МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Момотюк Олег Володимирович

Програмна підсистема пошуку оберненого елементу за модулем для асиметричних криптосистем/ The software search subsystem of a reverse element by module for asymmetric cryptosystem

спеціальність: 6.050102 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерні системи та мережі

Випускна кваліфікаційна робота

Виконав: студент групи КСМ-42/1
Момотюк Олег Володимирович

_____

Науковий Керівник
к.т.н. Касянчук М.М.

_____

Тернопіль – 2019

РЕЗЮМЕ

Бакалаврська робота містить 55 сторінок пояснюючої записки, 18 рисунки, 20 таблиць, 3 додатки. Обсяг графічного матеріалу – 3 аркуші формату А3.

Метою дипломної роботи є розробка програмної підсистеми пошуку оберненого елементу за модулем для асиметричних криптосистем.

Методи досліджень – методи програмування, методи захисту інформації.

У дипломній роботі розроблено програмну підсистему пошуку оберненого елементу за модулем для асиметричних криптосистем. На основі аналізу криптографічних методів шифрування встановлено їх найбільш поширені арифметичні операції, визначено основні переваги та недоліки їх реалізації.. На основі векторно-модульного методу розроблено алгоритм пошуку залишку. На основі формування вимог до реалізації операцій модулярного множення та експоненціювання розроблено алгоритмічне забезпечення. На основі розробленого алгоритмічного забезпечення побудовано UML Use-case діаграму програмної підсистеми. На основі UML Use-case діаграм розроблено програмну підсистему модулярного множення та експоненціювання, тестування якого підтвердило збільшення швидкодії виконання вказаних операцій у порівнянні з класичними алгоритмами в асиметричних криптосистемах.

Ключові слова: ПРОГРАМНА ПІДСИСТЕМА, ЗАЛИШОК, МОДУЛЯРНЕ МНОЖЕННЯ, МОДУЛЯРНЕ ЕКСПОНЕНЦІЮВАННЯ, UML USE-CASE ДІАГРАМА.

RESUME

The Bachelor paper contains 86 pages of explanatory notes, 18 drawings, 20 tables, 3 appendices. The volume of graphic material - 3 sheets of A3 format.

The aim of the thesis is to develop a software subsystem for finding the inverse element by module for asymmetric cryptosystems.

Methods of research - methods of programming, methods of information security, methods of graph theory.

In the thesis the software sub-system of modular exponentiation for asymmetric cryptosystems has been developed. Based on the analysis of cryptographic encryption methods, their most common arithmetic operations were determined, the main advantages and disadvantages of their implementation were determined. On the basis of the vector-modular method, an algorithm for finding the remainder was developed. Based on the formation of requirements for the implementation of operations of modular multiplication and exponentiation, algorithmic support is developed. Based on the developed algorithmic support, the UML Use-case diagram of the software subsystem was built. On the basis of the UML Use-case diagrams, a software subsystem of modular multiplication and exponentiation was developed, the testing of which confirmed the increase in the performance of these operations compared with the classical algorithms in asymmetric cryptosystems.

Keywords SOFTWARE SUBMISSION, SCALE, MODULAR MANNING, MODULAR EXPONENTATION, UML USE-CASE DIAGRAM.

ЗМІСТ

ВСТУП

Сучасні системи захисту інформаційних потоків забезпечують необхідний рівень стійкості до різних типів атак. Їх функціювання здійснюється в реальному масштабі часу [1]. Це обумовлює потребу в задоволені програмними бібліотеками криптографічних перетворень [2] високої швидкодії базових операцій [3-4] в асиметричних системах захисту на різних етапах їх роботи, таких як генерування ключів, шифрування і дешифрування. В цьому плані найбільш трудомісткою і поширеною операцією є пошук обернених елементів у кільці лишків по модулю [5].

Вказана операція характеризується своїм частим використанням при виконанні множення точки еліптичної кривої на число у афінних координатах над полем GF(p) [6], в протоколі Діффі-Хелмана [4], криптографічних алгоритмах RSA, схемі Ель-Гамаля [7], криптосистемі Рабіна [8], кодуванні даних [12], а також системі залишкових класів [9-11]. До того ж пошук обернених елементів у кільці лишків по модулю допомагає в підвищенні ефективності роботи безпровідних сенсорних мереж [13] і оптимізації результатів обчислень в методах виявлення найбільшого спільного дільника. Описана операція використовується при модулярному множенні [3], експоненціюванні [4] та факторизації [14]. Саме тому дослідження існуючих і розробка нових методів пошуку мультиплікативно оберненого елемента у кільці лишків по модулі є на сьогодні актуальною задачею.

Для вирішення поставленого завдання можуть використовуватись не тільки алгоритмічні (математичні), але й програмно-апаратні методи оптимізації. При цьому отримання найкращих результатів відбувається при паралельному застосовувані обох цих підходів.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОШУКУ ОБЕРНЕНОГО ЕЛЕМЕНТА ЗА МОДУЛЕМ

1.1 Огляд симетричних і асиметричних криптосистем

Алгоритм – це набір інструкцій, в яких описується порядок дій виконавця, направлених на досягнення результату розв’язування задачі за певну скінченну кількість дій. Під цим терміном розуміється система правил виконання дискретного процесу, завдяки якій можна досягти поставленої цілі за визначений період часу.

В комп’ютерних науках алгоритм – це перелік деталізованих інструкцій, які допомагають реалізувати процес обчислення. Починаючи з початкового стану, він відбувається за допомогою певних послідовних станів, закінчуючись кінцевим станом.

Використання алгоритмів шифрування і дешифрування відбувається в різних сферах комп’ютерної техніки. Такі деталізовані інструкції є основою функціонування систем захисту конфіденційної і комерційної інформації від потрапляння її до третіх осіб. Основний принцип алгоритмів шифрування і дешифрування полягає в тому, що одержувач повідомлення наперед знає алгоритм шифрування, маючи при цьому відповідний ключ. Без останнього отримана інформація виглядає, як набір символів, в якому відсутній будь-який сенс.

Криптографія – це окрема галузь знань, що займається вивченням тайнопису і способів його розкриття (криптоаналізу) [15].

Криптосистема – це завершена комплексна модель, яка може здійснювати двосторонні криптографічні перетворення над даними різного обсягу, підтверджуючи при цьому час відправки повідомлення. Ця схема, володіючи системою транспортного кодування, використовує механізми перетворення паролів і ключів (рисунок 1.1).

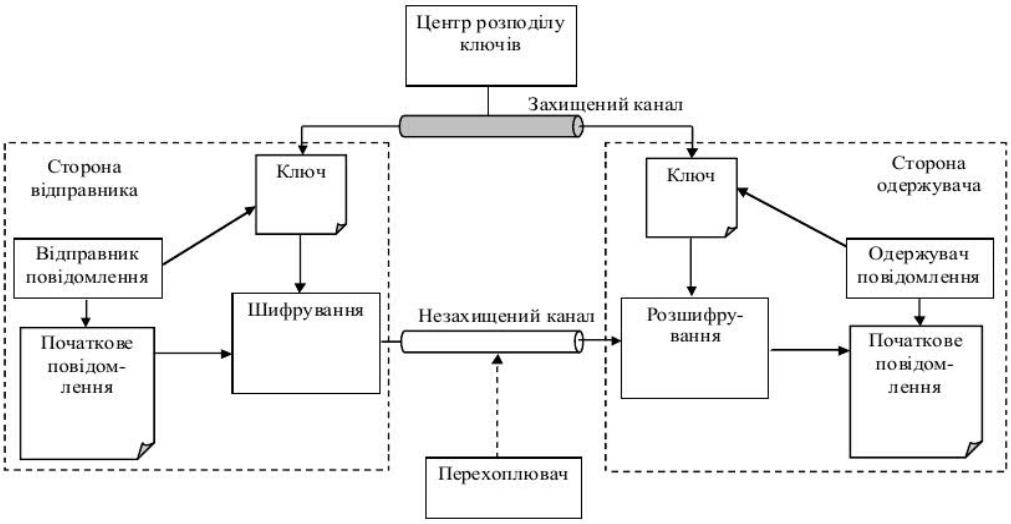


Рисунок 1.1 – Принципова схема формування та передачі даних

Загалом на сьогоднішній день існує два основних типи криптосистем, основаних на методах, в яких процеси шифрування і дешифрування здійснюються з використанням симетричних або асиметричних ключів шифрування [16].

Симетричні криптосистеми – це методи шифрування, в процесі виконання яких під час шифрування і дешифрування використовуються один і той же самий криптографічний ключ. Він зберігається в секреті як отримувачем повідомлення, так і відправником. Симетричні криптосистеми були єдиним способом шифрування довгий час до розробки асиметричних систем.

Симетричні криптоалгоритми функціонують за допомогою перетворень невеликих по своєму обсягу блоків даних. Ця система захисту дозволяє прочитати повідомлення тільки в тому випадку, коли користувач знає відповідний секретний ключ шифрування (рисунок 1.2).

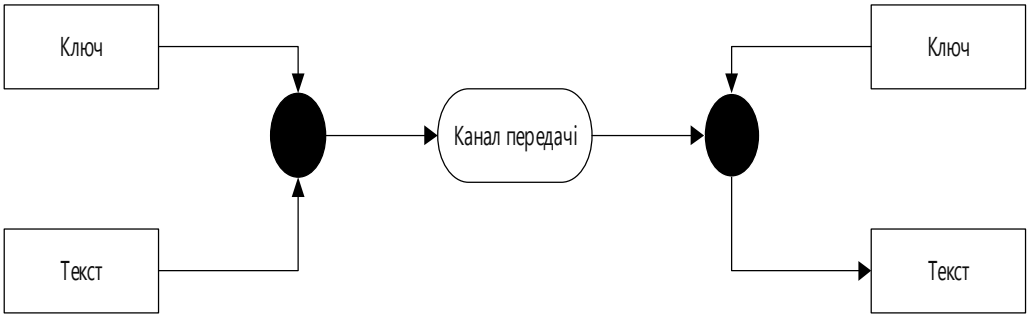


Рисунок 1.2 – Принцип обробки інформації симетричним алгоритмом шифрування

Перші розробки асиметричних шифрувань можна знайти в роботі «Нові напрямки в сучасній криптографії», написаній Уітфілдом Діффі разом з Мартіном Хелманом. Описання функціонування цієї системи було здійснено в 1976 році. Воно відбулося під впливом роботи Ральфа Меркле, який дещо раніше запропонував способи використання відкритих ключів [17]. Саме на основі цієї теорії американські вчені продумали метод отримання секретних ключів, використовуючи при цьому відкриті канали даних. Цей спосіб експоненціального обміну секретного ключа на сьогодні більш відомий, як обмін ключами Діффі-Хелмана. Він став першим опублікованим методом, призначеним для встановлення поділу секретного ключа між попередньо завіреними користувачами одного каналу. Уже в 2002 році Хелман визнав вагомий внесок Ральфа Меркле в розробку даної методики, запропонувавши називати її алгоритмом «Діффі-Хелмана-Меркле».

У 1977 році Рональд Райвест, Аді Шамір і Леонард Адлеман розробили алгоритм шифрування, в основі якого була проблема розкладання на множники. Нова на той час система RSA була названа в честь цих американські вчених Массачусестського Технічного Інституту. Вона стала першим алгоритмом, який міг використовуватись як для шифрування, так і для цифрового підпису.

RSA – аббревіатура, створена першими буквами прізвищ розробників данного метода. Ця система по своїй суті являється одним з криптографічних алгоритмів, що мають відкритий ключ. Він базується на обчислювальній

складності задачі факторизації великих цілих чисел. Особливості цього алгоритму дозволяють його на сьогодні застосовувати до великої кількості криптографічних застосунків.

Основні етапи алгоритму RSA такі:

- генерація ключів;
- шифрування повідомлення;
- розшифрування повідомлення;
- розповсюдження ключів.

Безпека використання алгоритму RSA напряму основана на складності здійснення факторизації великих цілих чисел. Система при своєму функціонуванні використовує два ключі – відкритий і закритий. Перший з них не потребує захисту. Його основна роль – шифрування даних. Якщо повідомлення було з його допомогою зашифровано, дешифрування в результаті можна здійснити тільки з використання секретного ключа.

Відкритий ключ разом з відповідним йому секретним ключом утворюють пару ключів.

Схема Ель-Гамаля – один з асиметричних алгоритмів шифрування, оснований на складності обчислення дискретних логарифмів в кінцевому полі. Ця криптографічна система включає як алгоритм шифрування, так і алгоритм цифрового підпису. Метод Ель-Гамаля певний час був основою стандартів цифрового електронного підпису в Сполучених Штатах Америки [18].

Описана схема була створена Тахером Ель-Гамалем в 1984 році. Цей американський криптограф єгипетського походження розробив один із варіантів алгоритму Діффі-Хелмана. Для цього була дещо удосконалена запропонована ними системам. Завдяки цьому вдалось отримати два різних алгоритмі, призначених для шифрування і аутентифікації. Тахер Ель-Гамаль не отримав патент на свою розробку. Через це запропонована ним система стала більш дешевим криптографічним методом, який не потребує сплати ліцензійних внесків за його використання. На сьогодні вважається, що даний спосіб шифрування підпадає під патент Діффі-Хелмана.

Таким чином, криптографи Уітфілд Діффі та Мартін Хелман – розробники концепції вирішення проблем керування відкритими ключами шифрування даних. Криптографія з асиметричним методом – це схема, в якій застосовуються пари ключів: відкритий (public key) і закритий (private key) [19]. Перший призначений для шифрування даних, другий – для їх дешифрування. Користувач, що має копію відкритого ключа, завжди може зашифрувати необхідну інформацію.

Незважаючи на те, що пара ключів математично зв'язана між собою, обчислення секретного ключа з відкритого практично здійснити неможливо. Користувач, який має відкритий ключ, може тільки зашифрувати інформацію. Розшифрувати дані може тільки та людина, що володіє секретним ключем.

Таким чином, особливості функціонування криптографічних систем з відкритими ключами зробили їх ефективними та надійними способами криптографічного захисту даних. Для різних дій в них застосовується пара ключів – відкритий і закритий. Звідси і назва систем – асиметричні.

Перший ключ симетричних криптосистем є відкритим. Він може бути опублікований для користувачів, яким необхідно зашифровувати свої особисті дані [20]. Розшифрування цієї інформації за допомогою доступного їм ключа неможливе (рисунок 1.3).

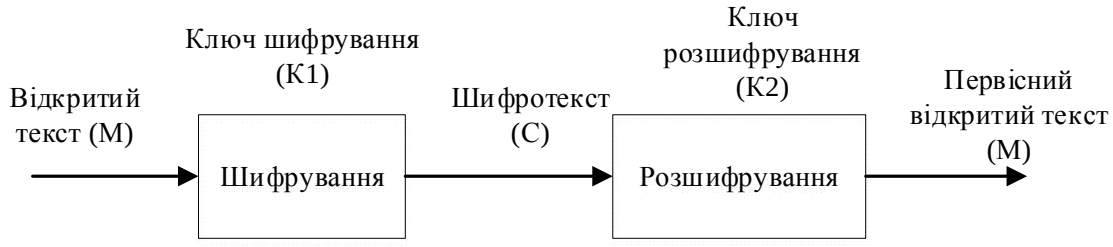


Рисунок 1.3 – Асиметричне шифрування інформації

Для того щоб розшифрувати отримані дані, потрібно використовувати другий – секретний – ключ. Він математично не може бути визначений на практиці з відкритого ключа шифрування.

1.2 Методи пошуку оберненого елемента за модулем

Обернений елемент за модулем для ненульового елемента *a*, що відноситься до простого поля *GF(p)* – це розв'язок рівняння $(a \cdot b) \bmod p = 1$. Воно задане в полі натуральних чисел $N\{+,\cdot\}$. При цьому розв'язок $a^{-1} \bmod p = b$ також є одним з елементів простого поля GF(p) [21].

Найбільш простим в виконанні способом пошуку оберненого елемента $a^{-1} \bmod p = b$ є перебір всіх елементів простого поля *GF(p)*, що не дорівнюють нулю. При цьому виконується їх перевірка згідно з поставленої умови $(a \cdot b) \bmod p = 1$. Цей спосіб відрізняється високою обчислювальною складністю.

Набагато частіше використовується методика пошуку мультиплікативного оберненого елемента за модулем, в основі якої лежить розширений алгоритм Евкліда [22]. Для того щоб скористатись цим способом, спочатку записується прямий алгоритм Евкліда. В відповідності з ним при $n>a=r_0$, де n і a –цілі числа, виконується наступна оперативна послідовність:

$$
\begin{aligned}
& n = r_0 \cdot q_1 + r_1, \;\; q_1 = a, \;\; 0 \le r_1 < r_0; \\
& r_0 = r_1 \cdot q_2 + r_2, \;\; 0 \le r_2 < r_1; \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& r_{n-3} = r_{n-2} \cdot q_{n-1} + r_{n-1}, \;\; 0 \le r_{n-1} < r_{n-2}; \\
& r_{n-2} = r_{n-1} \cdot q_n + r_n, \;\; 0 \le r_n < r_{n-1}; \\
& r_{n-1} = r_n \cdot q_{n+1} + 0.
\end{aligned}
\qquad (1.1)
$$

Через те, що цілі числа a і n є взаємно простими, r_n=1. При використання розширеного алгоритму Евкліда застосовується проходження прямого алгоритму у зворотному порядку:

$$1 = r_n = r_{n-2} - q_n r_{n-1} = r_{n-2} - q_n\left(r_{n-3} - q_{n-1} r_{n-2}\right) = r_{n-2} - q_n r_{n-3} + q_n q_{n-1} r_{n-2} = \\ -q_n r_{n-3} + \left(1 + q_n q_{n-1}\right) r_{n-2} = -q_n r_{n-3} + \left(1 + q_n q_{n-1}\right)\left(r_{n-4} - q_{n-2} r_{n-3}\right) = \ldots \quad (1.2)$$

Процедура пошуку продовжується до того часу, поки не вдасться отримати вираз $v \cdot n + t \cdot r_0 = 1$, де величина $b = t \bmod n = a^{-1} \bmod n$ і буде шуканим оберненим елементом. Таблиця 1 ілюструє процесс пошуку оберненого елемента за модулем з використання розширеного алгоритму Евкліда.

Таблиця 1.1 – Пошук оберненого елемента $41^{-1} \bmod 157$ на основі 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; $41^{-1} \bmod 157 = 23$

Описана методика відрізняється необхідністю здійснення великої кількості операцій, таких як ділення з остачею, перемноження і підстановки. Незважаючи на це, даний способі характеризується набагато меншою складністю в порівнянні з альтернативними.

Для того щоб знайти обернений елемент за модулем, можна використати теорему Ейлера ($a^{\varphi(p)} \equiv 1 \bmod p$). Вона є вірною тільки в тому випадку, коли a і p взаємно прості. Звідси $a^{\varphi(p)-1} \bmod p \equiv a^{-1} \bmod p$. В результаті даний спосіб зводиться до задачі модулярного експоненціювання. Це значно ускладнює

знаходження оберненого елемента за модулем по відношенню до багаторозрядних чисел.

1.3 Постановка задачі

Пошук оберненого елемента за модулем $a^{-1} \bmod p = b$ на сьогодні широко застосовується в асиметричніх криптографічних системах RSA і Ель Гамаля. Він характеризується високою обчислювальною складністю на початковому етапу, коли проходить процес генерування ключів.

Існують такі основні способи пошуку оберненого елемента за модулем $a^{-1} \bmod p = b$:

a) брутальна атака;

b) використання функції Ейлера;

c) застосування наслідків алгоритму Евкліда [23].

Перечисленні способи відрізняються своєю громіздкістю. Вони характеризуються високою обчислювальною та часовою складністю. Це обумовлено потребою виконання великої кількості операцій, таких як ділення з остачею, піднесення до степеню та знаходження функції Ейлера. До того ж вказані дії використовуються по відношенню до багаторозрядних чисел. Це може стати причиною переповнення розрядної сітки.

Для того щоб зменшити часову складність використання вище зазначених способів існує необхідність в розробці нової методики пошуку оберненого елемента за модулем за допомогою операції додавання модуля та залишку. Таким чином можна набагато простіше вирішувати поставлені задачі без потреби використання складних обчислювальних дій, таких як множення та ділення з остачею.

Додатково потрібно виконати наступне:

a) побудувати аналітичні вирази часових складностей розробленого способу та класичних методик пошуку оберненого елемента за модулем;

b) провести експериментальні дослідження, направленні на визначення часових характеристик програмної та апаратної реалізації як нового, так і стандартних способів пошуку оберненого елемента за модулем;

c) усунути основні недоліки і визначити мету проведеної роботи.

2 РОЗРОБКА МЕТОДІВ ПОШУКУ ОБЕРНЕНОГО ЕЛЕМЕНТА ЗА МОДУЛЕМ

2.1 Теоретичні основи методу пошуку оберненого елемента за модулем на основі додавання модуля

Процес пошуку мультиплікативного оберненого елемента за модулем на сьогодні є необхідною умовою для вирішення великої кількості завдань в різноманітних сферах. До них відноситься сучасна теорія чисел, обчислювальна та прикладна математика, криптографічних асиметричних систем, таких як RSA і метод Ель-Гамаля.

Пошук оберненого елемента за модулем за допомогою алгоритму Евкліда здійснюється наступним чином. Умова $a \cdot b - n \cdot t = 1,$ де t є часткою від ділення $a \cdot b$ на просте число p. До того ж вказаний спосіб дозволяє відшукати найбільший спільний дільник $НСД(a,n)$ для чисел a і n, а також g і h такі, що $a \cdot g + n \cdot h = НСД(a,n)$. У зв'язку з тим, щоб обернений елемент за модулем може існувати лише у випадку, коли a і p є взаємно простими, $a \cdot g + n \cdot h = 1.$

Вираз $a \cdot b \bmod n = 1$ можна записати наступним чином: $a \cdot b = k \cdot n + 1$, де k – деяке ціле число. З цього слідує, що для визначення оберненого елемента за модулем потрібно до модуля додати 1 і перевірити, чи можна отримане в результаті число націло поділити на a. Якщо дана дія без остачі не виконується, до отриманого числа послідовно необхідно додавати модуль до того часу, поки після ділення кожного наступного результату не вдасться отримати ціле число. Математично описати дану процедуру можна наступним чином:

$$n_0 = n + 1;\ b_0 = (n+1)/a;$$
$$n_1 = 2 \cdot n + 1;\ b_1 = (2 \cdot n + 1)/a;$$
$$\ldots.$$
$$n_i = (i+1) \cdot n + 1;\ b_i = ((i+1) \cdot n + 1)/a;\ b_i \in Z. \qquad (2.1)$$

Блок-схема алгоритму пошуку оберненого елемента на основі додавання модуля представлена на рисунку 2.1, а в таблиці 2.1 – приклад його застосування.

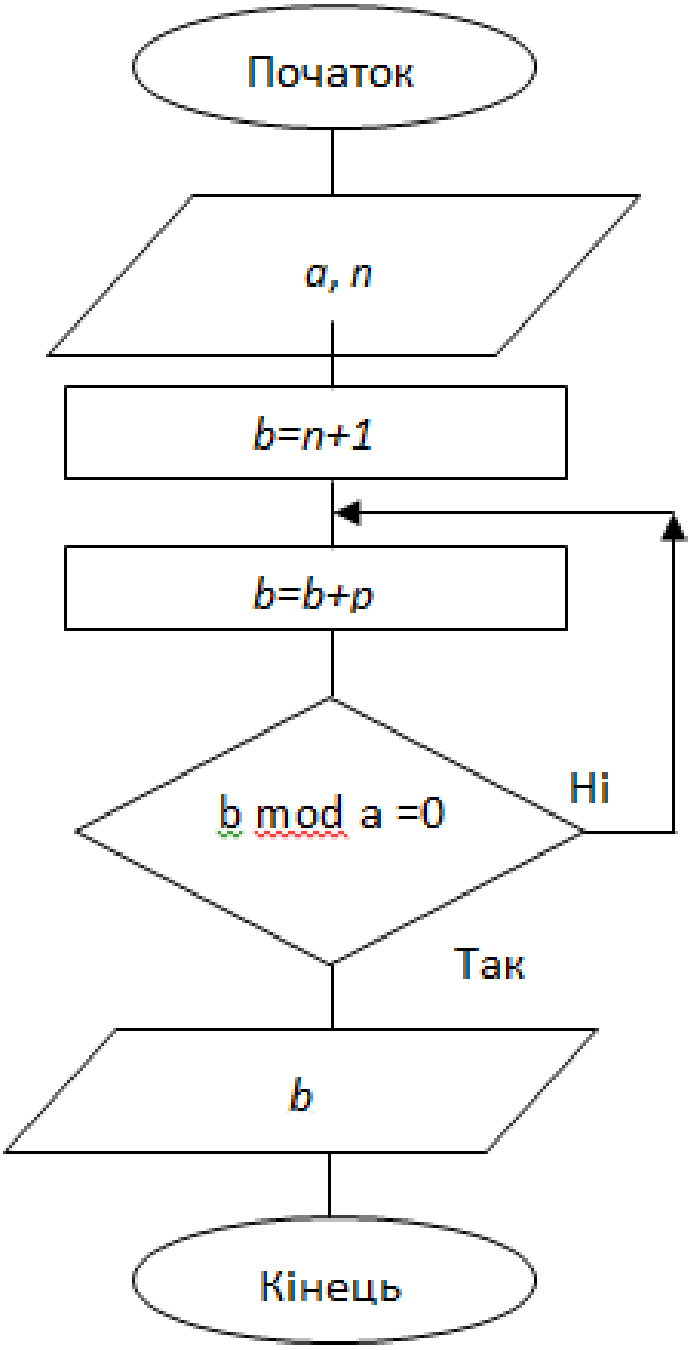


Рисунок 2.1 – Блок-схема пошуку оберненого елемента на основі додавання модуля

Таблиця 2.1 – Пошук оберненого елемента $41^{-1}\text{mod } 157$ на основі додавання модуля

i	0	1	2	3	4	5
n_i	158	315	472	629	786	943
b_i	3,85…	7,68…	11,51…	15,34…	19,17…	23

Таким чином, $41^{-1}\text{mod } 157=23$. Цей результат був отриманий без необхідності проведення досить громіздких операції, таких як множення та ділення з остачею.

2.2 Метод пошуку оберненого елемента за модулем на основі додавання залишку

Додатково для пошуку оберненого елемента за модулем $a^{-1}\text{mod } n$, де $a<n$ і НСД(a, n)=1 можна також застосовувати наступний метод. Спершу потрібно обчислити $a_0=n \text{ mod } a\neq 0$. Після цього необхідно виконати просту обчислювальну операцію додавання (формула 2.2):

$$a_1=(a_0+1)\text{mod } a;$$
$$a_2=(a_1+a_0)\text{mod } a=(2\cdot a_0+1)\text{mod } a;$$
$$\dots ;$$
$$a_i=(a_{i-1}+a_0)\text{mod } a=(i\cdot a_0+1)\text{mod } a. \quad (2.2)$$

Проводити перечисленні дії необхідно до того часу, поки наступне знайдене число a_i не стане рівним нулю. Після цього з його використання можна визначити обернений елемент за модулем за допомогаю наступної формули 2.3:

$$K = a^{-1} \bmod n = \frac{i \cdot n + 1}{a}. \qquad (2.3)$$

Рисунок 2.2 ілюструє блок-схему роботи використаного алгоритму. В таблиці 2.2 наведений приклад пошуку $101^{-1}\bmod 167$ описаним способом. В першому рядку вказані значення кількості етапів виконання алгоритму, а в другому – отримані значення a_i. Цей процес продовжується до того часу, поки не вдасться отримати деяке число $a_i=0$. В представленому прикладі є необхідність виконання 26 етапів, потрібних для виконання поставленої задачу – а саме обчислення оберненого елемента за модулем.

Таблиця 2.2 – Приклад пошуку $101^{-1}\bmod 167$ запропонованим методом

i	0	1	2	3	4	5	6	7	8
a_i	66	67	32	98	63	28	94	59	24
i	9	10	11	12	13	14	15	16	17
a_i	90	55	20	86	51	16	82	47	12
i	18	19	20	21	22	23	24	25	26
a_i	78	43	8	74	39	4	70	35	0

В відповідності до виразу (2.3), з таблиці 2.2 слідує наступне:

$$K = 101^{-1} \bmod 167 = \frac{26 \cdot 167 + 1}{101} = 43.$$

Найбільш часто використовуваними операціями в процесі пошуку оберненого елемента за модулем є визначення залишків і сум додавання. Першу з них (n mod a) краще проводити нижче описаним способом. Спершу потрібно число p записати у двійковій формі: $n=n_{r-1}\cdot 2^{n-1}+\ldots+n_i\cdot 2^i+\ldots+n_1\cdot 2^1+n_0\cdot 2^0$, де $n_i=0,1$. Після цього необхідно сформувати відповідно таблицю 2.3, у якій $n_{1i}=2^i \bmod a$.

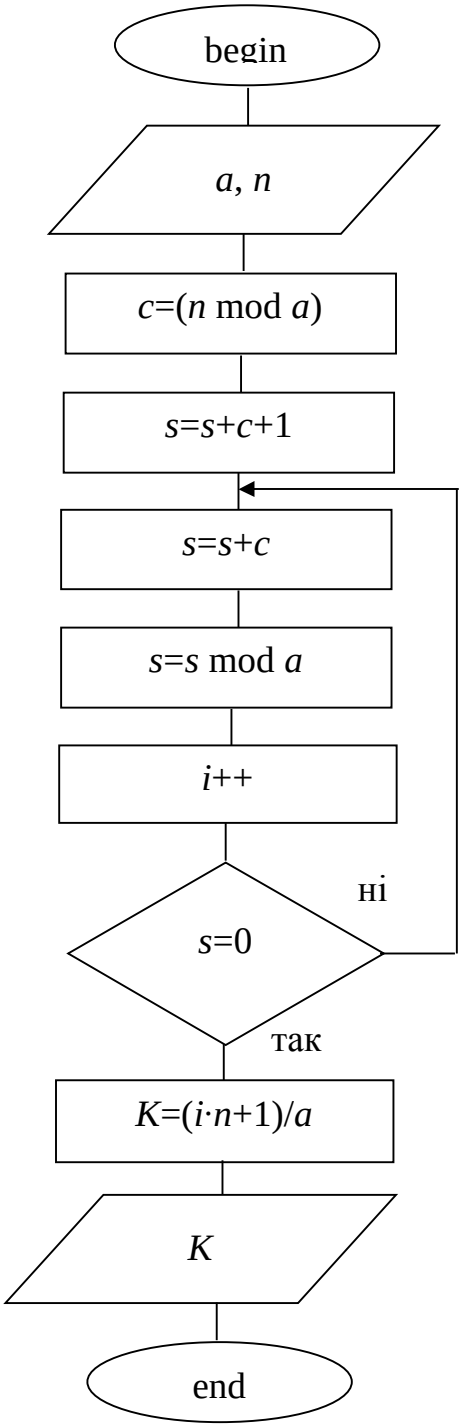


Рисунок 2.2 – Блок-схема роботи запропонованого алгоритму

Таблиця 2.3 – Знаходження залишків степенів двійки

2^{r-1}	2^{r-2}	…	2^{i}	…	2	1
n_{r-1}	n_{r-2}	…	n_{i}	…	n_{1}	n_{0}
$n_{1\,r-1}$	$n_{r\,n-2}$	…	$n_{1\,i}$	…	$n_{1\,1}$	$n_{1\,0}$

Для пошуку елемента n_{1i} потрібно здійснити множення на 2 попереднього елемента n_{1i-1}. Для цього у двійковій формі необхідно дописати 0 в молодший

розряд числа $n_{1i\text{-}1}$. Отриманий елемент в результаті порівнюється за формулою 2.4 з модулем n:

$$n_{1i} = \begin{cases} 2 \cdot n_{1\ i-1}, & 2 \cdot n_{1\ i-1} < n \\ 2 \cdot n_{1\ i-1} - n, & 2 \cdot n_{1\ i-1} \geq n. \end{cases} \tag{2.4}$$

Залишок, пошук якого здійснюється, в результаті буде дорівнювати сумі тих n_{1i}, для яких відповідні n_i рівні 1. Таблиця 2.4 ілюструє процесс пошуку залишку 167 mod 101 вище описаним методом без необхідності використання громіздкої по своїй суті операції ділення.

Таблиця 2.4 – Приклад пошуку залишку 167 mod 101

p_{1i}	27	64	32	16	8	4	2	1
p_i	1	0	1	0	0	1	1	1

Виходячи з написаного, можна зробити висновок, що 167 mod 101=(27+32+4+2+1) mod 101=66.

Запропоновані способи пошуку оберненого елемента за модулем дозволяють розділи цей процес на кілька паралельних один одному потоків додавання модуля та залишку, що неможливо здійснити при використанні алгоритму Евкліда. Для того щоб почати обчислення в кожному з них, потрібно відповідно використовувати наступні формули:

$$N_0 = \left(\left[\frac{(j-1)a}{z} \right] + 1 \right) n + 1; \tag{2.5}$$

$$N_1 = \left(\left(\left[\frac{(j-1)a}{z} \right] + 1 \right) n_{00} + 1 \right) \bmod a \tag{2.6}$$

де j - номер потоку;

z - кількість потоків.

Максимальна кількість ітерацій (повторень однієї і тієї ж операції) в кожному з потоків буде становити $a/z+1$.

2.3 Дослідження часової складності методів пошуку оберненого елемента за модулем

Для того щоб розрахувати часову складність описаного методу пошуку оберненого елемента за модулем, потрібно виділити його базові операції. Найбільш обчислювально складними серед них є множення і визначення залишків. В таблиці 2.5 наведена часова складність проведених дій.

Таблиця 2.5 – Часова складність базових операцій запропонованого алгоритму з додаванням залишку

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

Таким чином, складність пошуку оберненого елемента описаним способом, вважаючи, що $\max i=log_2 r$, обчислюється за допомогою наступної формули: $O1\left(\frac{r \cdot \log_2 r}{2} + \log_2 r \cdot \log_2 r/2\right)$. При використанні розширеного алгоритму Евкліда складність пошуку буде такою: $O\left(r^3\right)$ [24].

Рисунок 2.3 ілюструє залежність часових складностей стандартного і запропонованого способів в логарифмічній шкалі з основою 10. При аналізі

першого методу графік різко зростає, при аналізі другого – його підйом здійснюється набагато повільніше.

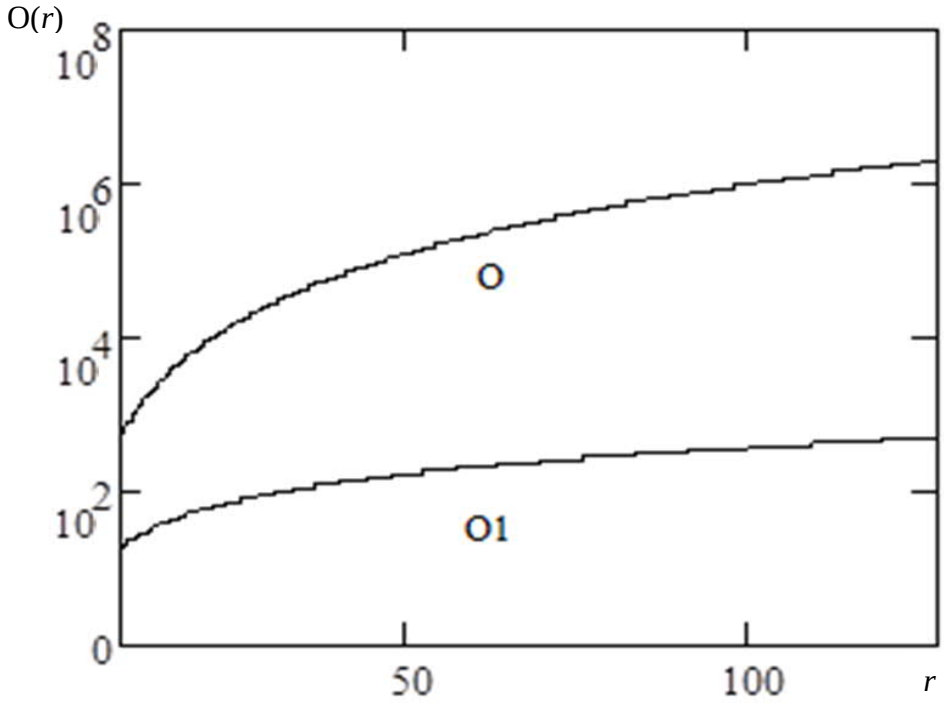


Рисунок 2.3 – Графічні залежності часових складностей класичного і запропонованого методів з додаванням залишку

Таким чином, запропонований метод пошуку оберненого елемента за модулем дозволяє повністю уникати обчислювально складних операцій. До них відноситься, в том числі, і ділення з остачею. Розроблений спосіб сприяє проведенню поставленої задачі над числами значно меншої розрядності в порівнянні з використанням стандартного алгоритму Евкліда та його наслідку.

3 ПРОГРАМНА ТА АПАРАТНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПОШУКУ ОБЕРНЕНОГО ЕЛЕМЕНТА ЗА МОДУЛЕМ

3.1 Апаратна реалізація методу додавання елемента

Для того щоб знайти обернений елемент за модулем, можна використовувати кілька можливих алгоритмів. В більшості випадків для цього застосовується теорема Ейлера, а також розширений алгоритм Евкліда разом з наслідком із нього. Перший спосіб характеризується високою степеню громіздкості, що робить його найменш оптимальним до розрахункових можливостей однокристальної ЕОМ. Саме тому в подальшому в роботі дана методика не буде розглядатись. Через це для обчислювальна сила алгоритму із додавання модуля та додавання залишку від ділення буде порівнюватись з обчислювальною силу алгоритму Евкліда.

Необхідність використання апаратних засобів для розв'язання спеціалізованих задач обумовлена можливістю підвищення продуктивності комп'ютерної системи.

Операції, що проводять в скінченних полях, можуть застосовуватись при потребі реалізації цілої низки криптографічних протоколів та алгоритмів. Вони на сьогодні широко використовуються у методах завадостійкого кодування.

Рисунок 3.1 ілюструє структурну схему блоку пошуку оберненого елемента за модулем. Алгоритм його обчислення наступний:

– $RG_C := p - 1$; $CT := 0$;

– $RG_C := (RG_C + a) \bmod p$; INC_CT;

– Якщо $(RG_C) = 0$, то $a \; ^1 \; CT$, інакше перейти до пункту 2.

Регістр A (RG_A) використовується для зберігання елемента a. Саме по відношенню до останнього і проходить пошук оберненого елемента. В регістрі C (RG_C) накопичуються результати проведених операцій. Початковим станом регістра C є $p - 1$. Після кожної наступної операції з додаванням відбувається

інкремент лічильника. Таким чином, по закінченню виконання алгоритму залишиться записаною кількість виконаних в процесі обчислювальних дій.

Коли на виході регістра *C* буде отриманий нульовий результат, в лічильнику *CT* збережеться число, що є оберненим до *a*. Таким чином, $CT = a^{-1}$.

Максимальна кількість дій, потрібних для знаходження оберненого елементу за модулем наступна: 3*p*-3. Розрахунок середньої кількості операції відбувається за такою формулою: t_1=0.875*p*-0.875.

З теореми про НДС слідує, що для будь-якого невід'ємного цілого числа *a* та будь-якого додатного цілого *b* є справедливим таке відношення: НДС (*a*,*b*) = НДС (*b*,*a* mod *b*). Саме цей висновок і ліг в основу алгоритма Евкліда.

Для того щоб зберегти отримані під час пошуку дані, при апаратній реалізації описаної процедури необхідно використати 6 регістрів. Їх назви наступні: u_1, u_2, v_1, v_2, t_1, та t_2.

Основні кроки використаного алгоритму такі:

1. Ініціалізація з присвоєнням u_1, u_2) <- (0,*p*), (v_1, v_2) <- (1,*a*).
2. Обчислення *q*<- (u_2/ v_2) та *r* <- mod (u_2/ v_2).
3. Якщо *r=0,* видання коду на вихід регістра v_1. Отриманий елемент і буде результатом, пошук якого проходив. Після цього виконується завершення алгоритму. В разі, коли даний пункт не виконаний, необхідно перейти до наступного кроку.
4. Виконання операції h_1<- u_1- v_1q,h_2<-*r*,(u_1/ u_2) <- (v_1/ v_2),(v_1/ v_2)<- (h_1/ h_2).

Описаний процес проілюстрований в пункті 2 таблиці 3.1.

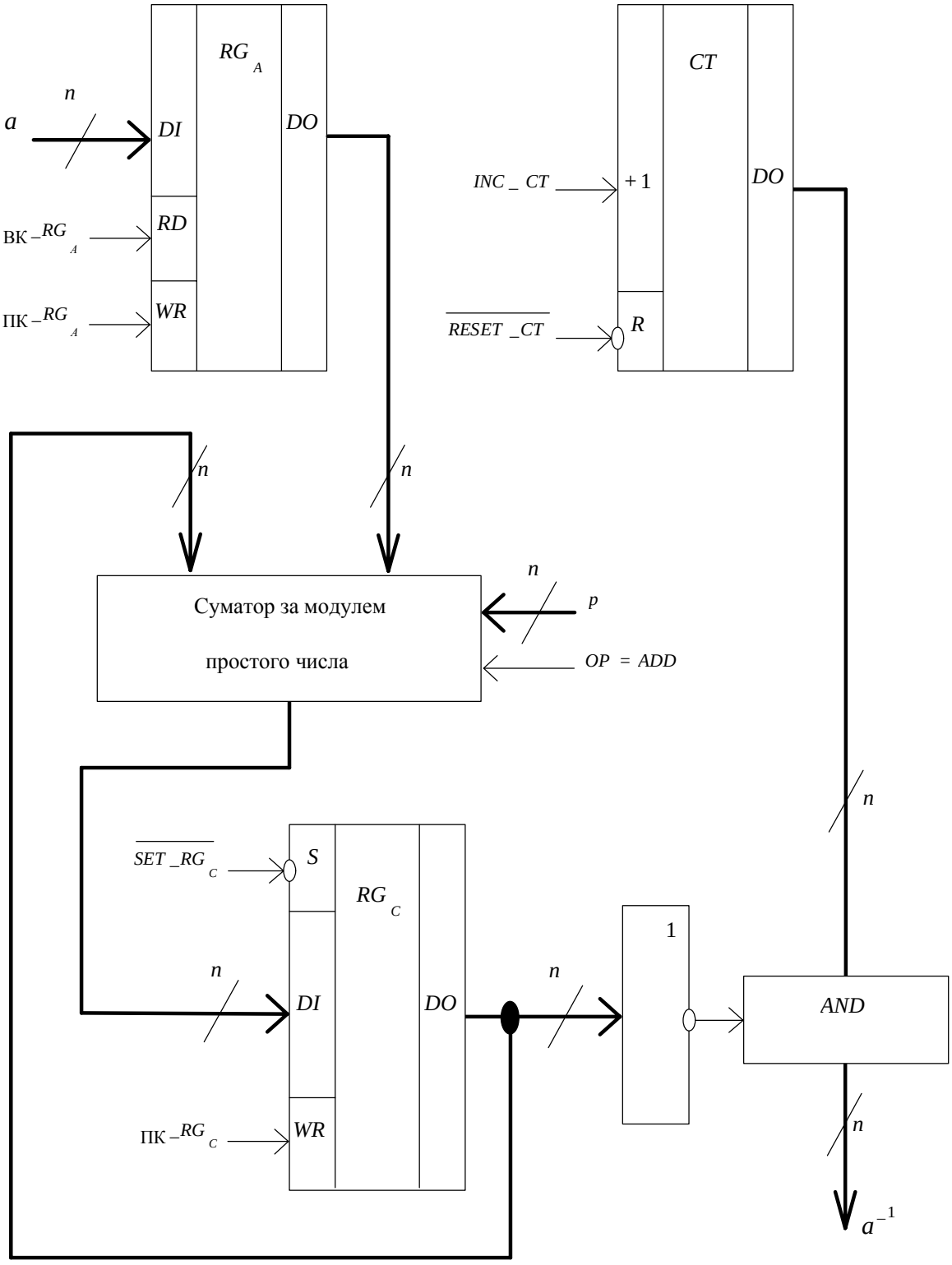


Рисунок 3.1 - Структурна схема блока обчислення оберненого елемента

Розглянемо приклад. Нехай a=4, p=23. На прикріпленій до роботи цифровій діаграмі таблиці 3.1 можна помітити, що НСП числа a, яке дорівнює 4 і модулю p, який дорівнює 23, є 1. Натомість елементом оберненим до 4 є 6.

Таблиця 3.1 – Цифрова діаграма роботи розширеного алгоритму Евкліда

q	r	u_1	u_2	v_1	v_2
-	-	0	23	1	4
5	3	1	4	-5	3
1	1	-5	3	6	1
3	0	-	-	-	-

Максимальна кількість ітерацій, які повинні бути проведені в ході виконання даного алгоритму, наступна: 2 ($\log_2 p$). Максимальна кількість операцій для пошуку оберненого елемента в відповідності з умовами прикладу така: p ($\log_2 p$) - 5 ($\log_2 p$) - 4 ($\log_2 p^2$). В свою чергу середня кількість ітерації, потрібних при знаходженні, наступна: ($\log_2 p$). Середня кількість операцій розраховується таким чином: $t_2 = 0.25p \log_2 p - 2.75 \log_2 p - 1.5\log_2 p^2$.

Процес реалізації пошуку оберненого елементу за модулем з використанням розширеного алгоритму Евкліда при умові, що шукане x потрібно взяти по модулю m навіть у випадку коли x від'ємне, виглядає так:

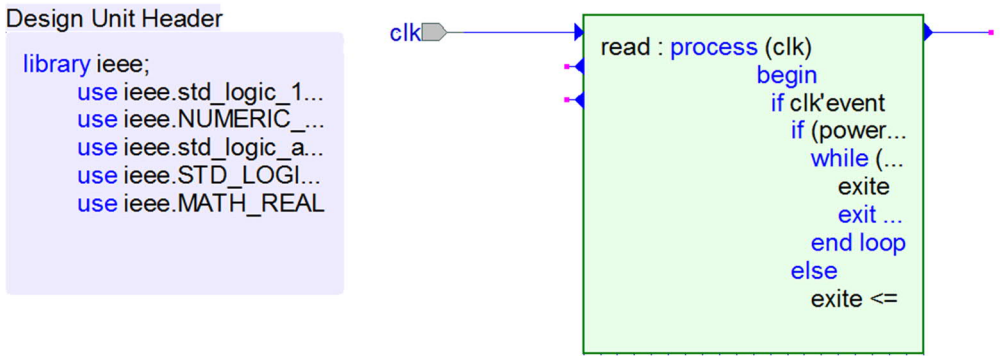
```
int x, y;
int g = gcdex (a, m, x, y);
   if (g != 1)
        cout << "no solution";
   else {

                x = (x % m + m) % m;
                cout << x;
        }
```

Асимптотика цього розв'язку наступна: $O(\log m)$.

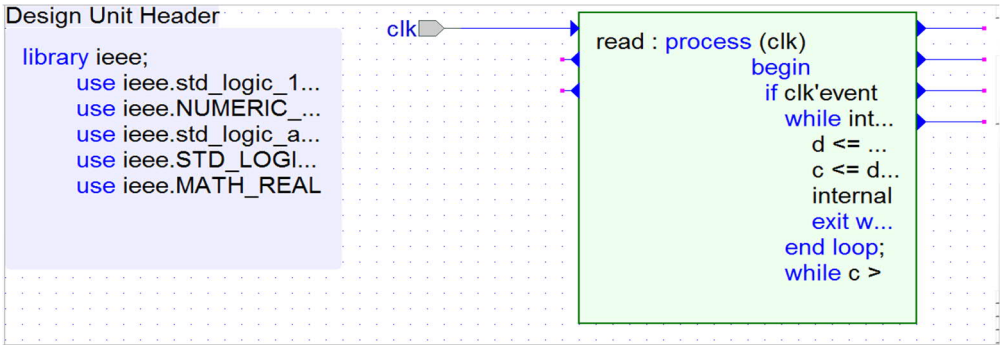
3.2 Апаратна реалізація методів пошуку оберненого елемента за модулем та дослідження часових характеристик

В ході проведення експериментального дослідження, ціллю якого було визначення потрібного періоду часу для знаходження оберненого елемента за модулем з застосуванням розширеного алгоритму Евкліда і запропонованого альтернативного способу для кожного з цих методів була проведена програмно-апаратна реалізація на базі середовища Aldec Active-HDL 9.1

Рисунок 3.2 ілюструє фрагменти UML-діаграм спроектованих пристроїв. На них вказані всі стандартні бібліотеки з компонентами, призначеними для їх правильного функціонування.



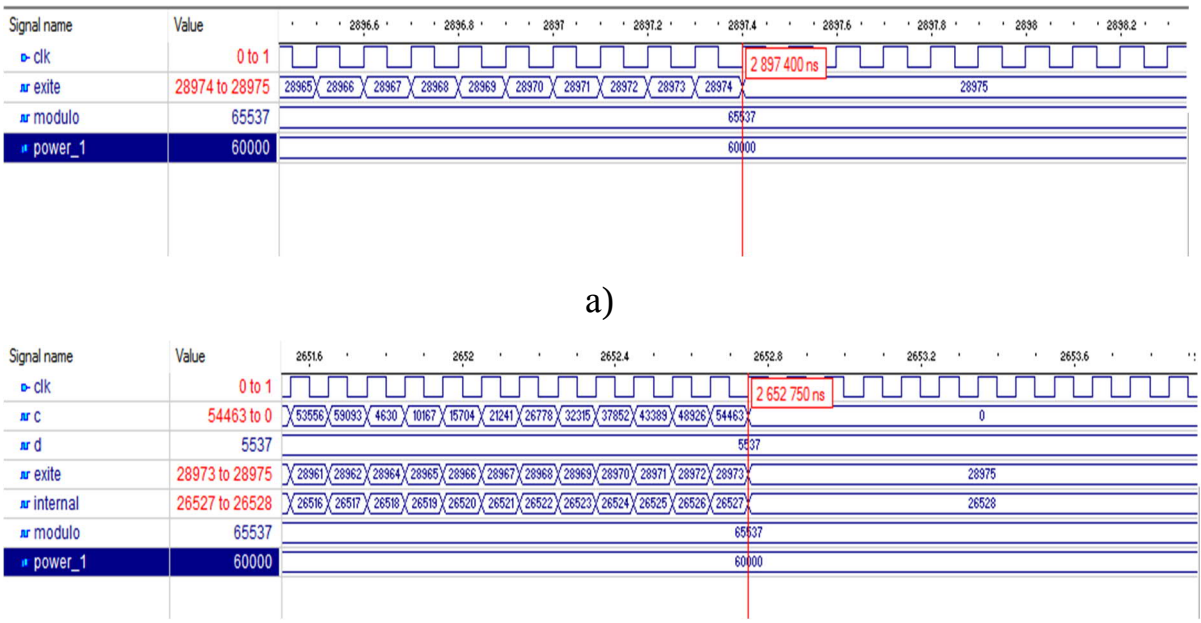
а)



б)

Рисунок 3.2 – Фрагменти UML-діаграм спроектованих пристроїв пошуку оберненого елемента за модулем: а) на основі алгоритму Евкліда; б) на основі запропонованого методу

На рисунку 3.3 зображені часові діаграми функціонування спроектованих пристроїв, призначених для визначення оберненого елемента за модулем. Для прикладу було вибрано $60000^{-1}\text{mod}65537$. Ілюстрації показують різницю між роботою розширеного алгоритму Евкліда та запропонованим методом.



а)

б)

Рисунок 3.3 – Часова діаграма пошуку оберненого елемента за модулем: а) на основі алгоритму Евкліда; б) на основі запропонованого методу

Згідно з наведеним прикладом, можна зробити висновок, що часовий термін, потрібний для функціонування системи, зменшився з 2,8974 мс (на основі алгоритму Евкліда) до 2,65275 мс (на основі запропонованого методу додавання залишків). Таким чином, він став коротшим приблизно в 1,09 разів.

В процесі виконання експериментальних досліджень часу, потрібного для програмно-апаратної реалізації двох описаних способів пошуку оберненого елементі, число p=65537 було вибрано простим і фіксованим. Таким чисном, для будь-якого $a<p$ існував обернений елемент. В свою чергу число a змінювалося з 5000 до 65000 з кроком 5000. Отримані результати відображені в таблиці 3.1.

Таблиця 3.2 –Порівняльні характеристики часових затрат обчислення оберненого елемента за модулем в середовищі розробки Aldec Active-HDL 9.1

N п/п	*a*	$a^{-1} \bmod p$	Час роботи, *ns*	
			Алгоритм Евкліда та його наслідок	Метод додавання залишку
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
Середній час	-	-	3343600	2280619

Рисунок 3.4 ілюструє графіки залежності витраченого часу для визначення оберненого елементу як стандартним (зображення 1), так і новим (зображення 2) способом в повній відповідності до таблиці 3.2. Також позначені середні показники отриманих результатів (графіки 3 та 4 відповідно).

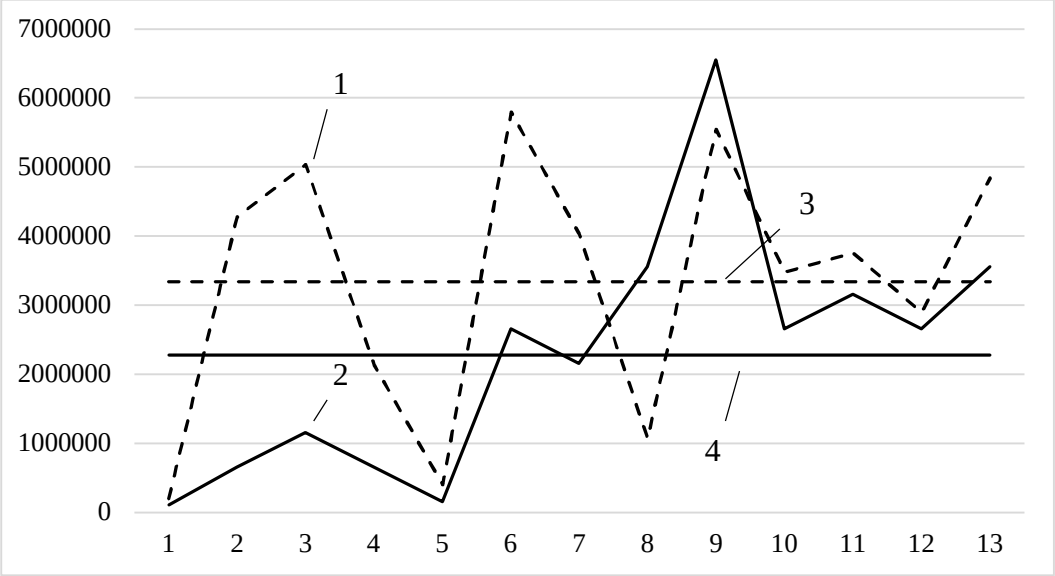


Рисунок 3.4 – Часові характеристики VHDL – реалізації класичного (штрихові лінії) та запропонованого (суцільні лінії) методів пошуку оберненого алгоритму за модулем.

Отриманні результати свідчать про те, що лише в 2 з 13 випадків при a=40000 та a=45000 запропонований альтернативний спосіб поступається в часі свого виконання класичному. Це обумовлено потребою виконання при таких значення задано великої кількість операції додавання.

Середній час пошуку оберненого елемента за модулем з використання алгоритму Евкліда (3343600 ns) в 1,47 разів більший за отримані показники при застосуванню методу додавання залишків (2280619 ns).

3.3 Програмна реалізація методів пошуку оберненого елемента за модулем та дослідження часових характеристик

Для проведення експерименту був вибраний портативний комп’ютер моделі Lenovo B50-70, оснащений процесором Intel Pentium 3558U з частотою 1.7 ГГц. Обсяг оперативної пам’яті даного приладу становив 4 GB. Для проектування програмного комплексу, завдяки якому проводились

обчислювальні операції, біла обрана мова програмування високого рівня C++. Вона характеризується можливістю трансформування кодів під різні архітектурі та ОС.

Можливості вибраного портативного комп'ютера не дозволяють провести паралельні обчислення на будь-яку кількість потоків. Через це, коли були визначенні межі, все операції проводились послідовно по відношенню один до одного. Натомість час визначення оберненого елемента обирався в тому потоці, де був знайдений результат пошуку.

На рисунку 3.5 зображений графік залежності середнього часу визначення оберненого елемента від кількості паралельних потоків *z* при *n*=1021 (рисунок 3.5а) та *n*=65521 (рисунок 3.5б) різними методами: додаванням модуля (штрихова лінія) та додаванням залишку (суцільна лінія). Вибрані значення є найбільшими простими шістнадцяти- та десятирозрядними числами. Число *a* пробігає значення від 2 до *n*-1 з кроком 1.

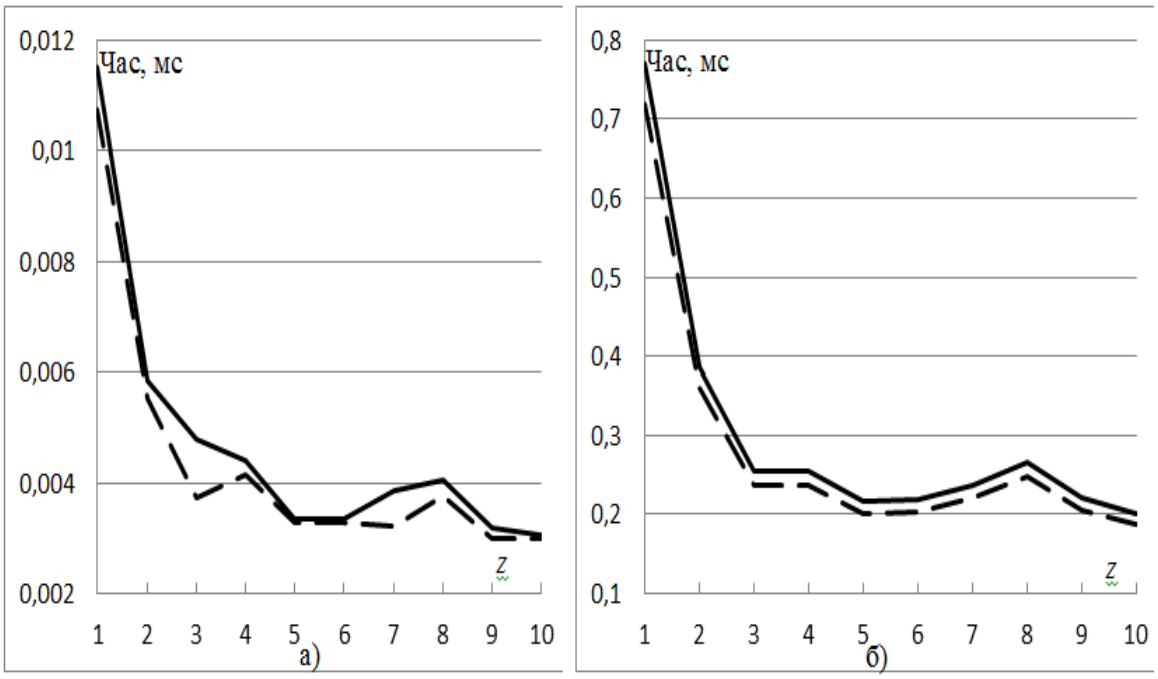


Рисунок 3.5 - Графічна залежність середнього часу пошуку оберненого елемента на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) від кількості паралельних потоків при *n*=1021 (рисунок 3.5а) та *n*=65521 (рисунок 3.5б)

Рисунок 3.5 ілюструє, що в кожному з випадків метод додавання модуля характеризується набагато більшою швидкодієї в порівнянні зі способом додавання залишку. При цьому характер графіків схожий між собою. Зменшення часу обчислень досягалось при проведенні операції паралельно між собою в двох або трьох потоках. При послідовному виконанні дій (z=1) потрібен був набагато більший часовий термін. Якщо проводилось подальше збільшення z, можна помітити, що інтенсивність зменшення часу спадає. Невеликий його підйом відбувається при z=7 та 8. Після цих показників графік повільно спадає. Саме тому для проведення подальшого експерименту був вибраний показник z=6.

Рисунок 3.6 показує графік залежності часу знаходження оберненого елемента на основі методу додавання модуля (штрихова лінія) і методу залишку (суцільна лінія) при z=6 та n=65521. В свою чергу таблиця 3.3 ілюструє цей фрагмент з частиною отриманих даних.

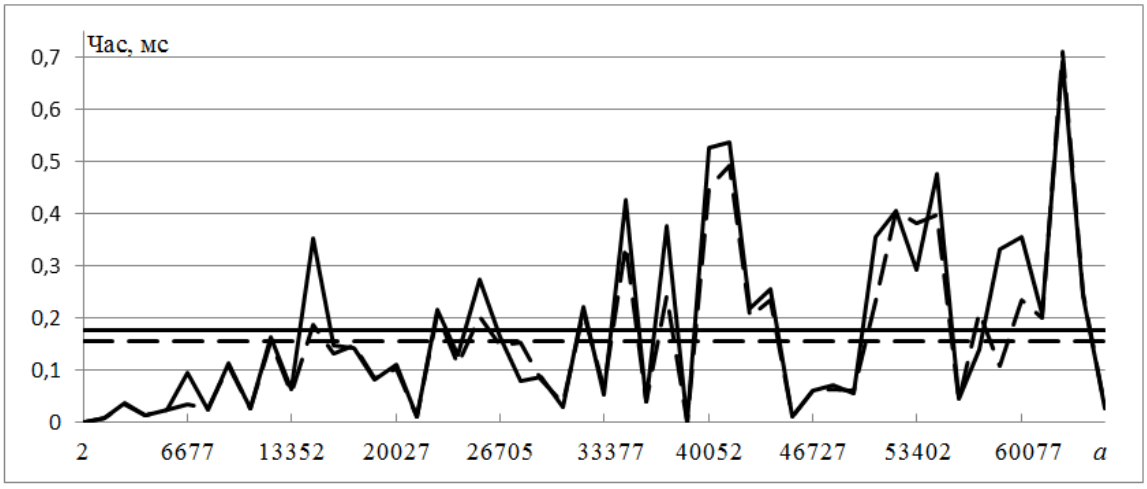


Рисунок 3.6 - Графічна залежність часу пошуку обернених елементів на основі додавання модуля (штрихова лінія) та залишку (суцільна лінія) при z=6 та n=65521 та середні часи обчислень (штрихова та суцільна прямі відповідно)

Зміна числа a відбувається від від 2 до 65417 з кроком 1335. Останній був вибраний для того, щоб процес розрахунку оберненого елемента за модулем відбувався для 50 різних значень. Додатково зазначений середній час обчислень

двома способами (штриховий графік – додавання модуля, суцільний графік – додавання залишку). Прямі 1 і 2 мають осцилюючий характер. Незважаючи на це, загальний тренд дозволяє зробити висновок щодо того, що ріст числа *a* зумовлює збільшення часу пошуку. У більшості випадків спосіб додавання модуля є в 1,136 рази більш швидким, ніж спосіб додавання залишку. Середній час обчислень при використанні першого методу становить 0,15499 мс, при застосування другого методу – 0,176144 мс. Таким чином,

Таблиця 3.3 – Фрагмент з частиною отриманих результатів

a		Час пошуку, мс	
	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

Рисунок 3.7 ілюструє графік залежності усередненого часу знаходження оберненого елемента від розрядності модуля на основі трьох методів: 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).

Модуль n вибирався, як найбільш простій. При цьому він менший за 2^k, де k змінється від 10 до 19,5 з кроком 0,5. Число a набувало 1000 різних значень.

Вибір кроку його зміни здійснювався так, щоб можна було охопити весь діапазон від 1 до *n*.

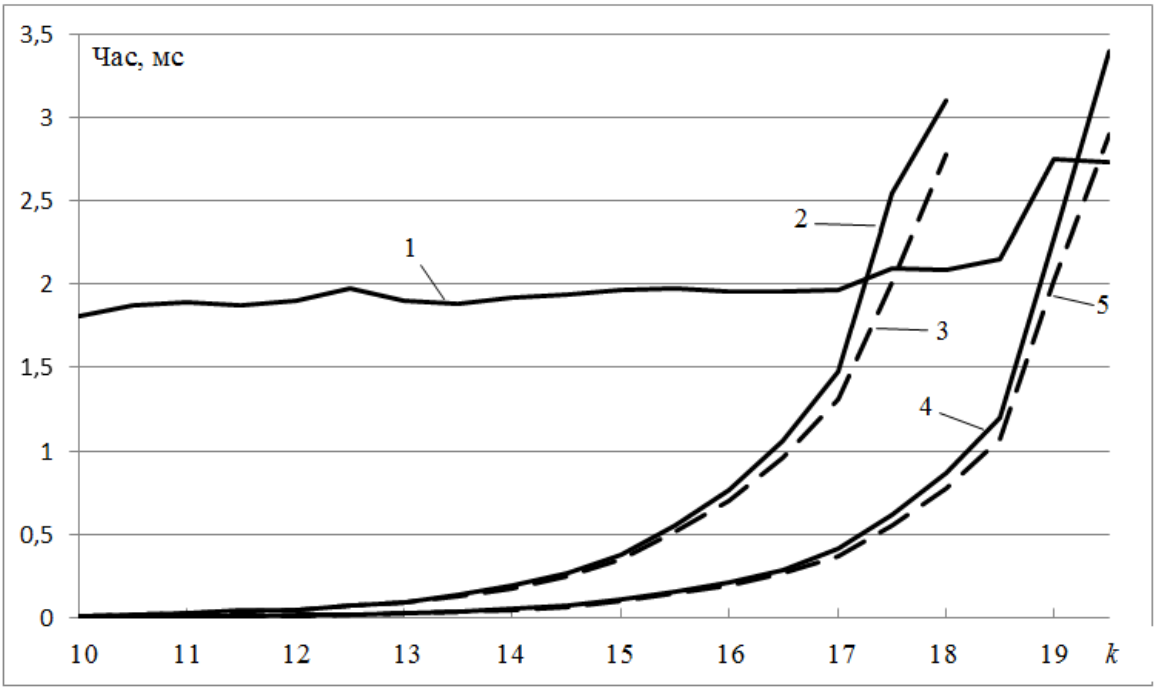


Рисунок 3.7 – Графічна залежність часу пошуку оберненого елемента на основі розширеного алгоритму Евкліда (крива 1) та запропонованих методів додавання модуля (криві 2, 3) та залишку (криві 4, 5) при *z*=1 (криві 2, 4) *z*=6 (криві 3,5) від розрядності модуля

На рисунку 3.7 видно, що в випадку застосування алгоритму Евкліда збільшення розрядності чисел приводить до досить повільного збільшення часового проміжку виконання завдання. При *k*=19 час різко зростає. Після цього можна помітити практично горизонтальну ділянку графіка. В свою очергу при застосуванні обох способів середній час зростал майже по параболічній кривій. При цьому алгоритм додавання модуля відрізняється більшою степенню швидкодії. При *z*=1 (коли не відбувається розпалелення), плгоритм Евкліда дещо випереджає альтернативні способи. Таку ж особливість можна спорстерігати для *z*=6 – при *k*=19,5. Це говорить про потребу в разпералеленні операцій обчисленнь задля зменшення терміну знаходження оберненого елемента за модулем.

Для того щоб зневілювати випадкові фактори впливу на час пошуку, експериментальні дослідження повторювались 100 разів.

ВИСНОВКИ

1. Відбувся аналіз існуючих криптосистем, їх недоліки та переваги.

2. На основі теоретичної бази пропонованого методу асиметричного медулярного експоненціювання зроблено аналітичний приклад із дослідженням дієздатності алгоритмічного підходу.

3. Беручи до основи розглянуто алгоритмічний підхід до вирішення завданння

4. Побудовано блоксхему класичного та пропонованого методу із врахуванням корекції частини, щоб виконання алгоритму видавав очікуваний результат, без помилок.

4. На основі розробленого алгоритмічного забезпечення побудовано блок-схему програмної підсистеми, що дозволило визначити основні компоненти що потрібно внести для розробки програмного рішення, щоб дослідити роботу класичного і пропонованого методів.

5. На основі блок-схеми розроблено програмний засіб для дослідження працездатності методів.

6. Здійснено дослідження методів за допомогою розробленого програмного засобу.

7. Отримані результати які були експортовані у вихідний файл проаналізовано та побудовано графіки.

8. Дослідження показало, що пропонований метод є набагато ефективнішим, як у розробці, так і у швидкодійності обчислення, на відмінну класичного, що потребую нагромадженої кількості додаткових перевірок, для корегування обчислення, що також має і вплив на його швидкодію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Якименко І.З., Касянчук М.М., Кінах Я.І., Власюк І.М., Суслін В.В. Удосконалення реалізації асиметричних криптоалгоритмів на основі системи залишкових класів. Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології» (АСІТ), 2018. С. 79.
2. 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. С. 178.
3. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М.: Радио и связь, 2001. 376 с.
4. Воробьев И.В. Защита информации в персональных ЭВМ. М.: Мир, 2008. 312 с.
5. Мафтик С. Механизмы защиты в компьютерных сетях. М.: Мир, 2013. 219 с.
6. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография. М.: Профессионал, 2005. 480 с.
7. Рябко Б.Я., Фионов С.В. Криптографические методы защиты информации: Учебное пособие для вузов. М.: Телеком, 2005. 229 с.
8. Панасенко С.П. Алгоритмы шифрования. М.: Академический Проект, 2006. 529 с.
9. Петров В.П., Петров С.В. Информационная безопасность человека и общества. М.: ЭНАС, 2007. 334 с.
10. Шнаер Б. Практическая криптография. М.: Вильямс, 2007. 424 с.
11. Задірака В.К., Олексюк О.С. Комп'ютерна криптологія. Тернопіль, Київ, 2002. 504 с.
12. Вербіцький О.В. Вступ до криптології. Львів: ВНТЛ, 1998. 247 с.

13. Иванов И.И. Криптология. М.: Высшая школа, 2004. 230 с.

14. Винокуров А.Ю. Блочные алгоритмы. М.: Монитор, 2005. 132 с.

15. Голубицкая Е.А., Жигульская Г.М. Экономика связи. М.: Радио и связь, 2003. 318 с.

16. Айерленд К. Классическое введение в современную теорию чисел. М.: Мир, 2007. 416 с.

17. Акушский И.Я., Юдицкий Д.М. Машинная арифметика в остаточных классах. М.: Сов. радио, 1968. 440с.

18. Вариченко Л.В. Абстрактные алгебраические системы и цифровая обработка сигналов. К.: Наука думка, 2006. 247 с.

19. Сидельников В.М. Криптография и теория кодирования. М.: ФИЗМАТЛИТ, 2008. 324 с.

20. Амербаев В.М. Теоретические основы машинной арифметики. Алма-Ата: Наука, 2006. 324 с.

21. Ворона В.А., Тихонов В.А. Системы контроля и управления доступом. М.: Горячая линия - Телеком, 2010. 272 с.

22. Деднев М.А., Дыльнов Д.В. Защита информации в банковском деле и электронном бизнесе. М.: КУДИЦ ОБРАЗ, 2004. 321 с.

23. Казарин О.В. Безопасность программного обеспечения компьютерных систем. М.: Высшая школа, 2013. 243 с.

24. Ярочкин В.И. Информационная безопасность. М.: Академический Проект, 2008. 544 с.

25. Барсуков В.С. Безопасность: технологии, средства, услуги. М.: КУДИЦ-ОБРАЗ, 2001. 496 с.

26. Белов Е.Б., Лось В.П., Мещеряков Р.В. Основы информационной безопасности. М.: Горячая линия - Телеком, 2006. 544 с.

27. Краснобаев В.А. Методы повышения надежности специализированных ЭВМ систем и средств связи. Харьков: ХВВКИУ РВ, 2000. 172с.

28. Ковалевский В. Криптографические методы. М.:Компьютер Пресс, 2013. 312 с.

29. Севастьянова Б.А. Совершенные шифры. М.: Гелиос АРВ, 2003. 160 с.

30. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2016.– 65 с.

31. Об’єктно орієнтована мова програмування *Java*: веб-сайт URL: https://uk.wikipedia.org/wiki/JAVA (дата звернення 5.02.2019)

32. Методичні вказівки до написання техніко-економічного розділу для дипломних проектів на здобуття освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» / І.Р.Паздрій. - Тернопіль: ТНЕУ, 2015.– 36 с.

33. Методичні вказівники до оформлення курсових проектів, звіт про проходження практики, випускних кваліфікаційних робіт Студентів спеціальності «Клмп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019.– 33с.