МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Кафедра комп'ютерної інженерії

ПЕТРИЦА Наталія Павлівна

Алгоритм факторизації багаторозрядних чисел на основі теоретико-числового базису Радемахера / Factorization Algorithm of Multi Numbers Based on Rademacher Theoretic Basis

спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

Магістерська робота

Виконала студентка групи
КІзм-21
Н. П. Петрица

Науковий керівник:
к.т.н., І. З. Якименко

Магістерську роботу допущено
до захисту:

"21" 05 2018 р.

Завідувач кафедри
О. М. Березький

ТЕРНОПІЛЬ - 2018

Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії
Освітній ступінь «магістр»
спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.М. Березький
"21" 01 2016 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ
Петрици Наталії Павлівни
(прізвище, ім'я, по батькові)

1. Тема магістерської роботи «Алгоритм факторизації багато розрядних чисел на основі теоретико-числового базису Радемахера/FactorizationAlgorithmofMultiNumbersBasedonRademacherTeoreticBasis»
керівник роботи к.т.н.,І.З. Якименко
затверджені наказом по університету від 23 жовтня 2017 р. № 1325.

2. Строк подання студентом роботи «15» січня 2018 року

3. Вихідні дані до магістерської роботи .

Об'єкт дослідження –розкладання на прості множники великорозрядних чисел для криптоаналізу асиметричних криптографічних систем захисту інформації.

Предмет дослідження – методи і алгоритми пошуку зображень по змісту на основі структурних і колірних ознак.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- проаналізувати методи перевірки криптостійкості шифрування з відкритим ключем та функції на яких вони базуються;
- дослідити процес факторизації натуральних чисел;
- проаналізувати основні методи факторизації;
- розробити та проаналізувати удосконалений метод факторизації Ферма;
- дослідити та порівняти обчислювальні складності розробленого та існуючих алгоритмів факторизації;
- розробка програмних модулів;
- виконати тестування та експериментальне дослідження програмного забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслен
   - мета роботи, об'єкт та задачі дослідження
   - наукові результати;
   - наукова новизна;
   - блок – схема алгоритму пошукузалишківвеликорозрядних чисел простих модулях;
   - блок-схема удосконаленого алгоритму факторизації на основімет Ферма;
   - складності основних операцій розробленого алгоритму факториз багато розрядних чисел;
   - обчислювальна складність розробленого алгоритму в порівнянн класичним;
   - приклад роботи класичного та вдосконаленого методу Ферма;
   - приклад роботи класичного методу Ферма та удосконаленого мет Ферма;

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Антиплагіат	Мельник Г.М., доцент		
Нормо-контроль	Гураль І. В., викладач		

7. Дата видачі завдання « 21 » листопада 2016 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Приміт
1	Захист інформації на основі асиметричних криптографічних алгоритмів та криптоаналіз	3.11.2016 – 1.01.2017	
2	Ефективні алгоритми та методи криптоаналізу асиметричних систем у криптосистемах	2.01.2017 – 31.05.2017	
3	Проектування та розробка програмного продукту	1.06.2017 – 25.01.2018	
4	Нормоконтроль, попередній захист	16.01.2018 – 1.02.2018	
5	Захист	2.02.2018	

Студент _____ Петрица Н.П.
(підпис)

Керівник магістерської роботи _____ к.т.н., доцент, І.З. Якименко
(підпис)

ЗМІСТ

ВСТУП

Актуальність роботи. Сьогодні проблема несанкціонованого доступу (НСД) до ресурсів інформаційних систем загострюється із розвитком інформаційних технологій. Розв’язання завдань розробки та вибору відповідних ефективних методів і засобів захисту значною мірою залежить від низки чинників, пов’язаних із самим процесом несанкціонованого доступу. Для вирішення цього завдання використовується цілий комплекс засобів, що включає в себе технічні, програмно-апаратні засоби і адміністративні заходи захисту інформації. Побудова надійного захисту комп'ютерної системи неможлива без попереднього аналізу загроз безпеки системи. Але існує низка недоліків при проектуванні систем безпеки, які стають причиною виникнення несанкціонованого доступу та порушення базових властивостей інформації і інформаційних ресурсів мережі.

Сучасний світ характеризується тенденцією постійного підвищення ролі інформації, яка має усе більш вагоме значення у функціонуванні державних і суспільних інститутів, в житті кожної людини. У сучасних умовах сформувався новий вид трудової діяльності, пов'язаний із здобуттям, поширенням і зберіганням інформації. Промислове суспільство трансформується в інформаційне. Інформатизація веде до створення єдиного світового інформаційного простору, до уніфікації інформаційних технологій різних країн. Нові технології обіцяють грандіозні перспективи. У той же час катастрофічно зростає ціна втрат в разі нештатного функціонування або зниження надійності систем обробки і передачі інформації. З підвищенням значущості і цінності інформації, відповідно зростає і важливість її захисту. Одним із можливих способів захисту інформації при її передачі та зберіганні є криптографічний захист.

Сьогодні найбільш розповсюдженою системою шифрування є система шифрування з відкритим ключем RSA. Система шифрування інформації RSA

є якісною. На сьогодні її можна зустріти у багатьох готових продуктах, що пропонуються на ринку програмного та апаратного забезпечення, зокрема у новому проекті RSA-OAEP. На сьогодні відсутні докази абсолютної стійкості для нового методу RSA – OAEP. Однак він достатньо стійкий проти адаптивних атак на фрагменти шифротексту, оскільки ця стійкість безпосередньо пов’язана із складністю обчислення оберненої односторонньої функції, яка застосовується у RSA алгоритмі. Тому вибрана задача криптоаналізу є перспективною. Доцільно значну частину досліджень спрямувати на зменшення витрат обчислювальних ресурсів, шляхом використання базису Крестенсона та символів Якобі.

Метою роботи є удосконалення методу факторизації Ферма в криптографічних системах захисту інформації для оцінки криптостійкості RSA-подібних шифрів.

Для досягнення даної мети ставились наступні **задачі**:

- проаналізувати методи перевірки криптостійкості шифрування з відкритим ключем та функції на яких вони базуються;
- дослідити процес факторизації натуральних чисел;
- проаналізувати основні методи факторизації;
- розробити та проаналізувати удосконалений метод факторизації Ферма;
- дослідити та порівняти обчислювальні складності розробленого та існуючих алгоритмів факторизації;
- розробка програмних модулів;
- тестування та експериментальне дослідження програмного забезпечення.

Об'єкт дослідження –розкладання на прості множники великорозрядних чисел для криптоаналізу асиметричних криптографічних систем захисту інформації.

Предмет дослідження – методи та засоби зменшення обчислювальної складності алгоритмів факт на основі використання ТЧБ Крестенсона.

Наукова новизна одержаних результатів визначається наступним чином:

– удосконалено метод факторизації Ферма для криптоаналізу асиметричних криптографічних систем захисту інформації;

– досліджено та проаналізовано обчислювальні складності існуючих алгоритмів факторизації;

– проаналізовано математичні особливості базису Крестенсона, що дозволило зменшити використання обчислювальних ресурсів при використанні операції факторизації.

Практична цінність одержаних результатів полягає в тому, що:

– обґрунтовано підхід та розроблено алгоритм факторизації, що дозволило застосувати відповідне програмне та апаратне забезпечення для реалізації запропонованого рішення;

– реалізовано програмне забезпечення для факторизації на основі середовища розробки C++ Builder 6.0 та мови програмування С.

Публікації та апробація ДР. За результатами наукових досліджень, проведених у магістерській роботі, підготовлено тези доповіді «Алгоритм завадостійкого кодування на основі циклічних кодів» обсягом 1 сторінка на Всеукраїнській школі-семінарі «Сучасні комп’ютерні інформаційні технології» (АСІТ-2017) [9].

1 ЗАХИСТ ІНФОРМАЦІЇ НА ОСНОВІ АСИМЕТРИЧНИХ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ ТА КРИПТОАНАЛІЗ

1.1 Передові технології у криптографічному захисті інформації

У зв’язку з інтенсивним розвитком інформатизації суспільства, що накладає певний відбиток на розвиток економіки, суттєво впливає на темпи науково-технічного прогресу, підвищення продуктивності праці і удосконалення соціально-економічних відносин, проблема захисту інформації, в цілому, стає більш значущою в забезпеченні безпеки розвинених держав.

Глобальна комп’ютеризація суспільства і розповсюдження інформаційно-обчислювальних мереж на великих географічних просторах породжують ряд суттєвих проблем, найбільш важливою з яких є захист інформації від витоку і порушення її цілісності. Висока концентрація інформації підвищує значущість шкоди у випадку викрадення або втрати. Відносна простота її отримання в окремих випадках і практична безкарність при різних несанкціонованих діях з нею, стимулювали процес становлення і розвитку «комп’ютерної злочинності». До цього процесу, як показує світовий досвід, залучені достатньо широкі кола, в які входять як окремі фізичні особи, так і представники державних структур.

Світові засоби масової інформації подають висновки аналізу спецслужб, що мають безпосереднє відношення до різних аспектів захисту інформації, про те, що до 80% всього обсягу шкоди, нанесеної незаконним отриманням і використанням інформації, відбувається саме через її витік технічними каналами. Розвиток спеціальних інформаційно-телекомунікаційних систем на сучасному етапі розвитку суспільства, який характеризується впровадженням новітніх технологій і систем є однією з глобальних науково–технічних стратегій, а одним з важливих завдань є не лише забезпечення

функціонування вже існуючих, але і впровадження нових, які б не поступалися світовим стандартам та відповідали велінню часу.

Важливим напрямком діяльності у сфері криптографічного та технічного захисту є наукова та науково-технічна діяльність, що зумовлено необхідністю:

а) постійного вдосконалення та розвитку існуючих мереж і комплексів спеціальних інформаційно-телекомунікаційних систем, розроблення сучасних вітчизняних засобів криптографічного та технічного захисту інформації, забезпечення безпеки державних інформаційних ресурсів в інформаційно–телекомунікаційних системах;

б) визначення загроз інформації, формування вихідних даних для оцінки можливостей технічних засобів розвідки і їх небезпеки, виявлення потенційних каналів витоку інформації.

Основу забезпечення інформаційної безпеки в інформаційно–телекомунікаційних системах складають криптографічні методи і засоби захисту інформації.

В основі криптографічних методів лежить поняття криптографічного перетворення інформації, вироблюваного на основі певних математичних законів, з метою виключити доступ до даної інформації сторонніх користувачів. Це криптографічне перетворення називається алгоритмом шифрування (шифром), під яким розуміється сімейство однозначно оборотних відображень множини відкритих повідомлень спільно з простором ключів в множину закритих повідомлень (криптограм). Де ключ – це конкретний секретний стан деяких параметрів алгоритму, який задає однозначне перетворення відкритого тексту [16].

Апаратно-програмні засоби криптографічного захисту інформації (наприклад у системі захисту СЕМП) забезпечують автентифікацію адресата та відправника електронних документів і службових повідомлень, гарантують їх достовірність та цілісність у результаті неможливості підробки або заміни документів у шифрованому вигляді. Криптографічний захист

інформації має охоплювати всі етапи обробки електронних документів, починаючи з часу їх створення до зберігання в архівах. Використання різних криптографічних алгоритмів на різних етапах обробки електронних документів дає змогу забезпечити безперервний захист інформації в інформаційній мережі, а також відокремлену обробку інформації стосовно різних завдань інформатизації.

Для забезпечення розв'язання завдань суворої автентифікації установ, підключених до інформаційної мережі, розроблено систему ідентифікації користувачів, яка є основою системи розподілу ключів криптографічного захисту. Відповідні ідентифікатори ключів записуються в електронні картки, які є носіями ключової інформації для апаратного шифрування. Для забезпечення захисту інформації від модифікації з одночасною суворою автентифікацією та безперервного захисту платіжної інформації та інших інформаційних задач з часу її формування система захисту СЕМП включає механізми формування перевірки ЕЦП на базі несиметричного алгоритму RSA.

Для забезпечення конфіденційності інформація СЕМП, що циркулює в інформаційній мережі, має пройти обробку АРМ – НБУ або АРМ-СТП. Програмно-апаратний комплекс АРМ-НБУ є єдиним шлюзом до всіх задач файлового обміну інформацією Національного банку. Генерація ключової інформації для апаратури захисту, її транспортування, контроль за її обліком і використанням, контроль за використанням апаратури захисту, техніко-експлуатаційне обслуговування та ремонт апаратури захисту, а також сертифікація відкритих ключів покладаються лише на службу захисту інформації Національного банку.

Важливим напрямком розвитку досліджень у галузі методів та програмно-апаратних засобів опрацювання цифрових даних є вдосконалення алгоритмів, які використовуються в сучасних комп’ютерних системах (КС), та розвиток математичних основ теорії чисел на базі кодових систем різних теоретико-числових базисів (ТЧБ), до яких належать: унітарний, Хаара, Радемахера,

Крестенсона, Уолша, Галуа тощо. Вдосконаленням цього класу обчислень на основі названих ТЧБ є створення високопродуктивних компонентів та спецпроцесорів міжбазисних перетворень та рішення задач теорії чисел. Розробка відповідних програмних та апаратних обчислювальних засобів дозволяє підвищити продуктивність, швидкодію, зменшити апаратну та обчислювальну складності методів опрацювання багаторозрядних чисел (БРЧ) при знаходженні набору модулів досконалої та модифікованої досконалої форм системи залишкових класів (СЗК) в різних прикладних задачах обчислювальної математики.

Однією з найбільш складних задач такого класу методів та спецпроцесорів є реалізація алгоритмів опрацювання БРЧ, модульних операцій, модулярного множення та експоненціювання, пошуку квадратичних лишків, пошуку багаторозрядних простих чисел (БПЧ), тестів на простоту тощо.

Успіхи сучасного розвитку мікроелектронної, комп'ютерної техніки та використання потужних багатоядерних суперпроцесорів та кластерів дозволяє ефективно вирішити багато прикладних задач теорії чисел на основі двійкової арифметики ТЧБ Радемахера шляхом створення відповідних алгоритмічних, програмних та мікропроцесорних інструментальних засобів.

Аналіз стану застосування в сучасній комп'ютерній техніці арифметики різних систем числення свідчить, що в цілому стан цієї задачі далекий від вирішення. Значний вклад у розвиток теорії методів та високопродуктивних алгоритмів опрацювання багаторозрядних чисел в двійковій системі внесли відомі українські та зарубіжні вчені: Акушський І.Й., Палагін О.В., Брюхович Є.І., Романов С.І., Тарасенко В.П., Николайчук Я.М., Мельник А.О., Червяков В.П., Краснобаєв В.А., B. Omondi, N. Szabo, M. Hosseinzadeh, Lenstra H. W., Lakhani G., L.-L. Yang, L. Hanzo та інші.

1.2 Особливості та переваги криптографії з відкритим ключем

У 1976 р. У.Діффі та М.Хеллманом [30] було запропоновано новий тип криптографічної системи - система з відкритим ключем [public key cryptosystem]. У схемі з відкритим ключем є два ключі, відкритий [public] і секретний [private, secret], обрані таким чином, що їх послідовне застосування до масиву даних залишає цей масив без змін. Шифруюча процедура використовує відкритий ключ, що дешифрує - секретний. Дешифрування коду без знання секретного ключа практично нездійсненно; зокрема, практично нерозв'язна задача обчислення секретного ключа за відомим відкритому ключу. Основна перевага криптографії з відкритим ключем - спрощений механізм обміну ключами. При здійсненні комунікації по каналу зв'язку передається тільки відкритий ключ, що робить можливим використання для цієї мети звичайного каналу і усуває потребу в спеціальному захищеному каналі для передачі ключа.

Шифрування послань відкритим ключем принципово не занадто відрізняється від симетричного шифрування з використанням секретного ключа, але все ж має ряд переваг. Наприклад, відкрита частина ключової пари може вільно поширюватися без побоювань, що це перешкодить використовувати особистий ключ. Не потрібно розсилати копію свого відкритого ключа всім кореспондентам; вони зможуть отримати його на сервері вашої компанії або у вашого провайдера.

Інша перевага криптографії з відкритим ключем в тому, що вона дозволяє автентифікувати відправника послання. Оскільки ви - єдиний, хто має можливість зашифрувати інформацію вашим особистим ключем, всякий, хто використовує ваш відкритий ключ для розшифровки послання, може бути впевнений, що воно від вас. Таким чином, шифрування електронного документа вашим особистим ключем схоже з підписом на паперовому документі. Але не забувайте: немає жодних гарантій, що крім одержувача ваше послання не прочитає хтось ще. Використання криптографічних алгоритмів з відкритим ключем для шифрування послань - це досить

повільний обчислювальний процес, тому фахівці з криптографії придумали спосіб швидко генерувати короткий, унікальну виставу вашого послання, зване дайджестом послання. Дайджест можна зашифрувати, а потім використовувати як ваш цифровий підпис.

З появою систем з відкритим ключем поняття про захист інформації, а разом з ним функції криптографії значно розширилися. Якщо раніше основним завданням криптографічних систем вважалося надійне шифрування інформації, в даний час область застосування криптографії включає також цифровий підпис (автентифікацію), ліцензування, нотарізацію (засвідчення), розподілене керування, схеми голосування, електронні гроші та багато іншого. Найбільш поширені функції криптографічних систем з відкритим ключем - шифрування і цифровий підпис.

Цифровий підпис використовується для автентифікації текстів, переданих по телекомунікаційних каналах. Він аналогічний звичайному рукописному підпису і володіє її основними властивостями: засвідчує, що підписаний текст виходить саме від особи, яка його поставила, і не дає самій цій особі можливості відмовитися від зобов'язань, пов'язаних з підписаним текстом. Цифровий підпис являє собою невелику кількість додаткової інформації, переданої разом з підписуваним текстом. На відміну від шифрування, при формуванні підпису використовується таємний ключ, а під час перевірки - відкритий.

Через особливості алгоритмів, що лежать в основі систем з відкритим ключем, їх швидкодію при обробці одиничного блоку інформації звичайно в десятки разів менше, ніж швидкодію систем з симетричним ключем на блоці тієї ж довжини. Для підвищення ефективності систем з відкритим ключем часто застосовуються змішані методи, що реалізують криптографічні алгоритми обох типів. При шифруванні інформації вибирається випадковий симетричний ключ, викликається алгоритм з симетричним ключем для шифрування вихідного тексту, а потім система з відкритим ключем для

шифрування симетричного ключа. Комунікаційному каналу передається текст, зашифрований симетричним ключем, і симетричний ключ, зашифрований відкритим ключем. Для розшифровки дії проводяться у зворотному порядку: спочатку за допомогою секретного ключа одержувача розшифровується симетричний ключ, а потім за допомогою симетричного ключа - отриманий по каналу зашифрований текст. Для формування електронного підпису по підпису текст обчислюється його односпрямована хеш-функція (дайджест) [one-way hash function, digest], що представляє собою один короткий блок інформації, що характеризує весь текст в цілому, завдання відновлення тексту за його хеш-функції або підбору іншого тексту, що має ту ж хеш-функцію, практично нерозв'язна. При безпосередньому формуванні підпису, замість шифрування секретним ключем кожного блоку тексту секретний ключ застосовується тільки до хеш-функції; по каналу передається сам текст і сформований підпис хеш-функції. Для перевірки підпису знову обчислюється хеш-функція від отриманого по каналу тексту, після чого за допомогою відкритого ключа перевіряється, що підпис відповідає саме даному значенню хеш-функції. Алгоритми обчислення односпрямованих хеш-функцій, як правило, логічно тісно пов'язані з алгоритмами шифрування з симетричним ключем.

Описані гібридні методи шифрування і цифрового підпису поєднують в собі ефективність алгоритмів з симетричним ключем і властивість незалежності від додаткових секретних каналів для передачі ключів, притаманне алгоритмам з відкритим ключем. Криптографічна стійкість конкретного гібридного методу визначається стійкістю найслабшої ланки в ланцюзі, що складається з алгоритмів з симетричним і з відкритим ключем, вибраних для його реалізації.

1.3 Рівень надійності та криптографічного захисту комп’ютерних систем та мереж

Розглянемо шість мережевих операційних системах клієнт-сервер, що застосовуються на машинах IBM PC в Україні: UNIX, NetWare, VINES, LAN Server, LAN Manager і Windows NT. З цих систем лише VINES володіла вбудованою системою шифрування даних на сервері і була досить надійна до проникнення. Дещо більше дефектів у порівнянні з іншими мережевими ОС в безпеці системи можна виявити у NetWare, хоча це явно пов'язано з її широкою популярністю і граничної вивченістю Ще одна позитивна властивість NetWare - передача пароля зі станції на сервер в добре зашифрованому вигляді, що ускладнює його розшифровку при перехопленні пакетів. Всі зазначені системи здатні забезпечити прийнятний рівень безпеки даних при належній експлуатації.

Зазвичай безпеку комп'ютерних систем класифікують відповідно до критеріїв Національного центру комп'ютерного захисту США, названими "помаранчевою книгою". Вони дозволяють оцінювати системи за відносним рівнем безпеки і визначають чотири рівні захисту A, B, C і D в порядку убування надійності. Так як фактично рівнів безпеки більше, то букви модифікуються цифрами від 1 до 3. Наприклад, класифікація A1 позначає найбільш безпечні системи, C2 - достатню безпеку і D - мінімальну. Опишемо коротко властивості систем різних рівнів, розглядаючи їх зверху вниз:

- рівень A. Рівень A1 є вищим у захисті систем. Він відрізняється від рівня B3 лише надійним розподілом і додатковими гарантіями безпеки;

- рівень B. Рівень B1 відрізняється від C2 тим, що доступ до даних контролюється не тільки правами передавального користувача, але і приймаючого. Тобто не можна передати права на доступ до даних особі, яка не має на це прав. У класі B1 реалізується стратегія захисту даних і утиліти захисту чітко відокремлені від інших програм;

- рівень B2 покращує захист рівня B1 тим, що користується математичним описом функцій захисту і реалізує правило найменших привілеїв, відповідно до якого для виконання наданих користувачеві прав він повинен володіти найменшими можливими привілеями найменший можливий час. На цьому рівні на захист втягуються всі об'єкти системи і в першу чергу апаратні засоби;

- рівень B3 розширює рівень B2 тим, що гарантії захисту суворіші. Потрібно виділений адміністратор захисту системи, а також процедури відновлення після збою та оповіщення адміністратора про порушення захисту;

- рівень C. На рівні C1 система забезпечує так званий дискреціонний захист. Це означає, що всі користувачі, спільно обробляючи дані, мають однакові права. Це попереджає ненавмисну порчу даних один одного, але не рятує від вторгнення хакера. В таких системах кожен користувач має свій пароль і може захищати свої дані, визначаючи, кому він довіряє доступ до них. Вважають, що прості UNIX системи мають рівень C1. На рівні C2 система дещо суворіше, так як забезпечує контрольований доступ. Це означає, що права доступу до даних можуть бути передані лише частково: комусь тільки читання, а комусь ще і мережевими. Важливий елемент захисту таких систем - захист даних в пам'яті і на диску навіть від підглядання сторонніми;

- рівень D. На рівні D до системи не пред'являється спеціальних вимог і може не проводитись сертифікація. Таким чином, рівень D - системи, що не відповідають вищим класам захисту, або незахищені. До таких систем відноситься, наприклад, MS DOS.

Ця класифікація ієрархічна в тому сенсі, що системи високих рівнів секретності забезпечені всіма механізмами захисту більш низьких рівнів. Хоча ця класифікація використовується в основному для військових систем, вона неминуче поширюється і на цивільні та комерційні системи. Ця

класифікація вже була застосована до операційної системи UNIX, яка зараз є обов'язковою для більшості військових та урядових проектів в США.

Операційні системи для однорівневих локальних мереж абсолютно всі від LANtastic і Personal NetWare до Windows for Workgroup не відповідають ніяким критеріям на безпеку. Більш того, навіть доцільність застосування на них криптографічного захисту у вигляді NDisk з NU викликає серйозний сумнів. Дуже просто проникнути на чужий секретний диск в NWLite, тому не можна такого роду секретні диски тримати під одноранговою мережею. З цих систем виділяється вітчизняний SECRET DOS, званий ще SDOS, який являє собою комплекс засобів забезпечення безпеки комп'ютерних систем, побудованих на базі IBM PC і операційної системи MS-DOS. SDOS може використовуватися як на автономних комп'ютерах, так і на ЕОМ у складі локальних обчислювальних мереж. Ця система призначена для захисту інформації від несанкціонованого доступу та протидії спробам порушення нормального режиму роботи обчислювальної системи. В цілому вона не дотягує до рівня С1 і має такі відмінні особливості не тільки рекламного характеру:

а) часткова апаратна підтримка механізму ідентифікації;

б) непогане розмежування доступу для користувачів і груп;

в) автоматичне стирання залишкової інформації;

г) малі необхідні ресурси - близько 4 кілобайти RAM.

Застосування SDOS під одноранговою мережею може істотно посилити її таємність, хоча, як і будь-яке рішення, не знімає всіх проблем. До безперечних достоїнств SDOS можна віднести і те, що користувачі охоче розлучаються з нею, коли вона розвалюється від збоїв і не вимагають її відновлення.

Слід зазначити, що ряд фірм випускає по-справжньому секретні мережеві програмно-апаратні засоби, чудово захищені криптографічно. Так, фірма Harris випускає захищену локальну обчислювальну мережу LAN / SX і програмні засоби управління, що формують захищену обчислювальну середу

SX для багатопроцесорних EOM NightHawk цієї ж фірми. Ціна апаратних і програмних засобів мережевого Центру безпеки приблизно $ 25000, контролерів індивідуальної криптографічного захисту $ 5000, а програмних засобів $ 2500 за робоче місце. Число чисто програмних рішень значно більше. Корпорація OpenVision продає продукт OpenVSecure на основі технології Kerberus, що забезпечує захист мереж, за ціною близько $ 300 за станцію і $ 3000 за сервер. Фірма CyberSafe, що виділилася з широко відомої компанії Open Computing Security Group, теж на основі технології Kerberus випускає для широкого спектра платформ систему Challenger за цінами від $ 100 для станцій і до $ 5000 для розширених серверів. Компанія Axent Technologies продає системи захисту для різних платформ під загальною назвою OmniGuard за цінами від $ 400 за станцію і до $ 4000 за сервер. За дійсно безпечну систему потрібно викласти кілька сотень тисяч доларів, тому не дивно, що купивши їх фірми зазвичай не можуть звести кінці з кінцями по завершенню фінансового року.

Тепер на прикладі NetWare стисло опишемо використання функцій забезпечення секретності. Захист інформації в NetWare 3.11 організована по перекриваються рівнями: файли - директорії - користувачі - групи. Адміністратор може згенерувати систему захисту настільки складною, наскільки вимагають конкретні умови. Хоча навряд чи вона здатна перевищити рівень С2 в стандартній поставці, але і це дуже непогано. Дуже важливо, що конкретний користувач може бути обмежений в роботі за часом, до доступним йому робочим станціям займаної дискової пам'яттю і числом спроб реєстрації. У автора з практики склалася думка, що число спроб реєстрації більше 2 шкідливо. Статистика показує, що в 97% випадках користувачі входять в систему з першої ж спроби набору пароля, в 2% з другої, а подальший перенабор пароля мало чим допомагає. У той же самий час блокувати подальші спроби набору потрібно на строк не менше години - обідньої перерви. Для залишився 1% найзручніше передбачити еквіваленти для разового використання, видавати їх і знищувати в кінці дня, змінюючи

пароль. NetWare 3.11 хороша також і тим, що розумно перевіряє пароль користувача, захищаючи від введення вже побували у вжитку. Довжина пароля може бути встановлена залежно від вимог секретності. Для підвищеної секретності довжина пароля не повинна бути коротше 12 символів, коли користувач змушений задавати паролі не словом, а хоча б фразою. Принципово слабке місце NetWare - атрибут, який забороняє копіювання файлів. Він не витримує навіть такої атаки, як застосування DiskEdit з Нортоновских утиліт. Тому потрібно пам'ятати, що будь-яка інформація, що надається користувачеві лише на перегляд, може бути їм скопійована.

Незважаючи на те, що безпека систем обґрунтовується лише теоретично, але перевіряється вона тільки практично. І немає кращого способу атестації безпеки системи, ніж запросити пару хакерів зламати її, не повідомляючи попередньо персонал мережі. Така практика стала широко розповсюджуватися в США, наприклад, компанією Price Waterhouse. Штурм триває від 2 до 8 тижнів при солідному гонорар. Наймані хакери в результаті надають конфіденційну доповідь з оцінкою рівня доступності інформації та рекомендаціями по поліпшенню захисту. Довести безпеку системи таким прийомом все ж не можна, але хоча б можна її спростувати при явні прорахунки, а це вже немало.

1.4 Вибір перспективних методів криптоаналізу асиметричних систем

Оскільки деякі методи факторизації можна розпаралелити, зокрема метод решета числового поля, то розповсюдження персональних комп'ютерів веде до дискредитації системи шифрування RSA. Збільшення довжини ключа компенсує будь-які удосконалення алгоритмів факторизації. Відкритим залишається питання верхньої межі асимптотичної оцінки часу роботи

алгоритмів факторизації. Проте коли задача ефективності згаданих алгоритмів, що пов'язана з питанням довжини ключа, буде розв'язана, то алгоритм RSA стане непрактичним в криптографії. На сьогодні розклад числа на прості множники вважається трудомісткою задачею [18]. Однак це твердження не є доведеним, дана невизначеність стимулює розвиток існуючих алгоритмів факторизації та створення нових. Реальною є можливість доведення трудомісткості факторизації. Такий розвиток подій гарантував би безпеку системи RSA в просторі ключів деякого діапазону. Проте поставлену задачу теоретично можна успішно розв'язати на квантовому комп'ютері.

Виникає питання, чи не можна розв’язати задачу визначення таємного ключа, обходячись без факторизації. За твердженням 2.3 [29], цю задачу можна переформулювати як задачу знаходження такого d, що ed-1 ділиться на $\psi(n)$, де значення функції $\psi(n)$ чисельно дорівнює найменшому спільному кратному чисел p-1 і q-1 для $n=pq$. Але для такого d число $m=ed$-1 можна використати для розкладу n на множники за допомогою імовірнісної процедури IV.5.2 [29]. Отже, визначення таємного ключа для системи RSA є таким же важким, як факторизація модуля n. Тобто знаходження таємного ключа для системи RSA є еквівалентним до факторизації чисел, що є добутком двох простих, відносно поліноміальної імовірнісної звідності [29].

Суть іншого способу криптоаналізу системи шифрування RSA полягає у добуванні коренів e-го степеня за модулем n з числа c, де $c = m^e \bmod n$, m - вихідне повідомлення. Такий криптоаналіз дозволив би читати зашифровані повідомлення та підробляти цифровий підпис. Відомо, що такий різновид атаки не еквівалентний до факторизації . Проте не виявлено ефективних методів, що реалізують даний вид атаки в загальному випадку.

В літературних джерелах, які стосуються системи RSA, часто допускається, що використання так званих сильних простих чисел p і q для породження модуля перетворення n ускладнює задачу факторизації внаслідок такої властивості: відсутність малих дільників чисел p+1 і p-1. Причиною

таких заходів застереження є наявність методу факторизації Полларда. За останні десять років спростовують переваги сильних простих чисел методи, що використовують еліптичні криві. Нові методи факторизації дозволяють однаково успішно розкладати на множники числа, які містять прості або сильні прості числа. Тому традиційний вибір сильних множників в складі модуля перетворення системи RSA зараз не посилює її безпеку. Отже, для підвищення рівня інформаційної безпеки системи шифрування RSA необхідно збільшувати довжину ключів[12]. Зазначимо, що факторизація натурального числа, тобто задача пошуку всіх його простих дільників, еквівалентна до задачі пошуку хоча б одного простого дільника. На межі сучасних можливостей є факторизація чисел із 150 десятковими цифрами. Розклад на множники чисел, які мають 200 десяткових цифр, поки що залишається справою майбутнього.

Згідно з твердженням 4.4 [29], обчисливши два квадратні корені y та y' з деякого числа x за модулем n, можна твердити: найбільший спільний дільник $(y+y', n)$ є одним з дільників p або q числа n, за умови, що [2]

$$y \neq \pm y' (mod\ n). \tag{1.1}$$

Таким чином, поставлена задача зводиться до розв'язання конгруенції [29]

$$y^2 = (y')^2 (mod\ n). \tag{1.2}$$

Серед ефективних методів розв'язку останньої конгруенції заслуговують на увагу метод квадратичного решета та метод решета числового поля. Звичайно, можна перебирати всі прості числа до $\sqrt{m}$ і, ділячи на них m, знайти необхідний розклад числа. Візьмемо до уваги, що кількість простих чисел у вказаному проміжку асимптотично дорівнює $2\sqrt{m} \cdot \ln^{-1}(m)$ [12]. З

врахуванням цього знаходимо, що для числа *m* довжиною в сто десяткових знаків кількість простих чисел перевищує $4 \cdot 10^{42}$, на котрі прийдеться ділити *m* при розкладі його на множники. Комп'ютеру, що виконує мільйон ділень в секунду, для розкладу числа $m > 10^{99}$ таким способом на прості множники необхідно не менше, ніж 10^{35} років роботи.

Факторизація є однією з найважливіших задач теорії чисел та сучасної асиметричної криптографії [9]. Її суть полягає у розкладі деякого цілого числа у добуток простих співмножників. Відомі методи факторизації, в залежності від їх продуктивності, розбиваються на дві групи: експоненціальні та субекспоненціальні. Всі вони досить громіздкі, тому вимагають значних обчислювальних ресурсів для опрацювання багаторозрядних чисел. Крім того, сьогоднішній інтерес до проблеми факторизації продиктований також невизначеністю щодо теоретичного обґрунтування стійкості до розкриття асиметричних криптосистем.

Найбільш розповсюдженими для факторизації є алгоритми що базуються на теоремі Ферма. Обчислювальна складність методу Ферма для великорозрядних чисел досить велика, оскільки кількість ітерацій може становити 2^{300} - 2^{400} і тільки на єдиному кроці можливе однозначне рішення задачі факторизації. Для спрощення цієї задачі доцільно використати систему залишкових класів (СЗК) [4], яка дозволить виконати розпаралелення обчислень.

Задачі факторизації RSA від RSA-896 до RSA-2048 залишаються відкритими, з нагородами відповідно від $ 20000 до $ 200000. Наявні на сьогодні результати наведено в табл. 1.1. Одиницею вимірювання складності задачі в даному випадку є MIPS-рік – обсяг роботи, виконуваної протягом року процесором, що здійснює обробку одного мільйона команд у секунду, що приблизно еквівалентно виконанню $3*10^{13}$ команд. Так, машина на базі Pentium з частотою 2000 МГц показує швидкість 500 MIPS.

Таблиця 1.1 – Прогрес у розв’язку задач факторизації модуля *N*

Число десяткових знаків	Приблизне число бітів	Дата вирішення	Необхідне число MIPS-років	Використаний алгоритм
100	332	Квітень 1991 р.	7	Квадратичне решето
110	365	Квітень 1992 р.	75	Квадратичне решето
120	398	Червень 1993 р.	830	Квадратичне решето
129	428	Квітень 1994 р.	5000	Квадратичне решето
130	431	Квітень 1996 р.	500	Загальне решето числового поля
140	464	Лютий 1999 р.		Загальне решето числового поля

Продовження таблиці 1.1

Число десяткових знаків	Приблизне число бітів	Дата вирішення	Необхідне число MIPS-років	Використаний алгоритм
150	498	Відкритий	Відкликаний	
155	514	Серпень 1999 р.		Загальне решето числового поля
160	531	Квітень 2003 р.		Загальне решето числового поля
174	576	Грудень 2003 р.		Загальне решето числового поля

193	640	Березень 2004 р.		Загальне решето числового поля
212	704	Листопад 2005 р.		Загальне решето числового поля
232	768	Січень 2006 р.		Загальне решето числового поля
270	896	Жовтень 2009		Загальне решето числового поля
309	1024	Відкритий		
463	1536	*Відкритий*		
617	2048	*Відкритий*		

Загроза ключам великої довжини тут подвійна: безупинне зростання обчислювальної потужності сучасних комп'ютерів і безупинне удосконалення алгоритмів розкладання на множники. Генерують випадкову 16 – бітову стрічку S і побітово додають її до найбільш вживаних фрагментів значення хеш функції H_2.

Сучасні комп’ютерні мережі потребують надійного захисту, оскільки із зростанням швидкодії мережі скорочується час виконання паралельної реалізації криптоаналітичних алгоритмів, це дає можливість несанкціонованого доступу в комп’ютерних мережах.

Система шифрування інформації RSA є якісною. На сьогодні її можна зустріти у багатьох готових продуктах, що пропонуються на ринку програмного та апаратного забезпечення, зокрема у новому проекті RSA-OAEP. На сьогодні відсутні докази абсолютної стійкості для нового методу RSA – OAEP. Однак він достатньо стійкий проти адаптивних атак на фрагменти шифротексту, оскільки ця стійкість безпосередньо пов’язана із складністю обчислення оберненої односторонньої функції, яка застосовується у RSA алгоритмі. Тому вибрана задача криптоаналізу є перспективною. Доцільно значну частину досліджень спрямувати на

зменшення витрат обчислювальних ресурсів, шляхом розпаралелення на комп’ютерну мережу.

Задачі факторизації RSA від RSA-968 до RSA-2048 залишаються відкритими. Оскільки деякі методи факторизації можна розпаралелити, зокрема методи решета числового поля, Ферма, то розповсюдження персональних комп'ютерів веде до дискредитації системи шифрування RSA. Збільшення довжини ключа компенсує будь-які удосконалення алгоритмів факторизації. Відкритим залишається питання верхньої межі асимптотичної оцінки часу роботи алгоритмів факторизації у мережі.

1.5 Постановка задачі

Серед сучасних програмних засобів опрацювання багато розрядних простих чисел(БРЧ) виділяються ряд бібліотек для реалізації відповідних алгоритмів [15], а саме: бібліотека для роботи з БРЧ Ленстра, Arageli, NTL, GMP, Crypto++ та інші. Наведемо особливості організації деяких з них.

В роботі [8] наведений порівняльний аналіз швидкодії реалізації алгоритмів опрацювання БРЧ з використанням різних бібліотек (таблиця 1.1). В результаті дослідження отримані відповідні значення RANK[8] та наведені в таблиці 1.1. Для кожного алгоритму було виконано 100 тестів та встановлено відносні показники продуктивності.

Таблиця 1.1 Характеристики швидкодії алгоритмів з використанням різних бібліотек опрацювання БРЧ

Бібліотека	$RANK(ALG_i)$	AES-192	TDES	RSA (2048)	ECDSA (F2m=283)	HMAC SHA-1	SHA-1 (k=256) RANK

	MUL	POWMOD	xGCD	ECMUL			*sign*	*verf*	*sign*	*Verf*			
NTL	1,01	1,18	1,0	-	-	-	-	-	-	-	-	-	1,06
MIRACLE	3,58	2,62	3,15	1	1,00	-	2,29	1,72	2,02	2,76	-	1,12	1,40
Botan	3,41	5,21	1,09	1,42	2,16	1,98	1,00	1,00	1,60	1,95	1,15	1,13	1,67
Crypto++	4,49	5,04	16,82	4,35	2,68	1,90	1,12	1,15	1,00	1,00	1,05	1,07	2,19
OpenSSL	2,80	2,43	12,49	1,15	4,49	3,39	1,68	1,51	1,85	2,53	1,00	1,00	2,26
OpenPGP	2,87	2,31	3,11	1,41	1,12	1,37	2,35	1,73	1,54	2,00	-	1,06	1,79
GNU Crypto	1,0	1,0	1,01	1,24	1,38	1,00	2,71	1,80	1,75	2,13	1,06	1,08	1,35
CryptLib	5,25	4,17	8,59	1,7	1,03	1,47	1,09	1,08	1,07	2,05	1,02	1,06	1,82
LenstraLib	1,02	1,15	1,03	1,3	-	-	-	-	-	-	-	-	1,12

Для роботи з БРЧ бібліотека Lenstra є досить гнучким інструментом, що забезпечує найбільш ефективне використання обчислювальних ресурсів, а також реалізацію основних арифметичних та криптографічних алгоритмів, тому для реалізації розроблених методів обрана саме вона.

Факторизацію доцільно застосовувати для підбору модулів у СЗК [86], яка на даний час є однією з альтернатив двійковій системі числення, що дозволяє застосовувати нові підходи до організації обчислювальних систем при виконанні елементарних математичних операцій [86]. Хоча СЗК не позбавлена недоліків, до яких відносяться, зокрема, відсутність ділення та порівняння чисел, необхідність визначення умов переповнення розрядної сітки, однак її успішно можна застосовувати для додавання, віднімання та множення цілих багаторозрядних чисел [86, 87]. Безсумнівною перевагою СЗК є можливість виконання операцій над числами, які менші за вибрані модулі, розпаралелення процесу обчислень та відсутність міжрозрядних переносів.

Проведемо аналіз існуючих методів факторизації:

1) Повний перебір можливих дільників.

Найпростішим алгоритмом факторизації є повний перебір можливих дільників. Кількість можливих варіантів при використанні такого методу дорівнює *(n-1)/2*, де *n* – добуток чисел для факторизації.

Обчислювальна складність повного перебору можливих дільників дорівнює $O(N^{1/2})$, де $N=log_2 n$. Недоліком такого методу факторизації є висока обчислювальна складність, яка базується на складності виконання операцій ділення, та велика кількість ітерацій за умови, якщо дільники однакової розрядності [28].

2) Дослідження методу факторизації Ферма.

Відомий метод факторизації багаторозрядних чисел [28], що ґрунтується на представленні відомого числа n, яке є добутком двох шуканих БРПЧ $n = p \cdot q$, що задовольняє співвідношення згідно теореми Ферма:

$$n = A^2 - B^2,$$

де $p = A + B,\ q = A - B$.

Суть методу полягає в тому, що визначається ціла частина від квадратного кореня з n, $m = \lfloor \sqrt{n} \rfloor$. Для різних чисел $x=1,2,\ldots$ послідовно багатократно визначається значення згідно виразу $q(x) = (m + x)^2 - n$ до тих пір, поки отримане значення не буде рівне повному квадрату у вигляді цілого числа згідно десяткової арифметики.

Таким чином, в алгоритмі факторизації Ферма на кожній ітерації 1 раз виконується операція добування квадратного кореня з БРЧ *n* та виділення цілої частини *m,* а в кожній ітерації - додавання $m + x$ та піднесення до квадрату, віднімання від отриманого результату числа *n,* визначення квадратного кореня з отриманого результату та перевірки отриманого результату на цілочисельну повноту.

Суттєвим недоліком такого методу є значна обчислювальна складність, яка обумовлена виконанням великого числа обчислювально-складних операцій

у позиційній системі числення над БРЧ, які включають операції піднесення до степеня та добування квадратного кореня над числами, розрядність яких експоненційно зростає, починаючи з розрядності числа *n*.

При реалізації обчислювальних операцій факторизації методом Ферма на основі використання двійкової позиційної системи числення виникає висока кількість наскрізних переносів, що приводить до низької швидкодії обчислювальних засобів. Особливо це стосується опрацювання БРЧ при виконанні операцій додавання, множення, піднесення до квадрату, добування квадратного кореня, перевірки отриманого результату на цілочисельність та віднімання згідно двійково-десяткової та двійкової арифметик [35].

Наприклад, при факторизації 31-розрядного десяткового числа *n* = 2752899074514048103193505027141 для визначення 16-розрядних десяткових чисел p= 1232144355415431 та q= 2234234213235411 необхідно виконати операцію визначення кореня квадратного $m = \lfloor \sqrt{n} \rfloor$ = 1659186268781792, визначити цілу частину 1659186268781792 та виконати x=74003015543628 ітерацій для визначення числа q(x).

На завершальному кроці визначається число $(m+x)^2 - n$ = 3003945095300461569704458176400, корінь квадратний з якого є повним квадратом, тобто $m + x = 1733189284325420$.

В [28] розглянуто алгоритм Ферма (додаток Е) для факторизації БРЧ, який характеризуються експоненційною обчислювальною складністю *O(exp(N))* у зв'язку з експоненційним зростанням квадратів, які повинні задовільнити співвідношення $n = A^2 - B^2$.

3) (p — 1)-метод Полларда.

В праці Ішмухаметова Ш.Т. [28] проведено дослідження методу Полларда, який базується на властивостях малої теореми Ферма, згідно якої для кожного $a,\ 1 \le a < p,$ виконується умова $a^{p-1} \equiv 1 (\mathrm{mod}\ p)$ (Додаток Е).

Недоліком даного методу є необхідність зберігати велику кількість попередніх значень x_j, що задовольняють умови $(x_j - x_i) \equiv 0 (\mathrm{mod}\ p)$,

$(f(x_j) - f(x_i)) \equiv 0(\bmod p)$, з яких отримується рішення задачі факторизації. Тому обчислювальна складність методу складає $O(N^{1/4})$.

4) (p+1)- метод Вільямса

Метод Вільямса базується на рекурентній послідовності Люка-Лемєра (Lucas) [28] u_n,, яка визначається співвідношеннями $u_0=0, u_1=u, u_{n+1}=P \cdot u_n - Q \cdot u_{n-1}$, де P, Q фіксовані цілі числа. Даний метод представлений в додатку Е і схожий на $(p - 1)$ - метод Полларда, основною ідеєю якого є припущення гладкості числа $p+1$, де p — простий дільник факторизовуваного числа n.

В результаті досліджень методів факторизації побудовано графік для порівняння обчислювальних складностей досліджуваних алгоритмів, представлений на рисунку 1.2.

Особливістю алгоритму Вільямса є застосування мультиплікативних степеневих функцій, які характеризуються особливою складністю обчислень та різким зростанням розрядів обчислювальних чисел. Крім того, даний метод передбачає зберігання та генерування великих масивів БРПЧ, а також передбачає достатньо складного обчислення операції НСД. Граничною ефективністю методу Віллямса є обчислювальна складність $O(N^{1/4})$.

Аналіз показує, що метод Ферма характеризується експоненціальною складністю, але є алгоритмічно найпростіший і характеризується формалізацією границь кількості ітерацій, піддається розпаралеленню, не є рекурентним та ймовірнісним, характеризується обмеженим числом типів операцій (додавання, множення, експоненціювання, порівняння).

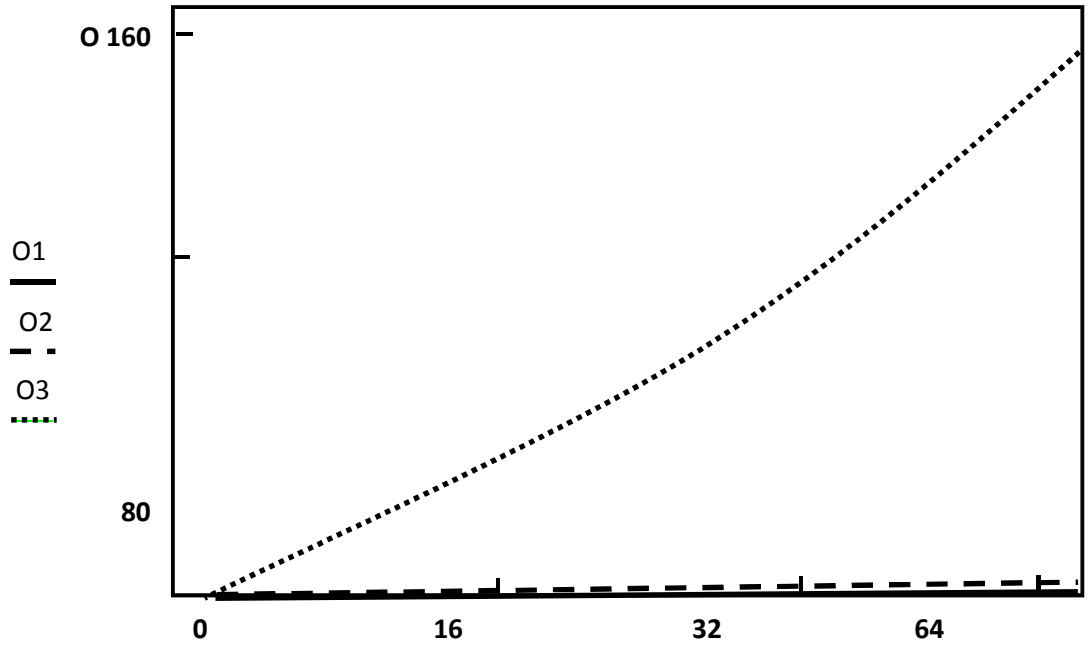

Рисунок 1.2 – Графіки обчислювальної складності алгоритмів факторизації: прямого перебору(O2), Поларда та Вільямса(O1), Ферма(O3).

Аналіз алгоритму факторизації БРЧ методами Поларда та Вільямса дозволяє встановити наступні їх характеристики обчислювальної, часової та прогнозно апаратної складності:

- методи Поларда та Вільямса в окремих випадках, коли факторизується добуток чисел різної розрядності, є більш ефективними у порівнянні з методом Ферма;

- розроблені на основі методу Поларда модифіковані алгоритми є ймовірнісними і не гарантують надійного кінцевого результату факторизації чисел;

- в даних методах не досліджені можливості глибокого розпаралелення обчислювальних операцій для підвищення швидкодії алгоритмів;

- не досліджені можливості застосування систем числення різних ТЧБ та відповідної модульної арифметики, в той час, коли в алгоритмі присутнє велике число модульних операцій.

При дослідження методу факторизації Ферма виявлено наступні переваги, що дозволяють підвищити його швидкодію та покращити характеристики

ефективності:

- метод дозволяє розпаралелення обчислювального процесу;
- Ферма-подібні методи відзначаються ефективністю при умові рівності множників;
- значні затрати пам’яті при факторизації БРЧ, що використовуються на опрацювання квадратів більших 1024 біт можна уникнути використовуючи властивості квадратичних лишків досліджені 2 пункті 2.4;
- експоненційну обчислювальну складність, яка виникає в процесі факторизації, можна спростити з допомогою використання СЗК ТЧБ Крестенсона.

Тому такі характеристики обґрунтовують перспективу його вдосконалення з використанням сучасних програмних засобів, у тому числі швидкодіючої модульної арифметики Крестенсона.

Програмні засоби, які повинні входити до складу розробки, мають бути оптимальними для реалізації існуючих задач. Програма розроблена за допомогою середовища програмування C++ Builder. В якості мови програмування вибрана C++, так як вона забезпечує і високу швидкість розробки, і добрі швидкісні характеристики розробленого програмного продукту. Також висуваються ще вимоги:

- IBM-сумісний комп'ютер на базі мікропроцесора Pentium II і вище;
- операційна система Windows XP;
- розмір оперативної пам'яті не менш 512 Mb;
- підтримка маніпулятора типу “миша”;

розмір простору на твердому диску 80 Гб.

2 ЕФЕКТИВНІ АЛГОРИТМИ ТА МЕТОДИ КРИПТОАНАЛІЗУ АСИМЕТРИЧНИХ СИСТЕМ У КРИПТОСИСТЕМАХ

2.1 Порівняльний аналіз методів факторизації

Криптоаналіз – наука, яка займається оцінкою сильних і слабких сторін методів шифрування, а також розробкою методів, що дозволяють зламувати криптосистеми.

Задача криптоаналізу полягає в тому, щоб визначити ймовірність злому шифру і, таким чином, оцінити його застосовність в тій чи іншій галузі.

Як бачимо, на сьогоднішній день для злому криптосистем з відкритим ключем успішно застосовуються методи повного перебору ключем, аналіз ключового генератора, факторизація / дискретне логарифмування і аналіз по побічним каналам.

Разом з виникненням в криптографії нових понять і методів розширилося і коло криптографічних додатків теорії чисел.

До елементарної та аналітичної теорії чисел все більш широко використовується алгебраїчна теорія чисел (тести на простоту з застосуванням сум Гауса і Якобі, криптосистеми, засновані на квадратичних полях, решето числового поля) і арифметично аналітична геометрія (факторизація за допомогою еліптичних кривих, криптосистеми, засновані на еліптичних і гіпереліптічних кривих і абелевих групах).

На рисунку 2.1 наведено класифікацію способів несанкціонованого доступу до інформації.

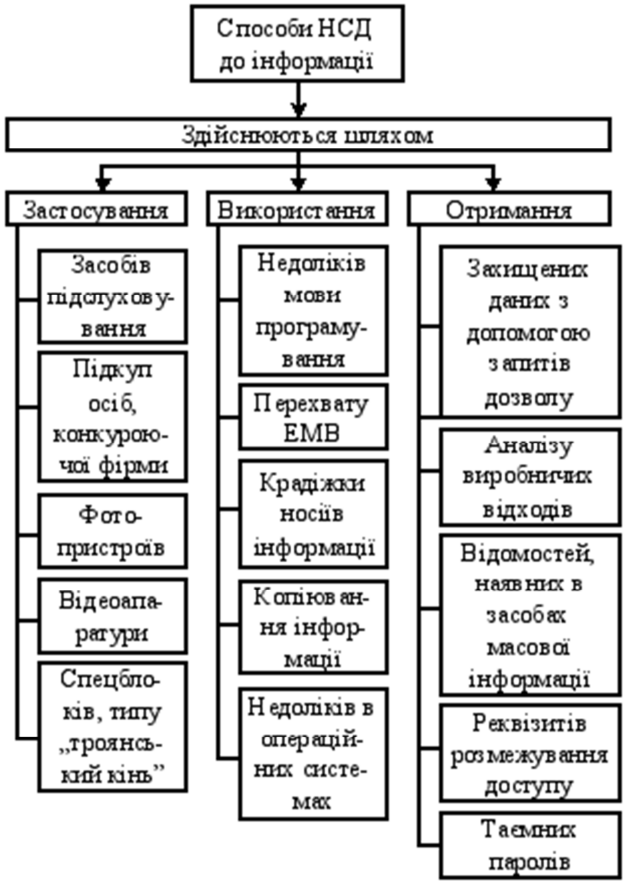

Рисунок 2.1 – Способи НСД

На даний момент широко використовується алгоритм з використанням квадратичного решета числового поля, використання якого є трудомістким і має експоненційну складність.

Натуральне число називається простим, якщо воно ділиться тільки на себе і на 1. Число, яка не є простим, називається складеним. Очевидно, що будь-яке просте число, не рівне 2, є непарним. Існують ознаки подільності цілих чисел на різні прості числа, наприклад, щоб число в десятковому вигляді ділилося на 3 і 9 достатньо, щоб сума його цифр ділилася на 3 і 9 відповідно. Щоб число ділилося на 5, достатньо, щоб його остання цифра була 0 або 5. Такі власні ознаки подільності можна використовувати, якщо потрібно зменшити множину кандидатів перевірки на простоту або відсікти завідомо складені числа. Альтернативним способом отримання простих чисел є решето Ератосфена, приписуване давньогрецькому вченому Ератосфену Киренському, що жив приблизно в 276 - 194 м до н.е. Для

знаходження безлічі простих до заздалегідь обраної верхньої межі B ми спочатку виписуємо послідовність всіх непарних чисел від 3 до B. Потім вибираємо перше число у списку, тобто трійку, і залишаючи його у списку, викреслюємо всі кратні 3, починаючи з 6. Потім переходимо до другого числа списку (п'ятірці) і викреслюємо його кратні, залишивши саму п'ятірку і т.д., поки не дійдемо до кінця списку. У списку що залишився будуть тільки прості числа.

Факторизацією натурального числа називається його розкладання в добуток простих множників. Існування і єдиність (з точністю до порядку слідування множників) такого розкладу випливає з основної теореми арифметики.

Ця задача має велику обчислювальну складність. Один з найпопулярніших методів криптографії з відкритим ключем, метод RSA, заснований на трудомісткості завдання факторизації довгих цілих чисел.

Залежно від складності алгоритми факторизації можна розбити на дві групи [16]. Перша група – експоненціальні алгоритми, складність яких експоненціально залежить від довжини вхідних параметрів (тобто від довжини самого числа в бінарному представленні). Для позначення їх складності прийнята O-нотація. Ця нотація дозволяє враховувати у функції f (n) лише найбільш значущі елементи, відкидаючи другорядні.

Наприклад, у функції $f(n) = 2n^2 - 5n + 1$, при досить великих n компонента n^2 буде значно перевершувати інші складові, і тому характерна поведінка цієї функції визначається саме цією компонентою. Решту компонент можна відкинути і умовно записати, що дана функція має оцінку поведінки (в сенсі швидкості росту її значень) виду $O(N^2)$. Фраза «алгоритм факторизації з обчислювальною складністю $O(N^{1/2})$ означає, що зі збільшенням параметра N , що характеризує кількість вхідної інформації алгоритму, час роботи алгоритму не може бути обмежено величиною, яка росте повільніше, ніж $N^{1/2}$.

До цієї групи належать алгоритми: перебір можливих дільників; метод факторизації Ферма; Р-алгоритм Поларда; метод квадратичних форм Шенкса; Метод Лемана.

Друга група – субекспоненціальні алгоритми, це алгоритми, які працюють більш, ніж за поліноміальний час, але менш, ніж за експоненціальний час («субекспоненціальні»). Для позначення їх складності прийнята L-нотація:

$$L_N(\alpha, c) \coloneqq \mathrm{O}(\exp((\mathrm{c} + \mathrm{o}(1))(\log N)^{\alpha} (\log\log N)^{1-\alpha})) , \qquad (2.1)$$

де N – число, що підлягає факторизації, $0 < \alpha < 1$ і c – деякі константи.

До них відносять: алгоритм Діксона; метод безперервних дробів; метод квадратичного решета; метод еліптичних кривих; решето числового поля; спеціальний метод решета числового поля; загальний метод решета числового поля.

Обчислювальна складність експоненціальних алгоритмів:

- перебір можливих дільників – найбільш тривіальний алгоритм факторизації з обчислювальною складністю $\mathrm{O}(\mathrm{N}^{1/2})$.
- Р-алгоритм Поларда має складність $\mathrm{O}(\mathrm{N}^{1/4})$;
- метод квадратичних форм Шенкса має складність $\mathrm{O}(\mathrm{N}^{1/4})$;
- метод Лемана має складність $\mathrm{O}(\mathrm{N}^{1/3})$.

Обчислювальна складність субекспоненціальних алгоритмів:

- алгоритм Діксона має складність $L_N(1/2, 2\sqrt{2})$;
- метод безперервних дробів має складність $L_N(1/2, \sqrt{2})$;
- метод квадратичного решета має складність $L_N(1/2, 1)$;
- метод еліптичних кривих має складність $L_p(1/2, \sqrt{2})$, де P – найменше просте, яке ділить.
- решето числового поля: спеціальний метод решета числового поля зі складністю $L_N(1/3, (32/9)^{\frac{1}{3}})$ (метод застосуємо для факторизації чисел

тільки спеціального виду); загальний метод решета числового поля зі складністю $L_N(1/3, (64/9)^{\frac{1}{3}})$ (метод застосуємо до всіх чисел).

Всі ці методи досить трудомісткі, тому вимагають значних обчислювальних ресурсів для чисел великої довжини. Однак теоретичне обґрунтування необхідної складності таких обчислень або, іншими словами, існування високих нижніх оцінок не доведено, тому питання про існування алгоритму факторизації з поліноміальною складністю на класичному комп'ютері для виконання факторизації є однією з важливих відкритих проблем сучасної теорії чисел.

2.2 Математичні основи базису Крестенсона

Сучасні комп’ютерні мережі є швидкісними та забезпечують високу продуктивність роботи під час обробки і збереження інформації. Проте вони потребують надійного захисту, оскільки із зростанням швидкодії мережі скорочується час виконання паралельної реалізації криптоаналітичних алгоритмів, це дає можливість несанкціонованого доступу в комп’ютерних мережах.

Система числення залишкових класів (СЗК) базису Крестенсона, розроблена Акушинським І.Я. та Юдіцьким Д.І., особливо її цілочисельна форма, широко використовувалась, починаючи з 70-х рокув минулого століття для побудови швидкодіючих спеціалізованих процесорів систем повітряної оборони колишнього СРСР. Нормалізована форма СЗК, запропонована науковою школою професора Николайчука Я.М., активно використовується та застосовується при дослідженні двовимірних (матричних) форм систем числення базису Радемахера та Галуа.

Відомо, що двійкова система числення, яка використовується в сучасних комп’ютерних системах, має певні недоліки – наявність між розрядних

зв'язків та велику розрядність. Тому актуальним є застосування непозиційних систем числення, в яких відсутні вказані недоліки. Саме такою є СЗК, або, як її ще називають, представлення чисел у базисі Крестенсона [7,11], причому найбільш фундаментально досліджено цілочисельну форму в системі залишкових класів[20]. Хоча вона не набула значного поширення у зв'язку з необхідністю визначення умов переповнення, складністю та громіздкістю зворотнього перетворення чисел у десяткову систему числення, а також складнощами реалізації операцій ділення та порівняння, але СЗК можна ефективно використовувати у мультибазисних процесорах, спеціалізованих обчислювальних машинах для виконання операцій додавання, віднімання та множення, наприклад, у задачах лінійної алгебри (матрично-векторні операції) тощо.

Необхідно відмітити, що ця система особливо ефективна при обчисленнях з великими числами [2, 15].

Фундаментальною основою СЗК є теорія чисел [11], зокрема, властивості китайської теореми про залишки. Будь-яке ціле додатне число N у десятковій системі числення представляється в СЗК у вигляді набору найменших додатних залишків від ділення цього числа на фіксовані цілі додатні попарно взаємно прості числа

Числа $p_1, p_2, \dots, p_n$ ($N_{10}=(b_1, b_2, \dots, b_n)_{p1, p2, \dots, pn}$, де $b_i=N \bmod p_i$), які називаються модулями (n – кількість модулів). При цьому повинна виконуватись умова $0 \le N \le P-1$, де $P=\prod_{i=1}^{n} p_i$.

Теоретичною базою цілочисельного перетворення СЗК базису Крестенсона є доведення існування в кільці а набору сукупності елементів y_1, y_2, … y_j, …, y_k, які задовольняють систему Діафантових рівнянь (порівнянь) [28]

$$\left.\begin{array}{c} y_j \equiv 1(\bmod A_j) \\ y_j \equiv 0(\bmod \prod^{k} A_j), \forall_j = \overline{1,k} \end{array}\right\},$$

де Aj – символ ідеалу причому

$$A_j + \prod_{i \neq j}^{k} A_i = A; \forall_j = \overline{1,k}.$$

В СЗК елементи Yj позначається через Bj [28] і називається ортогональними базисами. При цьому {Bj} повинні задовольняти систему порівнянь

$$\left.\begin{array}{c} \sum_{j=1}^{k} B_j \equiv 1(\mathrm{mod} \prod_{j=1}^{k} P_j); \\ B_j \equiv 1(\mathrm{mod}\, P_j) \end{array}\right\}, \tag{2.2}$$

де Pj – попарно взаємно прості модулі, звідки слідує, що:

$$\left.\begin{array}{c} B_j \equiv 1(\mathrm{mod} \prod_{j=1}^{k} P_j; \\ B_j \equiv A(\mathrm{mod} P_j), i \neq j, \forall = \overline{1,k} \end{array}\right\}. \tag{2.3}$$

Рішення (2.2) означає, що кожне Bj ділиться без остачі на всі Pi, якщо $i \equiv j$ і кожне Bj ділиться без остачі на добуток Pj, тобто

$$B_j = m_j \prod_{i \neq j}^{k} P_{i,}$$

(2.4)

де mj- деяке ціле число у діапазоні $1 \leq m_j \leq P_j - 1$.

При цьому діапазон однозначного представлення чисел Nj у цілочисельних СЗК знаходиться у границях

$$0 \leq N_j \leq \prod_{j=1}^{k} P_j - 1.$$

Якщо позначити

$$P = \prod_{j=1}^{k} P_j \quad \text{то} \quad B_j = m_j \frac{P}{p_j} \tag{2.5}$$

mj може бути знайдено з рішення діафантового рівняння[4]

$$m_j \frac{P}{p_j} \equiv 1 (\mathrm{mod}\, P_j). \tag{2.6}$$

Викладені теоретичні положення визначають пару перетворень цілочисельної СЗК у вигляді прямого

$$N_j = res \sum_{j=1}^{k} b_j B_j (\mathrm{mod}\, P) \tag{2.7}$$

та зворотного

$$b_j = res N_k (\mathrm{mod}\, P_j); \forall_j = \overline{1,k}, \tag{2.8}$$

де bj – найменший невід'ємний залишок числа Nj по модулю Pj, тобто $0 \le b_j \le P_j - 1$.

Представимо розглянуту СЗК у вигляді системи ортогональних функцій Віленкіна-Крестенсона (рис. 2.1).

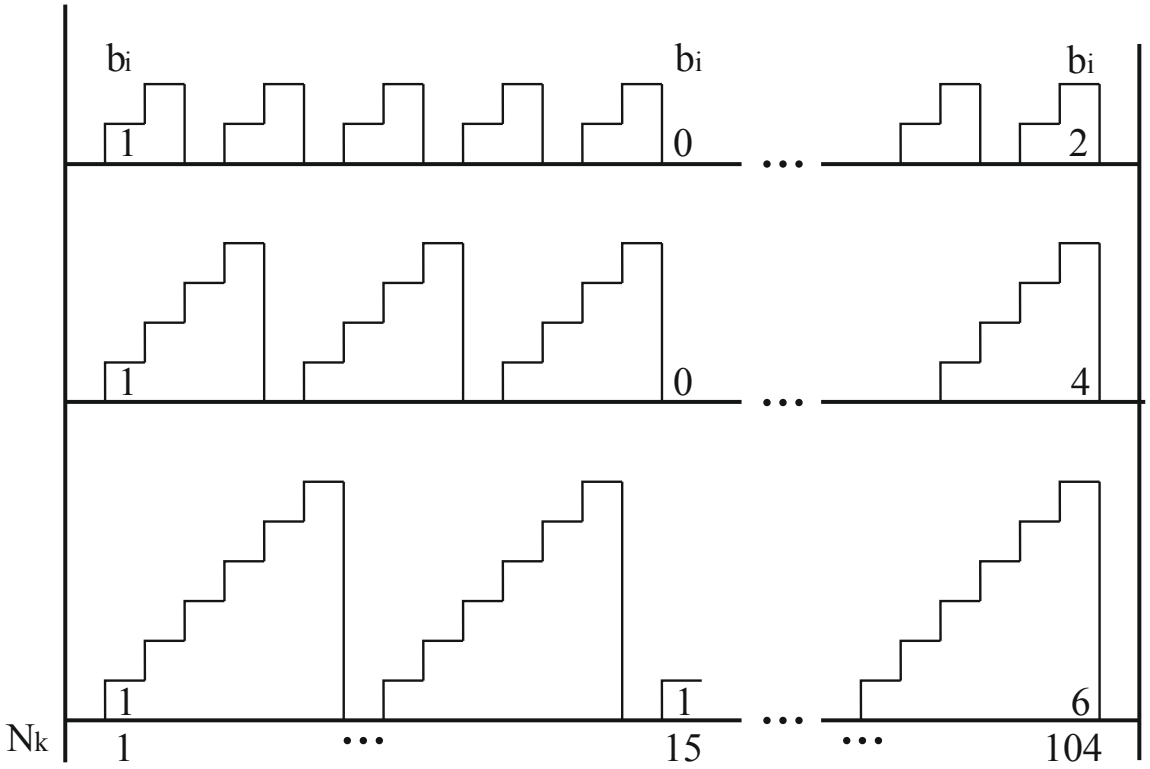

Рисунок 2.1 – Сукупність ортогональних функцій Віленкіна-Крестенсона у системі модулів СЗК:P_1=3; P_2=5; P_3=7.

З рисунку 2.1. видно розраховане число N_1=1 подано кодом Галуа в СЗК (1,1,1), ортогональне базисне число $B_1 = 15 = (0,0,1)$, а $N_k = P - 1 = 104 = (P_1 - 1,\ P_2 - 1, P_3 - 1) = (2,4,6)$.

Традиційною галуззю застосування такого класу кодів Галуа та цілочисельної форми СЗК є побудова високопродуктивних процесорів та обчислювальних машин, які працюють у модульній арифметиці СЗК[4, 11].

Особливістю такої арифметики залишкових класів є відсутність наскрізних переносів, які існують у відомій 2-й системі числення базису Радемахера. Тобто:

додавання виконується згідно наступного виразу P_1 P_2… P_j…P_k

$$\frac{x=(a_1 a_2 \ldots, a_j, \ldots, a_k)}{y=(b_1 b_2 \ldots, b_j, \ldots, b_k)} \qquad (2.9)$$
$$z=(c_1 c_2 \ldots, c_j, \ldots, c_k)$$

де $c_j = res(a_j + b_j) mod P_j, j \epsilon \overline{1,k}.$

Подібно без наскрізних переносів виконується операція множення цілих чисел P_1 P_2… P_j…P_k

$$\begin{array}{l} x=(a_1 a_2 \ldots, a_j, \ldots, a_k) \\ \underline{y=(b_1 b_2 \ldots, b_j, \ldots, b_k)} \\ z=(d_1 d_2 \ldots, d_j, \ldots, d_k) \end{array} \qquad (2.10)$$

де $d_j = res(a_j * b_j) mod P_j, j \epsilon \overline{1,k}.$

Наприклад x=13; y=7; (P1 P2 P3)=(3, 5, 7).

Визначимо коди СЗК чисел x та y згідно (2.6):

x → res13(mod3)=1; a_1 =1;
res13(mod5)=3; a_2 =3; x=(1,3,6).
res13(mod7)=6; a_3 =6;

y → res7(mod3)=1; b_1 =1;
res7(mod5)=2; b_2 =2; y=(1,2,0).
res7(mod7)=0; b_3 =0;

Виконаємо операції додавання та множення в кодах СЗК згідно (2.9) та (2.10):

$$\begin{array}{c} P_1\ P_2 P_3 \\ x=(1,\ 3,\ 6) \\ \underline{+\,y=(1,\ 2,\ 0)} \\ z=(2,\ 0,\ 6) \end{array}; \quad \begin{array}{c} P_1\ P_2 P_3 \\ x=(1,\ 3,\ 6) \\ \underline{\times\, y=(1,\ 2,\ 0)} \\ z=(1,\ 1,\ 0) \end{array}.$$

Перевіряємо результати згідно виразу (2.7):

$z = (2 \cdot 70 + 0 \cdot 21 + 6 \cdot 15) \bmod 105 = 20;\ D = (1 \cdot 70 + 1 \cdot 21 + 0 \cdot 15) \bmod 105 = 91.$

В окремих випадках, коли з якихось умов відомо, що $x \geq y$ в СЗК однозначно виконується операція віднімання. Тобто:

$$\begin{array}{c} P_1 P_2 P_3 \\ x = (1, 3, 6) \\ \underline{y = (1, 2, 0)} \\ V = (0, 1, 6) \end{array}$$

і $x - y = (0 \cdot 70 + 1 \cdot 21 + 6 \cdot 15) \bmod 105 = 6$.

Подібним чином, якщо відомо, що У- взаємно-прості числа з модулями в СЗК можливо виконати операцію ділення шляхом еквівалентного множення згідно виразу:

$$\begin{array}{c} D=(d_1,d_2,d_3) \\ \underline{Y=(y_1,y_2,y_3)} \\ X=(a_1,a_2,a_3) \end{array} \qquad (2.11)$$

Використання особливих наборів модулів цілочисельної СЗК (система числення залишкових класів) у діапазоні кодування чисел Ферма та Мерсена [28] забезпечили ефективну реалізацію швидкого перетворення Фур'є-Галуа і відповідне ефективне застосування ТЧБ в галузі захисту інформації [14, 2].

2.3 Метод Ферма

Метод факторизації Ферма - алгоритм факторизації (розкладання на множники) непарного цілого числа, запропонований П'єром Ферма (1601-1665) в 1643 році [1].

Метод заснований на пошуку таких цілих чисел x та y , які задовольняють співвідношенню $x^2 - y^2 = n$, що веде до розкладання $n = (x + y)(x - y)$.

Метод Ферма заснований на теоремі про представлення числа у вигляді різниці двох квадратів: якщо $n > 1$ непарна, то існує взаємно однозначна відповідність між розкладаннями на множники $n = a \cdot b$ і уявленнями у

вигляді різниці квадратів $n = x^2 - y^2$ с $x > y > 0$, задане формулами $x = \frac{a+b}{2}$, $y = \frac{a-b}{2}$ $a = x + y$ $b = x - y$ [6].

Якщо задана факторизація $n = a \cdot b$, то має місце співвідношення: $n = a \cdot b = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$. Таким чином, виходить представлення у вигляді різниці двох квадратів.

Зворотно, якщо дано, що $n = x^2 - y^2$, то праву частину можна розкласти на множники: $(x + y)(x - y)$.

Для розкладання на множники непарного числа n шукається пара чисел (x, y) таких, що $x^2 - y^2 = n$, або $(x + y)(x - y) = n$. При цьому числа $(x + y)$ та $(x - y)$ є множниками n, можливо, тривіальними (тобто одне з них дорівнює 1, а інше $-n$).

Рівність $x^2 - y^2 = n$ рівносильно $x^2 - n = y^2$, тобто тому, що $x^2 - n$ є квадратом.

Пошук квадрата такого виду починається з $x = \lceil\sqrt{n}\rceil$ - найменшого числа, при якому різниця $x^2 - n$ невід'ємна. Для кожного значення $k \in \mathbb{N}$ починаючи з $k = 1$, обчислюють $(\lceil\sqrt{n}\rceil + k)^2 - n$ і перевіряють, чи не є це число точним квадратом. Якщо не є - то k збільшують на одиницю і переходять на наступну ітерацію.

Якщо $(\lceil\sqrt{n}\rceil + k)^2 - n$ є точним квадратом, тобто $x^2 - n = \left(\lceil\sqrt{n}\rceil + k\right)^2 - n = y^2$ то отримано розкладання:

$n = x^2 - y^2 = (x + y)(x - y) = a \cdot b$ в якому $x = \left(\lceil\sqrt{n}\rceil + k\right)$.

Якщо воно є тривіальним та єдиним, то n - просте.

На практиці значення виразу на $(k + 1)$ -ому кроці обчислюється з урахуванням значення на k-ому кроці :

$$(s + 1)^2 - n = s^2 + 2s + 1 - n, \tag{2.12}$$

де $s = \lceil\sqrt{n}\rceil + k$.

Таким чином алгоритм Ферма має експоненційну оцінку і не ефективний для розкладання довгих чисел [6]. Можна поліпшити метод Ферма, виконавши спочатку пробне ділення числа n на числа від 2 до деякої константи B, виключивши тим самим малі подільники n до B включно, і тільки потім виконати пошук методом Ферма.

2.4 Удосконалений алгоритм Ферма

В даному методі Ферма доцільно скористатися теоретико-числовим базисом Крестенсона [20], який дозволяє зменшити обчислювальну складність за рахунок зменшення розрядностей чисел, над якими проводяться операції.

Тобто в рівнянні:

$$x^2 = y^2 - n \tag{2.13}$$

робимо наступне перетворення:

$$x^2 mod\ p = y^2 - n\ mod\ p, \tag{2.14}$$

в результаті отримали $x^2 \equiv (y^2 - n) mod\ p$

Для рішення даного порівняння доцільно скористатися символами Якобі, які дозволяють однозначно вказувати, чи обчислюється корінь за модулем.

Нехай p – просте, a – ціле число. Символ Лежандра $\left(\frac{a}{p}\right)$ визначається так:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, \text{якщо } p \text{ ділиться на } a \\ 1, \text{ якщо } a \in Q_p \\ -1, \text{ якщо } a \in \bar{Q}_p \end{cases}.$$

Число a, яке не ділиться на непарне просте p, є квадратичним лишком за модулем p тоді і тільки тоді, коли $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, і квадратичним нелишком тоді і тільки тоді коли $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

За теоремою Ферма [25, 27] $a^{p-1} \equiv 1 \pmod{p}$ при НСД(a, p) = 1 та НСД(2, p) = 1. Або:

$$\left(a^{\frac{p-1}{2}} + 1\right) * \left(a^{\frac{p-1}{2}} - 1\right) \equiv 0 \bmod p.$$

Звідси вираз в одній із дужок ділиться на p. Обидві дужки не можуть ділитися на p, оскільки тоді на p ділилася б і їх різниця, яка дорівнює 2, а за умовою теореми p – непарне просте число. Якщо a є квадратичним лишком, то $a = x^2 \pmod{p}$ для деякого такого x, що НСД(x, p) = 1. Маємо: $a^{\frac{p-1}{2}} \equiv \left(x^2\right)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \bmod p$, тобто $a^{\frac{p-1}{2}} \equiv 1 \bmod p$ або $a^{\frac{p-1}{2}} - 1$ ділиться на p. Якщо a є квадратичним нелишком, то $a^{\frac{p-1}{2}} - 1$ не ділиться на p, звідки $a^{\frac{p-1}{2}} + 1$ повинно ділитися на p, або $a^{\frac{p-1}{2}} \equiv -1 \bmod p$.

$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$. Якщо число a є квадратичним лишком за модулем p, то за означенням символа Лежандра $\left(\frac{a}{p}\right) = 1$, а за критерієм Ейлера $a^{\frac{p-1}{2}} \pmod{p} \equiv 1$. Відповідно якщо число a є квадратичним нелишком за модулем

p, то $\left(\frac{a}{p}\right) = -1$ і $a^{\frac{p-1}{2}} \pmod p \equiv -1$, звідки і випливають наступні властивості символа Лежандра [12, 16]:

1. $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p$. Вказана властивість є наслідком критерію Ейлера.

Зокрема $\left(\frac{1}{p}\right) = 1$ та $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$.

Отже $-1 \in Q_p$ якщо $p \equiv 1 \pmod 4$ та $-1 \in \overline{Q}_p$ якщо $p \equiv 3 \pmod 4$.

2. $\left(\frac{a*b}{p}\right) = \left(\frac{a}{p}\right)*\left(\frac{b}{p}\right)$. Властивість випливає з послідовності очевидних порівнянь:

$$\left(\frac{a*b}{p}\right) \equiv (ab)^{\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} * b^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right)*\left(\frac{b}{p}\right) \pmod p.$$

Зокрема, якщо $a \in Z_p^*$, то $\left(\frac{a^2}{p}\right) = 1$ та $\left(\frac{a^2 b}{p}\right) = \left(\frac{b}{p}\right)$.

3. Якщо $a \equiv b \pmod p$, то $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$. Властивість випливає з того, що числа одного класу є одночасно або квадратичними лишками, або нелишками. Випливаючи з цієї властивості, можна записати:

$$\left(\frac{a}{p}\right) = \left(\frac{a+pt}{p}\right), t \in Z.$$

4. $\left(\frac{1}{p}\right) = 1$. Одиниця є квадратичним лишком для довільного непарного простого p. Ця властивість випливає з того, що порівняння $x^2 \equiv 1 \pmod p$ завжди має розв'язки $x = \pm 1 \pmod p$.

5. $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$.

Якщо p = 8k ± 1, то $\frac{p^2-1}{8} = \frac{(8k\pm1)^2-1}{8} = \frac{64k^2\pm16k}{8} = 8k^2 \pm 2k$ – парне число.

Якщо $p = 8k \pm 3$, то $\frac{p^2-1}{8} = \frac{(8k\pm3)^2-1}{8} = \frac{64k^2\pm48k+8}{8} = 8k^2 \pm 6k + 1$ – непарне число.

Отже $\left(\frac{2}{p}\right)$=1, якщо $p \equiv 1$ або 7 (mod 8) та $\left(\frac{2}{p}\right)$= -1, якщо $p \equiv 3$ або 5 (mod 8).

6. Закон взаємності непарних простих чисел. Якщо p – просте непарне число, відмінне від q, то

$$\left(\frac{p}{q}\right) * \left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}.$$

Помноживши цю рівність на $\left(\frac{p}{q}\right)$, отримаємо: $\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}} * \left(\frac{p}{q}\right)$. Якщо виконується хоча б одна з рівностей p (mod 4) ≡ 1 чи q (mod 4) ≡ 1, то $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$, інакше $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$.

Символ Якобі є узагальненням символу Лежандра на випадок коли n є непарним, але не обовя’язково простим.

Нехай n – непарне ціле число, n ≥ 3. Відомо, що $n = p_1^{k_1} p_2^{k_2} ... p_t^{k_t}$, де p_i – прості числа. Символ Якобі $\left(\frac{a}{n}\right)$ визначається так:

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{k_1}\left(\frac{a}{p_2}\right)^{k_2}\cdots\left(\frac{a}{p_t}\right)^{k_t}.$$

Зазначимо, що якщо n просте, то символ Якобі стає символом Лежандра.

Властивості символа Якобі

1. $\left(\frac{a}{n}\right)$ може приймати одне з трьох значень: -1, 0 чи 1. При цьому $\left(\frac{a}{n}\right)=0$ тоді і тільки тоді коли НСД(a, n) $\neq$ 1.

2. $\left(\frac{a*b}{n}\right)=\left(\frac{a}{n}\right)*\left(\frac{b}{n}\right)$. Якщо $a \in Z_n^*$, то $\left(\frac{a^2}{n}\right)=1$.

3. $\left(\frac{a}{m*n}\right)=\left(\frac{a}{m}\right)*\left(\frac{a}{n}\right)$.

4. Якщо $a \equiv b \pmod n$, то $\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right)$.

5. $\left(\frac{1}{n}\right)=1$.

6. $\left(\frac{-1}{n}\right)=(-1)^{\frac{n-1}{2}}$. Отже $\left(\frac{-1}{n}\right)=1$, якщо $n \equiv 1 \pmod 4$ та $\left(\frac{-1}{n}\right)=-1$, якщо $n \equiv 3 \pmod 4$.

7. $\left(\frac{2}{n}\right)=(-1)^{\frac{n^2-1}{8}}$.

Отже $\left(\frac{2}{n}\right)=1$, якщо $p \equiv 1$ або 7 (mod 8) та $\left(\frac{2}{n}\right)=-1$, якщо $p \equiv 3$ або 5 (mod 8).

8. $\left(\frac{m}{n}\right)=\left(\frac{n}{m}\right)(-1)^{\frac{(m-1)(n-1)}{4}}$.

З властивостей символу Якобі випливає, що якщо n непарне, а число a подати у вигляді $a = 2^k a_1$, де a_1 – непарне число, то

$$\left(\frac{a}{n}\right) = \left(\frac{2^k}{n}\right)\left(\frac{a_1}{n}\right) = \left(\frac{2^k}{n}\right)\left(\frac{n \bmod a_1}{a_1}\right)(-1)^{\frac{(a_1-1)(n-1)}{4}}$$

Ця формула дає можливість обчислити значення символа Якобі не маючи розкладу числа n на прості множники.

На відміну від символу Лежандра, символ Якобі $\left(\frac{a}{n}\right)$ не визначає, чи є число a квадратичним лишком за модулем n. Справді, якщо $a \in \mathrm{Q_n}$, то $\left(\frac{a}{n}\right) = 1$, але з того що $\left(\frac{a}{n}\right) = 1$ не випливає $a \in \mathrm{Q_n}$.

Нехай n – непарне ціле число, $\mathrm{n} \geq 3$. Число a будемо називати псевдопростим за модулем n, якщо $\left(\frac{a}{n}\right) = 1$. Множину псевдопростих чисел позначатимемо через $\widehat{Q}_n = \mathrm{J}_n - \mathrm{Q}_n$, де $\mathrm{J}_n = \{a \in \mathrm{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\}$.

На рисунку 2.1 наведено блок-схему розробленого алгоритму, що забезпечує процес факторизації за допомогою використання символів Якобі.

Тоді удосконалення методу Ферма полягає в тому, що ми завідома відкидаємо, ті значення $(y^2 - n)$ для яких не існує корінь за модулем і зменшуємо розрядність обчислень за рахунок модульної операції.

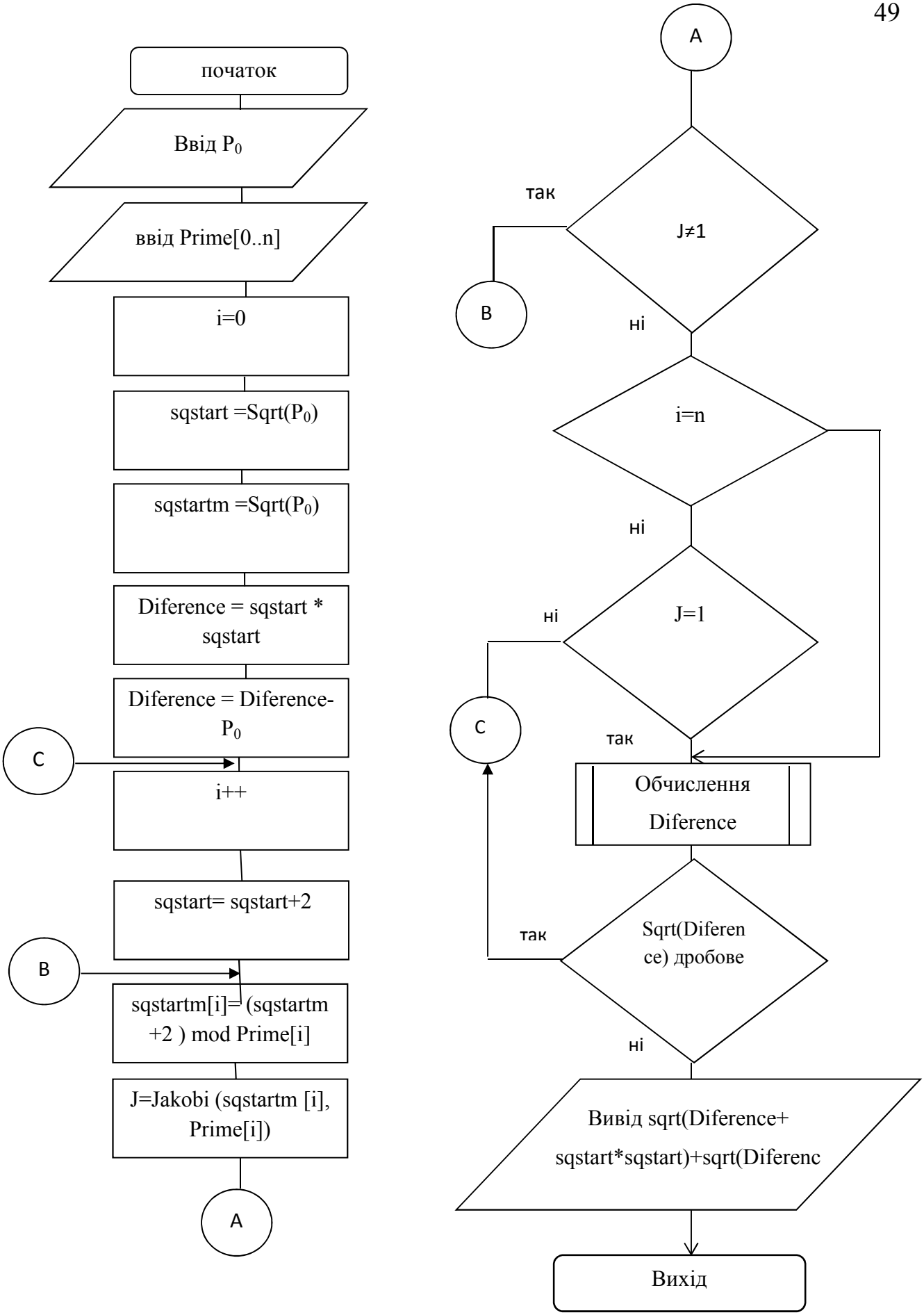

Рисунок 2.1 – Блок-схема удосконаленого алгоритму факторизації методу Ферма

Тобто x знаходимо з наступного виразу:

$$x \equiv \sqrt{(y^2 - n)} mod\ p \ . \qquad (2.15)$$

Удосконалений алгоритм факторизації буде мати вигляд, зображений на рисунку 2.1.

Незважаючи на підвищення швидкодії за рахунок використання ТЧБ Крестенсона та символів Якобі, не для всіх видів чисел даний алгоритм буде ефективний. При факторизації чисел виду $2^n - 1$ доцільно використати теоретико-числовий базис Радемахера.

Застосування на практиці різних методів розкладання чисел показало, що час виконання алгоритму безпосередньо залежить від його типу та обчислювальної складності.

У даному розділі розглянуто деякі алгоритми факторизації натуральних чисел, а також провели порівняльну характеристику по групах структур.

Таким чином, можна зробити висновок, що для факторизації натуральних чисел існує досить багато способів, але це завдання далеко не тривіальне, і досить затратне в плані часу, про це говорить оцінка складності алгоритмів факторизації. Деякі алгоритми можуть вирішувати поставлену задачу століттями. Але вирішення даної проблеми не стоїть на місці, оскільки великі відкриття і нові досягнення в цій галузі з'являються знову і знову. Ці досягнення обумовлені не тільки розвитком обчислювальної потужності комп'ютерів і мереж, а й розвитком тих областей теоретичної математики, які донедавна служили областю інтересів лише професійних математиків-фахівців в абстрактній алгебрі і теорії чисел.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Обґрунтування вибору програмного засобу

C++ Builder являє собою SDI-додаток, головне вікно якого містить персоналізовану інструментальну панель (ліворуч) і палітру компонентів (праворуч). Крім цього, за умовчанням при запуску C++ Builder з'являються вікно інспектора об'єктів (ліворуч) і форма нового додатка (справа). Під вікном форми програми знаходиться вікно редактора коду [3, 17].

Форми є основою додатків C++ Builder. Створення користувальницького інтерфейсу додатка полягає в додаванні у вікно форми елементів об'єктів C++ Builder, званих компонентами. Компоненти C++ Builder розташовуються на палітрі компонентів, виконаної у вигляді багатосторінкового блокнота. Важлива особливість C++ Builder полягає в тому, що він дозволяє створювати власні компоненти і налаштовувати палітру компонентів, а також створювати різні версії палітри компонентів для різних проектів.

Компоненти розділяються на видимі (візуальні) і невидимі (невізуальні). Візуальні компоненти з'являються під час виконання точно так само, як і під час проектування. Прикладами є кнопки і редаговані поля. Невізуальні компоненти з'являються під час проектування як піктограми на формі. Вони ніколи не помітні під час виконання, але мають певну функціональність (наприклад, забезпечують доступ до даних, викликають стандартні діалоги Windows та ін.).

Для додавання компонента у форму можна вибрати мишею потрібний компонент в палітрі і клацнути лівою клавішею миші в потрібному місці проектованої форми. Компонент з'явиться на формі, і далі його можна переміщати, міняти розміри та інші характеристики.

Кожен компонент C++ Builder має три різновиди характеристик: властивості, події та методи.

Якщо вибрати компонент з палітри і додати його до форми, інспектор об'єктів автоматично покаже властивості та події, які можуть бути використані з цим компонентом. У верхній частині інспектора об'єктів є список, що випадає, що дозволяє вибирати потрібний об'єкт із наявних на формі.

Властивості є атрибутами компонента, що визначають його зовнішній вигляд і поведінку. Багато властивостей компонента в колонці властивостей мають значення, яке встановлюється за замовчуванням (наприклад, висота кнопок). Властивості компонента відображаються на сторінці властивостей (Properties). Інспектор об'єктів відображає опубліковані (published) властивості компонентів. Крім published-властивостей, компоненти можуть і найчастіше мають загальні (public), опубліковані властивості, які доступні тільки під час виконання програми. Інспектор об'єктів використовується для установки властивостей під час проектування. Список властивостей розташовується на сторінці властивостей інспектора об'єктів. Можна визначити властивості під час проектування або написати код для видозміни властивостей компонента під час виконання додатку.

При визначенні властивостей компонента під час проектування потрібно вибрати компонент на формі, відкрити сторінку властивостей в інспекторі об'єктів, вибрати обумовлена властивість і змінити його за допомогою редактора властивостей (це може бути просте поле для введення тексту або числа, що випадає список, що розкривається список, діалогова панель і т.д.).

Сторінка подій (Events) інспектора об'єктів показує список подій, розпізнаваних компонентом (програмування для операційних систем з графічним призначеним для користувача інтерфейсом, зокрема, для Windows 95 або Windows NT вважає опис реакції проекту на ті чи інші події, а сама операційна система займається постійним опитуванням комп'ютера з метою виявлення настання якої-небудь події). Кожен компонент має свій власний набір обробників подій. В C ++ Builder варто писати функції, звані обробник подій, і зв'язувати події із цими функціями. Створюючи оброблювач тієї чи

іншої події, ви доручаєте програмі виконати написану функцію, якщо ця подія відбудеться.

Для того щоб додати оброблювач подій, потрібно вибрати на формі за допомогою миші компонент, якому необхідний оброблювач подій, потім відкрити сторінку подій інспектора об'єктів і двічі клацнути лівою клавішею миші на колонці значень поряд з подією, щоб змусити C ++ Builder згенерувати прототип обробника подій і показати його в редакторі коду. При цьому автоматично генерується текст порожній функції, і редактор відкривається в тому місці, де слід вводити код. Курсор позиціонується всередині операторних дужок. Далі потрібно ввести код, який повинен виконуватися при настанні події. Оброблювач подій може мати параметри, які вказуються після імені функції в круглих дужках.

Метод є функцією, яка пов'язана з компонентом, і яка оголошується як частина об'єкта. Створюючи обробники подій, можна викликати методи, використовуючи наступну нотацію: ->, наприклад:

Edit1-> Show ();

Відзначимо, що при створенні форми пов'язані з нею модуль і заголовний файл з розширенням * .h генеруються обов'язково, тоді як при створенні нового модуля він не зобов'язаний бути пов'язаний з формою (наприклад, якщо в ньому містяться процедури розрахунків). Імена форми і модулі можна змінити, причому бажано зробити це відразу після створення, поки на них не з'явилося багато посилань в інших формах і модулях.

Файли, що утворять додаток - форми та модулі - зібрані в проект. Менеджер проектів показує списки файлів і модулів проекту та дозволяє здійснювати навігацію між ними. Можна викликати менеджер проектів, вибравши пункт меню View / Project Manager. За замовчуванням знову створений проект одержує ім'я Project1.cpp [23, 26].

За замовчуванням проект спочатку містить файли для однієї форми та вихідного коду одного модуля. Однак більшість проектів містять кілька форм і модулів. Щоб додати модуль або форму до проекту, потрібно клацнути

правою кнопкою миші і вибрати пункт New Form з контекстного меню. Можна також додавати існуючі форми та модулі до проекту, використовуючи кнопку Add контекстного меню менеджера проектів і вибираючи модуль або форму, яку потрібно додати. Форми і модулі можна видалити в будь-який момент протягом розробки проекту. Однак, через те, що форма зв'язана завжди з модулем, не можна видалити одне без видалення іншого, за винятком випадку, коли модуль не має зв'язку з формою. Видалити модуль з проекту можна, використовуючи кнопку Remove менеджера проектів.

Якщо вибрати кнопку Options у менеджері проектів, відкриється діалогова панель опцій проекту, у якій можна вибрати головну форму проекту, визначити, які форми будуть створюватися динамічно, які параметри компіляції модулів (у тому числі створених в Delphi, так як C ++ Builder може включати їх в проекти) і компонування.

Важливим елементом середовища розробки C ++ Builder є контекстне меню, що з'являється при натисканні на праву клавішу миші та дозволяє швидкий доступ до найчастіше використовуваних команд.

Зрозуміло, C ++ Builder володіє вбудованою системою контекстно-залежної допомоги, доступної для будь-якого елементу інтерфейсу і є великим джерелом, довідкової інформації про C ++ Builder.

Створення додатків в C ++ Builder: першим кроком у розробці проекту C ++ Builder є створення проекту. Файли проекту містять згенерований автоматично вихідний текст, що стає частиною проекту, коли воно скомпільовано та підготовлене до виконання. Щоб створити новий проект, потрібно вибрати пункт меню File / New Application.

C ++ Builder створює файл проекту з ім'ям за замовчуванням Project1.cpp, а також make-файл з ім'ям за замовчуванням Project1.mak. При внесенні змін до проекту, таких, як додавання нової форми, C ++ Builder оновлює файл проекту.

Проект або додаток звичайно мають кілька форм. Додавання форми до проекту створює наступні додаткові файли:

- файл форми з розширенням .DFM, що містить інформацію про ресурси вікон для конструювання форми;

- файл модуля з розширенням .CPP, що містить код на C ++;

- заголовний файл з розширенням .H, що містить опис класу форми.

Коли ви додаєте нову форму, файл проекту автоматично оновлюється.

Для того щоб додати одну або більше форм до проекту, виберіть пункт меню File / New Form. З'явиться порожня форма, що буде додана до проекту. Можна скористатися пунктом меню File / New, вибрати сторінку Forms і вибрати підходящий шаблон з репозиторія об'єктів.

Для того, щоб просто відкомпілювати поточний проект, з меню Compile потрібно вибрати пункт меню Compile. Для того щоб відкомпілювати проект і створити виконує файл для поточного проекту, з меню Run потрібно вибрати пункт меню Run. Компілювання проекту є інкрементним (перекомпілюються тільки модулі).

Якщо при виконанні проекту виникає помилка, C ++ Builder робить паузу у виконанні програми та показує редактор коду з курсором, встановленим на операторі, що є джерелом помилки. Перш ніж робити необхідну корекцію, слід перезапустити програму, вибираючи пункт меню Run з контекстного меню або з меню Run, закрити додаток і лише потім вносити зміни в проект. У цьому випадку зменшиться ймовірність втрати ресурсів Windows.

3.2 Проектування програмного продукту

При розробці інтерфейсу необхідно користуватись наступними принципами:

а) розробка інтерфейсу зазвичай починається з визначення завдання або набору завдань, для яких продукт призначений;

б) просте повинно залишатися простим;

в) користувачі не повинні замислюватись над тим, як влаштована програма;

г) інтерфейс повинен бути орієнтованим на людину;

д) передбачати поведінку та звички користувачів;

е) розробляти інтерфейс виходячи з принципу найменшої можливої кількості дій з боку користувача.

При проектуванні програмного продукту було створено форму, що на рисунку 3.1.

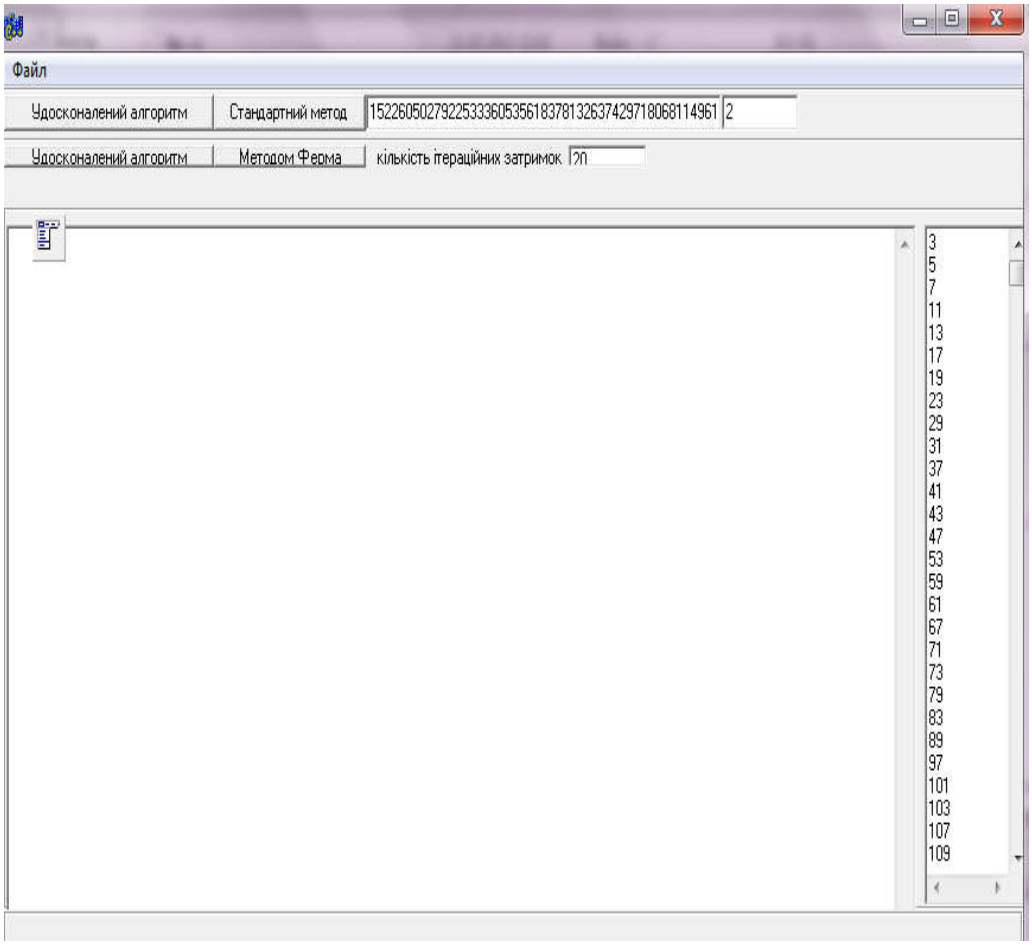

Рисунок 3.1 – Головне вікно додатку на етапі проектування

При проектуванні додатку були використані компоненти наступних типів:

- TpageControl - TPageControl (Набір сторінок) являє собою складені одна на іншу сторінки, причому доступ до кожної сторінки, яка містить свій набір елементів управління, здійснюється через вкладки, на яких можна написати назву. Даний елемент управління зручний тим, що дозволяє ефективно використовувати обмежений простір екрана, створюючи ефект книги, яку можна розкрити на будь-якій сторінці.

- TtabSheet - компонент TTabSheet представляє собою окрему закладку об'єкта TPageControl. В C ++ Builder компонент TPageControl зберігає індексований масив закладок в свою властивість Pages.

- Tbutton – кнопка;

- Tmemo - поля Memo можна лише частково розглядати як класичний список, що складається з незалежних рядків. З одного боку, властивість Text являє вміст цього об'єкта як один довгий символьний рядок, що містить керуючі символи ("перехід рядка", "Повернення каретки", горизонтальна табуляція). Ручне або автоматичне включення таких символів в текст дозволяє відображати довгий рядок у вигляді багаторядкового документа. З іншого боку, об'єкт Memo розташовує властивістю Lines, що забезпечує доступ до кожного рядка по її індексу;

- TstatusBar - компонент зручний для відображення різної службової та налагоджувальної інформації, яка дозволяє вести візуальний контроль за діями додатка. Компонент StatusBar можна поділити на будь-яку кількість окремих міні-статусних вікон.

Для роботи з змінними великого розміру була використана спеціалізована бібліотека А. Ленстра, яка складається з Lip.h та Lip.c [26]. Бібліотека Lenstra є досить гнучким інструментом, що забезпечує найбільш ефективне використання обчислювальних ресурсів, а також реалізацію основних арифметичних та криптографічних алгоритмів, тому для реалізації розроблених методів обрана саме вона.

Сама бібліотека є оптимізованою для використання в криптосистемах та містить в собі наступні функції, що були використані в проектованому додатку: zsadd, zadd, zsub,zsmul, ,zsq, zsqin, zsmod, zmod, zlshift, zrshift, zbit, zgetbits, zsetbit, ziszero, z2log.

Для отримання розкладу числа на дільники була створена наступна функція:

```
firstmod=0;
from_memo_to_vector(&prime_mods,Memo2);
bool finded=false;

zsread(Edit1->Text.c_str(),&P0);
zsqrt1(P0,&sqrtP0,&sqrtP0diference);
zsadd(sqrtP0,1,&sqrtP0);
print(P0," P0 ");
print(sqrtP0," sqrtP0 ");
print(sqrtP0diference," sqrtP0diference ");
find_end_element(sqrtP0,one,&endstep);
find_end_element(sqrtP0,one,&firststep);
print(endstep," endstep ");
print(firststep," firststep ");
from_memo_to_vector(&prime_mods,Memo2);

long sqrtP0diference0=0;
long step0=0;
long prevsize=0;
long count_of_modulus=StrToInt(Edit2->Text);
vector<bool>stepvector;

AnsiString s=""
//1000
```

```
;
//*************************************
for (long i=0;i<count_of_modulus;i++)
{
s="";
stepvector.resize(prime_mods[i]);
//globalvector.resize(1);
sqrtP0diference0=zsmod(sqrtP0diference,prime_mods[i]);
s=s+IntToStr(is_square_modulus(sqrtP0diference0,prime_mods[i]))+" ";
stepvector[0]=is_square_modulus(sqrtP0diference0,prime_mods[i]);
step0=zsmod(firststep,prime_mods[i]);
   for (long j=1;j<prime_mods[i];j++){
   sqrtP0diference0=(sqrtP0diference0+step0)%prime_mods[i];
   s=s+IntToStr(is_square_modulus(sqrtP0diference0,prime_mods[i]))+" ";
   stepvector[j]=is_square_modulus(sqrtP0diference0,prime_mods[i]);
   step0=(step0+2)%prime_mods[i];
   };

   if (i==0){
   globalvector.resize(stepvector.size());
   for(int h=0;h<stepvector.size();h++)globalvector[h]=stepvector[h];
    //  from_vector_memo1(globalvector,Memo1);
   }else
   {
   prevsize=globalvector.size();
      globalvector.resize(stepvector.size()*globalvector.size());

for(int h=0;h<globalvector.size();h++)
if (globalvector[h%prevsize]&&(!stepvector[h%stepvector.size()]))
{globalvector[h]=false;}else {globalvector[h]=globalvector[h%prevsize];};
```

```
    };
   //  Memo1->Lines->Add(s);

};
from_vector_memo1(globalvector,Memo1);
long i=1;
verylong total=0;
verylong temp=0;
verylong dif=0;
verylong final=0;
bool limit=false;
//sqrtP0
zcopy(sqrtP0diference,&final);
while (!limit&&i<100){
      zadd(final,endstep,&final);
      zsadd(endstep,2,&endstep);
      if (globalvector[i%globalvector.size()]==1) {
           // Memo1->Lines-
>Add(IntToStr(globalvector[i%globalvector.size()]));
            zsqrt(final,&temp,&dif);

            if (ziszero(dif)==1){limit=true;zsadd(endstep,-2,&endstep);};

      };
     // print(final," final ");
      i++;
};
     Memo1->Lines->Add("Факторизовано ...");
     print(endstep," endstep ");
```

Для виведення змінних великого розміру була розроблена наступна функція:

```
void from_vector_memo1(vector<bool>& temp,TMemo *Memo1){
    AnsiString s="";
    for (long i=0;i<(temp.size());i++)
    {
    s=s+IntToStr(temp[i])+"";
    };
    Memo1->Lines->Add(s);
};
```

Для перевірки залишків, на можливість квадратисності створена наступна функція:

```
bool TForm1::is_square_modulus(long part,long modul){
if (part!=0){
long m=1;
bool is=false;
for(long i=1;i<=modul/2;i++){
m=(i*i)%modul;
if (part==m){is=true;break;}
};
return is;   }else{return true;};
};
```

Для обчислення кінцевого елемента створена наступна функція:

```
void TForm1::find_end_element(verylong sqrtp0,verylong count,verylong *endelement){

zadd(sqrtp0,count,endelement);
```

```
zsmul(*endelement,2,endelement);
zsub(*endelement,one,endelement);
};
```

Для створення вектора ключів створена наступна процедура:

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{
from_memo_to_vector(&prime_mods,Memo2);
//globalvector=0;
makevectorkey(Edit1->Text,prime_mods,globalvector);
from_vector_memo1(globalvector,Memo1);
}
```

3.3 Тестування та відлагодження програмного забезпечення

Тестування програмного забезпечення — це процес, що використовується для виміру якості розроблюваного програмного забезпечення. Зазвичай, поняття якості обмежується такими поняттями, як коректність, повнота, безпечність, але може містити більше технічних вимог, які описані в стандарті ISO 9126. Тестування - це процес технічного дослідження, який виконується на вимогу замовників, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись. До цього процесу входить виконання програми з метою знайдення помилок.

Основна ідея запропонованого методу полягає в тому, що квадрати цілих чисел можна представити у вигляді суми непарних чисел, кількість яких дорівнює даному числу [29]:

$$n^2 = \sum_{i=1}^{n} (2i-1). \qquad (3.1)$$

Відобразимо залишки квадратів цілих чисел по декількох простих модулях p_j, тобто $a_1(p_1, p_2, ..., p_m) = b_1^1, b_2^1, \ldots, b_m^1$, $a_2(p_1, p_2, ..., p_m) = b_1^2, b_2^2, \ldots, b_m^2$, $a_n(p_1, p_2, ..., p_m) = b_1^n, b_2^n, \ldots, b_m^n$, де $a_i = i^2$, $b_j^i = a_i (mod\ p_j), 1 \le i \le n,\ 1 \le j \le m$, m – кількість модулів.

Використовуючи властивість циклічності залишків квадратів, шукані залишки отримуємо за допомогою рекурентної формули:

$$b_j^i = \left(b_j^{i-1} + z_i^j\right) mod\ p_j,$$

де $z_j^i = z_i\ mod\ p_j$, z_i=2i-1.

Відповідні результати по модулях 3, 5, 7, 11 представлені в таблиці 3.1 і показують, що кількість квадратичних лишків для кожного модуля становить (p_j+1)/2 (включаючи 0). Це випливає з рівності $n^2 mod\ p_j$=($-n^2$) $mod\ p_j$=(p_j-n)^2 $mod\ p_j$.

Далі використовується властивість, коли квадрат числа по будь-якому простому модулю є квадратичним лишком по цьому ж модулю [14].

Відповідно, знайшовши залишки даного числа за простим модулем p_j, можна отримати для нього одне із значень часткового ключа факторизації y_n^j, яке дорівнює 0 або 1, причому значенню 1 відповідає випадок, коли $\Delta_n^j = \left(\Delta_n\right)^2\ mod\ p_j = \left(\Delta_{n-1}^j + z_j^i\right) mod\ p_j$ є квадратичним лишком по p_j і тоді Δ_n може бути цілим числом.

Таблиця 3.1 - Результати по модулях 3, 5, 7, 11

N	z_i	a_n	p_1=3		p_2=5		p_3=7		p_4=11	
			z_1^i	b_1^i	z_2^i	b_2^i	z_3^i	b_3^i	z_4^i	b_4^i
1	1	1	1	1	1	1	1	1	1	1
2	3	4	0	1	3	4	3	4	3	4
3	5	9	2	0	0	4	5	2	5	9
4	7	16	1	1	2	1	0	2	7	5

Продовження таблиці 3.1

N	z_i	a_n	p_1=3		p_2=5		p_3=7		p_4=11	
			z_1^i	b_1^i	z_2^i	b_2^i	z_3^i	b_3^i	z_4^i	b_4^i
5	9	25	0	1	4	0	2	4	9	3
6	11	36	2	0	1	1	4	1	0	3
7	13	49	1	1	3	4	6	0	2	5
8	15	64	0	1	0	4	1	1	4	9
9	17	81	2	0	2	1	3	4	6	4
10	19	100	1	1	4	0	5	2	8	1
11	21	121	0	1	1	1	0	2	10	0

Значення залишків в процесі факторизації для числа n визначається так: $Y_n = y_n^1 \wedge y_n^2 \wedge \ldots \wedge y_n^m$. Отримані результати представлено в таблиці 3.2.

Таблиця 3.2 - Ознаки квадратичності по модулях 3, 5, 7, 11

k	N	z_i	$(\Delta_n)^2$	p_1=3			p_2=5			p_3=7			p_4=11			Y_n
				z_1^i	Δ_n^1	y_n^1	z_2^i	Δ_n^2	y_n^2	z_3^i	Δ_n^3	y_n^3	z_4^i	Δ_n^4	y_n^4	
1	62	123	33	0	0	1	3	3	0	4	5	0	2	0	1	0
2	63	125	158	2	2	0	0	3	0	6	4	1	4	4	1	0
3	64	127	285	1	0	1	2	0	1	1	5	0	6	10	0	0
4	65	129	414	0	0	1	4	4	1	3	1	1	8	7	0	0
5	66	131	545	2	2	0	1	0	1	5	6	0	10	6	0	0
6	67	133	678	1	0	1	3	3	0	0	6	0	1	7	0	0
7	68	135	813	0	0	1	0	3	0	2	1	1	3	10	0	0
8	69	137	950	2	2	0	2	0	1	4	5	0	5	4	1	0
9	70	139	1089	1	0	1	4	4	1	6	4	1	7	0	1	1

Вектор $Y(Y_1, Y_2, \ldots, Y_n)$, в якому Y_i=0 або 1, утворює загальний ключ факторизації. В даному прикладі Y(000000001). Це означає, що на дев'ятій ітерації можливий випадок, коли Δ_n буде цілим числом, що і підтверджується обчисленням: $\sqrt{1089} = 33$.

Слід відмітити, що частковий ключ отриманий в процесі факторизації y_n^j володіє властивістю циклічності, період якої дорівнює модулю p_j, тому при розрахунках немає необхідності визначати всі поточні значення $(\Delta_n)^2$ [35]. Число $(\Delta_n)^2$, корінь квадратний з якого може бути цілим числом (Y_n=1), отримується згідно формули обчислення суми арифметичної прогресії, яку утворює послідовність z_i, за таким виразом:

$$(\Delta_n)^2 = (z_{i\ min} + k)(k-1) + (\Delta_n)^2_{min}. \qquad (3.2)$$

Метод реалізується згідно таких кроків:

1) вхід: число P_0;

2) обчислюється $\sqrt{P_0}$;

3) заокруглюється до більшого цілого $\lceil \sqrt{P_0} \rceil$;

4) шукається $\lceil \sqrt{P_0} \rceil^2 - P_0$;

5) присвоюється $i=1$;

6) шукається $pN = (2\sqrt{P_0} - 1)\ mod\ m$;

7) присвоюється $sN=1$;

8) обчислюється $pN=pN+2$;

9) шукається $sN= sN+ pN$;

10) обчислюється $pN=(pN\ mod\ m)$; $sN =(sN\ mod\ m)$;

11) присвоюється $i =i+1$;

12) записується ознака приналежності залишку sN множині залишків цілих квадратів, якщо sN є залишком квадрату, тоді в ключі факторизації відбувається присвоєння Y_i=1, інакше Y_i=0;

13) якщо $i<m$, то перехід на крок 10;

14) присвоюється $pN = (2\sqrt{P_0} - 1) \bmod m$;

15) присвоюється $K=1$;

16) якщо $Y[K \bmod Y.\text{size}] = 0$, то відбувається перехід на крок 25;

17) STEP=$2\lfloor\sqrt{P0}\rfloor + 1$;

18) присвоюється $i=0$;

19) присвоюється $sum=0$;

20) присвоюється $sum=sum+step$;

21) присвоюється $step=step+2$;

22) i++;

23) якщо $i<K$, то відбувається перехід на крок 20;

24) якщо корінь квадратний з *sum* є цілим числом, тоді число факторизовано, крок 27;

25) присвоюється $K = K + 1$;

26) відбувається перехід на крок 16;

27) вихід

Приклад факторизації на основі властивостей залишків квадратів:

P_0=3811 , для генерації ключа обирається модуль $m=7$:

$\sqrt{3811} \approx 62 + 1 = 63$;

$63^2 = 3969$;

$3969 - 3811 = 33$;

$pN = 2 \cdot 63 - 1$;

$pN = 125$;

$sN = 33 \bmod 7 = 5$;

$pN \bmod m = 125 \bmod 7 = 6$;

$sN=(6+5) \bmod 7=4$;

Оскільки sN є лишком квадрату, то $Y[1]=1$;

$pN = pN + 2 \bmod m = (6 + 2) \bmod 7 = 1$;

sN=(sN+pN) mod 7=(1+4) mod 7=5;

Оскільки sN не є лишком квадрату, то $Y[2]=0$;

$pN = pN + 2 \ mod \ m = (1+2) \ mod 7 = 3$;

sN=(sN+pN) mod 7=(5+3) mod 7=1;

Оскільки sN є лишком квадрату, то $Y[3]=1$;

$pN = pN + 2 \ mod \ m = (3+2) \ mod 7 = 5$;

sN=(sN+pN) mod 7=(1+5) mod 7=6;

Оскільки sN не є лишком квадрату, то $Y[4]=0$;

$pN = pN + 2 \ mod \ m = (5+2) \ mod 7 = 0$;

sN=(sN+pN) mod 7=(0+6) mod 7=6;

Оскільки sN не є лишком квадрату, то $Y[5]=0$;

$pN = pN + 2 \ mod \ m = (0+2) \ mod 7 = 2$;

sN=(sN+pN) mod 7=(6+2) mod 7=1;

Оскільки sN не є лишком квадрату, то $Y[6]=1$;

$pN = pN + 2 \ mod \ m = (2+2) \ mod 7 = 4$;

sN=(sN+pN) mod 7=(1+4) mod 7=5;

Оскільки sN не є лишком квадрату, то $Y[7]=0$;

$pN = pN + 2 \ mod \ m = (4+2) \ mod 7 = 6$.

В результаті отримується вектор $Y[]=0100101$, наступні елементи якого можна формувати, використовуючи властивість циклічності.

Запропоноване технічне рішення при достатньо великому модулю m, який складений з великої кількості простих модулів, дозволяє уникнути операції перевірки кореня квадратного, а вектор Y з ознаками однозначно вкаже на розв'язок рівняння.

Таким чином, метод факторизації з використанням СЗК дозволяє змінити зону розрядностей обчислювальних ресурсів на декілька порядків нижче по шкалі квадратів та замінити операцію знаходження кореня квадратного, на якій базується обчислювальна складність алгоритму Ферма, на операцію порівняння згенерованого ключа факторизації, який формується на основі ознак квадратичності.

Після запуску додатку користувачеві необхідно обрати і ввести в наступному числовому полі кількість простих модулів, що будуть брати участь в обчисленнях, кількість простих модулів для великого числа може значно скоротити обчислення, а для маленького може стати надлишковою, якщо їх добуток буде більшим ніж саме число для факторизації.

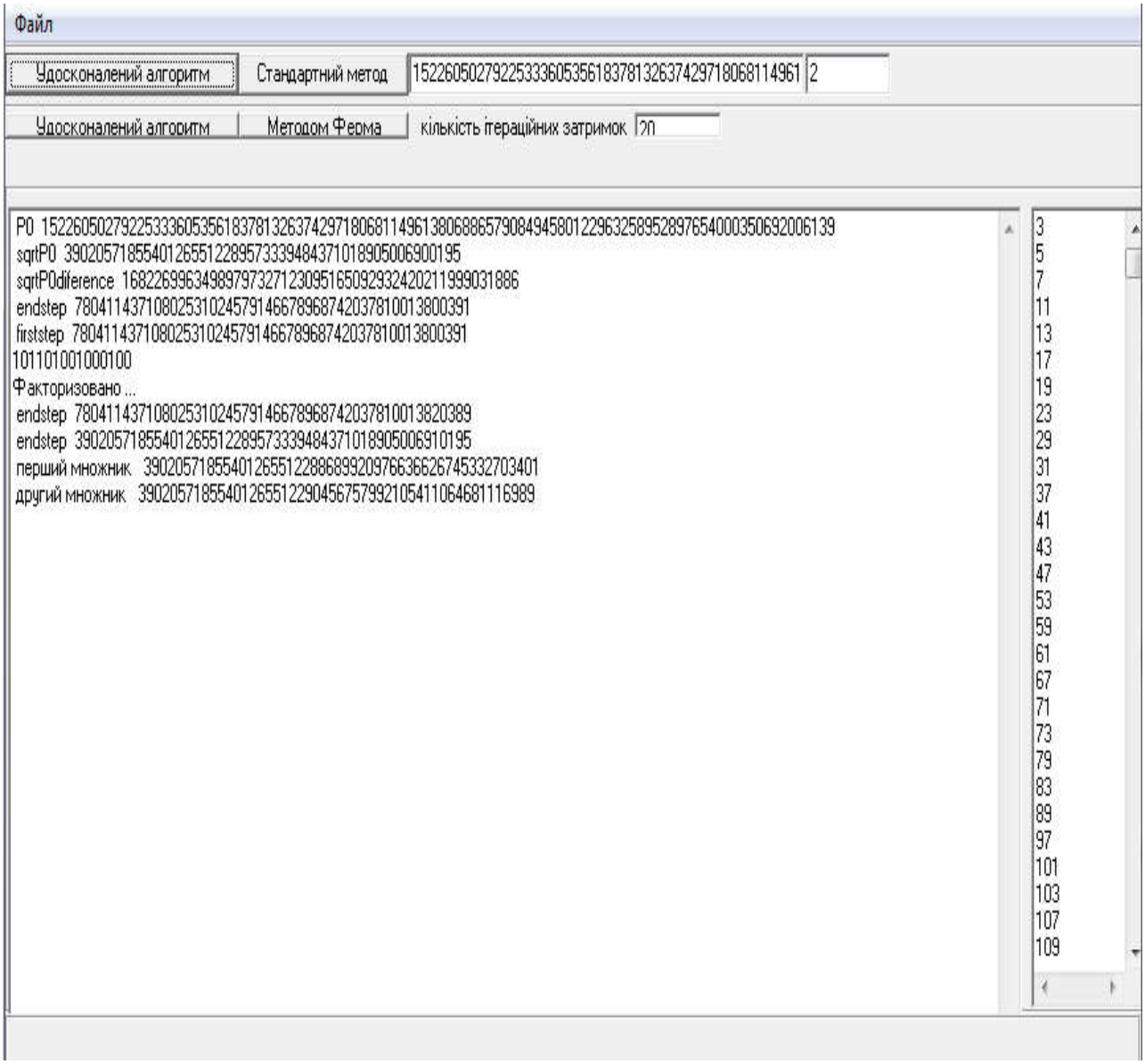

Рисунок 3.2 – Додаток на етапі тестування

Для факторизації числа його необхідно ввести в текстове поле та натиснути кнопку Удосконалений алгоритм, яка знаходиться у першому полі.

При обчисленні відображається службова інформація, що використовується при пошуку наступного цілого квадрата.

- P0 3811 – саме число для факторизації;

- sqrtP0 62 – заокруглений корінь;

- sqrtP0diference 33 різниця між наступним цілим квадратом та числом для

факторизації;

- endstep 125 – службова змінна, що відображає на якому кроці закінчуються обчислення, адже квадрат числа складається з суми непарних чисел;

- firststep 125 - крок з якого починають обчислення;

- 00000000100000000000000000000000000010000000000000000000000000000000000000000000000010000000000000000000 – ключ за допомогою якого уникають надлишковості перевірок на цілісність кореня, він застосовується циклічно і його розмір залежить від кількості та розміру простих модулів, що обрані для обчислень;

- 103 – розв’язок задачі факторизації.

Для порівняння методів факторизації реалізовано ітераційну затримку для виявлення переваг удосконаленого алгоритму факторизації. Додаток забезпечує можливість внесення простих модулів для генерації ключа факторизації, що виключає виконання операції визначення квадратного кореня.

Також існує можливість вибрати лише декілька перших модулів, в прикладі наведено використання лише перших двох простих модулів.

Додаток інформує про перший крок до цілого квадрату та завершальний, з якого обчислюється значення чисел, що факторизуються. В інформаційному полі наводиться інформація про генерований ключ факторизації.

Класичний алгоритм факторизації Ферма є подібним до методу «пробних ділень», тобто методу перебору всіх дільників. Тому складність даного методу оцінюється $O(n_0 (\log n)^2)$. Параметр n_0 визначається з конкретних обчислювальних можливостей. Чим більше значення n_0, тим

більше число можливих дільників буде перевірено. Проте на відмінну від методу «пробних ділень», за допомогою якого знаходять найменший дільник, метод Ферма дозволяє знаходити найбільший дільник числа *n* що не перевищує $\sqrt{n}$.

Для оцінки складності розробленого алгоритму, необхідно визначити кількість базових операцій, а саме: ділення, множення та додавання. Слід зазначити, що основною трудомісткою операцією розробленого алгоритму є пошук символу Якобі, перевірки чи добувається корінь квадратний за модулем та добування кореня від числа. Складність обчислення символу Якобі потребує $O(log^2 n)$ бітових операцій, а пошук кореня $O(n)$.

В таблиці 3.1 представлені основні кроки виконання розробленого алгоритму та оцінка їх складностей.

Таблиця 3.1 – Складності основних операцій розробленого алгоритму факторизації багато розрядних чисел

№ п/п	Основні кроки виконання розробленого алгоритму факторизації багато розрядних чисел	Складності основних операцій (в тактах)
1	Ввід Prime[0..n]	1
2	i=0	1
3	sqstart =Sqrt(P0)	n
4	Diference = (sqstart+1) * (sqstart+1)	$n*log_2 n$
5	Diference = Diference- P0	n
6	i++	1
7	sqstart= sqstart+2	1
8	sqstartm[i]= (sqstartm +2) mod Prime[i]	$log_2 n$
9	Якщо символ Якобі (sqstartm[i], Prime[i])≠1 тоді крок 8	$log^2 n$
10	Якщо i=n крок 14	1
11	Якщо символ Якобі (sqstartm[i], Prime[i])=1 тоді i++, крок 10	$log^2 n$
12	Якщо Sqrt(Diference) дробове тоді крок 6	n
13	Вивід sqrt(Diference+sqstart*sqstart)+sqrt(Diference)	n
14	Вихід	1

Отже, загальна складність розробленого алгоритму обчислюється як $O(nlogn+2log^2 n+4n+3) \approx O(nlogn)$. Результати досліджень показують, що

розроблений алгоритм характеризується меншою обчислювальною складністю в порівнянні з класичними, що ілюструє рисунок 3.3.

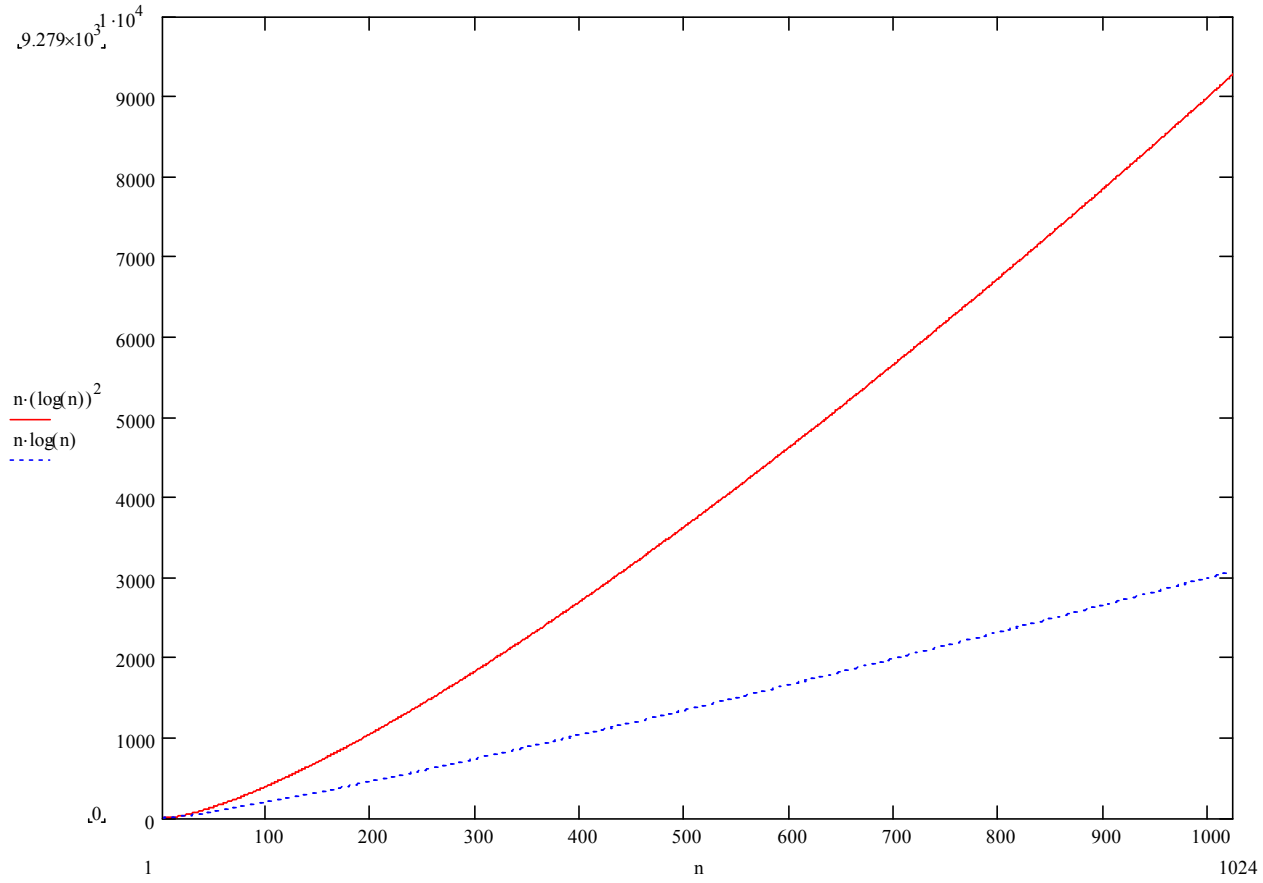

Рисунок 3.3 – Обчислювальна складність розробленого алгоритму в порівнянні з класичним

Ефективність алгоритмів факторизації в роботі оцінюється їх обчислювальною складністю.

Виграш в ефективності запропонованого в роботі алгоритму факторизації відносно класичного визначимо як співвідношення обчислювальних складностей:

$$E(n) = \frac{n(\log_2 n)^2}{n(\log_2 n)} = log_2 n. \qquad (3.3)$$

Отже, виграш у ефективності вдосконаленого методу при зростанні розрядності чисел зростає в $log_2 n$ разів.

3.3 Аналіз роботи розробленого програмного продукту

Проведено аналіз роботи розробленого програмного продукту, для цього було реалізовано класичний метод Ферма, описаний в пункті три другого розділу даного дипломного проекту.

В таблиці 3.2 представлено час роботи розробленого методу та час роботи класичного методу факторизації Ферма для розрядностей чисел від 10-ти до 72-ух біт.

Таблиця 3.2 – Приклад роботи класичного та вдосконаленого методу Ферма

Десяткове число	Кількість бітів	Час роботи класичного алгоритму	Час роботи модифікованого алгоритму
529	10	4,97685185188446E-7	8,21759259250765E-7
543789	20	3,93518518521097E-7	7,63888888893893E-7
543787793	30	4,97685185184976E-7	3,47222222222329E-7
746378175493	40	5,90277777781645E-7	4,51388888891846E-7
987463781754953	50	5,78703703707495E-7	1,82870370370181E-6
698740637810754953	60	7,87037037028315E-7	2,18750000000045E-6
3716987406378107546731	72	5,55555555545317E-7	1,68981481482589E-6

В програмному середовищі C++ Buider 6.0 реалізовано основні компоненти даного алгоритму, які відповідають теоретично розрахованим параметрам і підтверджують правильність та результативність

запропонованого наукового підходу по вдосконаленню алгоритму факторизації багаторозрядних чисел в комп'ютерних мережах за умови застосування базису Крестенсона та символів Якобі.

Експериментальні дослідження швидкодії стандартного та запропонованого алгоритмів для чисел різної розрядності (10, 20, 30, 40, 50, 60, 72-бітні) проводилися на комп'ютері з двох'ядерним процесором типу Intel з тактовою частотою 2,3 ГГц, оперативна пам'ять становила 2 Гбайти.

На рисунку 3.4 наведено приклад роботи класичного методу Ферма та удосконаленого методу Ферма на основі застосування базису Крестенсона та символів Якобі.

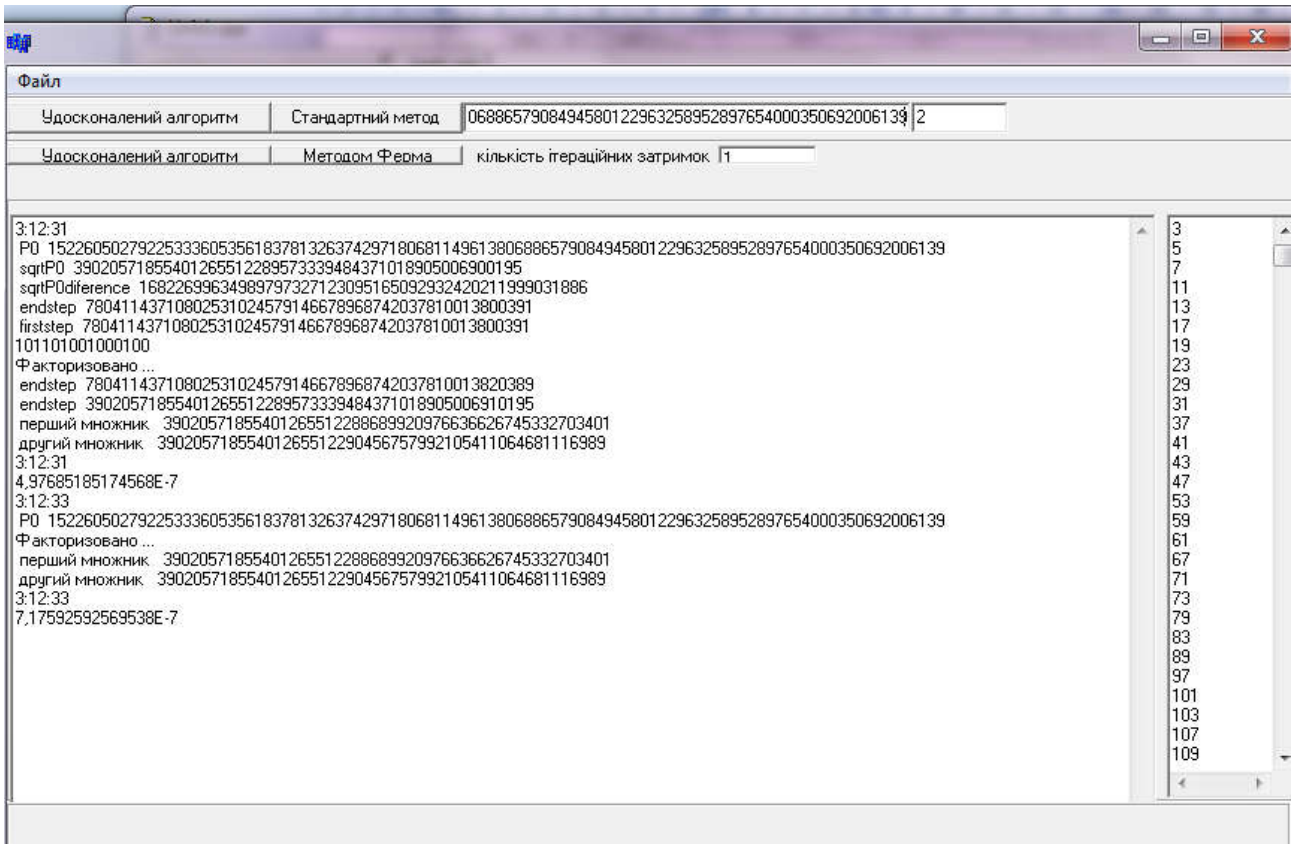

Рисунок 3.4 – Приклад роботи класичного методу Ферма та удосконаленого методу Ферма

Як видно з таблиці та рисунку, запропонований алгоритм для чисел невеликої розрядності значно не відрізняється від класичного за часовою

характеристикою, проте при збільшенні розрядності чисел виграш у ефективності алгоритму значно зростає.

З результатів досліджень видно, що алгоритм характеризується меншою обчислювальною складністю, тому існує доцільність його використання в асиметричних системах захисту інформації.

Даний алгоритм програмно реалізований для великорозрядних чисел, обчислювальна складність по відношенню до класичного алгоритму на порядок менша.

Для переведення чисел, над якими будуть виконані операції, в базис Крестенсона необхідно мінімум ресурсів, оскільки для цього достатньо побітно зчитати числа з пам’яті, використавши вказівники.

Отже, в програмному середовищі C++ Buider 6.0 реалізовано основні компоненти, які відповідають теоретично розрахованим параметрам і підтверджують правильність та результативність запропонованого наукового підходу по вдосконаленню алгоритму факторизації багато розрядних чисел в комп’ютерних мережах за умови застосування базису Крестенсона.

Отже, для реалізації поставленого в дипломній роботі завдання було обґрунтовано обрану мову програмування C++ та середовище програмування C++ Buider 6.0.

Розроблено інструментальні програмні засоби реалізації алгоритму методу Ферма факторизації багаторозрядних чисел, а також досліджено вплив розрядностей чисел на часові характеристики запропонованого алгоритму.

ВИСНОВОК

1. Удосконалено метод факторизації Ферма для криптоаналізу асиметричних криптографічних систем захисту інформації, обчислювальна складність якого в $log_2 n$ разів менша за класичний метод Ферма.

2. Досліджено та проаналізовано обчислювальні складності існуючих алгоритмів факторизації, що дозволило обґрунтувати необхідність розробки удосконаленого алгоритму факторизації та довести ефективність його використання.

3. Проаналізовано математичні особливості базису Крестенсона, що дозволило зменшити використання обчислювальних ресурсів при використанні операції факторизації та який успішно застосовується для побудови спец процесорів стиснення інформації та реалізації високопродуктивних процесорів опрацювання інформаційних потоків, у системах повітряної оборони та для опрацювання великорозрядних чисел у системах криптозахисту інформації.

4. Обґрунтовано підхід та розроблено алгоритм факторизації, що дозволило застосувати відповідне програмне та апаратне забезпечення для реалізації запропонованого рішення;

5. Реалізовано програмне забезпечення для факторизації на основі середовища розробки C++ Builder 6.0 та мови програмування С.

6. Реалізовано основні компоненти, які відповідають теоретично розрахованим параметрам і підтверджують правильність та результативність запропонованого наукового підходу по вдосконаленню алгоритму факторизації багато розрядних чисел в комп’ютерних мережах за умови застосування базису Крестенсона.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Айерлэнд К. Классическое введение в современную теорию чисел.// К. Айерлэнд, М. Роузен/ — М.: Мир.– 1987- 416 с.

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

3 Архангельский А.Я. Программирование в C++ Builder. – М.: Изд. «Бином», 2010 г. – 447 с.

4 Берлекэмп, Э. Алгебраическая теория кодирования [Текст] / Э. Берлекэмп ; под ред. С. Д. Бермана ; пер.с англ. – М. : Мир, 1971. – 477 с.

5 Боброва-Голикова Л.П. Эргономика и безопасность труда / Л.П. Боброва-Голикова, О.М. Мальцева, Н.А. Коханова, А.Н. Строкина. – М.: Машиностроение, 1985 г. – 112 с.

6 Вербіцький О.В. Вступ до криптології/ О.В. Вербіцький. – Львів: ВНТЛ, 1998 г. – 248 с.

7 Василенко О.Н. Теоретико-числовые алгоритмы в криптографии/ О.Н. Василенко. - М.:МЦНМО, 2003.—328 с.

8 Головатюк С.С. Методические указания и задания к расчетам по промсанитарии и противопожарной безопасности в дипломных проектах для студентов специальностей 10.05, 10.06, 10.07, 10.10, 21.03 / Сост. С.С. Головатюк. – Одесса: ОПИ, 1992. – 44 с.

9 Якименко І.З. Алгоритм факторизації на основі теореми Ферма/ І.З. Якименко, С.В.Івасьєв, Н.П. Петрица // Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп’ютерні інформаційні технології» АСІТ-2017; 19-20 травня 2017 р. – Тернопіль, 2017. – С. 215.

10 Горшков С.И. Производственная эргономика /Под ред. С. И. Горшкова; АМН.–М.: Медицина, 1979. 312 с.

11 Демиденко Г.П. Защита объектов народного хозяйства от оружия массового поражения / Г.П. Демиденко, Е.П. Кузьменко, П.П. Орлов, В.А. Пролыгин, Н.А. Сидоренко.- Киев: Головное издательство издательского объединения «Выща школа», 1989 г. - 145 с.

12 Касянчук М.М. Теорія та математичні закономірності досконалої форми системи залишкових класів // Праці Міжнародного симпозіуму „Питання оптимізації обчислень (ПОО–XXXV)”. Т.1. – Київ–Кацивелі.– 2009.– С. 306–310.

13 Каутинхо С. Введение в теорию чисел. Алгоритм RSA / С. Каутинхо. – Москва: Постмаркет 2001 г. – 328 с.

14 Николаенко О.В. Анализ методов факторизации в криптографических системах защиты информации / О.В. Николаенко, Л.Н. Тимошенко, К.В. Вербик. – Матеріали III міжнародної науково-практичної конференції ІУСТ-Одеса-2014. – С. 173-175.

15 Николайчук Я.М. Теорія джерел інформації. – Тернопіль: ТзОВ„Терно–граф”, 2010. – 536 с.

16 Николайчук Я.М. Теоретичні основи побудови спецпроцесорів у базисі Крестенсона / Я.М. Николайчук, О.І. Волинський, С.В. Кулина // Вісник Хмельницького національного університету - 2007. - № 3. Т.І(93). - С. 85-90.

17 Смарт Н. Криптография / Н. Смарт. – Москва: Издательский дом «ТЕХНОСФЕРА», - 2005 г. 526 с.

18 Стенли Б. С++ для начинающих / Б. Стенли, Липпман: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.

19 Тимошенко Л.М. Современные методы решения криптоаналитических задач/ Л.Н. Тимошенко, К.В. Вербик. – Матеріали ВШСМВС СКІТ.-Тернопіль.-2014.-С. 220-222

20 Тимошенко Л.М. Шляхи вдосконалення алгоритму факторизації чисел Тимошенко, К.В. Вербик. – Матеріали ВШСМВС СКІТ.-Київ.-2014.-С. 220-222

21 Тимошенко Л.М. Удосконалений метод Ферма факторизації чисел / Л.М. Тимошенко, К.В. Вербик, С.В., Івасьєв - Зб.мат. пробл-наук. міжн. конф. ПНМК-2014. Львів.- 2014. - С.348-350.

22 Тимошенко Л.М. Алгоритми факторизації криптоаналізу асиметричних систем / Л.М. Тимошенко, К.В. Вербик, Я.М. Николайчук, С.В. Івасьєв //«Інформатика та математичні методи в моделюванні».- Одеса, ОНПУ. – 2014. - № (Том). – С.

23 Тимошенко Л.М. Удосконалення алгоритму факторизації для криптографічних систем захисту інформації/ Л.М. Тимошенко, С.В. Івасьєв, К.В. Вербик //«Сучасна спеціальна техніка».- Київ, ДНДЕКЦ МВС України. – 2014. - №1(36). – С.

24 Фаронов, В. В. Система программирования Delphi [Текст] / В. В. Фаронов. – С-Пб. : БХВ-Питер, 2006. – 888 с.

25 Фомин, С. В. Системы счисления [Текст] / С. В. Фомин. – Изд. 5-е. – М. : Наука, 1987. – 47 с.

26 Хинчин, А. Я. Великая теорема Ферма [Текст] / А. Я. Хинчин. – 3-е изд. – М. : ЛКИ, 2007. – 80 с. – Режим доступу: http://library.tneu.edu.ua/images/stories /zmist/2012/літв/великая теорема ферма хинчин 2007.pdf.

27 Шилдт Г. Самоучитель C++: Пер. с англ. / Г. Шилдт. - Санкт-Петербург: BHV-Санкт-Петербург, 1998. - 620с.

28 Шушляпин, Пётр Великая теорема Ферма [Текст] / Пётр Шушляпин. – М. : Русаки, 2010. – 76 с. – Режим доступу : http://library.tneu.edu.ua/images/ stories/zmist/ 2012/літв/великая теорема ферма шушляпин 2010.pdf.

29 Якименко І.З. Алгоритм знаходження системи модулів модифікованої досконалої форми системи залишкових класів/ І.З. Якименко, М.М. Касянчук, Л.М.Тимошенко, С.В. Івасьєв, Я.М. Николайчук - Матеріали МНПК СІЕТ.- Одеса.-2014.-С. 115-117.

30 Яглом, А. М. Вероятность и информация [Текст] : 5-е изд., стереотип. / А.М. Яглом, И.М. Яглом. – М.: КомКнига, 2007. – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2012/літв/вероятность и информация яглом 2007.pdf.

31 Ященко В.В. Введение в криптографию/ В.В. Ященко, Н.П. Варновский, Ю.В. Нестеренко, Г.А. Кабатянский и другие. - Питер, - 2001. 271 с.

32 Gbolagade K.A. Residue Number System Operands to Decimal Conversion for 3-Moduli Sets, / K.A.Gbolagade, S. D. Cotofana // Proceedings of 51st IEEE Midwest Symposium on Circuits and Systems (MWSCAS-08). - Knoxville, USA. – 2008. - pp. 791-794.

33 Yang L. L. A residue number system based parallel communication scheme using orthogonal signaling: Part II—Multipath fading channels / L. L. Yang, L. Hanzo // IEEE Trans. Veh. Technol. – 2002. - Vol. 51. – P. 1541-1553.

34 Yang L. L. Minimum-distance decoding of redundant residue number system codes / L. L. Yang, L. Hanzo // Proc. IEEE ICC '2001. – Helsinki (Finland) – 2001. - P. 2975-2979.

35 Yang L. L. Performance analysis of coded M-array orthogonal signaling using errors-and-erasures decoding over frequency-selective fading channels / L. L. Yang, L. Hanzo // IEEE J. Select. Areas Community. – 2001.- Vol. 19. - P. 211-221.

36 Lakhani G. Some Fast Residual Arithmetic Adders / G. Lakhani // International Journal of Electronics. - 1994. - pp. 225-240.

37 Lenstra H. W. Divisors in residue classes / H. W. Lenstra // Math. Comput.- 1984. -Vol. 42. - N 165. - P.331-340.

38 Sousa L. Efficient Method for Magnitude Comparison in RNS Based on Two Pairs of Conjugate Moduli / L. Sousa // Electrical and Computer Engineering Department, INESC-ID/IST, TU Lisbon.

39 Ma Q. An integrated framework for information security management / Q. Ma, B.M. Schmidt, J.M. Pearson // Review of Business. Retrieved – 26 October 2013.– pp.58–69.

40 Andrijchuk V.A. Modern Algorithms and Methods of the Person Biometric Identification. Proceedings / V.A. Andrijchuk, I.P. Kuritnyk, M.M. Kasyanchuk, M.P. Karpinski // Third IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2005) – Sofia, Bulgaria. – 2005. – P.403–406.

41 Tsmots I. Basic Components of Neuronetworks with Parallel Vertical Group Data Real-Time Processing / I. Tsmots, V. Teslyuk, T. Teslyuk, I. Ihnatyev // Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing, vol. 689, Springer, Cham. – pp. 558 – 576.

42 Nykolaychuk Ya. M. Theoretical Foundations of the Modified Perfect form of Residue Number System / Ya.M. Nykolaychuk, M.M. Kasianchuk, I.Z. Yakymenko // Cybernetics and Systems Analysis, 2016, p. 219-223.

43 Kasianchuk M.N. Theory and Methods of Constructing of Modules System of the Perfect Modified Form of the System of Residual Classes / M.N. Kasianchuk, Ya.N. Nykolaychuk, I.Z. Yakymenko // Journal of Automation and Information Sciences. 2016, Vol.48, №8, p.56-63.

44 Stallings W. Cryptography and Network Security: Principles and Practice / W. Stallings /. 5th Prentice Hall Press Upper Saddle River, NJ, USA.– 2010.–719 p.

45 Karpiński M. Advanced method of factorization of multi-bit numbers based on Fermat's theorem in the system of residual classes / M. Karpiński, S. Ivasiev, I. Yakymenko, M. Kasianchuk, T. Gancarczyk // Proc. of 16th International Conference on Control, Automation and Systems (ICCAS–2016), Gyeongju, Korea, V.1, October, 2016, p.1484–1486.

46 Fischer W. Note on fast computation of secret RSA exponents / W. Fischer, J.-P. Seifert // Information Security and Privacy (ACISP 2002), Vol. 2384 of Lecture Notes in Computer Science, Springer-Verlag, 2002, p. 136–143.

47 Yakymenko I. Matrix Algorithms of Processing of the Information Flow in Computer Systems Based on Theoretical and Numerical Krestenson’s Basis / I.Yakymenko, M.Kasyanchuk, Ya. Nykolajchuk. // Proceedings of the X–th

International Conference ”Modern Problems of Radio Engineering, Telecommunications and Computer Science” (TCSET–2010).–L’viv–Slavske.– 2010. – P.241.

48 Kozaczko D. Vector Module Exponential in the Remaining Classes System / D.Kozaczko, M.Kasianchuk, I.Yakymenko, S.Ivasiev // Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2015). - Warsaw, Poland. - V.1, September – 2015. - P.161–163.

49 Abdallah M. On MultiModuli residue number systems with moduli of forms ra, rb-1, rc+1 / M. Abdallah and A. Skavantzos // IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS, VOL. 52, NO. 7 , 2005.

50 Hosseinzadeh M. Design Residue Number System Circuits in Current mode / M. Hosseinzadeh, K. Navi and S. Timarchi // 14th Iranian Conference of Electrical Engineering, 2006.

51 Методичні рекомендації до виконання дипломної роботи з освітньо-кваліфікаційного рівня “Магістр”. Спеціальність „Комп’ютерні системи та мережі” / О.М. Березький, Л.О. Дубчак /Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2012.– 47 с.