення $N=p\cdot q$ в трьохмодульній СЗК при фіксованому множнику p=65536 з двома різними системами модулів (перший випадок - модулі мало

відрізняються один від одного: p_1=1625=$\left\lfloor \sqrt[3]{65536^2} \right\rfloor$, p_2=1626, p_3=1627 - пунктирні лінії, другий випадок - модулі відрізняються сильно: p_1=163, p_2=1627, p_3=16381 - суцільні лінії) [344].

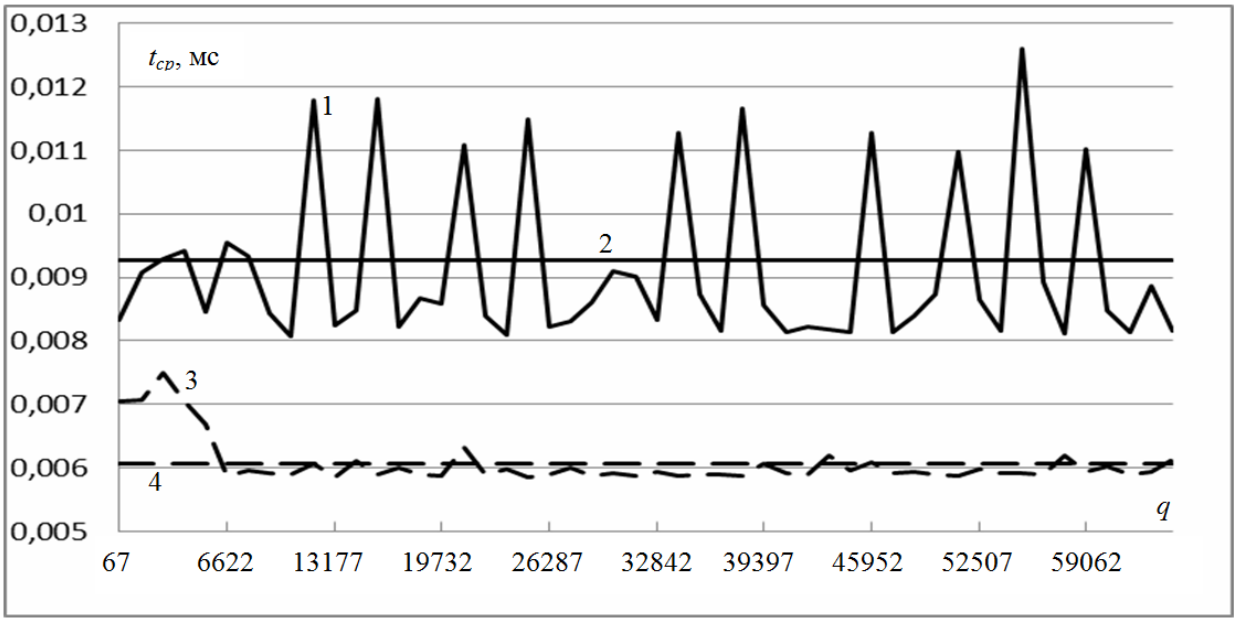


Рис. 7.27 – Часові характеристики виконання операції множення в трьохмодульній СЗК

Другий множник q змінювався від значення 67 до p з кроком 1311. Останній визначав кількість отриманих обчислень, яке дорівнювало 50. Добуток модулів обох систем перевищує 2^{32}.

Як видно з рис. 7.27, графік 1 носить осцилюючий характер. Середній час виконання операції множення (лінія 2) дорівнює 0,009259 мс. У другому випадку, крім початкових значень q, час виконання множення (графік 3) не зазнає суттєвих коливань. Середній час (лінія 4) складає 0,006066 мс, що в 1,53 рази менше, ніж в попередньому випадку. Тому для підвищення швидкодії в СЗК попарно взаємно прості модулі необхідно вибирати так, щоб вони якомога менше відрізнялися один від одного.

Для дослідження МДФ СЗК система модулів зі значною різницею між ними (p_1=651, p_2=691, p_3=11246) вибиралася за формулою (4.26), яку можна записати таким чином:

$$p_3 = p_1 + \frac{p_1^2 \pm 1}{p_2 - p_1}. \qquad (7.7)$$

При побудові трьохмодульної МДФ СЗК за формулою (7.7) не може бути вибрана система модулів однаковою розрядності. Найменша різниця між модулями буде за такої умови:

$$p_{2,3} = 2p_1 \pm 1. \qquad (7.8)$$

Виходячи із цього, були вибрані такі модулі: p_1=1025, p_2=2049, p_3=2051. Знову ж добуток модулів в обох випадках перевущує 2^{32}.

Вхідні параметри були ті ж самі, що і для звичайної СЗК. Обчислення проводилися згідно такого виразу для МДФ СЗК:

$$A = \left(-b_1 P_1 + b_2 P_2 + b_3 P_3\right) \bmod P, \qquad (7.9)$$

де b_i – залишки.

Отримані результати представлені на рис. 7.28. Суцільна лінія показує час виконання множення (крива 1) і середній час (лінія 2) для 50 значень p при p_1=651, p_2=691, p_3=11246, пунктирна (графіки 3, 4) - відповідно для p_1=1025, p_2=2049, p_3=2051 [344].

Видно, що в обох випадках при малих значеннях q амплітуда коливань велика, при збільшенні q вона зменшується за винятком невеликого відрізка в другій половині діапазону змін значення q. Середній час для системи модулів p_1=651, p_2=691, p_3=11246 складає 0,002293 мс (лінія 2), а для p_1=1025, p_2=2049, p_3=2051 - 0,002169 мс (лінія 4), що в 1,057 рази менше, ніж в попередньому випадку. Порівняння рис. 7.27 та 7.28 показує суттєве підвищення швидкодії за рахунок використання МДФ СЗК.

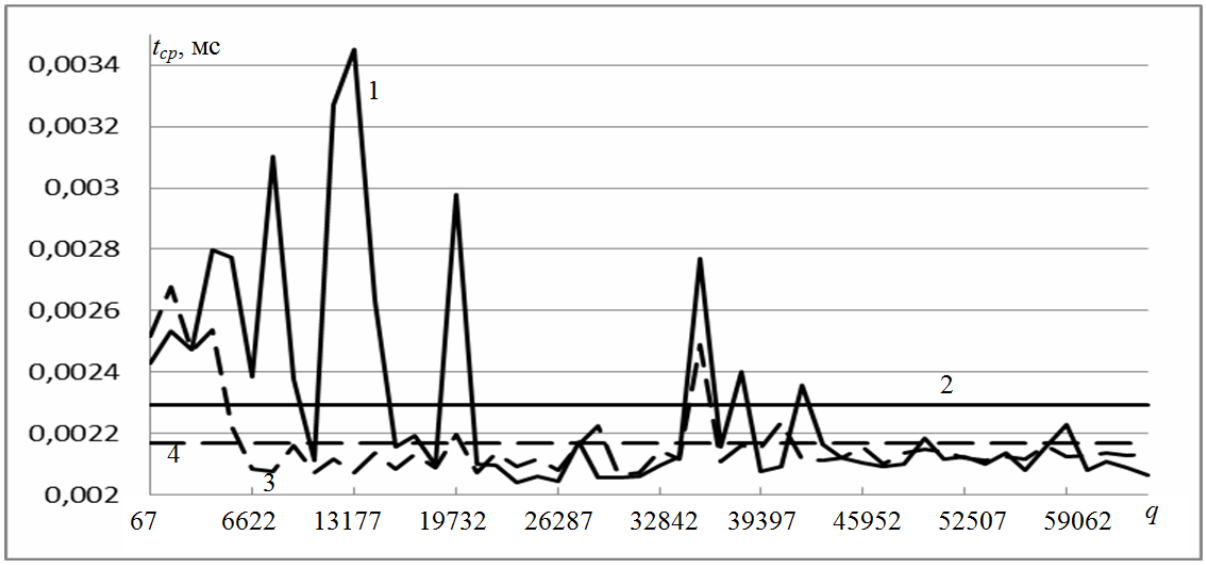


Рис. 7.28 - Часові характеристики виконання операції множення в трьохмодульній МДФ СЗК

Подальше дослідження проводилося для чисел, розрядність n_0 яких змінювалася від 16 до 24 біт. Були розглянуті чотири випадки побудови системи модулів:

1) модулі СЗК сильно відрізняються один від одного;

2) модулями є три послідовних числа, перше і третє з яких непарні: $p_1 \approx \left[\sqrt[3]{2^{2n_0}}\right]$, $p_2 = p_1+1$, $p_3 = p_1+2$;

3) модулі обчислюються за такими формулами: $p_2 = p_1+1$, $p_3 = p_1(p_1+1)-1$;

4) модулі обчислюються за такими виразами: $p_2 = 2p_1-1$, $p_3 = 2p_1+1$.

У всіх випадках добуток модулів є мінімальним, але перевищує 2^{2n_0}. У третьому і четвертому випадках системи модулів утворюють МДФ СЗК. Перший множник в добутку $N=p\cdot q$ був фіксованим: $p=2^{n_0}-1$, що відповідає максимальному числу заданої розрядності. Другий множник q змінювався від початкового значення $q = 2^{n_0} - \left[\frac{2^{n_0}}{1000}\right] + 1$ з кроком $\left[\frac{2^{n_0}}{1000}\right]$. Таким чином отримано 1000 різних значень числа q і, відповідно, час виконання 1000

операцій множення $A=p\cdot q$ при фіксованому значенні p і біжучому q. Далі для кожної розрядності визначався середній час виконання операції.

Для нівелювання випадкових впливів на роботу комп'ютера усі обчислення повторювалися 100 разів. Відповідні набори модулів, а також середній час обчислень для чисел різних розрядностей представлені в табл. 7.10 [344].

Таблиця 7.10 - Набори модулів і середній час обчислень для чисел різних розрядностей

n_0		16	17	18	19	20	21	22	23	24
Випадок 1	p_1	163	235	341	501	737	1093	1627	2429	3641
	p_2	1627	2587	4097	6503	10323	16387	26009	41287	65539
	p_3	16381	28413	49165	84541	144523	245807	416147	701881	1179703
$t_{ср}$, мкс		8,154	8,562	8.526	9,319	9,28	9,671	9,749	10,105	10,69
Випадок 2	p_1	1625	2581	4095	6501	10321	16385	26007	41285	65537
	p_2	1626	2582	4096	6502	10322	16386	26008	41286	65538
	p_3	1627	2583	4097	6503	10323	16387	26009	41287	65539
$t_{ср}$, мкс		5,891	5,95	5,962	5,966	5,934	5,93	5,982	7,185	7,304
Випадок 3	p_1	256	362	512	724	1024	1448	2048	2896	4096
	p_2	257	363	513	725	1025	1449	2049	2897	4097
	p_3	65791	131405	262655	524899	1049599	2098151	4196351	8389711	16781311
$t_{ср}$, мкс		5,461	5,98	5,618	5,689	5,65	5,684	5,57	5,732	5,593
Випадок4	p_1	1025	1626	2581	4097	6502	10322	16385	26008	41286
	p_2	2049	3251	5161	8193	13003	20643	32769	52015	82571
	p_3	2051	3253	5163	8195	13005	20645	32771	52017	82573
$t_{ср}$, мкс		5,551	5,351	5,33	5,364	5,365	5,44	5,956	5,959	6,679

На рис. 7.29 представлені графіки залежності середнього часу виконання операції множення від розрядності n_0 чисел, які використовуються згідно табл. 7.10 (номер графіка відповідає номеру випадку в табл. 7.10).

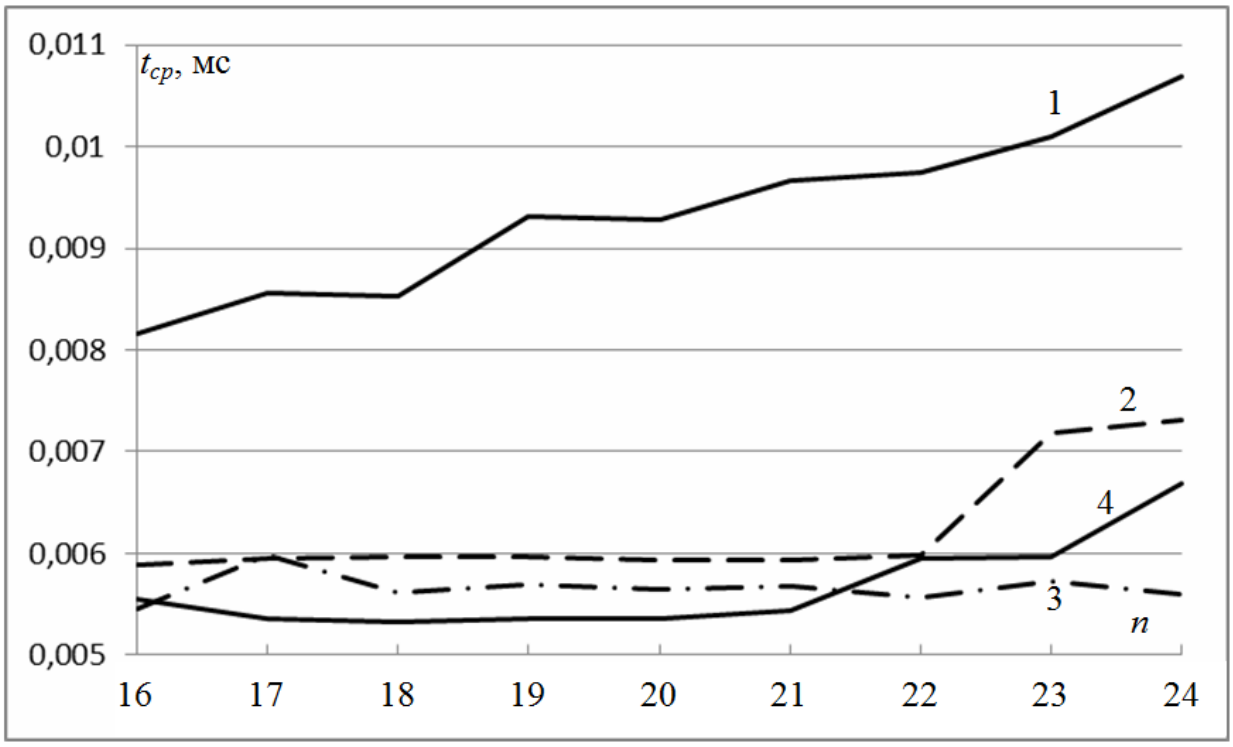


Рис. 7.29 - Графіки залежності середнього часу виконання операції множення в СЗК від розрядності множників

З рис. 7.29 випливає, що найбільший час витрачається для звичайної СЗК в першому випадку, коли модулі сильно відрізняються один від одного. Причому графік росте практично лінійно із збільшенням розрядності. Графіки 2 і 4 при малих розрядних майже лінійні, часові скачки спостерігаються при n_0=22 та n_0=23 відповідно. І третій графік практично лінійний на всьому розглянутому діапазоні. Аналіз рис. 7.29 свідчить про те, що використання модулів, які або утворюють МДФ СЗК, або мало відрізняються один від одного, дозволяє збільшити швидкодію обчислювальних систем.

На рис. 7.30 представлені графіки залежності середнього часу виконання операції множення від розрядності n_0 для третього (крива 1) і четвертого (крива 2) випадку табл. 7.10, модулі в яких утворюють МДФ СЗК, при тих самих вхідних параметрах з використанням формули (7.9).

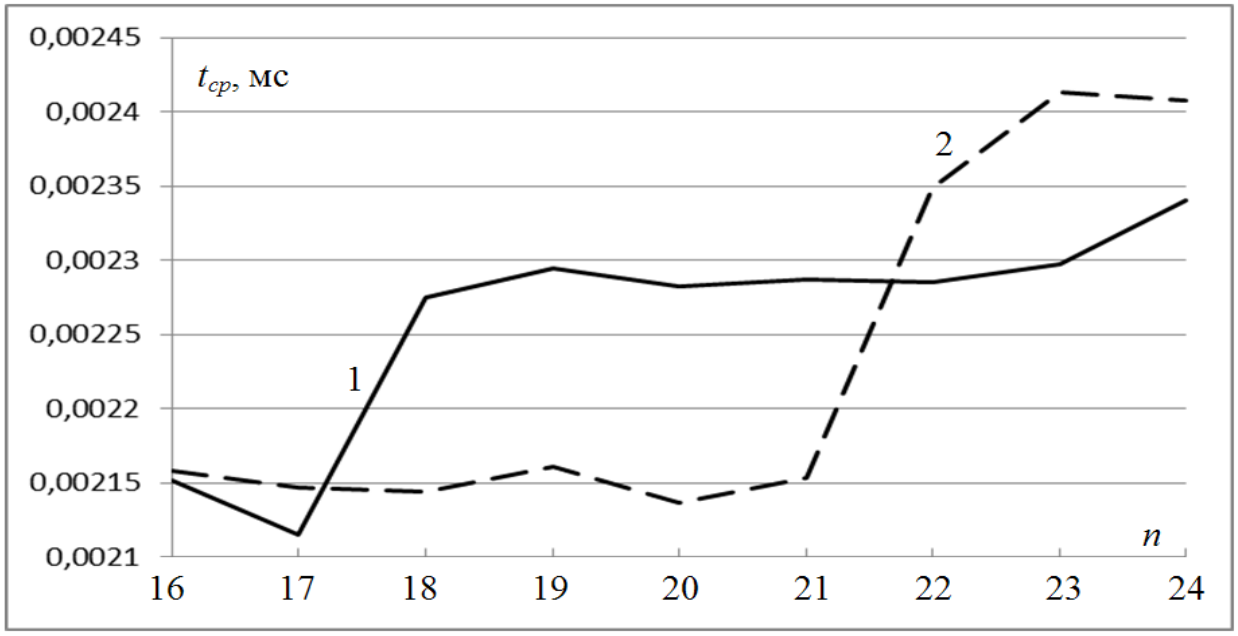


Рис. 7.30 - Графіки залежності середнього часу виконання операції множення від розрядності при використанні формули (7.9)

Аналіз рис. 7.29, 7.30 показує, що середній час обчислень в МДФ СЗК зменшується приблизно в 2,5-3 рази в порівнянні із звичайною цілочисельною формою СЗК.

7.8 Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах

Створення методології опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі запропонованих методів дозволить побудувати єдину стратегію використання СЗК, а також матрично- та векторно-модульних методів модулярного множення та експоненціювання для зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах, ефективно будувати швидкодіючі криптографічні системи захисту інформації [345]. Викладена в [346] методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах (її структурно-

аналітичне відображення представлено на рис. 7.31) містить вісім етапів: 1) формування множини блоків відкритого тексту; 2) формування вимог до параметрів криптосистем та захищеності інформації; 3) вибір асиметричної криптосистеми; 4) *ф*ормування множини базових операцій; 5) вибір методу виконання операцій; 6) вибір форми СЗК; 7) вибір методів побудови ДФ та МДФ СЗК; 8) реалізація основних асиметричних криптосистем. Опишемо більш детально кожен етап [346].

Етап 1. Формування множини блоків відкритого тексту. На першому етапі користувачу необхідно сформувати множину блоків відкритого тексту $BT = \left\{ \bigcup_{i=1}^{z_1} BT_i \right\} = \left\{ BT_1, BT_2, ..., BT_{z_1} \right\}$, (z_1 – кількість блоків відкритого тексту). Їх числові значення в основних асиметричних криптосистемах RSA, Рабіна, Ель-Гамаля мають бути меншими від відповідного параметру відкритого ключа, діленням на який числового значення відкритого тексту визначається величина z_1. Крім того, визначаються можливі загрози, які виникають при передачі даних у вигляді блоків зашифрованого тексту каналами зв'язку. В результаті формується визначена користувачем множина загроз $МЗ = \left\{ \bigcup_{i=1}^{z_2} МЗ_i \right\} = \left\{ МЗ_1, МЗ_2, ..., МЗ_{z_2} \right\}$, де z_2 – їх кількість.

Етап 2. Формування вимог до параметрів криптосистем та захищеності інформації. На основі сформованих множин загроз та блоків відкритого тексту відбувається формування вимог до кількісних показників захищеності інформації $ЗІ = \left\{ \bigcup_{i=1}^{z_2} \bigcup_{j=1}^{z_3} ЗІ_{ij} \right\}$, які записуються у вигляді матриці, та параметрів криптосистем $ВП = \left\{ \bigcup_{i=1}^{z_4} ВП_i \right\} = \left\{ ВП_1, ВП_2, ..., ВП_{z_4} \right\}$.

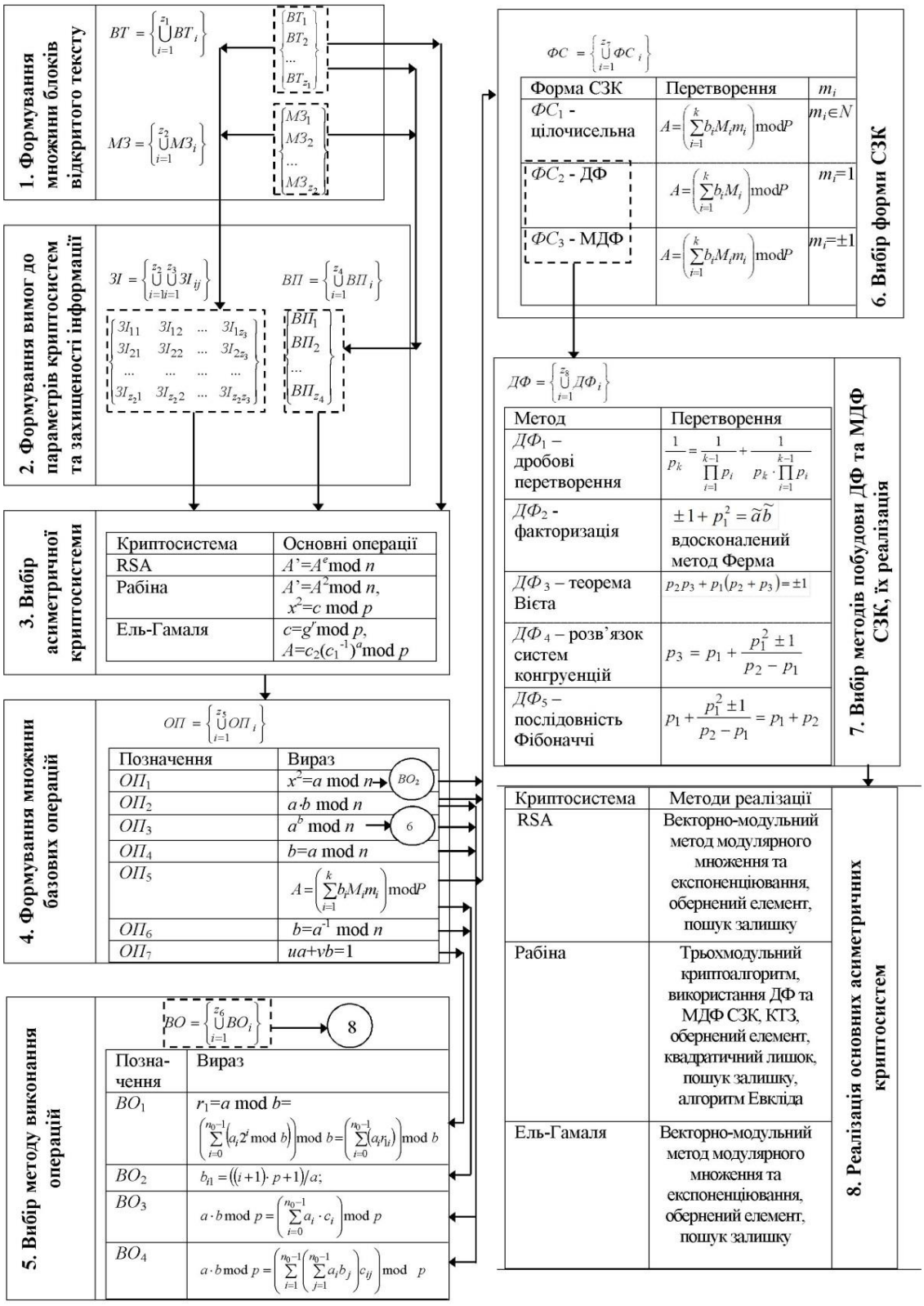


Рис. 7.31 – Структурно-аналітичне відображення методології опрацювання багаторозрядних чисел в асиметричних криптосистемах

Для кожного блоку відкритого тексту $ВТ_i, i \in [1, z_1]$, та кожної можливої загрози $МЗ_i, i \in [1, z_2]$, ставиться у відповідність множина показників захищеності $ЗІ_i = \left\{ \bigcup_{j=1}^{z_3} ЗІ_{ij} \right\} = \{ЗІ_{i1}, ЗІ_{i2}, ..., ЗІ_{iz_3}\}$, де z_3 – кількість можливих показників для i-ої загрози, та множина вимог до параметрів криптосистеми $ВП_i, i \in [1, z_4]$, де z_4 – кількість вимог.

Етап 3. Вибір асиметричної криптосистеми. Третій етап призначений для вибору асиметричної криптосистеми, найбільш поширеними з яких є RSA, Рабіна та Ель-Гамаля, згідно визначених у попередньому етапі вимог до їх параметрів.

Для генерації ключів криптосистем, шифрування та розшифровування використовуються такі операції: пошук залишку, квадратного кореня за модулем, найбільшого спільного дільника, оберненого елемента, використання розширеного алгоритму Евкліда, відновлення десяткового числа за його залишками (китайська теорема про залишки), модулярне множення та модулярне експоненціювання.

Етап 4. Формування множини базових операцій. На четвертому етапі формується відповідна множина базових операцій $ОП = \left\{ \bigcup_{i=1}^{z_5} ОП_i \right\} = \{ОП_1, ОП_2, ..., ОП_{z_5}\}$, $i \in [1, z_5]$, де z_5 – кількість операцій.

Етап 5. Вибір методу виконання операцій. На п’ятому етапі відбувається формування множини методів виконання операцій, визначених на попередньому етапі: $BO = \left\{ \bigcup_{i=1}^{z_6} BO_i \right\} = \{BO_1, BO_2, ..., BO_{z_6}\}$, $i \in [1, z_6]$, де z_6 – кількість методів. Зокрема, пошук залишку, модулярне множення та експоненціювання може виконуватися матрично- або векторно-модульним методом, усі інші – методом додавання модуля або добутку модулів.

Етап 6. Вибір форми СЗК. Більшість з описаних на четвертому етапі операцій можна виконувати на основі СЗК, множина форм якої формується

на шостому етапі: $ФС = \left\{ \bigcup_{i=1}^{z_7} ФС_i \right\} = \left\{ ФС_1, ФС_2, ..., ФС_{z_7} \right\}$, $i \in [1, z_7]$, де z_7 – кількість форм СЗК. Поряд із звичайною цілочисельною формою, найбільш перспективними для застосування в асиметричних криптосистемах є ДФ та МДФ СЗК.

Етап 7. Вибір методів побудови ДФ та МДФ СЗК. На сьомому етапі формується множина методів побудови ДФ та МДФ СЗК: $ДФ = \left\{ \bigcup_{i=1}^{z_8} ДФ_i \right\} = \left\{ ДФ_1, ДФ_2, ..., ДФ_{z_8} \right\}$, $i \in [1, z_8]$, де z_8 – кількість методів. У дисертації розроблені методи на основі дробових перетворень, факторизації, теореми Вієта, розв'язку систем конгруенцій та послідовності Фібоначчі.

Етап 8. Реалізація основних асиметричних криптосистем. Восьмий етап передбачає реалізацію вибраної асиметричної криптосистеми (RSA, Рабіна, зокрема трьохмодульної, або Ель-Гамаля) з базовими модулярними операціями (етап 4) на основі вибору методу їх виконання (етап 5) або з використанням відповідних форм СЗК (етап 6), методи побудови яких визначені на етапі 7.

Розроблена методологія за рахунок використання матрично та векторно модульних методів пошуку залишку, модулярного множення та експоненціювання, знаходження оберненого елемента на основі додавання модуля, а також використання ДФ та МДФ СЗК, дозволяє забезпечити зменшення обчислювальної складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах

Висновки до сьомого розділу

1. Здійснено програмну реалізацію методів модулярного експоненціювання для застосування в криптосистемі RSA, експериментальні

дослідження якої показали переваги використання МДФ СЗК при розрядностях, більших 1024 біти.

2. Запропоновано метод сумісного виконання алгоритму Евкліда та перемноження двох багаторозрядних чисел для використання у криптосистемі Рабіна та проведено експериментальне дослідження часових характеристик класичним та розробленим методами над числами різної розрядності. Показано, що в переважній більшості розглянутих випадків запропонований метод характеризується більшою швидкодією, середній час виконання операцій зменшується приблизно в 1,3 рази.

3. На основі експериментальних досліджень часових характеристик програмної реалізації методів факторизації встановлено, що запропонований метод факторизації особливо ефективним є для множників різної розрядності.

4. На основі дослідження часових характеристик програмної та програмно-апаратної реалізації методів пошуку оберненого елемента встановлено, що час обчислень зменшується приблизно в 1,5 разів.

5. На основі експериментального дослідження часових характеристик VHDL-моделей трьохмодульного криптоалгоритму Рабіна встановлено, що використання МДФ СЗК дозволяє зменшити час шифрування/дешифрування приблизно в 1,5 разів.

6. Розроблено методологію опрацювання багаторозрядних чисел на основі запропонованих методів, яка дозволяє забезпечити зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах. Застосування цієї методології дає можливість використовувати розроблені методи в єдиній стратегії опрацювання багаторозрядних чисел у галузі асиметричних криптосистем і ефективно будувати швидкодіючі криптографічні системи захисту інформації

Основні результати сьомого розділу відображені у роботах [302, 303, 315-317, 328-334, 336, 338-346].

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-прикладну проблему, яка полягає у підвищенні ефективності опрацювання багаторозрядних чисел на основі використання матрично- та векторно-модульного підходу до виконання операцій модулярного множення та експоненціювання, ДФ та МДФ СЗК для зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах. При цьому отримано такі основні теоретичні й практичні результати і наукові висновки:

1. Проведено аналіз сучасних методів, алгоритмів та засобів опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики. Встановлено, що відомі методи модулярного множення та експоненціювання характеризуються значною часовою та/або апаратною складністю і не володіють властивістю розпаралелення процесу обчислень. Крім того, при опрацюванні багаторозрядних чисел все більше стали проявлятися недоліки двійкової системи числення, яка використовується в сучасних обчислювальних системах. Результати проведеного аналізу дали можливість визначити завдання дисертаційного дослідження щодо розробки методів та засобів опрацювання багаторозрядних чисел в асиметричних криптосистемах.

2. Удосконалено методи модулярного множення, модулярного експоненціювання та пошуку найбільшого спільного дільника з використанням векторно-модульних перетворень у розмежованій системі числення залишкових класів, який дає можливість замінити операції піднесення до степеня та множення на, відповідно, множення та додавання малорозрядних залишків. Це дозволило, у порівнянні з відомими методами, зменшити обчислювальну складність модулярного множення та експоненціювання в 2-4 рази. Розроблено алгоритмічне забезпечення пошуку

найбільшого спільного дільника, криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу модулярного множення та експоненціювання.

3. Розроблено методи пошуку мультиплікативного оберненого елемента за модулем та реалізації китайської теореми про залишки на основі додавання модуля та додавання залишку, які не містять громіздких операцій ділення з остачею та піднесення до степеня, дозволяють виконувати розпаралелення процесу обчислень та зменшити обчислювальну складність даної операції при застосуванні в асиметричних криптосистемах. Встановлено, що при програмно-апаратній реалізації час обчислень зменшується майже в 1,5 рази в порівнянні з класичними методами.

4. Розроблено метод пошуку мультистепеневої функції за модулем, який забезпечує уникнення операції піднесення багаторозрядних чисел до степеня за модулем шляхом переходу до розв'язування лінійної конгруенції, виконання арифметичних дій над операндами, меншими від заданого модуля, та, відповідно, зменшення часової складності при опрацюванні мультистепеневої функції в асиметричних криптосистемах.

5. Розроблено загальний метод побудови спеціальних модулів системи залишкових класів у вигляді $p_1=2^u-1$, $p_2=2^u+1$, …, $p_i = 2^{u\cdot 2^{i-2}}+1$, …, $i=3, 4, \ldots$, який забезпечує аналітичний пошук коефіцієнтів базисних чисел при відновленні десяткового числа із системи залишкових класів, уникнувши операції знаходження мультиплікативного оберненого елемента за модулем.

6. Запропоновано методи побудови модулів досконалої форми системи залишкових класів на основі дробових перетворень та факторизації, що дозволяє уникнути виконання операцій знаходження мультиплікативного оберненого елемента за модулем та множення на нього при переведенні чисел із системи залишкових класів у десяткову систему числення.

7. Розроблено методи побудови трьох- та багатомодульної модифікованої досконалої форми системи залишкових класів на основі факторизації, теореми Вієта, розв'язку систем конгруенцій. Дана форма

забезпечує уникнення виконання операції пошуку мультиплікативного оберненого елемента за модулем та множення на нього при переведенні чисел із системи залишкових класів у десяткову систему числення. За допомогою програмної реалізації встановлено, що використання трьохмодульної модифікованої досконалої форми при перемноженні двох чисел підвищує швидкодію обчислень в 2,5-3 рази. Обґрунтовано доцільність її використання в асиметричних криптосистемах і показано, що при програмній реалізації криптосистеми RSA збільшення швидкодії наступає при використанні чисел розрядністю не менше 1024 біти в порівнянні з відомими методами.

8. Удосконалено метод Ферма для факторизації багаторозрядних чисел, який в порівнянні з класичним методом Ферма забезпечує заміну громіздкої операції пошуку квадратного кореня операцією додавання, зменшення розрядності операндів та спрощення методу факторизації. На основі програмної реалізації встановлено, що запропонований метод ефективніший від класичного методу Ферма для множників різної розрядності.

9. Розроблено метод сумісного виконання алгоритму Евкліда та перемноження двох багаторозрядних чисел для використання у криптосистемі Рабіна та проведено експериментальне дослідження часових характеристик класичним та розробленим методами над числами різної розрядності. Показано, що в переважній більшості розглянутих випадків запропонований метод характеризується більшою швидкодією, середній час виконання операцій зменшується приблизно в 1,3 рази.

10. Розроблено трьохмодульну криптосистему Рабіна на основі звичайної цілочисельної та модифікованої досконалої форм системи залишкових класів, яка дозволяє розширити блок шифрування відкритого тексту. Показано, що застосування модифікованої досконалої форм системи залишкових класів забезпечує зменшення значень операндів та кількість арифметичних операцій, необхідних при розшифруванні. Отримані за допомогою VHDL-моделей часові характеристики показують, що

використання модифікованої досконалої форм зменшує час шифрування та розшифрування приблизно в 1,5 рази.

11. Розроблено методологію опрацювання багаторозрядних чисел на основі запропонованих методів, яка дозволяє забезпечити зменшення часової складності, підвищення швидкодії алгоритмів, спеціалізованого програмного і апаратного забезпечення в асиметричних криптосистемах. Застосування цієї методології дає можливість використовувати розроблені методи в єдиній стратегії опрацювання багаторозрядних чисел у галузі асиметричних криптосистем і ефективно будувати швидкодіючі криптографічні системи захисту інформації.

12. Комп'ютерне моделювання за допомогою розробленого програмного та програмно-апаратного забезпечення на основі застосування спеціалізованої бібліотеки А.Ленстра, об'єктно орієнтованої мови С++, програмного інтерпретатора Python, засобів VHDL для проектування на ПЛІС підтверджує основні результати, які отримані теоретично.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hoffstein J., Pipher J., Silverman J. An Introduction to Mathematical Cryptography. Springer Science+Business Media, LLC, 2008. 524 p.
2. Jeffrey H., Jill P., Joseph H. An Introduction to Mathematical Cryptography. Berlin: Springer, 2008. 540 p.
3. Задірака В.К., Олексюк О.С. Комп'ютерна криптологія. Тернопіль, Київ, 2002. 504 с.
4. Фергюссон Н., Шнайер Б. Практическая криптография. М.: Вильямс, 2005. 424 с.
5. Вербіцький О.В. Вступ до криптології. Львів: ВНТЛ, 1998. 247 с.
6. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2006. 336 с.
7. Shoup V. A Computational Introduction to Number Theory and Algebra. Cambridge University Press, 2005. 517 p.
8. Stillwell J. Elements of Number Theory. Springer, 2010. 256 p.
9. Бородін О.І. Теорія чисел. К.: Вища школа, 1970. 275 с.
10. Бухштаб А.А. Теория чисел. М.: Просвещение, 1966. 384 с.
11. Hardy G.H., Wright E.M., Wiles A. An Introduction to the Theory of Numbers. Oxford University Press, 2008. 656 p.
12. Виноградов И.М. Основы теории чисел. Москва-Ижевск: НИЦ «Регулярная и хаотическая динамика», 2003. 176 с.
13. 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)*. 2016. Vol. 8, №11. P. 9-18.
14. Parthasarathy S. Multiplicative inverse in mod(m). *Algologic Technical Report*. 2012. №1. P. 1-3.

15. Дичка І.А., Онай М.В. Способи знаходження мультиплікативного оберненого елемента в скінченних полях. *Наук. техн. журн. «Наукові вісті НТУУ «КПІ»*. 2015. Вип. 2. С. 160-165.
16. Дичка І.А., Онай М.В., Бартков'як А.Ю. Застосування k-арного методу Евкліда для пошуку мультиплікативного оберненого елемента у кільці лишків за модулем m. *Наук. техн. журн. «Наукові вісті НТУУ «КПІ»*. 2016. Вип. 5. С. 248-252.
17. Lorencz R. New Algorithm for Classical Modular Inverse. *Cryptographic Hardware and Embedded Systems:* International Workshop. 2002. P. 57-70.
18. Sorenson J. Two fast GCD algorithms. *Journal of Algorithms*. 1994. P. 110-144.
19. Vallee B. Dynamical analysis of a class of Euclidean algorithms. *Theoretical Computer Science*. 2003. V.297. P. 447–486.
20. Vallee B. Dynamics of the binary Euclidean algorithm: functional analysis and operators. *Algorithmica*. 1998. Vol.22, №4. P. 660–685.
21. Гаусс К. Ф. Труды по теории чисел / Общая редакция академика И. М. Виноградова. М.: Изд-во АН СССР, 1959. 297 с.
22. Valach M., Svoboda A. Origin of the code and number system of remainder classes. *Stroje Na Zpracovani Informaci*. 1955. Vol. 3. P. 121-134.
23. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М. : Радио и связь, 2001. 376 с.
24. Столлингс В. Криптография и защита сетей. Принципы и практика. М.: Вильямс. 2001. 669 с.
25. Diffie W., Hellman M. New directions in cryptography. *IEEE Transactions on Information Theory*. 1976. Vol. 22, №6. P. 644–654.
26. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*. 1978. Vol. 21, Iss. 2. P. 120–126.
27. Ян С. Криптоанализ RSA. Ижевск: РХД. 2011. 312 с.

28. Srivastava A., Mathur A. The Rabin cryptosystem and analysis in measure of chinese reminder theorem. *International Journal of Scientific and Research Publications*. 2013. Vol. 3 (6). P. 1-4.
29. Hayder R.H. H-Rabim Cryptosystem. *Journal of Mathematics and Statistics*. 2014. Vol. 10 (3). P. 304-308.
30. Коблиц Н. Курс теории чисел и криптографии. М.: ТВП. 2001. 254с.
31. ElGamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*. 1985. Vol. 31 (4). P. 469–472.
32. Adki V., Hatkar S. A Survey on Cryptography Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2016. Vol. 6 (6). P. 469-475.
33. Silverman J.H. The arithmetic of elliptic curves. New York: Springer. 2007. Vol. 106. P. 17-40.
34. Washington L. Elliptic Curves Number Theory and Cryptography. Series Discrete Mathematics and Its Applications, Chapman & Hall/CRC, 2008. 524 p.
35. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир. 1979. С. 37 - 40.
36. Montgomery P. L. Modular multiplication without trial division. *Mathematics of Computation*. 1985. Vol. 44. P. 519-521.
37. 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..
38. 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..

39. 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.
40. Касянчук М.М., Якименко І.З., Николайчук Я.М., Івасьєв С.В. Векторно-модульний метод множення багаторозрядних чисел в базисі Радемахера-Крестенсона. *Захист інформації і безпека інформаційних систем – 2014*: Матеріали Міжнародної конференції. Львів. 2014. С. 53-54.
41. 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.
42. Kasianchuk M., Yakymenko I., Ivasiev S., Nykolaychuk Ya. Efficient methods for modular multiplication through the use of Rademacher-Krestenson TNB. *Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET–2014)*: Proceedings of the XI–th International Conference. L'viv–Slavske. 2014. P.93-94.
43. 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.
44. 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.
45. Минаев В.А., Хренов В.П. Безопасность в сфере конфиденциальной информации и закон формирования простых чисел. *Спецтехника и связь*. 2008. № 3. С. 45–48.

46. Song Y. Cryptanalytic attacks on RSA. Springer Science and Business Media, Inc., 2008. 255 p.
47. Авдошин С.М., Савельева А.А. Криптоанализ: современное состояние и перспективы развития. М.: Машиностроение, 2007. 124 с.
48. Горбенко І.Д., Долгов В.І., Потій А.В., Федорченко В.Н. Аналіз каналів вразливості системи RSA. *Безпека інформації*. 1995. №2. С. 22-26.
49. Лазарева С.В., Овчинников А.А. Математические основы криптологии: тесты простоты и факторизация. СПб.: СПбГУАП, 2006. 65 с.
50. Venturi D. Lecture Notes on Algorithmic Number Theory. Springer-Verlag, New-York, Berlin, 2009. 217 p.
51. Ишмухаметов Ш.Т., Бойко А.А., Зиятдинов Д.Б. Об одном подходе к проблеме факторизации натуральных чисел. *Известия вузов. Математика*. 2011. №4. 15-22 с.
52. Ишмухаметов Ш.Т. Методы факторизации натуральных чисел. Казань: Изд-во Казан. Ун-т, 2011. 190 с.
53. Dridi R., Alghassi H. Prime factorization using quantum annealing and computational algebraic geometry. *Scientific Reports*. 2017. Vol.7. P. 158-167.
54. Нестеренко Ю.В. Теория чисел. М.: Академия, 2008, 273 с.
55. Серовайский С.Я. Простые числа от Пифагора до криптографии. *Математика. Республиканский научно-методический журнал*. 2009. №1-3. С. 12-23.
56. Ингам А.Э. Распределение простых чисел. М.: УРСС, 2005. 160 с.
57. Каленикова Н.А., Минаев В.А., Хренов В.П. Ускорение факторизации в методе Ферма. *Вестник Российского нового университета*. 2010. №3. С. 12-16.
58. 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. P. 108–111.

59. Somsuk K., Tientanopajai K. An Improvement of Fermat's Factorization by Considering the Last m Digits of Modulus to Decrease Computation Time. *International Journal of Network Security*. 2017. Vol.19 (1). P.99-111.
60. Dixon J.D. Asymptotically fast factorization of integers. *Math. Comp.* 1981. Vol.36 (153). P. 255–260.
61. Pollard J.M. Theorems on factorization and primality testing. Proc.Cambridge Phil.Society. 1974. Vol.76. P. 521-578.
62. Crandall R., Pomerance C. Prime Numbers: A Computational Perspective. Springer, 2001. P. 227–244.
63. Gower J., Wagstaff S. Square Form Factorization. *Mathematics of Computation*. 2008. Vol. 77 (261). P. 551-588.
64. Dixon J.D. Asymptotically fast factorization of integers. *Mathematics of Computation*. 1981. Vol.36 (153). P. 255–260.
65. Lehmer D.H., Powers R.E. On Factoring Large Numbers. *Bulletin of the American Mathematical Society*. 1931. Vol. 37 (10). P. 770–776.
66. Morrison M., Brillhart J. A Method of Factoring and the Factorization of F_7. *Mathematics of Computation*. 1975. Vol.29 (129). P. 183–205.
67. Pomerance C. Analysis and comparison of some integer factoring algorithms. In Mathematisch Centrum Computational Methods in Number Theory, Part 1. 1982. P. 89-139.
68. Pomerance C. Tale of Two Sieves. Notices of AMS. 1996. P. 1473–1485.
69. Pomerance C. Smooth Numbers and the Quadratic Sieve. MSRI publications. 2008. Vol.44. P. 69–82.
70. Pomerance C., Smith J., Tuler R. A pipeline architecture for factoring large integers with the quadratic sieve algorithm. *SIAM J. Comput.* (*Special issue on cryptography*). 1988. Vol. 17. P. 387–403.
71. Pomerance C. Analysis and comparison of some integer factoring algorithms. In Mathematisch Centrum Computational Methods in Number Theory, Part 1. 1982. P. 89-139.

72. Ишмухаметов Ш.Т., Зиятдинов Д.Б., Рубцова Р.Г. О проблеме выбора полинома в методе решета числового поля. *Информационные технологии в системе социально–экономической безопасности России и ее регионов*: Труды III Всероссийской конференции. Казань, 2010. С. 177-183.
73. Brent R.P. Some integer factorization algorithms using elliptic curves. *Austral.Comput.Sci.Comm.* 1986. Vol. 8. P. 149–163.
74. Coblitz N., Menezes A., Vanstone S. The state of elliptic cryptography. *Design, Codes and Cryptography*. 2000. Vol. 19. P. 103–123
75. Lenstra H.W. Factoring integers with elliptic curves. *Ann.Math.* 1987. Vol.126. P. 649–674.
76. Montgomery P.L. An FFT-extension of the Elliptic Curve Method of Factirization. Doctoral Dissertation, Univ.Calif. USA. 1992. 118 p.
77. 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. P. 48–64.
78. Montgomery P.L. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*. 1987. Vol.48 (177). P.234–264.
79. Silverman R. Optimal Parameterization of SNFS. *J. Mathematical Cryptology*. 2007. Vol.1. P. 105–124.
80. Горбенко І.Д., Єсіна М.В. Алгоритм решета числового поля. *Радиотехника*. 2012. Вып. 171. С. 116-122.
81. Крэндалл Р., Померанс К. Простые числа. Криптографические и вычислительные аспекты. М.: УРСС, 2017. 664 с.
82. Wu H. Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis. *IEEE Transactions on Computers*. 2002. Vol.51 (7). P. 151-155.
83. Алиев Ф.А., Ларин В.Б. Алгоритм факторизации дискретных матричных полиномов второго порядка. *Proceedings of IAM*. 2017. Vol.6 (2). P. 159-172.

84. Ивашов Д.С. Об алгоритме факторизации полиномов многих переменных. *Вестник ТГУ*. 2012. Т.17 (2). С. 591-597.
85. Shoup V. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*. 1995. Vol. 20. P. 363-397.
86. Ivashov D.S. Factorization of polynomial of several variables. *Tambov University Reports. Series: Natural and Technical Science*. 2011. Vol. 16 (1). P. 133-137.
87. Gerver J.L. Factoring large numbers with a quadratic sieve. *Math. Comp*. 1983. Vol. 41. P. 287-294.
88. Винничук С.Д., Жилин А.В., Мисько В.Н. Факторизация числа $N=pq$ при простых p и q методом дискретного логарифмирования. *Электронное моделирование*. 2013. Т.35 (5). С. 3-10.
89. Макаренко А.В., Пыхтеев А.В., Ефимов С.С. Параллельная реализация и сравнительный анализ алгоритмов факторизации в системах с распределённой памятью. *Математические структуры и моделирование*. 2012. В.12. С. 94-109.
90. Кузьмин И.В., Кедрус В.А. Основы информации и кодирования. К.: Вища школа, 1986. 238 с.
91. Курко А.М., Решетник В.Я. Введення в теорія інформації. Тернопіль: Вид-во ТНТУ ім. Івана Пулюя, 2017. 108 с.
92. Жураковський Ю.П., Полторак В.П. Теорія інформації і кодування. К.: Вища школа, 2001. 255 с.
93. Рабинович З.Л., Раманаускас В.А. Типовые операции в вычислительных машинах. К.: Техніка, 1980. 264 с.
94. Хетагуров Я.А. Проектирование автоматизированных систем обработки информации и управления. М.: Высшая школа. 2006. 223 с.
95. Roy R., Datta D., Bhagat S., Saha S., Sinha A. Comparative Study and Analysis of Performances among RNS, DBNS, TBNS and MNS for DSP Applications. *Journal of Signal and Information Processing*. 2015. Vol. 6. P. 49-65

96. Ghosh A., Singha S., Sinha A. A New Architecture for FPGA Implementation of a MAC Unit for Digital signal Processors using Mixed number System. *Computer Architecture News*. 2012. Vol. 40. P. 33-38.
97. Kundu P., Bandyopadhyay O., Sinha A. An Efficient Architecture of RNS Based Wallace Tree Multi- plier for DSP Application. *Circuits and Systems*. 2008. Vol. 48. P. 221-224.
98. Singha S., Ghosh A., Sinha A. A New Architecture for FPGA based Implementation of Conversion of Binary to Double Base Number System (DBNS) Using Parallel Search Technique Singha. *Computer Architecture News*. 2011. Vol. 39. P. 12-18.
99. Deryabin M., Chervyakov N., Tchernykh A., Babenko M., Shabalina M. High Performance Parallel Computing in Residue Number System. *International Journal of Combinatorial Optimization Problems and Informatics*. 2018. Vol. 9 (1). P. 62-67.
100. Грицык В.В. Распараллеливание алгоритмов обработки информации в системах реального времени. К.: Наукова думка, 1981. 216 с.
101. Lin K.–L., Kemma A., Hosticka B. Modular low–power high–speed CMOS analog to digital converter for embedded systems. Boston: Kluwer Academic Publishers, 2008. 254 p.
102. Гофф М.К. Сетевые распределенные вычисления: достижения и проблемы. М.: Кудиц–образ, 2006. 320 с.
103. Гриценко В.И., Урсатьев А.А. Распределенные информационные системы. Состояния. Перспективы развития. *Управляющие системы и машины*. 2003. №4. С. 11–21.
104. Кондалев А.И. Высокопроизводительные преобразователи формы информации. К.: Наукова думка, 2007. 280 с.
105. Стемпковский А.Л., Корнилов А.И., Семенов М.Ю. Особенности реализации устройств цифровой обработки сигналов в интегральном

исполнении с применением модулярной арифметики. *Информационные технологии*. 2004. №2. С. 2–9.

106. Корнилов А.И., Семенов М.Ю., Калашников В.С. Методы аппаратной оптимизации сумматоров для двух операндов в системе остаточных классов. *Изв. ВУЗов. Электроника*. 2004. №1. С. 75–82.
107. Svoboda A. Rational numerical system of residual classes. *Stroje Na Zpracovani Informaci*. 1957. Vol. 1. P. 33-48.
108. Svoboda A. The numerical system of residue classes in mathematical machine. *Inform. processing*. 1960. Vol. 2. P. 81-100.
109. Акушский И.Я. Арифметические операции в системе остаточных клас сов. *Вопросы радиоэлектроники*. 1960. Серия VII, в.3. 254 с.
110. Акушский И.Я., Амербаев В.М., Пак И.Т. Основы машинной арифметики комплексных чисел. Алма-Ата: Наука, 1970. 248 с.
111. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. М.: Сов. радио, 1968. 460 с.
112. Торгашев В.А. Система остаточных классов и надежность ЦВМ. М.: Сов. Радио, 1973. 120 с.
113. Амербаев В.М. Теоретические основы машинной арифметики. Алма−Ата: Наука, 1976. 324 с.
114. Амербаев В.М., Стемпковский А.Л., Широ Г.Э. Быстродействующий согласованный фильтр, построенный по модулярному принципу. *Информационные технологии*. 2004. №9. С.5−12.
115. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Макоха А.И. Нейрокомпьютеры в остаточных классах. М: Радиотехника, 2003. 272 с.
116. Червяков Н.И., Бабенко М.Г., Ляхов П.А., Лавриненко И.Н. Приближенный метод сравнения модулярных чисел и его применение для деления чисел в системе остаточних клас сов. *Кибернетика и системный анализ*. 2014. Т. 50 (6). С. 176-186.

117. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Ряднов С.А. Модулярные параллельные вычислительные структуры нейропроцессорных систем. М: Физматлит, 2003. 288 с.
118. 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.
119. Барсов В. И., Сорока Л.С., Краснобаев В.А. Методология параллельной обработки информации в модулярной системе счисления. Харьков: УИПА, 2009. 268 с.
120. 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), 2015. №5 (136). P. 51-62.
121. 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*. 2016. Vol. 52, №1. P. 145–150.
122. Николайчук Я.М., Федорович Ю.С. Теоретичні основи базисних перетворень СЗК. *Автоматика 2000*: Матеріали наукової конференції. Львів. 2000. С. 120.
123. Николайчук Я.М., Возна Н.Я., Пітух І.Р. Проектування спеціалізованих комп'ютерних систем. Тернопіль: ТзОВ: Терн граф, 2010. 392 с.
124. Николайчук Я.М. Теорія джерел інформації. Тернопіль: ТзОВ: Терн граф, 2010. 536 с.
125. Николайчук Я.М. Коди поля Галуа: теорія та застосування. Тернопіль: ТзОВ: Терн граф, 2012. 576 с.
126. Omondi A., Premkumar B. Residue number systems: theory and implementation. London: Imperial College Press, 2007. 296 p.
127. Vinod A., B. Premkumar. A memoryless reverse converter for the 4-moduli superset $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$. *J. Circuits, Syst., Comput*. 2000. Vol. 10 (1-2). P. 85–99.

128. Premkumar B. A format framework for conversion from binary to residue numbers. *IEEE Trans. Sircuit Syst*. 2002. Vol. 49. P. 135-144.
129. Ananda Mohan P.V. RNS to binary conversion using diagonal function and Pirlo and Impedovo monotonic function. *Circuits Syst. Signal Process*. 2016. Vol. 35. P. 1063-1076.
130. Ananda Mohan P.V. Residue Number Systems: Theory and Applications. Birkhäuser, 2016. 351 p.
131. Финько О.А. Модулярная арифметика параллельных логических вычислений. М.: Институт Проблем Управления РАН, 2003. 214 с.
132. Яцків В.В. Теоретичні основи створення і структурна організація компонентів безпровідних сенсорних мереж підвищеної ефективності: дис. д-ра техн. наук: 05.13.05. Львів: НУ «ЛП», 2016. 327 с.
133. Завало С.Т., Костарчук В.Н., Хацет Б.І. Алгебра і теорія чисел. К.: Вища школа, 1977. 400 с.
134. Минеев М.П., Чубариков В.Н. Об одном применении китайской теоремы об остатках к шифру Виженера. *Доклады Академии наук*. 2010. Т.430, №1. С. 21-22.
135. 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*. 2014. Vol.3 (3). P. 2584-2588.
136. Шенец Н. Н. Модуляроное разделение секрета и системы электронного голосования. *Вестник БГУ*. 2011. № 1. С. 101-104.
137. Ростовцев В. С., Зорин Е.И., Грачёв Е.А. Модулярные высокоточные параллельные вычисления с использованием нейросетевых технологий. *Вестник СибГАУ: Математика, механика, информатика*. 2013. №4(50). С. 71-74.
138. Калмыков И.А., Калмыков М.И. Структурная организация параллельного спецпроцессора цифровой обработки сигналов,

использующего модулярные код. *Теория и техника радиосвязи*. 2014. №2. С. 60-66.

139. Pikh V., Kimak V., Krulikovskyi B. Synthesis of High-performance Components of Spectral Analyzers and Special Processors for Data Encryption in Rademacher-Krestenson's Theoretical-numerical Basis. *Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2015*): Proceedings of the 13th International Conference. 2015. P.182-184.
140. Vozna N., Nykolaichuk Ya., Zastavnyy O., Pikh V. System complexity criteria and synthesis of high-performance multifunctional parallel ADC in Rademacher's and Haar-Krestenson's theoretical and numerical bases. *Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2017*): Proceedings of the 14th International Conference. 2017. P. 218-221.
141. Chang C., Yeu - Pong L. A division algorithm for residue numbers. *Appl. Math. and Comput*. 2006. Vol. 172 (1). P. 368–378.
142. Червяков Н.И., Бабенко М.Г., Ляхов П.А., Лавриненко И.Н., Лягин А.М. Умножение и деления чисел в системе остаточних клас сов с использованием полей Галуа GP(p). *Научно-технические ведомости СПбГПУ*. 2014. Т.3 (198). С. 65-76.
143. Labafniya M., Eshghi M. Non-iterative RNS Division Algorithm. *International multiconference of engineers and computer scientists:* Proceedings. 2012. Vol. 1. P. 132-135.
144. 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*. 2013. №76. P. 117-126.
145. Labafniya M. Eshghi M. RNS division algorithm for 2^n-1 and 2^n dividers. *22nd Iranian Conference on Electrical Engineering (ICEE):* Proceedings. 2014. P. 111-114.

146. Lu M. Chiang J.-S. A novel division algorithm for the residue number system. *IEEE Transactions on Computers*. 1992. Vol.41(8). P. 1026-1032.
147. Bajard J.-C., Didier L.-S., Muller J.-M. A new Euclidean division algorithm for residue number systems/ *J. VLSI Signal Processing*. 1998. Vol. 19 (2). P. 167–178.
148. Hitz M.A., Kaltofen E. Integer division in residue number systems. *IEEE Trans. Comput*. 1995. Vol. 44 (8). P. 983–989.
149. Vivek N., Anusudha K. Design of RNS Based Addition Subtraction and Multiplication Units. *International Journal of Engineering Trends and Technology*. 2014. Vol. 10 (12). P. 593-596.
150. Beuchat J.-L. Some Modular Adder and Multipliers for Field Programmable Gate Arrays. *Parallel and Distributed Processing:* IEEE Proceedings of International Symposium. 2010. Vol.17. P.8-11.
151. Lalitha K.V., Sailaja V. High performance adder using residue number system. *International Journal of VLSI and Embedded Systems*. 2014. Vol. 05. P. 1323-1332.
152. Vergos H. On the design of efficient modular adders. *J. Circuits, Syst. and Comput*. 2005. Vol. 14 (5). P. 965–972.
153. 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. P. 6–9.
154. 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*. 2016. Vol. 5 (9). P. 1-5.
155. Isupov K., Knyazkov V. RNS-based Data Representation for Handling Multiple-Precision Integers on Parallel Architectures. *Engineering and Telecommunication:* Proceedings of the International Conference. 2016. P. 76-79.

156. 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*. 2017. Vol. 12 (16). P. 6159-6163.
157. Reshadinezhad M., Samani F.K. A Novel Low Complexity Combinational RNS Multiplier Using Parallel Prefix Adder. *International Journal of Computer Science*. 2013. Vol. 10 (3). P. 430-440.
158. 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*. 2002. Vol. 51. P. 1541-1553.
159. Harvey D. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*. 2014. Vol. 60. P. 113–119.
160. Николайчук Я.М. Теорія цифрових перетворень мультибазисного супершвидкодіючого процесора. *Искусственный интеллект*. 2008. №4. С. 387-394.
161. Tomczak T. Hierarchical residue number systems with small moduli and simple converters. *International Journal of Applied Mathematics and Computer Science*. 2011. Vol. 21 (1). P. 173–192.
162. Краснобаев В. А., Загуменная Е.В., Мороз С.А., Жадан В.О. Математическая модель процесса табличной реализации операций алгебраического умноження в классе вычетов. *Радіоелектронні і комп'ютерні системи*. 2012. Т.11 (2). С. 281-287.
163. Кошман С.А., Деренько Н.С. Метод реализации арифметических операций в модулярной арифметике на основе использования малоразрядных двоичных сумматоров. *Радіоелектронні і комп'ютерні системи*. 2007. Т.7 (26). С. 219-221.
164. Волинський О.І. Швидкодія міжбазисних перетворювачів Радемахера-Крестенсона. *Юриспруденція та проблеми інформаційного суспільства (ЮПІС - 2011)*: збірник матеріалів проблемно-наукової міжгалузевої конференції. 2011. С.71-75.

165. Jablonski J. Pipeline processing for serial realization of basical arithmetical operations. *Discrete-Event System Design (DESDes'01)*: Proceedings of The International Workshop. 2001. P. 85-90.
166. Николайчук Я.М., Волинський О.І., Кулина С.В. Теоретичні основи побудови та структура спецпроцесорів в базисі Крестенсона. *Вісник Хмельницького національного університету*. 2007. №3 (1). С. 85-90.
167. Hiasat A.A., Zohdy H.A.A. Semi-custom VLSI design and implementation of a new efficient RNS division algorithm. *Computer Journal*. 1999. Vol.42 (3). P.232-240.
168. Краснобаев В.А., Кошман С.А., Тыртышников А.И., Гаркавенко Н.С. Концепция создания отказоустойчивых компьютерных систем обработки информации в системе остаточных классов на основе применения ПЛИС. *Системи обробки інформації*. 2013. В. 7(114). С. 79-82
169. Краснобаев В.А., Кошман С.О., Маврина М.О. Метод повышения достоверности контроля данных, представленных в системе остаточных клас сов. *Кибернетика и системний анализ*. 2014. Том 50 (6). С.167 –175.
170. Су Цзюнь, Яцкив В.В., Саченко А.А., Ху Чежньбин. Спецпроцессор кодирования изображений в системе остаточных клас сов. *Современные информационные и электронные технологии (СИЭТ-2012):* Труды МНПК. Одесса, 2012. С.95.
171. Цаволык Т.Г., Яцкив В.В. Метод исправления ошибок на основе модулярных корректирующих кодов. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2015. № 5 (850). С. 36 – 38.
172. Яцків В.В. Виявлення та виправлення багатократних помилок на основі модулярних коректуючих кодів. *Інформаційні технології та комп'ютерна інженерія*. 2015. Том 33, №2. С.77-82. 88.

173. Яцків В.В., Цаволик Т.Г. Двовимірні коректуючі коди на основі модулярної арифметики. *Вісник Хмельницького національного університету. Технічні науки*. 2015. № 4 (227). С.144-148.
174. Яцків В.В. Метод підвищення надійності передачі даних в безпровідних сенсорних мережах на основі системи залишкових класів. *Радіоелектроніка та інформатика*. 2010. №2. С.32–35.
175. 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*. 2015. Vol 21(1). P. 76-81.
176. Roshanzadeh M., Saqaeeyan S. Error Detection & Correction in Wireless Sensor Networks By Using Residue Number Systems. *International Journal of Computer Network and Information Security*. 2012. №2. P. 29-35.
177. Aremu I.A., Gbolagade K.A. Information encoding and decoding using Residue Number System for $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ moduli sets. *International Journal of Advanced Research in Computer Engineering & Technology*. 2017. Vol. 6 (8). P. 1260-1267.
178. Xiao H., Garg H., Hu J., Xiao G. New Error Control Algorithms for Residue Number System Codes. *Electronics and Telecommunications Research Institute*. 2016. Vol. 38 (2). P.326-336.
179. Lo H., Lin T. Parallel Algorithms for Residue Scaling and Error Correction in Residue Arithmetic. *Wireless Eng. Technol*. 2013. Vol. 4 (4). P. 198–213.
180. 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*. 2016. Vol. 52 (1). P. 145–150.
181. Исупов К.С. Об одном алгоритме сравнения чисел в системе остаточных классов. *Вестник Астраханского государственного технического университета*. 2014. №3. С. 40-49.
182. Torabi Z., Jaberipur G. Low-Power/Cost RNS Comparison via Partitioning the Dynamic Range. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2016. Vol. 24 (5). P. 1849-1857.

183. Keir Y.A., Cheney P.W., Tanenbaum M. Division and overflow detection in residue number systems. *IRE Trans. Electron. Comput*. 1962. Vol. EC-11. P. 501- 507.
184. Rouhifar M., Hosseinzadeh M., Bahanfar S., Teshnehlab M. Fast Overflow Detection in Moduli Set $\{2^n-1, 2^n, 2^n+1\}$. *International Journal of Computer Science Issues*. 2011. Vol. 8 (3). P. 407-414.
185. Askarzadeh M., Hosseinzadeh M., Navi K. A New approach to overflow detection in moduli set $\{2^n-3, 2^n-1, 2^n+1, 2^n+3\}$. *Computer and Electrical Engineering:* Proceedings of the International Conference. 2009. P. 439-442.
186. Rouhifar M., Hosseinzadeh M., Teshnehlab M. A new approach to overflow detection in moduli set $\{2^n-1, 2^n, 2^n+1\}$. *International Journal of Computational Intelligence and Information Security*. 2011. Vol. 2 (3). P. 35-43.
187. Singh N. An overview of Residue Number System. *Devices, Circuits & Communication:* Proceedings of the National Seminar. 2008. P. 132-135.
188. Garner H.L., Harvey L.G. The Residue Number System. *IRE Transactions on Electronic Computers*. 1959. Vol. EC-8 (2). P. 140-147.
189. Sharoun A.O. Residue number system. *Electrical Engineering*. 2013. №76. P. 265-270.
190. Bavya V., Uthira Devi R. Optimizing the Precision of Digital Signal Processors Using Residue Number System. *Imperial Journal of Interdisciplinary Research*. 2016. Vol.2 (4). P. 1113-1122.
191. Kornerup P., Matula D.W. Finite Precision Number Systems and Arithmetic. Cambridge University Press, 2010. 699 p.
192. Исупов К.С., Князьков В.С. Арифметика многократной точности на основе систем остаточных классов. *Программные системы: теория и приложения*. 2016. №1(28). С. 61–97.
193. Кошман С.А. Концепция реализации немодульных операций в модулярной системе счисления. *Проблеми інформації:* Матеріали другої міжнародної науково-технічної конференції. 2014. С. 94-95.

194. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов. *Инфокоммуникационные технологии*. 2011. №4. С.4-12.
195. Лавриненко А.Н., Червяков Н.И. Исследование немодульных операций в системе остаточных классов. *Научные ведомости Белгородского государственного университета. Серия: Информатика*. 2012. №1 (120), Выпуск 21/1. С.110-122.
196. Fin'ko O.A. Large Systems of Boolean Functions: Realization by Modular Arithmetic Methods. *Automation and Remote Control*. 2004. Vol. 65 (6). P. 871–892.
197. Vassalos E., Bakalis D., Vergos H.T. SUT-RNS Residue-to-Binary Converters Design. *Digital System Design:* Proceedings of the 15th Euromicro Conference. 2012. P.65-72.
198. Vassalos E., Bakalis D., Vergos H.T. SUT-RNS forward and reverse converters. *IEEE Comput. Society Annual Symp* VLSI*:* Proceedings. 2010. P. 11–16.
199. Cardarilli G.C., Nannarelli A., Re A. Residue Number System for low-power DSP applications. *Signals, Systems and Computers*: Proceedings of the 41st Asilomar Conference. 2007. P. 1412–1416.
200. Madhukumar A., Chin F. Enhanced architecture for Residue Number System-based CDMA for high-rate data transmission. *IEEE Trans. Wireless Commun*. 2004. Vol. 3 (5). P. 1363–1368.
201. Wei S., Shimizu K. A novel residue arithmetic hardware algorithm using a Signed-Digit number representation. *IEICE Trans. Inform. Systems*. 2012. Vol. E83-D (12). P. 2056–2064.
202. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. *J. Signal Process. Syst*. 2009. Vol. 56 (1). P. 1–15.

203. Сиора А.А., Краснобаев В.А., Харченко В.С.Отказоустойчивые системы с версионно-информационной избыточностью в АСУ ТП: Монография. Х.: МОН, НАУ им. Н.Е. Жуковского (ХАИ), 2009. 320 с.
204. Jaberipur G., Parhami B., Ghodsi M. Weighted two-valued digit-set encodings: unifying efficient hardware representation schemes for redundant number systems. *IEEE Trans. Circuits Syst*. 2005. Vol. 52 (7). P. 1348–1357.
205. Краснобаев В.А., Янко А.С., Кошман С.А., Курчанов В.Н., Бендес Ю.П. Расчет и сравнительный анализ производительности компьютерной системы обработки целочисленных данных, представленных в системе остаточных классов. *Системи обробки інформації*. 2015. В 3 (128). С. 57-61.
206. Жихарев В.Я., Илюшко Я.В., Кравец Л.Г., Краснобаев В.А. Методы и средства обработки информации в непозиционной системе счисления в остаточных классах. Житомир: Изд-во "Волынь", 2005. 220 с.
207. Исупов К.С. Параллельные вычисления над многоразрядными числами в системе остаточных классов. Труды Международной суперкомпьютерной конференции (Новороссийск). 2011. С. 534-540.
208. Исупов К.С., Князьков В.С. Программный пакет высокоточных модулярно-позиционных вычислений с плавающей точкой. *Advanced Science*. 2013. №3. С. 150-171.
209. Исупов К.С., Князьков В.С. Система остаточных классов как инструмент для выполнения параллельных высокоточных численных расчетов. *Математическое моделирование развивающейся экономики, экологии и биотехнологий (ЭКОМОД-2010)*: Труды V Всероссийской научной конференции. 2010. С. 79-88.
210. Исупов К.С., Князьков В.С. Инструментальный комплекс для проектирования параллельных масштабируемых программ численных расчетов. *Научно-технический вестник СПбГУ*. 2010. № 6 (70). С. 68-72.

211. Abdullah M., Skavantzos A. A systematic approach for selecting practical moduli sets for residue number systems. *System Theory*: Proceedings of the 27th IEEE International Symposium. 1995. P. 445-449.
212. Wang W.M., Swamy N.S., Ahmad M.O. Moduli selection in RNS for efficient VLSI implementation. *Circuits System*: Proceedings of the IEEE International Symposium. 2003. V.4. P. 512-515.
213. Liu Y., Lai E. Moduli Set Selection and Cost Estimation for RNS-Based FIR Filter and Filter Bank Design. *Design Automation for Embedded Systems*. 2004. V.9. P.123-139.
214. Cao B., Chang C.-H., Srikanthan T. An efficient reverse converter for the 4-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ based on the new Chinese Remainder Theorem. *IEEE Trans. Circuits Syst*. 2003. Vol. 50(10). P.1296–1303.
215. Molahosseini A., Navi K., Dadkhah C., Kavehei O., Timarchi S. Efficient reverse converter designs for the new 4-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ based on New CRTs. *IEEE Trans. Circuits Syst*. 2010. Vol. 57 (4). P. 823–835.
216. 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. P. 2103–2106.
217. Kalampoukas L., Nikolos D., Efstathiou C., Vergos H.T., Kalamatianos J. High-speed parallel-prefix modulo 2^n-1 adders. *IEEE Transactions on Computers*. 2000. V. 49 (7). P. 673-679.
218. Efstathiou C., Vergos H.T., Nikolos D. Fast parallel-prefix modulo 2^n+l adder. *IEEE Transactions on Computers*. 2004. V.53 (9). P. 1211-1216.
219. Patel R A., Benaissa M., Powell N., Boussakta S. Novel power-delay-area-efficient approach to generic modular addition. *IEEE Transactions on Circuits and Systems*. 2007. V.54. P. 1279-1292.
220. Hiasat A.A. High-speed and reduced area modular adder structures for RNS. *IEEE Transactions on Computers*. 2002. V.51. P.84-89.

221. Zarandi A.A.E., Molahosseini A.S., Hosseinzadeh M. Modern Residue Number System Moduli Set: Efficiency vs. Complexity. *Нейрокомпьютеры: разработка, применение*. 2014. №9. С. 7-12.
222. Szabo N., Tanaka R. Residue Arithmetic and its Applications to Computer Technology. New York: McGraw-Hill, 1967. 319 p.
223. Premkumar B. An RNS to binary converter in 2n+1, 2n, 2n-1 moduli set. *IEEE Transactions on Circuits and Systems*. 1992. V.39. P. 480-482.
224. Pourbigharaz F., Yassine H.M. A signed-digit architecture for residue to binary transformation. *IEEE Transactions on Computers*. 1997. V.46. P. 1146-1150.
225. Hiasat A.A., Abdel-Aty-Zohdy H.S. Residue-to-binary arithmetic converter for the moduli set (2^k, 2^k-1, 2^{k-1}-1). *IEEE Transactions on Circuits and Systems*. 1998. V.45. P.204-208.
226. Mathew J., Radhakrishnan D., Srikanthan T. Fast residue-to-binary converter architectures. *Circuits and Systems:* Proceedings of IEEE International Midwest Symposium. 1999. P. 1090-1093.
227. Molahosseini A.S., Navi K., Rafsanjani M.K. A new residue to binary converter based on mixed-radix conversion. *Information and Communication Technologies: From Theory to Applications:* Proceedings of IEEE International Conference. 2008. P. 394-399.
228. Hariri A., Navi K., Rastegar R. A new high dynamic range moduli set with efficient reverse converter. *Elsevier Journal of Computers and Mathematics with Applications*. 2008. V.55 (4). P. 660-668.
229. Molahosseini A.S., Navi K., Hashemipour O., Jalali A. An efficient architecture for designing reverse converters based on a general three-moduli set. *Elsevier Journal of Systems Architecture*. 2008. V.54. P. 929-934.
230. Bhardwaj M., Srikanthan T., Clarke C.T. A reverse converter for the 4-moduli superset {2^n-1, 2^n, 2^n+1, 2^{n+1}+1}. *Computer Arithmetic:* Proceedings of IEEE Symposium. 1999. P. 316-321.

231. Vinod A.P., Premkumar B. A residue to binary converter for the 4-moduli superset $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$. *Journal of Circuits, Systems and Computers*. 2000. V.10. P. 85 -99.
232. Sheu M.H., Lin S.H., Chen C., Yang S.W. An Efficient VLSI Design for a Residue to Binary Converter for General Balance Moduli $\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$. *IEEE Transactions on Circuits and Systems*. 2004. V.51 (3). P. 152-155.
233. Zhang W., Siy P. An efficient design of residue to binary converter for four moduli set $\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$ based on new CRT II. *Elsevier Journal of Information Sciences*. 2008. V. 178 (1). P. 264-279.
234. Molahosseini A.S., Navi K. A Reverse Converter for the Enhanced Moduli Set $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$ Using CRT and MRC. *IEEE Computer Society Annual Symposium on VLSI:* Proceedings. 2010. P. 456 - 457.
235. 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*. 2014. V.10 (1). P. 143-148.
236. Hiasat A.A. VLSI implementation of new arithmetic residue to binary decoders. *IEEE Transactions on Very Large Scale Integration Systems*. 2005. V.13 (1). P. 153-158.
237. Cao B., Chang C.-H., Srikanthan T. A residue-to-binary converter for a new five-moduli set. *IEEE Transactions on Circuits and Systems I*. 2007. V. 54(5). P. 1041-1049.
238. Molahosseini A.S., Dadkhah C., Navi K. A new five-moduli set for efficient hardware implementation of the reverse converter. *IEICE Electronics Express*. 2009. V.6 (4). P. 1006-1012.
239. Pettenghi K., Chaves R., Sousa L. RNS Reverse Converters for Moduli Sets With Dynamic Ranges up to (8n+l)-bit. *IEEE Transactions on Circuits and Systems*. 2013. V. 60 (6). P. 1487-1500.

240. Pettenghi H., Chaves R., Sousa L. Method to Design General RNS Reverse Converters for Extended Moduli Sets. *IEEE Transactions on Circuits and Systems*. 2013. V. 60 (12). P. 877-881.
241. Parhami B. On Equivalences and Fair Comparisons among Residue Number Systems with Special Moduli. *Signals, Systems, and Computers:* Proceedings of 44th Asilomar Conference. 2010. P. 1690-1694.
242. Zarandi A. A. E., Molahosseini A.S., Hosseinzadeh M., Sorouri S., Antao S., Sousa L. Reverse Converter Design via Parallel-Prefix Adders: Novel Components, Methodology and Implementations. *IEEE Transactions on Very Large Scale Integration Systems:* Proceedings. 2014. P. 834-838.
243. Chokshi R., Berezowski K.S., Shrivastava A., Piestrak S.J. Exploiting residue number system for power-efficient digital signal processing in embedded processors. *Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09)*: Proceedings of the 2009 International Conference, Grenoble, France. 2009. P. 19–28.
244. Piestrak S., Berezowski K. Design of residue multipliers-accumulators using periodicity. *IET Irish Signals and Systems Conference (ISSC 2008)*: Proceedings of the International Conference, Galway, Republic of Ireland. 2008. P. 380–385.
245. Piestrak S.J., Berezowski K. Architecture of efficient RNS-based digital signal processor with very low-level pipelining. *IET Irish Signals and Systems Conference (ISSC 2008)*: Proceedings of the International Conference, Galway, Republic of Ireland. 2008. P. 127–132.
246. Wnuk M. Remarks on hardware implementation of image processing algorithms. *International Journal of Applied Mathematics and Computer Science*. 2008. Vol.18(1). P. 105–110.
247. Wang W., Swamy M.N.S., Ahmad M.O. RNS application for digital image processing. *System-on-Chip for Real-Time Applications (IWSOC'04)*: Proceedings of the 4th IEEE International Workshop, Banff, Alberta, Canada. 2004. P. 77–80.

248. Conway R., Nelson J. Improved RNS FIR filter architectures. *IEEE Transactions on Circuits and Systems*. 2004. Vol. 51(1). P. 26–28.
249. Mohan A.P.V. Residue Number Systems: Algorithms and Architectures. Kluwer Academic Publishers, Norwell, MA, 2002. 254 p.
250. Soderstrand M.A., Jenkins W.K., Jullien G.A., Taylor F.J. Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, Piscataway, NJ, 1986. 382 p.
251. Yassine H.M. Hierarchical residue numbering system suitable for VLSI arithmetic architecture. *Circuits and Systems (ISCAS '92)*: Proceedings of the IEEE International Symposium, San Diego, CA, USA. 1992. P. 811–814.
252. Skavantzos A., Abdallah M. Implementation issues of the two-level residue number system with pairs of conjugate moduli. *IEEE Transactions on Signal Processing*. 1999. Vol. 47(3). P. 826–838.
253. Bajard J., Eynard J., Merkiche N. Multi-fault attack detection for RNS cryptographic architecture. *Computer Arithmetic (ARITH 2016):* Proceedings of the 23nd IEEE Symposium, Silicon Valley, CA, USA. 2016. P. 16–23.
254. Perin G., Imbert L. Electromagnetic analysis on RSA algorithm based on RNS. *Euromicro Conference on Digital System Design (DSD)*: Proceedings. 2013. Vol.1. P. 345–352.
255. Schinianakis D., Stouraitis T. Hardware-fault attack handling in RNS-based Montgomery multipliers. *Circuits and Systems (ISCAS2013)*: Proceedings of the IEEE International Symposium. 2013. P. 3042–3045.
256. 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.
257. Fournaris A.P., Papachristodoulou L., Batina L., Sklavos N. Secure and Efficient RNS Approach for Elliptic Curve Cryptography. *Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016*): Proceedings of the 6th Conference, Barcelona. 2016. P. 121-126.

258. Fournaris A.P., Klaoudatos N., Sklavos N., Koulamas C. Fault and power analysis attack resistant RNS based edwards curve point multiplication. *Cryptography and Security in Computing Systems:* Proceedings of the 2nd Workshop, Amsterdam, Netherlands. 2015. P. 43–46.
259. 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 Conference. 2016. P. 1–4.
260. 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. P. 48–64.
261. Fadulilahi I.R., Bankas E.K., Ansuura J.B.A.K. Efficient Algorithm for RNS Implementation of RSA. *International Journal of Computer Applications*. 2015. Vol. 127 (5). P. 14-19.
262. Laurent I., Jean-Claude B. A Full RNS Implementation of RSA. *Transactions on computers*. 2004. Vol.53 (5). P. 1-6.
263. Nobuhiro T., Teruki I. Design of High-Speed RSA Encryption Processor Based on the Residue Table for Redundant Binary Numbers. *Systems and Computers in Japan*. 2002. Vol. 33(5). P. 423-432.
264. Якименко І.З., Тимошенко Л.М., Касянчук М.М. Вибір параметрів еліптичних кривих у задачах шифрування інформаційних потоків. *Сучасна спеціальна техніка*. 2018. № 2. С. 63–71.
265. 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.

266. Касянчук М.М., Михалюк І.В., Самарик П.С. Програмна реалізація захищеного каналу обміну повідомленнями з використанням апарату еліптичних кривих. *Сучасні комп'ютерні інформаційні технології* (*ACIT-2015*): Матеріали V Всеукраїнської школи-семінару молодих вчених і студентів. Тернопіль, 2015. С. 175-176.
267. 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*. 2011. Vol. 17 (2). P. 31–36.
268. Краснобаев В.А., Кошман С.А. Метод быстрой реализации криптографических преобразований на основе поразрядной табличной реализации. *Системи обробки інформації*. 2009. №7 (79). С.63-68.
269. Краснобаев В.А., Кошман С.А., Сомов С.В., Крючко Е.А. Метод быстрой обработки криптографической информации в модулярной системе счисления. *Системи обробки інформації*. 2013. №6 (113). С.194-198.
270. Краснобаев В.А., Кошман С.А., Курчанов В.Н., Гарамась А.В. Метод контроля криптографической информации, представленной в модулярной системе счисления. *Збірник наукових праць Харківського університету Повітряних сил ім.І.Кожедуба*. 2013. Вип.3 (36). С.104-107.
271. Krasnobayev V.A., Tyrtyshnikov O.I., Sliusar I.I., Kurchanov V.N., Koshman S.A. The model and the method of implementation of integer arithmetic operations within the RSA crypto algorithms. *Системи обробки інформації*. 2014. №1 (117). С.117-122.
272. Krasnobayev V.A., Tyrtyshnikov O.I., Somov S.V., Koshman S.A., Sokol G.V., Rvachova N.V. Mathematical model and tabular method implementation of modular arithmetic operations with crypto transformations in the residue class. *Системи обробки інформації*. 2014. №2 (118). С.119-123.

273. Стіренко С.Г., Марковський О.П., Лефтерис З., Міщенко Л.Д. Спосіб прискореного обчислення модулярної експоненти. *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка*. №65. С. 118-122.
274. Самофалов К.Г., Рамзи Анвар Салиба Сунна. Ускоренная реализация модулярного экспоненцирования на малоразрядных микропроцессорах и встроенных микроконтроллерах. *Проблеми інформатизації та управління*: Збірник наукових *праць*. 2005. Вип. 4(15). С.144- 153.
275. Xiang C. Verifiable and Secure Outsourcing Schemes of Modular Exponentiations Using One Untrusted Cloud Server and Their Application. IACR Cryptology. 2014. 500 p.
276. Markovskyi O.P., Bardis N., Doukas N., Kirilenko S. Secure Modular Exponentiation in Cloud Systems. *Information Technology, Computational and Experimental Physics (CITCEP 2015)*: Proceedings of the Congress, Krakow, Poland. P. 266-269.
277. Sharifi S., Esmaeildoust M., Taheri M., Navi K. Efficient Implementation of RNS Montgomery Multiplication Using Balanced RNS Bases. *Journal of mathematics and computer science*. 2014. Vol.12. P.51-64.
278. Bajard J.-C., Eynard J., Merkiche N. Montgomery Reduction within the Context of Residue Number System. *Arithmetic Journal of Cryptographic Engineering*. 2017. № 2. P. 121-132.
279. Bajard J.-C. Eynard J., Gandino F. Fault Detection in RNS Montgomery Modular Multiplication. *Computer Arithmetic*: Proceedings of the 21st IEEE Symposium. 2013. P.119-126.
280. 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.
281. Касянчук М.М. Концепція теоретичних положень досконалої форми перетворення Крестенсона та його практичне застосування. *Оптико-*

електронні інформаційно-енергетичні технології. 2010. №2 (20). С. 43-47.

282. 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. P.168-171.
283. Касянчук М.М., Якименко І.З., Давлетова А.Я., Долинюк Т.М., Рендзеняк Н.А. Метод знаходження модулів досконалої форми системи залишкових класів. *Теорія прийняття рішень*: Праці VII Міжнародної школи-семінару, Ужгород. 2014. С. 122-123.
284. 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.
285. Kasianchuk M. Conception of theoretical bases of the accomplished form of Krestenson's transformation and its practical application. *Advanced Computer Systems and Network: Design and Application* (*ACSN–2009*): Proceedings of the 4–th International Conference. L'viv. 2009. P.299–301.
286. Касянчук М.М. Теорія та математичні закономірності досконалої форми системи залишкових класів. *Питання оптимізації обчислень* (*ПОО–XXXV*): Праці Міжнародного симпозіуму. Київ–Кацивелі. 2009. Т.1. С. 306–310.
287. Касянчук М.М. Теорія перетворення досконалої форми системи залишкових класів базису Крестенсона. *Інформаційні проблеми комп'ютерних систем, юриспруденції, економіки та моделювання (ПНМК–2009)*: Матеріали проблемно–наукової міжгалузевої конференції. Тернопіль-Бучач. 2009. С.133–137.
288. Касянчук М. Алгоритми побудови модифікованої досконалої форми системи залишкових класів. *Спеціалізовані комп'ютерні технології в*

інформатиці: Колективна монографія. Під ред. Я.Николайчука. Тернопіль: Бескиди, 2017. С. 580-604. Р. 11.

289. Nykolaychuk Ya.M., Kasianchuk M.M., Yakymenko I.Z. Theoretical Foundations of the Modified Perfect form of Residue Number System. *Cybernetics and Systems Analysis*. 2016. Vol. 52, №2. P. 219-223.
290. 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*. 2016. Vol.48, №8. P.56-63.
291. Касянчук М. Построение модифицированной совершенной формы системы остаточных классов с использованием факторизации. *Радиоэлектроника, информатика, управление*. 2017. Vol.42, №3. P.53-59
292. Касянчук М.М. Побудова модифікованої досконалої форми системи залишкових класів на основі розв'язку систем конгруенцій. *Науковий збірник Національного лісотехнічного університету України*. 2016. Т.26, №7. С. 372-377.
293. Касянчук М.М. Побудова трьохмодульної модифікованої досконалої форми системи залишкових класів на основі розв'язку квадратного рівняння. *Інформатика та математичні методи в моделюванні*. 2016. Т.6, №1. С. 19–25.
294. Касянчук М. М. Досконала форма системи залишкових класів: методи побудови та застосування (Монографія). Тернопіль: Економічна думка (ТНЕУ), 2019. 224 с.
295. Волинський О. І. Методи побудови високопродуктивних спецпроцесорів на основі теоретико-числового базису Крестенсона: дис. канд. техн. наук: 05.13.05. Тернопіль: ТНЕУ, 2013. 210 с.
296. 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. Ceske Budejovice (Czech Republic). 2019. P. 175-178

297. 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.
298. Якименко І. З. Методи та алгоритми опрацювання інформаційних потоків в комп'ютерних мережах за умови застосування еліптичних кривих: дис. канд. техн. наук: 05.13.05. Тернопіль: ТНЕУ, 2011. 193 с.
299. 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. P.161–163.
300. Kasjanchuk M., Yakymenko I., Ivasiev S., Nykolaychuk Ya. Fundamental theoretical and algorithmic principles of the applied tasks decision of theory of numbers and construction of the high-performance special processors on their basis. *The Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2011*): Proceedings of the XI International Conference. Polyana-Svalyava. 2011. P.168-169.
301. Касянчук М.М. Математичні проблеми вдосконалення теорії матричних числень. *Інформаційні проблеми комп'ютерних систем, юриспруденції, економіки та моделювання (ПНМК–2008)*: Матеріали проблемно–наукової міжгалузевої конференції. Тернопіль-Бучач. 2008. С.25–29.
302. 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.

303. 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.
304. Rajba T. Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of Time Characteristics of Search Methods of Inverse Element by the Module. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (*IDAACS–2017*): Proceedings of the 2017 IEEE 9th International Conference. Bucharest, Romania. V.1. September, 2017. P.82–85.
305. Dasgupta S., Papadimitriou C., Vazirani U. Algorithms. McGraw-Hill Science, Engineering, Math. 2006. 336 p.
306. Касянчук М.М., Якименко І.З., Волинський О.І., Пітух І.Р. Теорія алгоритмів RSA та Ель–Гамаля в розмежованій системі числення Рademахера–Крестенсона. *Вісник Хмельницького національного університету. Технічні науки.* 2011. №3. С. 265-273.
307. 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. P.550-554.
308. Касянчук М.М., Якименко І.З., Николайчук Я.М. Теорія алгоритмів пошуку найбільшого спільного дільника у базисі Крестенсона. *Вісник Тернопільського національного технічного університету.* 2011. Т.16, №1. С. 154–161.
309. Касянчук М.М., Николайчук Я.М., Якименко І.З. Теорія алгоритмів перетворень китайської теореми про залишки в матрично розмежованому базисі Рademахера–Крестенсона. *Вісник Національного університету «Львівська політехніка» «Комп'ютерні системи та мережі»*. 2010. №688. С. 118–124.

310. Касянчук М.М. Фундаметальні основи китайської теореми про залишки в теоретико–числовому базисі Крестенсона–Радемахера. *Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління* (*ПНМК-2012*): Матеріали проблемно-наукової міжгалузевої конференції. Бучач. 2012. С. 95-100.
311. Задірака В.К., Николайчук Я.М., Касянчук М.М. Теоретичні основи та високопродуктивний алгоритм обчислення мультистепеневої функції в базисі Крестенсона *Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління* (*ПНМК-2010*): Матеріали проблемно-наукової міжгалузевої конференції. Бучач. 2010. с.30-32.
312. 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.
313. 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.
314. Николайчук Я.М., Ивасьев С.В., Якименко И.З., Касянчук М.Н. Метод факторизации многоразрядных чисел на основе свойств квадратичности вычетов в системе остаточных клас сов. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2015. № 5(95). С. 45–45.
315. 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. P.1484–1486.

316. Николайчук Я.М., Івасьєв С.В., Якименко І.З., Касянчук М.М. Алгоритм факторизації на основі теореми Ферма за допомогою використання властивостей квадратичності лишків. *Юриспруденція та проблеми інформаційного суспільства (ЮПІС–2016)*: Матеріали проблемно–наукової міжгалузевої конференції. Тернопіль-Бучач. 2016. С.133–138.
317. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. *Advanced Computer Information Technology* (*ACIT–2018*): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235.
318. Волинський О.І. Методи міжбазисних перетворень на основі розмежованої системи числення залишкових класів. *Вісник національного університету "Львівська політехніка", "Комп'ютерні системи та мережі"*. 2010. №688. С.53-59.
319. 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*. 2014. Vol. 50, № 5. P. 649-654.
320. Николайчук Я.М., Касянчук М.М., Якименко І.З., Тимошенко Л.М., Долинюк Т.М. Алгоритм знаходження системи модулів модифікованої досконалої форми системи залишкових класів. *Сучасні інформаційні та електронні технології*: Матеріали Міжнародної науково-практичної крнференції. Одеса. 2014. С. 115-116.
321. Stakhov A. The Generalized Principle of the Golden Section and its Applications in Mathematics, Science and Engineering. *Chaos, Solitons & Fractals*. 2005. V. 26 (2). p. 263-289.
322. 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.

323. 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.
324. Кучерук І.М., Горбачук І.Т., Луцик П.П. Загальний курс фізики. К.: Техніка, 2001. 452 с.
325. Заставний О.М., Николайчук Я.М. Методологія побудови автономних сенсорів для розподілених комп'ютерних мереж. *Вісник Технологічного університету Поділля*. 2002. Т1, №3. С. 142-146.
326. 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.
327. Касянчук М.М., Чирка М.І., Николайчук Я.М., Якименко І.З. Метод побудови розподіленого температурного сенсора на основі системи числення базису Крестенсона. *Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління ПНМК-2011*: Матеріали проблемно-наукової міжгалузевої конференції. Бучач-Яремча. 2011. С. 143-149.
328. 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.
329. 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.
330. Kasianchuk M. Yakymenko I., Pazdriy I., Melnyk A., Ivasiev S. 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. P.222-224.

331. 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.
332. Grynchyshyn T., Yakymenko I., Nykolaychuk Ya., Kasyanchuk M. The Theoretical Basis of Bisignal Formation of Information Flow in Computer Systems with Open Optical Signals. *Modern Problems of Radio Engineering, Telecommunications and Computer Science* (*TCSET–2010*): Proceedings of the X–th International Conference. L'viv–Slavske. 2010. P.222.
333. 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. P.403–406.
334. Карпінський М., Кінах Я., Яциковська У., Якименко І., Касянчук М. Удосконалення архітектури комп'ютерної мережі для програмної реалізації криптоаналітичних алгоритмів. *Інформаційні моделі, системи та технології*: Матеріали V науково-технічної конференції. Тернопіль, 2018. С. 93.
335. Stewart J. Python for Scientists. Cambridge: Cambridge University Press, 2014. 230 p.
336. 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.
337. Задірака В.К., Олексюк О.С. Комп'ютерна арифметика багаторозрядних чисел. К.: 2003. 264 с.

338. 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.
339. Николайчук Я.Н., Ивасьев С.В., Якименко И.З., Касянчук М.Н. Метод компактной кодировки простых многоразрядных чисел в двоичной системе исчисления. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2016. №5 (96). С. 21-23.
340. 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, Telecommunica-tions and Computer Science* (*TCSET–2016*): Proceedings of the XIII–th International Conference. L'viv–Slavske. 2016. P.534-536.
341. Николайчук Я.М., Якименко І.З., Івасьєв С.В., Касянчук М.М. Метод збереження простих великорозрядних чисел у базисі Радемахера. *Питання оптимізації обчислень* (*ПОО-XXXVII*): Матеріали Міжнародної молодіжної математичної школи. Київ: Інститут кібернетики імені В.М. Глушкова НАН України. 2015. С. 159-161..
342. Якименко І.З., Касянчук М.М., Івасьєв С.В. Метод зберігання простих великорозрядних чисел у базисі Радемахера. *Питання оптимізації обчислень* (*ПОО-XXXVII*): Матеріали Міжнародної молодіжної математичної школи. Київ: Інститут кібернетики імені В.М. Глушкова НАН України. 2013. С. 142-144.
343. Якименко І.З., Касянчук М.М., Волинський О.І., Івасьєв С.В. Теоретичні основи аналітики та алгоритми оптимізації обчислень простих чисел. *Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління (ПНМК-2010)*: Матеріали проблемно-наукової міжгалузевої конференції. Бучач-Яремча. 2010. С.33-36.

344. Касянчук М.Н. Экспериментальное исследование программной реализации системы остаточных классов и ее модифицированной совершенной формы. *Вестник Брестского государственного технического университета. Физика, математика, информатика*. 2017. №5 (97). С. 53-57.
345. Яциковська У.О. Касянчук М.М., Трембач Р.Б. Удосконалена система захисту комп'ютерної мережі на підставі асиметричного шифрування. *Вісник Східноукраїнського національного університету імені Володимира Даля*. 2009. №6(136). Ч.1. С. 57–60.
346. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах. *Захист інформації*. 2019. Т.21, №2. С. 65- 73.

Додаток А

Документи, що підтверджують впровадження результатів дисертаційної роботи

ДЕРЖАВНА СЛУЖБА СПЕЦІАЛЬНОГО ЗВ'ЯЗКУ
ТА ЗАХИСТУ ІНФОРМАЦІЇ УКРАЇНИ

Управління в Тернопільській області

46002, м. Тернопіль, проспект С. Бандери, 21,
тел. (0352) 52-14-19, факс (0352) 52-19-98, e-mail: ternopil@dsszzi.gov.ua

АКТ

впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики»

Даний акт складений про те, що результати дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.21 – Системи захисту інформації, опрацьовані Управлінням Державної служби спеціального зв'язку та захисту інформації України в Тернопільській області, визнані такими, що мають актуальність та практичне значення і можуть бути використані для збільшення швидкості процесу шифрування та надійності передачі зашифрованої конфіденційної інформації засобами спеціального зв'язку.

Т.в.о. начальника Управління С.М. Скрепецький

8.07.2019р.

ДЕРЖАВНА СЛУЖБА УКРАЇНИ З НАДЗВИЧАЙНИХ СИТУАЦІЙ

Управління Державної служби України
з надзвичайних ситуацій у Тернопільській області

вул.Л.Українки,6, м. Тернопіль, 46011 тел.: (0352) 43-43-30, факс: 24-37-30
код ЄДРПОУ 38535547 E-mail: ternopil@mns.gov.ua

АКТ
впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі медулярної арифметики»

Даний акт складений про те, що результати дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багато розрядних чисел в асиметричних криптосистемах на основі медулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.21 - Системи захисту інформації, опрацьовані Управлінням, визнані такими, що володіють актуальністю та практичним значенням і плануються до впровадження у роботі Управління. Зокрема, може бути використано:

- алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу медулярного множення та модулярного експоненціювання;
- метод пошуку мультистепеневої функції за модулем;
- програмна реалізація та дослідження виконання множення в цілочисельній та модефікованій досконалій формі системи залишкових класів.

Начальник Управління В.М. Маслей

INTEGRAL®
UNINTERRUPTABLE POWER SUPPLY DEVICES

Товариство з обмеженою відповідальністю
"Науково – виробнича фірма "Інтеграл"

Україна, 04208, м. Київ
п-кт. Правди, буд. 66, к. А, оф. 2
тел. 26 – 19 – 55

р\р 26002010044563 в
АБ "Південний"
МФО 328209 ЗКПО 32220131

Вих. № 2507-01
від 25 липня 2019р.

АКТ

про впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича на тему «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що ТОВ НВФ «ІНТЕГРАЛ» передані для впровадження результати дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», які включають:

- алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу модулярного множення та модулярного експоненціювання;
- програмне забезпечення для реалізації та дослідження методів модулярного експоненціювання в криптосистемі RSA та факторизації;
- алгоритмічне забезпечення для пошуку модулів досконалої та модифікованої досконалої форм системи залишкових класів на основі дробових перетворень, факторизації, теореми Вієта, розв'язку систем конгруенцій, що дозволило уникнути виконання операції оберненого елемента за модулем та множення на нього при переведенні чисел із системи залишкових класів в десяткову систему числення.

Передані для впровадження матеріали використані для забезпечення необхідного рівня захисту у спеціалізованих комп'ютерних системах при шифруванні, генерації цифрового підпису, обробці цифрових даних у реальному часі.

Директор ТОВ « НВФ «Інтеграл» **О. С. Колос**

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«ТЕРНОПІЛЬСЬКЕ КОНСТРУКТОРСЬКЕ БЮРО РАДІОЗВ`ЯЗКУ «СТРІЛА»

46020, м. Тернопіль, вул 15 Квітня, 6, тел/факс – 28-75-00, 28-72-00, tkbr_strila@ukr.net, tkbr_strila@mail.ru
р/р 26009308101054 в ТВБВ 10019/08 Філія Тернопільського обласного управління Ощадбанку,
МФО 338545, код 14042350
№ 288 від 19.07.2019р.

ЗАТВЕРДЖУЮ
Директор ТОВ ТКБР «Стріла»
_____ О.О.Рафалюк
«__» _____ 2019 р.

АКТ

впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що результати дисертаційної роботи Касяччука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики» передані для впровадження ТОВ ТКБР «Стріла» для забезпечення захисту передачі даних між об'єктами енергетики і унеможливлення зовнішнього втручання в роботу енергетичної системи. Результати даної роботи впроваджені при вдосконаленні комплексу дистанційного керування технологічними процесами «СТРІЛА-М», зокрема:

- алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу модулярного множення та модулярногоекспоненціювання;
- схемотехнічна реалізація трьохмодульної криптосистеми Рабіна;
- програмна реалізація операції множення в цілочисельній та модифікованій досконалій формі системи залишкових класів.

Застосування вказаних реалізацій та алгоритмічного забезпечення дозволило підвищити швидкодію опрацювання даних та рівень захищеності інформації від несанкціонованого доступу.

Головний інженер С.О.Піскун

АКТ

впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Результати дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», які направлені на підвищення ефективності опрацювання багаторозрядних чисел в асиметричних криптосистемах, використані в розробках компанії «CONNECT».

Запропоновані автором дисертації підходи до збільшення швидкодії алгоритмів, спеціалізованого програмного та апаратного забезпечення в асиметричних криптосистемах дозволили зменшити час опрацювання багаторозрядних чисел та підвищити рівень захисту інформаційних потоків при розробці інтелектуальної системи DataDet за рахунок використання векторно-модульних методів та розпаралелення процесу обчислень на основі модифікованої досконалої форми системи залишкових класів.

Директор
10.07.2019р.

Р.А.Яконюк

ГРОМАДСЬКА ОРГАНІЗАЦІЯ
«МІЖНАРОДНА АКАДЕМІЯ ІНФОРМАЦІЇ»

Код ЄДРПОУ 42154579
03022, м. Київ, вул. Михайла Максимовича, 7, 168
Тел. +38-073-418-96-76
e-mail: ngo-mai@ukr.net
https://interacademy.info

АКТ

про впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича на тему «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.21 – Системи захисту інформації

Даний акт складений про те, що результати дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики» впроваджені у науковий процес, що здійснюється у Громадській організації «Міжнародна академія інформації» (Public organization "INTERNATIONAL ACADEMY OF INFORMATION").

Складові результатів цієї дисертаційної роботи, а саме: метод пошуку модулів системи залишкових класів, який дозволяє аналітично обчислювати коефіцієнти базисних чисел, програмна реалізація операції множення в цілочисельній та модифікованій досконалій формі системи залишкових класів, трьохмодульна криптосистема Рабіна та її VHDL-модель використовуються під час тренінгових занять в частині використання запропонованих механізмів для кіберзахисту інформаційних ресурсів.

Голова
ГО «Міжнародна академія інформації» **Тетяна Марущак**

«10» 07 2019 р.

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Тернопільського національного
економічного університету,
д.е.н., проф. З.-М.В.Задорожний
_____
«11» 07 2019 р.

АКТ

про використання результатів докторської дисертації Касянчука Михайла Миколайовича на тему «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики»,

Комісія у складі декана факультету комп'ютерних інформаційних технологій, д.т.н., проф. Дивака М.П. (голова комісії), т.в.о. начальника науково-дослідної частини Русин Т.Т., начальника планово-фінансового відділу Кушніра О.Р. склали цей акт про те, що дослідження та результати дисертаційної роботи Касянчука М.М. використані під час виконання науково-дослідних робіт на кафедрах комп'ютерної інженерії, спеціалізованих комп'ютерних систем 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 розпаралелення алгоритмів захисту інформації;

2) науково-дослідної роботи на тему: "Опрацювання багаторозрядних чисел в системі залишкових класів" (керівник), номер державної реєстрації 0115U001607, у якій автором розроблено методи модулярного множення та експоненціювання багаторозрядних чисел;

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 розроблено теоретичні основи опрацювання багаторозрядних чисел;

4) госпдоговірної теми «Світлодіодне підсвічування зовнішньої реклами» (виконавець), у якій автором розроблено методи розподілу світлодіодних імпульсів;

5) науково-дослідної роботи на тему: "Теоретичні основи та апаратні засоби підвищення продуктивності роботи безпровідних сенсорних мереж" (відповідальний виконавець), номер державної реєстрації 0117U000414, у якій автором розроблено методи побудови наборів модулів модифікованої досконалої форми системи залишкових класів.

Голова комісії:
Декан факультету комп'ютерних
інформаційних технологій М.П.Дивак

Члени комісії:
Т.в.о.начальника НДЧ Т.Т.Русин

Начальник планово-
фінансового відділу О.Р.Кушнір

University
of Bielsko-Biala

UNIVERSITY OF BIELSKO-BIALA
DEPARTMENT OF COMPUTER SCIENCE
AND AUTOMATICS
2 Willowa St. Bielsko-Biala, 43-309 Poland
tel. 33 827 92 64

Department of Computer Science and Automatics

АКТ
від 16 липня 2019 року

про впровадження результатів дисертаційної роботи
пана канд. фіз.-мат. наук Михайла Миколайовича Касянчука
"Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах
на основі модулярної арифметики"
на здобуття наукового ступеня доктора технічних наук

Основні результати наукового дослідження пана канд. фіз.-мат. наук Михайла Миколайовича Касянчука на здобуття наукового ступеня доктора технічних наук апробовані та впроваджені у наукових роботах кафедри інформатики та автоматики, а також використані у навчальному процесі при викладанні дисципліни «Безпека інформаційних технологій» для студентів напряму "Інформатика".

Використання матеріалів дисертаційної роботи канд. фіз.-мат. наук Михайла Миколайовича Касянчука у викладенні вказаних дисциплін сприяє підвищенню якості підготовки фахівців.

Комісія в складі:
Заст. завідувача кафедри інформатики та автоматики,
доктор технічних наук, професор Василь Марценюк

професор кафедри інформатики та автоматики,
доктор технічних наук Станіслав Анджей Райба

43-309 Bielsko-Biała, ul. Willowa 2
phone: (33) 8279 264, fax: (33) 8279 264
Regon: 072728961, NIP: 547-19-43-784
kinf@ath.bielsko.pl, www.ath.bielsko.pl

University
of Bielsko-Biala

UNIVERSITY OF BIELSKO-BIALA
DEPARTMENT OF COMPUTER SCIENCE
AND AUTOMATICS
2 Willowa St, Bielsko-Biala, 43-309 Poland
tel. 33 827 92 64

Department of Computer Science and Automatics

ZAŚWIADCZENIE
z dnia 16 lipca 2019 roku

o wdrożeniu wyników pracy dysertacyjnej
Pana dra inż. Mykhaila Kasianchuka syna Mykoly
"Metody opracowania liczb wielocyfrowych w kryptosystemach asymetrycznych
na podstawie arytmetyki modularnej"
na uzyskanie stopnia naukowego doktora habilitowanego nauk technicznych

Główne wyniki badania naukowego Pana dra inż. Mykhaila Kasianchuka syna Mykoly będące podstawą do uzyskania stopnia naukowego doktora habilitowanego nauk technicznych aprobowano i wdrożono w pracach Katedry Informatyki i Automatyki, a także zastosowano w procesie dydaktycznym przy prowadzeniu przedmiotów "Bezpieczeństwo technologii informatycznych" dla studentów kierunku "Informatyka".

Zastosowanie rezultatów rozprawy dysertacyjnej Pana dra inż. Mykhaila Kasianchuka syna Mykoly w prowadzeniu powyższych przedmiotów przyczynia się do poprawy jakości kształcenia wysokowykwalifikowanych fachowców.

Komisja w składzie:
Zast. kierownika Katedry Informatyki i Automatyki
Profesor Katedry Informatyki i Automatyki
prof. dr hab. Vasyl Martsenyuk

Profesor Katedry Informatyki i Automatyki
dr hab. inż. Stanisław Andrzej Rajba

43-309 Bielsko-Biała, ul. Willowa 2
phone: (33) 8279 264, fax: (33) 8279 264
Regon: 072728961, NIP: 547-19-43-784
kinf@ath.bielsko.pl, www.ath.bielsko.pl

ЗАТВЕРДЖУЮ
Перший проректор
Тернопільського національного
економічного університету
_____ М.І.Шинкарик
«___» _____ 2019 р.

АКТ

впровадження у навчальний процес результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що результати, отримані в дисертаційній роботі Касянчука Михайла Миколайовича, зокрема: побудова трьохмодульної криптосистеми Рабіна та її VHDL-модель, вдосконалений метод Ферма для факторизації багаторозрядних чисел, алгоритмічне забезпечення для реалізації криптосистем RSA та Ель-Гамаля на основі векторно-модульного методу, програмні та апаратні рішення методів пошуку оберненого елемента за модулем, програмна реалізація операції множення в цілочисельній та модифікованій досконалій формі системи залишкових класів використовуються в навчальному процесі факультету комп'ютерних інформаційних технологій Тернопільського національного економічного університету при підготовці бакалаврів та магістрів спеціальності «Комп'ютерна інженерія» при викладанні дисциплін «Захист інформації в комп'ютерних системах», «Комп'ютерна криптографія», спеціальності «Кібербезпека» при викладанні дисциплін «Основи кібербезпеки», «Дослідження і проектування систем захисту інформації», спеціальності «Інженерія програмного забезпечення» при викладанні дисципліни «Методи та засоби захисту програмних продуктів».

Декан факультету комп'ютерних
інформаційних технологій, д.т.н., проф. М.П.Дивак

Завідувач кафедри комп'ютерних наук,
к.т.н., доц. А.В.Пукас

Завідувач кафедри кібербезпеки
д.т.н., доц. В.В.Яцків

УКРАЇНА
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ГНАТЮКА
вул. М.Кривоноса, 2, м. Тернопіль, 46027,
тел. (0352)43-58-80, факс (0352)43-60-55,
e-mail: info@tnpu.edu.ua, код ЄДРПОУ 02125544

UKRAINE
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
TERNOPIL VOLODYMYR HNATIUK NATIONAL PEDAGOGICAL UNIVERSITY
2 M. Kryvonosa st., Ternopil, 46027, Ukraine
tel. +38 0352 43 60 67, fax: +38 0352 43 60 55
e-mail: info@tnpu.edu.ua

від "15" 07 2019 р. № 928-33/03

АКТ

впровадження у навчальний процес результатів дисертаційної роботи Касянчука Михайла Миколайовича «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що результати, отримані в дисертаційній роботі Касянчука Михайла Миколайовича, зокрема: методи модулярного множення та експоненціювання на основі векторно-модульного підходу; методи пошуку оберненого елемента за модулем та виконання китайської теореми про залишки; метод пошуку мультистепеневої функції за модулем; методи пошуку наборів модулів досконалої та модифікованої досконалої форм системи залишкових класів; математичне забезпечення трьохмодульної криптосистеми Рабіна використовуються в навчальному процесі фізико-математичного факультету Тернопільського національного педагогічного університету імені Володимира Гнатюка при підготовці бакалаврів та магістрів спеціальності «Інформатика» при викладанні дисциплін «Захист інформації та шифрування даних», «Основи кібербезпеки» та спеціальності «Математика» при викладанні дисциплін «Алгебра і теорія чисел», «Комп'ютерна математика», «Математичні методи дослідження операцій».

Проректор з наукової роботи
та міжнародного співробітництва Г.І. Фальфушинська

Декан фізико-математичного факультету М.І.Громяк

10565

АКТ

про впровадження результатів дисертаційної роботи Касянчука Михайла Миколайовича на тему «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений про те, що результати дисертаційної роботи Касянчука Михайла Миколайовича на тему «Методи опрацювання багаторозрядних чисел в асиметричних криптосистемах на основі модулярної арифметики» впровадженні в навчальний процес Академії ГУСПОЛ (Чеська Республіка).

Складові результатів цієї дисертаційної роботи, а саме: алгоритмічне забезпечення для паралельної реалізації криптосистеми RSA за допомогою системи залишкових класів, програмне забезпечення для реалізації та дослідження методів медулярного експоненціювання в криптосистемі RSA, алгоритмічне забезпечення для пошуку модулів досконалої та модифікованої досконалої форм системи залишкових класів враховуються під час складання тематичних планів навчальної дисципліни «Національна безпека» з метою створення необхідних умов забезпечення належного рівня захисту при передачі по комп'ютерних мережах та обробці цифрових даних у реальному часі.

Проректор з міжнародних зв'язків
Академії ГУСПОЛ (Чеська Республіка),
експерт з національної безпеки
доктор юридичних наук, професор **Ігор Копотун**

10 07 2019 р.

Додаток Б

Лістинги кодів програмних та програмно-апаратних модулів

```
package edu.tneu.tdm.logic;

import java.util.Comparator;
import java.util.LinkedList;
import java.util.TreeSet;

public class ModulesCalculator {
   private static TreeSet<Pair>
generateModules(LinkedList<Integer> list,
            int[] modules) {
        TreeSet<Pair> data = new TreeSet<>(new
Comparator<Pair>() {
            @Override
            public int compare(Pair p1, Pair p2) {
                return p1.a - p2.a + p1.b - p2.b;
            }
        });

        // додаємо до множників одиницю
        if (!list.contains(1))
            list.add(1);
        // перебір можливих комбінацій множників
        for (int count = 0; count < list.size();
count++) {
            for (int i = 1; i < list.size(); i++) {
                int dobA = 1, dobB = 1;
                int j = 0;
                // порахувати добутки
                for (Integer el : list) {
                    if (j < i) {
                        dobA *= el;
                    }

                    else {
                        dobB *= el;
                    }
                    j++;
                }
                // перевірити чи задовільняють числа
задану умову і додати в
                // список
                if (test(dobA, modules) && test(dobB,
modules)) {
                    int p4 = calcP45(dobA, modules);
                    int p5 = calcP45(dobB, modules);
                    data.add(new Pair(p4, p5));
                }
            }
```

```
            // перемістити останній елемент на перше
місце
            int last = list.getLast();
            list.removeLast();
            list.addFirst(last);
        }
        return data;
    }

    public static class Pair {
        public final int a;
        public final int b;

        public Pair(int a, int b) {
            if (a < b) {
                this.a = a;
                this.b = b;
            } else {
                this.b = a;
                this.a = b;
            }
        }
    }

    public static TreeSet<Pair> calculate(int[] modules)
{
        int ab = calcAB(modules);
        LinkedList<Integer> fab = primeFactor(ab); //
факторизуємо добудок
        if (fab.size() > 1)
            return generateModules(fab, modules);//
повертаємо обчислені модулі
        else
            return null;
    }

    private static int calcAB(int m[]) {
        return (6 * m[2] * m[3]) * (6 * m[2] * m[3])
                - (m[3] * (m[2] - 6) - 6 * m[2]);
    }

    private static int calcP45(int ab, int m[]) {
        return (6 * m[2] * m[3] + ab) / (m[3] * (m[2] -
6) - 6 * m[2]);
    }

    private static boolean test(int ab, int m[]) {
        int rez = (6 * m[2] * m[3] + ab) % (m[3] * (m[2]
- 6) - 6 * m[2]);
        return rez == 0;
    }
```

```
   public static LinkedList<Integer> primeFactor(int
posInt) {
        LinkedList<Integer> primesInt = new
LinkedList<>();
        for (int i = 2; i <= posInt; i++) {
             int powerDegree = 0;

             while (posInt % i == 0) {
                  posInt /= i;
                  powerDegree++;
             }
             for (int j = 0; j < powerDegree; j++) {
                  primesInt.add(i);
             }
        }
        return primesInt;
   }
}
package edu.tneu.tdm.mods;

import java.util.Arrays;

public class Modules extends AbstractModules {
   private int[] modules;

   public Modules(int[] mods) {
        setModules(mods);
   }

   @Override
   public void recalculateParams() {
        mul = 1;
        sum = 0;
        for (int i = 0; i < modules.length; i++) {
             double powed = 1.0 / modules[i];
             sum += powed;
             mul *= powed;
        }
   }

   @Override
   public int[] getModules() {
        return modules;
   }

   @Override
   public void setModules(int[] mods) {
        Arrays.sort(mods);
        this.modules = mods;
        recalculateParams();
   }

   @Override
```

```
        public double getModulesMul() {
             return mul;
        }

        @Override
        public double getModulesSum() {
             return sum;
        }

        @Override
        public void setModule(int i, int value) {
             if (i < modules.length) {
                  modules[i] = value;
                  recalculateParams();
             } else
                  throw new IllegalArgumentException("Index
    is too high!");

        }
    }
```

Програмний код для апаратної реалізації трьохмодульної криптосистеми Рабіна

```
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;
import java.security.SecureRandom;
import java.util.Random;

import static java.math.BigDecimal.ROUND_HALF_UP;

public class RabinDecimal {
    private static Random r = new SecureRandom();
    private static BigDecimal Negative_One =
BigDecimal.valueOf(-1);
    private static BigDecimal k1 = BigDecimal.valueOf(0);
    private static BigDecimal k2 = BigDecimal.valueOf(0);
    private static BigDecimal k3 = BigDecimal.valueOf(0);
    private static BigDecimal ZERO = BigDecimal.valueOf(0);
    private static BigDecimal TWO = BigDecimal.valueOf(2);
    private static BigDecimal THREE = BigDecimal.valueOf(3);
    private static BigDecimal FOUR = BigDecimal.valueOf(4);
    private static BigDecimal p;
    private static BigDecimal q;
    private static BigDecimal z;
    private static BigDecimal n;
    private static int bitKey = 10;

    public static BigDecimal[] genKey(int bitLength) {
```

```
        while
(p.add((p.multiply(p).add(BigDecimal.ONE)).divide(q.subtract(p),
2)).remainder(BigDecimal.ONE) != BigDecimal.ZERO) {
            p = new BigDecimal(BigInteger.probablePrime(bitKey,
r));
            q = new BigDecimal(BigInteger.probablePrime(bitKey,
r));

            if
(p.add((p.multiply(p).add(BigDecimal.ONE)).divide(q.subtract(p),
2)).remainder(BigDecimal.ONE) == BigDecimal.ZERO) {
                z =
p.add((p.multiply(p).add(BigDecimal.ONE)).divide(q.subtract(p),
2));
            } else {
                z =
p.add((p.multiply(p).subtract(BigDecimal.ONE)).divide(q.subtract
(p), 2));
            }

            bitKey = bitKey++;
        /*BigDecimal p = BigDecimal.valueOf(191);
        BigDecimal q = BigDecimal.valueOf(229);
        BigDecimal z = BigDecimal.valueOf(1151);*/

            n = p.multiply(q).multiply(z);
        }
        return new BigDecimal[]{n, p, q, z};
    }


    public static BigDecimal encrypt(BigDecimal word, BigDecimal
n) {

        return
BigDecimal.valueOf(word.toBigInteger().modPow(TWO.toBigInteger()
, n.toBigInteger()).longValue());

    }

    public static BigDecimal[] decrypt(BigDecimal crypted,
BigDecimal p, BigDecimal q, BigDecimal z) {

        BigDecimal n = p.multiply(q).multiply(z);
        BigDecimal radix1 = BigDecimal.ONE;
        BigDecimal radix2 = BigDecimal.ONE;
        BigDecimal radix3 = BigDecimal.ONE;


        if
(k1.multiply(k1).remainder(p).compareTo(crypted.remainder(p)) !=
0) {
```

```
            while
(k1.multiply(k1).remainder(p).compareTo(crypted.remainder(p)) !=
0) {
                k1 = k1.add(BigDecimal.ONE);
                radix1 = k1;
            }
        } else {
            radix1 = k1;
        }

        if
(k2.multiply(k2).remainder(q).compareTo(crypted.remainder(q)) !=
0) {

            while
(k2.multiply(k2).remainder(q).compareTo(crypted.remainder(q)) !=
0) {
                k2 = k2.add(BigDecimal.ONE);
                radix2 = k2;
            }
        } else {
            radix2 = k2;
        }

        if
(k3.multiply(k3).remainder(z).compareTo(crypted.remainder(z)) !=
0) {

            while
(k3.multiply(k3).remainder(z).compareTo(crypted.remainder(z)) !=
0) {
                k3 = k3.add(BigDecimal.ONE);
                radix3 = k3;
            }
        } else {
            radix3 = k3;
        }

        System.out.println("radix1=" + radix1);
        System.out.println("k1=" + k1);
        System.out.println("radix2=" + radix2);
        System.out.println("k2=" + k2);
        System.out.println("radix3=" + radix3);
        System.out.println("k3=" + k3);

        BigDecimal p_a = p.subtract(radix1);
        BigDecimal p_b = q.subtract(radix2);
        BigDecimal p_z = z.subtract(radix3);

        BigDecimal m1 = n.divide(p).multiply(Negative_One);
        BigDecimal m2 = n.divide(q);
        BigDecimal m3 = n.divide(z);
```

```
        //y_p*p*m_q + y_q*q*m_p (mod n)
        BigDecimal out1 =
((m1.multiply(radix1)).add(m2.multiply(radix2)).add(m3.multiply(
radix3))).remainder(n);

        if (out1.compareTo(BigDecimal.ZERO) == -1) {
            out1 = out1.add(n);
        }

        BigDecimal out2 =
((m1.multiply(radix1)).add(m2.multiply(radix2)).add(m3.multiply(
p_z))).remainder(n);

        if (out2.compareTo(BigDecimal.ZERO) == -1) {
            out2 = out2.add(n);
        }

        BigDecimal out3 =
((m1.multiply(radix1)).add(m2.multiply(p_b)).add(m3.multiply(rad
ix3))).remainder(n);

        if (out3.compareTo(BigDecimal.ZERO) == -1) {
            out3 = out3.add(n);
        }

        BigDecimal out4 =
((m1.multiply(radix1)).add(m2.multiply(p_b)).add(m3.multiply(p_z
))).remainder(n);

        if (out4.compareTo(BigDecimal.ZERO) == -1) {
            out4 = out4.add(n);
        }

        BigDecimal out5 =
((m1.multiply(p_a)).add(m2.multiply(radix2)).add(m3.multiply(rad
ix3))).remainder(n);

        if (out5.compareTo(BigDecimal.ZERO) == -1) {
            out5 = out5.add(n);
        }

        BigDecimal out6 =
((m1.multiply(p_a)).add(m2.multiply(radix2)).add(m3.multiply(p_z
))).remainder(n);

        if (out6.compareTo(BigDecimal.ZERO) == -1) {
            out6 = out6.add(n);
        }

        BigDecimal out7 =
((m1.multiply(p_a)).add(m2.multiply(p_b)).add(m3.multiply(radix3
))).remainder(n);
```

```
        if (out7.compareTo(BigDecimal.ZERO) == -1) {
            out7 = out7.add(n);
        }

        BigDecimal out8 =
((m1.multiply(p_a)).add(m2.multiply(p_b)).add(m3.multiply(p_z)))
.remainder(n);

        if (out8.compareTo(BigDecimal.ZERO) == -1) {
            out8 = out8.add(n);
        }

        System.out.println("out1=" + out1);
        System.out.println("out2=" + out2);
        System.out.println("out3=" + out3);
        System.out.println("out4=" + out4);
        System.out.println("out5=" + out5);
        System.out.println("out6=" + out6);
        System.out.println("out7=" + out7);
        System.out.println("out8=" + out8);

        return new BigDecimal[]{out1, out2, out3, out4, out5,
out6, out7, out8};
    }

}

//код програми для перевірки роботи системи
import java.io.UnsupportedEncodingException;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.nio.charset.Charset;

public class RabinTestDecimal {
    public static void main(String[] args) throws
UnsupportedEncodingException {
        int counter = 0;
        int trials = 1;
        for(int i=0;i<trials;i++) {
            BigDecimal[] key = RabinDecimal.genKey(512);
            BigDecimal n = key[0];
            BigDecimal p = key[1];
            BigDecimal q = key[2];
            BigDecimal z = key[3];
            System.out.println("ss");
            String s = "c";
            BigInteger word = new
BigInteger(s.getBytes(Charset.forName("ascii")));
            BigDecimal crypted =
RabinDecimal.encrypt(BigDecimal.valueOf(word.longValue()), n);

            System.out.println("N="+n);
```

```
            System.out.println("m="+word);
            System.out.println("p="+p);
            System.out.println("q="+q);
            System.out.println("z="+z);
            System.out.println("crypted="+crypted);

            boolean worked = false;
            BigDecimal[] m2 = RabinDecimal.decrypt(crypted, p,
q, z);

            for(BigDecimal b:m2) {
                String dec = new
String(b.toBigInteger().toByteArray(),
Charset.forName("ascii"));
                if(dec.equals(s)) {
                    worked = true;
                    System.out.println("N="+n);
                    System.out.println("m="+word);
                    System.out.println("p="+p);
                    System.out.println("q="+q);
                    System.out.println("z="+z);
                }
            }
            if(worked)counter++;
        }

        System.out.println("worked "+counter+"/"+trials+"
times");
    }
}


//код програмно-апаратної реалізації для плати MAX II
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_textio.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_signed.all;
use ieee.numeric_std.all;
use ieee.numeric_bit.all;
use IEEE.std_logic_arith.all;
use IEEE.MATH_REAL.ALL;
use IEEE.MATH_complex.ALL;

entity comp is

     port (
     clk : in std_logic;
     v1,v2,v3,v4,v5,v6,v7,v8 : out real
     );

end comp;
```

```
architecture arch of comp is
     signal p1: real ;
     signal p2: real ;
     signal p3: real ;
    signal n: real :=0.0 ;
     signal w: real :=0.0 ;
     signal word: real ;
     signal sqrt1: real :=0.0 ;
     signal sqrt2: real :=0.0 ;
     signal sqrt3: real :=0.0 ;
     signal g1: real  :=1.0;
     signal g2: real  :=1.0;
     signal g3: real  :=1.0;
     signal p1_g1: real :=0.0 ;
     signal p2_g2: real :=0.0 ;
     signal p3_g3: real :=0.0 ;
     signal k1: real :=0.0 ;
     signal k2: real :=0.0 ;
     signal k3: real :=0.0 ;
     signal m1: real  ;
     signal m2: real  ;
     signal m3: real ;

     begin
          read: process(clk)
               begin
                    if clk'event and clk='1' then
                         n<=p1*p2*p3;

                         w<=(word**2.0) mod n ;

                         sqrt1<=w mod p1;
                         sqrt2<=w mod p2;
                         sqrt3<=w mod p3;

                               if  sqrt(sqrt1) mod 1.0/= 0.0 then
                         while sqrt(sqrt1+k1*p1) mod 1.0 /= 0.0
                         loop
                              k1<=k1+1.0;
                              exit when sqrt(sqrt1+k1*p1) mod 1.0
< 1.0;
                         end loop;
                         g1<=sqrt(sqrt1+k1*p1) mod p1;
                       else g1<=sqrt(sqrt1)mod p1;
                       end if;

                       if  sqrt(sqrt2) mod 1.0/= 0.0 then
                         while sqrt(sqrt2+k2*p2) mod 1.0 /= 0.0
                         loop
                              k2<=k2+1.0;
                              exit when sqrt(sqrt2+k2*p2) mod 1.0
< 1.0;
                         end loop;
```

```
                        g2<=sqrt(sqrt2+k2*p2) mod p2;
                      else g2<=sqrt(sqrt2)mod p2;
                      end if;



                     if  sqrt(sqrt3) mod 1.0/= 0.0 then
                        while sqrt(sqrt3+k3*p3) mod 1.0 /= 0.0
                        loop
                             k3<=k3+1.0;
                             exit when sqrt(sqrt3+k3*p3) mod 1.0
< 1.0;
                        end loop;
                        g3<=sqrt(sqrt3+k3*p3) mod p3;
                      else g3<=sqrt(sqrt3)mod p3;

                      end if;

                        p1_g1<= p1-g1;
                        p2_g2<= p2-g2;
                        p3_g3<= p3-g3;

                        m1<=n/p1;
                        m2<=n/p2;
                        m3<=n/p3;

                        v1<=(-m1*g1+m2*g2+m3*g3) mod n;

                        v2<=(-m1*g1+m2*g2+m3*p3_g3) mod n;

                        v3<=(-m1*g1+m2*p2_g2+m3*g3) mod n;

                        v4<=(-m1*g1+m2*p2_g2+m3*p3_g3) mod n;

                        v5<=(-m1*p1_g1+m2*g2+m3*g3) mod n;

                        v6<=(-m1*p1_g1+m2*g2+m3*p3_g3) mod n;

                        v7<=(-m1*p1_g1+m2*p2_g2+m3*g3) mod n;

                        v8<=(-m1*p1_g1+m2*p2_g2+m3*p3_g3) mod n;

                    end if;
          end process;
end arch;
```

Лістинг коду програмної реалізації модулярного експоненціювання

```
from task.task_generator import TaskGenerator
```

```
    from calculation.rns_calculator import RnsCalculator
    from       calculation.power_decrease_cubic       import
PowerDecreaseCubicCalculation
    from           calculation.power_simple           import
SimplePowerCalculation

    from rns.rns import RnsNumberFactory
    from          calculation.power_decrease          import
PowerDecreaseCalculation
    from  calculation.rns_power_decrease_calculator  import
RnsPowerDecreaseCalculator
    from   calculation.rns_parallel_power_decrease   import
RnsParallelPowerDecreaseCalculator
    from rns.mpf_rns import MpfRnsNumberFactory
    from table_printer import TablePrinter
    import test_bench

    if __name__ == '__main__':
        calculations = [
            ('PWR-RNS',          RnsPowerDecreaseCalculator,
MpfRnsNumberFactory),
            ('SMP-RNS',          RnsPowerDecreaseCalculator,
RnsNumberFactory),
            ('DEC-CUB',       PowerDecreaseCubicCalculation,
None),
            ('PWR-DEC', PowerDecreaseCalculation, None)
        ]

        bits_count = (8, 16, 32, 64 , 128, 256, 512 , 1024,
2048, 4096)

        tasks_count = 1

        input_deltas = [0 , 10, 30, 50, 80]  # in %

        def _create_table_printer():
            rows  =  [calculation[0]  for  calculation  in
calculations]
            columns = [str(count) for count in bits_count]
            return TablePrinter(rows, columns, 7)


        def _run_tests_for_delta(input_delta):
            result_table = _create_table_printer()

            for bits in bits_count:
                tasks         =         TaskGenerator(bits,
input_delta).generate(tasks_count)

                # tasks[0].print(bits)

                for calculation_entry in calculations:
                    name,                calculation_class,
```

```
number_factory_class = calculation_entry

                    calculation                                 =
calculation_class(number_factory_class, bits)

                    time,              result                   =
test_bench.run_test(tasks, calculation)
                    result_table.set_data(name,   str(bits),
str(time))

            result_table.print()


        print('Starting research \nTasks per case count: %d
\nTime in microseconds' % tasks_count)
        for delta in input_deltas:
            print('\tDELTA = %d' % delta)
            _run_tests_for_delta(delta)

    from datetime import datetime
    from task.task import Task


    def run_test(tasks, calculator):
        result = None

        t1 = datetime.now()
        for task in tasks:
            result = calculator.calculate(task)
        t2 = datetime.now()

        execution_time = (t2 - t1).microseconds
        return execution_time, result


    class PowerDecreaseCalculation:
        def     __init__(self,     number_factory_class=None,
bits_count=8):
            pass

        def calculate(self, task):
            a = task.a
            power = task.x
            add_mul = 1

            while power > 1:
                if power % 2 == 0:
                    power //= 2
                else:
                    power = (power - 1) // 2
                    add_mul *= a
                    if add_mul > task.p:
                        add_mul %= task.p
```

```
                a = a ** 2 % task.p
            return (a * add_mul) % task.p


    class RnsCalculator:
        def __init__(self, rns_class_factory, bits_count):
            self.rns_factory                                    =
rns_class_factory(bits_count)

        def calculate(self, task):
            a = self.rns_factory.create_number(task.a)
            b = self.rns_factory.create_number(task.a)

            # calculate a^x in rns
            for i in range(task.x):
                a.multiply(b)

            return a.get_number() % task.p

    from .power_decrease import PowerDecreaseCalculation
    from task.task import Task


    class RnsPowerDecreaseCalculator:
        def __init__(self, rns_class_factory, bits_count):
            self.rns_factory                                    =
rns_class_factory(bits_count)
            self.power_decrease                                 =
PowerDecreaseCalculation()

        def calculate(self, task):
            a = self.rns_factory.create_number(task.a)

            for i in range(len(a.number)):
                sub_task    =    Task(a.number[i],    task.x,
a.moduli_generator.moduli[i])
                a.number[i]                                     =
self.power_decrease.calculate(sub_task)

            return a.get_number() % task.p


    class MpfRnsModuliGenerator:
        def __init__(self, bits):
            k = 2 ** (int((bits - 2) / 3.0) + 1) + 1
            p1 = k + 1
            self.moduli = (p1, 2 * p1 - 1, 2 * p1 + 1)

            self.multiply = 1
            for m in self.moduli:
                self.multiply *= m

            self.base_numbers = []
```

```
            for m in self.moduli:
                base_number = self.multiply // m
                if base_number % m != 1:
                    base_number *= -1
                self.base_numbers.append(base_number)


    class MpfRns:
        def __init__(self, number, moduli_generator):
            self.moduli_generator = moduli_generator
            # convert number to rns
            self.number = [number % m for m in
moduli_generator.moduli]

        def get_number(self):
            number = 0
            for i, ni in enumerate(self.number):
                number += ni *
self.moduli_generator.base_numbers[i]
            return number % self.moduli_generator.multiply

        def multiply(self, argument):
            for i, ni in enumerate(self.number):
                self.number[i] = (self.number[i] *
argument.number[i]) % self.moduli_generator.moduli[i]
            return self

        def partial_multiply(self, b, i):
            self.number[i] = (self.number[i] * b) %
self.moduli_generator.moduli[i]
            return self.number[i]


    class MpfRnsNumberFactory:
        def __init__(self, initial_bits=8):
            self._bits_count = initial_bits
            self._moduli_generator =
MpfRnsModuliGenerator(initial_bits)

        def create_number(self, number):
            return MpfRns(number, self._moduli_generator)


    class Rns:
        def __init__(self, number, moduli_generator):
            self.moduli_generator = moduli_generator
            self.number = [number % m for m in
self.moduli_generator.moduli]

        def get_number(self):
            n = 0
            for p, ni in zip(self.moduli_generator.moduli,
self.number):
```

```
                m_i = self.moduli_generator.multiply // p
                #  print('%d^-1  mod  %d  =  %d'  %  (m_i,  p,
rec))
                n += ni * m_i * self._reciprocal(m_i, p)
            return n % self.moduli_generator.multiply

        def multiply(self, argument):
            for                    i                    in
range(len(self.moduli_generator.moduli)):
                self.partial_multiply(argument, i)
            return self

        def partial_multiply(self, b, i):
            self.number[i]   =   (self.number[i]   *   b)   %
self.moduli_generator.moduli[i]

        def egcd(self, a, b):
            x, y, u, v = 0, 1, 1, 0
            while a != 0:
                q, r = b // a, b % a
                m, n = x - u * q, y - v * q
                b, a, x, y, u, v = a, r, u, v, m, n
            gcd = b
            return gcd, x, y

        def _reciprocal(self, a, m):
            gcd, x, y = self.egcd(a, m)
            if gcd != 1:
                return  None   #  modular  inverse  does  not
exist
            else:
                return x % m


    class RnsModuliGenerator:
        def __init__(self, bits=8):
            k = 2 ** int((bits + 0.2) / 3.0 + 1)
            if k % 2 == 0:
                p2 = k + 1
            else:
                p2 = k
            c = 2 ** (bits // 4)
            self.moduli = (p2 - c, p2, p2 + c)

            self.multiply = 1
            for m in self.moduli:
                self.multiply *= m
            return


    class RnsNumberFactory:
        def __init__(self, initial_bits=8):
            self._bits_count = initial_bits
```

```
            self._moduli_generator                          =
RnsModuliGenerator(initial_bits)

        def create_number(self, number):
            return Rns(number, self._moduli_generator)


    class Task:
        def __init__(self, a, x, p):
            self.a = a
            self.x = x
            self.p = p

        def __str__(self):
            return "(a=%d, x=%d, p=%d) " % (self.a, self.x,
self.p)

        def __unicode__(self):
            return "(a=%d, x=%d, p=%d) " % (self.a, self.x,
self.p)

        def print(self, bits_count):
            print("size:  %d  \na=%d  \nx=%d  \np=%d  \n"  %
(bits_count, self.a, self.x, self.p))

    from random import random
    from .task import Task
    from rns.mpf_rns import MpfRnsModuliGenerator


    class TaskGenerator:
        def __init__(self, bits_count=8, delta=0):
            self.bits_count = bits_count
            self.delta = delta
            self.moduli_generator                           =
MpfRnsModuliGenerator(bits_count)

        def random_number(self, bits_count):
            max_number = (1 << bits_count) - 1
            delta_diff  =  round(random()  *  round(bits_count
* self.delta / 100))
            # delta_diff = round(bits_count * self.delta /
100)
            return max_number - delta_diff

        def generate(self, task_count=10):
            # generate a, x and p: (a^x mod p);
            return [Task(
                self.random_number(self.bits_count - 3),
                self.random_number(self.bits_count),
                self.moduli_generator.multiply
            ) for i in range(task_count)]
```

Лістинг коду програмної реалізації модулярного експоненціювання

```
#include <iostream>
#include "time.h"
#include <windows.h>
#include <math.h>
#include <fstream>
#include <lip.h>

using namespace std;
const int thread = 8;
ofstream fout;
ifstream fin;

//метод підбору
long long int MP(long long int x, long long int y)
{
  int rez = 1;
  while ((rez*x-1)%y != 0)
  {
       rez++;
  }
  return rez;
}

//метод №1
long long int M1(long long int x, long long int y)
{

  __int64 start, end, tps;
  int step = x*y/thread;

  QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
  QueryPerformanceCounter((LARGE_INTEGER *)&start);

  long long int y1 = y+1;

  for (y1; y1<=x*y; y1+=y)
  {
       if (y1 >= step)
       {
            step *= 2;
            QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
            QueryPerformanceCounter((LARGE_INTEGER *)&start);
       }
       if (y1%x == 0.)
       {
```

```
                y1/=x;
                QueryPerformanceCounter((LARGE_INTEGER *)&end);
                break;
           }
      }

      cout << "M1: rez = " << y1 << endl;
      fout << "|M1\t" << y1;
      fout << '\t' << ((double)(end - start) / tps) * 1000. << "
ms" << "\t|";

      cout << "time: " << fixed << ((double)(end - start) / tps) *
1000./8. << " ms" << endl << endl;

      return 0;
   }

   long long int M2_mod(long long int x, long long int y)
   {
     __int64 start, end, tps;

     QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
     QueryPerformanceCounter((LARGE_INTEGER *)&start);

     long long int k=1, z=(y-x)+1;
     int i;
     double step = ceil(x/thread+0.5);

     for (i=1; i<=x; i++)
     {
          if (z <= 0)
          {
               break;
          }

          if (i == step)
          {
               step*=2;
               QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
               QueryPerformanceCounter((LARGE_INTEGER *)&start);
               k = i;
               z = ((k*y)+1)%x;
          }

          if (z >= x)
          {
               z %= x;
          }
          else
          {
               z += y-x;
               k++;
          }
```

```
    }

    z = ((k*y)+1)/x;

    QueryPerformanceCounter((LARGE_INTEGER *)&end);

    fout << "|M2\t" << z;
    fout << '\t' << ((double)(end - start) / tps) * 1000. << "
ms" << "\n";
    cout << "M2: rez = " << z << endl;
    cout << "time: " << fixed << ((double)(end - start) / tps) *
1000./8. << " ms" << endl << endl;

    return z;
  }

  //метод Евкліда
  long long int MEV(long long int a, long long int n)
  {
    long long int b=n, x=0, d=1, q, y;
    int k=0;
    while (a > 0)
    {
        q=b/a;
        y=a;
        a=b%a;
        b=y;
        y=d;
        d=x-q*d;
        x=y;
        k++;
    }
    x=x%n;
    if (x < 0)
    {
        x = (x+n)%n;
    }
      return x;
  }

  int main()
  {
    cout.precision(8);
    long long int x,y;
    __int64 start, end, tps;
    int i=0;

    fin.open("in.txt");
    fout.open("out.txt");

    fout.precision(5);

  x = 0;
```

```
   y = 65537;
   //while (fin >> x >> y)
   while (x <= y-1000)
   {
     x+=1000;

     cout << "Number " << x << ": ->" << endl;
     fout << x << '\t' << y << '\t';
   //MP
     QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
     QueryPerformanceCounter((LARGE_INTEGER *)&start);

     cout << "#MP: rez = " << MP(x,y) << endl;
     fout << "|MP\t" << MP(x,y);

     QueryPerformanceCounter((LARGE_INTEGER *)&end);
     cout << "time: " << ((double)(end - start) / tps) * 1000. <<
" ms" << endl << endl;
     fout << '\t' << ((double)(end - start) / tps) * 1000. << "
ms" << "\t|";


   //алгоритм Евкліда
     QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
     QueryPerformanceCounter((LARGE_INTEGER *)&start);

     cout << "MEV: rez = " << MEV(x,y) << endl;
     fout << "|Euclid\t" << MEV(x,y);

     QueryPerformanceCounter((LARGE_INTEGER *)&end);
     cout << "time: " << ((double)(end - start) / tps) * 1000. <<
" ms" << endl << endl;
     fout << '\t' << ((double)(end - start) / tps) * 1000. <<
"\t";


   //M1
     M1(x,y);

   //метод №2
     M2_mod(x,y);


   /*алгоритм Ейлера
     QueryPerformanceFrequency((LARGE_INTEGER *)&tps);
     QueryPerformanceCounter((LARGE_INTEGER *)&start);

     cout << "ME: rez = " << ME(x,y) << endl;
   //     fout << "ME = " << ME(x,y);

     QueryPerformanceCounter((LARGE_INTEGER *)&end);
     cout << "time: " << ((double)(end - start) / tps) * 1000. <<
" ms" << endl << endl;
```

```
   //     fout << '\t' << ((double)(end - start) / tps) * 1000.
<< " ms" << "\t|";
   */
   }

     fin.close();
     fout.close();
     system("pause");
     return 0;
   }
```

Програмний код для апаратної реалізації пошуку оберненого елемента

```
   //метод додавання залишку від ділення
   library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_unsigned.all;
   use IEEE.MATH_real.ALL;

   entity comp is
   port (
   clk : in std_logic
        );

   end comp;

   architecture arch of comp is
   signal power_1 : INTEGER;
   signal exite : INTEGER;
   signal modulo : INTEGER :=0;
   signal internal : INTEGER :=0;
   signal d : INTEGER :=0;
   signal c: INTEGER;
     begin
        read: process(clk)
             begin
                  if clk'event and clk='1' then
                          while internal<1 loop
                            d<=modulo mod power_1;
                            c<=d+1;
                            internal<=1;
                            exit when internal<1;
                          end loop;
```

```
                        while c>0 loop
                                c<=(d+c) mod power_1;

  exite<=((internal*modulo)+1)/power_1;
                                internal<=internal+1;
                                exit when c>0;
                        end loop;

             end if;
  end process;
end arch;
//метод додавання модуля до числа
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.MATH_real.ALL;

entity comp is
port (
  clk : in std_logic
     );

end comp;

architecture arch of comp is
signal power_1 : INTEGER;
signal exite : INTEGER;
signal modulo : INTEGER :=0;
signal internal : INTEGER :=0;
  begin
     read: process(clk)
          begin
               if clk'event and clk='1' then

               if modulo mod power_1 /= 0 then
               while (modulo*internal + 1) mod power_1 /=0
                    loop

                         internal<=internal+1;
                         exit when internal+1>internal;

                    end loop;
               else exite <= 1;
               end if;
```

```
                exite<=(modulo*internal+1)/power_1;
                end if;
    end process;
  end arch;

  //розширений алгоритм евкліда для знаходження оберненого
елементу за модулем
  library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_unsigned.all;
  use IEEE.MATH_REAL.ALL;

  entity comp is

    port (
       clk : in std_logic
       );

  end comp;

  architecture arch of comp is
    signal modulo: INTEGER ;
    signal power_1: INTEGER  ;
    signal exite: INTEGER :=0 ;
  begin
    read: process(clk)
    begin
       if clk'event and clk='1' then
            if (power_1-1) mod modulo /= 0 then
                while (exite*power_1 - 1) mod modulo /=0
                     loop
                     exite<=exite+1;
                     exit when exite+1>exite;
                end loop;
            else exite<= 1;
            end if;
       end if;
    end process;
  end arch;
```