МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПЗВО «МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ
імені ПИЛИПА ОРЛИКА»
Економіко-технологічний факультет
Кафедра інженерних технологій

Кваліфікаційна робота
на здобуття освітнього ступеня магістра
за освітньою програмою «Комп'ютерна інженерія»
зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «АЛГОРИТМИ ЗАХИСТУ ДАНИХ ІЗ ВИКОРИСТАННЯМ МЕХАНІЗМІВ ЗАПЕРЕЧУВАНОГО ШИФРУВАННЯ ДАНИХ»

Виконав:
здобувач ІІ курсу, групи КІ -20-24
Білозерцев Руслан Сергійович

Керівник:
к.ф-м.н., доцент кафедри інженерних технологій
АРАМЯН Армен Матрикович

Миколаїв – 2024

ЗМІСТ

ВСТУП

Актуальність дослідження. В умовах високого темпу розвитку та інтеграції методів машинної обробки даних у різні галузі діяльності людини, об’єм інформаційного простору зростає значними темпами. Це призводить до появи проблем пов’язаних з обробкою та захистом інформації, яка обробляється в даному просторі. Встановлено, що у вирішенні питань з безпекою даних набули поширення засоби програмного захисту інформації. Їх надійність ґрунтується на обчислювальні стійкості алгоритмів, які забезпечують захист від більшої кількості криптоаналітичних атак. Однак, темпи розвитку комп’ютерної техніки та поява нових видів атак, зокрема атак на основі примусу, призводять до зменшення їх надійності. Внаслідок цього, задачі щодо розробки, створення практичних реалізації та впровадження прогресивних засобів захисту інформації є одним з актуальних напрямів досліджень. Так у теперішній час створено велику кількість перспективних алгоритмів і методів захисту даних, які ґрунтуються на використанні сучасних методів машинної обробки даних. Разом з тим встановлено, що практичне використання окремих ефективних криптографічних алгоритмів, зокрема алгоритмів заперечуваного шифрування, унеможливлене. Визначено, що з-поміж інших причин, основною проблемою є високі вимоги до обчислювальних ресурсів автоматизованих систем. Згідно з відкритими джерелами, на сьогодні не створено ефективних практичних реалізацій для використання вказаних засобів захисту. Враховуючи значний інтерес до інформації з боку третіх осіб і обмежену можливість захисту даних за допомогою існуючих рішень визначено, що задачі щодо розробки ефективних підходів методів захисту даних, зокрема алгоритмів заперечуваного шифрування, і їх практичних реалізацій є досить актуальними для галузей інформаційних технологій та інформаційної безпеки.

Метою роботи є розробка способів ефективної організації обчислень, які лежать в основі існуючих алгоритмів заперечуваного шифрування даних.

З метою досягнення вказаного результату розв’язано наступні **завдання:**

1) дослідити проблемну область щодо методів підвищення швидкодії криптографічних алгоритмів шляхом оптимізації процедур кодування та застосування паралельних і розподілених обчислень;

2) розробити модель ефективного виконання обчислень, які лежать в основі алгоритмів заперечуваного шифрування;

3) розробити прототип програмного забезпечення, який реалізує перетворення закладені у вихідну, паралельні та розподілену моделі заперечуваного шифрування.

Об'єктом дослідження є механізми перетворення даних, які використовуються в сучасних засобах криптографічного захисту інформації, зокрема механізми заперечуваного шифрування.

Предметом дослідження є способи і методи ефективного перетворення даних, які не призводять до внесення змін у вихідні алгоритми заперечуваного шифрування.

Методи дослідження. Для досягнення необхідного результату були використані наступні методи:

1) вивчення існуючих програмних і математичних рішень шляхом аналізу відкритих джерел;

2) декомпозиція і аналіз існуючих алгоритмів шифрування, кодування та обробки даних;

3) моделювання процесів перетворення даних і пошук вузлів, які потребують оптимізації обчислювального процесу;

4) оцінка ефективності запропонованих рішень шляхом аналізу експериментальних результатів.

Наукова новизна одержаних результатів. У магістерській роботі дістало подальший розвиток вирішення проблеми практичного застосування механізмів заперечуваного шифрування даних. Вивчення існуючих алгоритмів перетворення та обробки даних дозволило розробити модель операцій заперечуваного шифрування, удосконалення якої у частині кодування інформації дозволило скоротити час, необхідний на перетворення окремих

типів даних, за рахунок зменшення кількості використаних у цьому процесі операцій.

Практичне значення отриманих результатів. Наукові результати,які були отримані в даній роботі полягають у створенні програмної реалізації моделі, яка дозволяє ефективніше виконувати процедури перетворення даних покладені в основу механізмів заперечуваного шифрування. Науково-технічний ефект полягає у зменшенні часу необхідного на перетворення даних за допомогою механізмів заперечуваного шифрування.

Особистий внесок здобувача. Безпосередньо автором здійснено: виконано інформаційний пошук та аналіз літературних даних за темою магістерського дослідження; застосовано методи паралельних і розподілених обчислень, кодування даних у вихідних алгоритмах шифрування даних;

Апробація. Основні положення роботи викладено та обговорено на науково-практичних конференціях: X Міжнародній науково-практичній конференції «Тенденції розвитку філологічної освіти в контексті інтеграції у європейський простір» (25 – 26 квітня 2024 р., м. Миколаїв), науково-практичній конференції «Магістерські читання» (26 березня 2024 р., м. Миколаїв), II Всеукраїнській науково-теоретичній конференції «Глобалізаційні процеси та їх вплив на соціально-економічний та правовий розвиток України» (20 грудня 2023 р., м. Київ), Всеукраїнському круглому столі «Європейський Союз після Лісабонського договору: від трьохопорної структури до конституціоналізму» (20 травня 2023 р., м. Глухів).

Публікації. Основні положення роботи знайшли відображення у одній публікації.

Структура роботи. Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаних джерел (61 позиція). Повний обсяг роботи становить – 75 сторінок.

РОЗДІЛ 1.
СПОСОБИ ПРОГРАМНОГО ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Методи криптографічного захисту інформації

Визначено, що криптографічні засоби захисту даних ґрунтуються на використанні спеціалізованих математичних методів перетворення інформації. В загальному випадку їх призначення зводиться до перетворення відкритих даних у шифрограми (криптограми) за завчасно погодженим порядком дій, оформлених у вигляді алгоритму [3].

Вказані алгоритми дозволяють виконувати пряме та зворотне перетворення даних. З метою унеможливити зворотного виконання вказаних процедур третіми особами та отримання несанкціонованого доступу до захищених даних, в алгоритмах шифрування використовують ключову інформацію [4].

Також надійність сучасних криптографічних алгоритмів ґрунтується на використанні складних математичних алгоритмів з високою обчислювальною стійкість до атак на основі грубої сили та методів криптографічного аналізу. Однак вони виконують виключно перетворення даних і не включають в себе процедури підготовки інформації. Тому враховуючи її об’єм та вимоги щодо швидкості перетворення (обробки), сучасні засоби криптографічного захисту тісно пов’язані з методами машинної обробки даних.

Симетричні криптографічні системи. В сучасних симетричних алгоритмах перетворення даних використовуються наступні класи перетворень [3, 5]: підстановка (використовується для заміни значень визначеними еквівалентами); перестановка (вихідні дані змінюють свій порядок розміщення лише в межах заданого блоку даних і по визначеним правилам); аналітичне перетворення (перетворення даних у шифрограму здійснюється відповідно до визначеної аналітичної залежності); комбіновані перетворення. Безпека даних, у симетричних криптографічних системах ґрунтується на використанні одного

ключа та стійкого до зламу алгоритму перетворення інформації. Вказані перетворення реалізовані у криптографічних алгоритмах ГОСТ 28147-89, Twofish, Serpent, AES (Rijndael), Blowfish, CAST5, RC4, TDES (3DES), IDEA та ін.

Комбінований алгоритм шифрування DES [6, 7] – перший національний стандарт шифрування США. Раундові перетворення даних, які лежать в його основі забезпечують високу стійкість навіть до сучасних атак. Відомо, що його застосовують для захисту даних на цифрових носіях і в системах електронних платежів, для обміну комерційної інформації та її автентифікації.

Удосконаленим варіантом алгоритму є Triple DES [8, 9], який має ключ більшої довжини та має складнішу технічну реалізацію. Вказані особливості негативно вплинули на продуктивність Triple DES у порівнянні з його попередником.

Блоковий алгоритм ГОСТ 28147-89 [10] має структуру подібну до DES. Але збільшення розміру його ключа дозволило підвищити надійність алгоритму в 4 рази. Незважаючи на нижчу продуктивність алгоритму, він має досить високу стійкість до атак. Відомо, що в теперішній час він використовується для захисту даних у банківській системі.

Симетричний блоковий алгоритм IDEA є міжнародним стандартом шифрування [11], який замінив DES. Головною відмінністю алгоритму від попередників є використання різних режимів шифрування для обробки кожного блоку, при багатократному перетворенні даних. Встановлено, що завдяки своїй надійності, алгоритм був включений до пакету програм шифрування PGP.

В основі блокового алгоритму Blowfish [12], лежить використання типових елементів симетричної криптографії: мережа Фейстеля, операція додавання на модулем 2, підстановка та додавання. Його особливість полягає у використанні спеціального механізму для регулювання кількості раундів шифрування та блоків S-box, що дозволяє керувати рівнем стійкості алгоритму.

Відомо, що алгоритм Blowfish набув практичного застосування в програмних засобах PuTTY, SSH, OpenVPN, GnuPG, а також використовується

в системах електронного листування, в лініях зв'язку, спеціалізованому мережевому обладнанні, окремих протоколах мережного і транспортного рівня.

В блоковому алгоритмі Twofish [13, 14] реалізовано комбіновані перетворення алгоритмів Blowfish, Safer, Square, яка реалізована у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують одна одну за допомогою криптографічних перетворень Адамара. В його основі лежить використання попередньо обчислених S-box і складна схема розгортки.

Встановлено, що складність його структури, аналіз на предмет слабких ключів, прихованих зв'язків і низький показник продуктивності стали причиною того, що він не набув поширення.

У симетричному блоковому алгоритмі Serpent [15] реалізована SP-мережа з 32 раундами перетворення даних. В її основі лежить використання верифікованих таблиць підстановки, які забезпечують стійкість алгоритму до новітніх способів криптографічного аналізу. Подвоєна кількість раундів призвела до збільшення рівня захисту алгоритмом TDES і досягти продуктивності перетворень на рівні DES.

Комбінований симетричний блочний алгоритм CAST-128 [17] ґрунтується на використанні мережу Фейстеля з 3 типами раундових функцій. Особливість алгоритму полягає у використанні бент-функцій та модулярної арифметики для генерації S-box розміром 8 16.

Потоковий алгоритм RC4 [14] досить сильно відрізняється від інших алгоритмів шифрування. Простота його реалізації призвела до отримання досить значних показників продуктивності перетворення даних, що дозволило використання алгоритму в протоколах TLS та WEP. Незважаючи на це, в його структурі були виявлені суттєві вразливості, які дозволяють зламати захист алгоритму за короткий проміжок часу (кілька годин або днів) [17]. Внаслідок цього, відповідно до подання IETF, використання алгоритму RC4 в протоколі TLS та його реалізаціях припинено [18].

Найбільш поширеним і вживаним є симетричний блоковий алгоритм Rijndael [19, 20]. На відміну від алгоритм AES, Rijndael підтримує широкий

діапазон розміру блоків з даними та ключів від 128 до 256 біт, з кроком у 32 біти. Так, при використанні ключа розміром 128 біт, алгоритм виконує 10 раундів шифрування інформації, в основі якого лежить використання спеціалізованих алгоритмів для перетворення даних: subBytes, shiftRows, mixcolumns, xorRoundKey та SubBytes.

Вітчизняним аналогом алгоритму Rijndael є блоковий симетричний алгоритм «Калина» [21, 22], в основі якого лежить використання ітеративної процедури з попередньою та фінальною рандомізацію даних і два різні ітеративні послідовні раунди шифрування. Його структура подібна до Rijndael, тому він включає подібні процедури XORRoundKey, Add32RoundKey, Kalina_S_boxes, ShiftRows, MixColumns, Kalina_keyExspansion. Але на відміну від Rijndael, до його складу входять процедури, які не мають аналогів, зокрема Add32RoundKey. ЇЇ використання дозволяє посилити нелінійність алгоритму, ввести додаткові проміжні значення, підвищити стійкість алгоритму до алгебраїчних атак, диференціального, лінійному та інших методів криптографічного аналізу. Однак, використання цієї процедури потребує додаткових витрат часу в порівнянні з AES. Вказані недоліки були усунені в алгоритмі «Калина», однак це в значній мірі вплинуло на час її виконання.

Разом з тим, на думки вчених вказаним недоліком можна знехтувати на користь стійкості цієї процедури та її разового використання. Відомо, що в теперішній час алгоритм прийнятий у якості державного стандарту шифрування в Україні.

Криптосистеми з відкритим ключем. Встановлено, що криптографічні системи захисту інформації складні та забезпечують надійний захист з точки зору їх обчислювальної стійкості [12]. Їх слабким місцем є розподіл ключової інформації. З метою забезпечення безпечного обміну конфіденційними даними між двома користувачами автоматизованої системи одному з них необхідно сформувати ключ шифрування даних і передати його іншому користувачеві без його попереднього розголошення. Вказане завдання вирішується за допомогою криптографічних систем з відкритим ключем.

На відміну від симетричних систем шифрування даних, системи шифрування з відкритим ключем ґрунтуються на використанні двох різних ключів для шифрування (відкритий) та дешифрування (закритий) даних [23], які пов’язані між собою функціональною залежністю.

Відновлення вихідних даних можливе лише у разі застосування парного закритого ключа шифрування, який знає лише адресант. Оскільки в криптографічних засобах захисту використовуються незворотні або односторонні функції, то можливість отримання вихідних даних з отриманої або перехопленої шифрограми є завдання з високою обчислювальною стійкістю.

В теперішній час розроблена велика кількість незворотних функцій, які використовуються в криптографічних системах з відкритим ключем. Але їх практична реалізація залежить від особливостей автоматизованих систем [1, 12]. Враховуючи це, до криптографічних систем захисту даних з відкритим ключем висуваються наступні вимоги [24]. Саме тому алгоритми шифрування з відкритим ключем набули більшого практичного застосування в сучасних автоматизованих системах.

В основі сучасних алгоритмів шифрування з відкритим ключем лежить використання складно вирішуваних задач: факторизації великих складних чисел, обчислення дискретного логарифму та вирішення степеневих порівнянь [6]. Прикладними способами застосування вказаних алгоритмів є перетворення та збереження даних і розподіл ключової інформації. Вони отримали практичне застосування в алгоритмах Independent Elements, Меркле-Хелмана, Шаміра, RSA, Рабіна, Вільямса, Ель-Гамаля, Діффі-Хеллмана, криптографічних системах на основі еліптичних кривих.

Криптографічна система з відкритим ключем Меркле-Хелмана [12] є рішенням широкого призначення, яке відноситься до класу ранцевих алгоритмів. Його надійність ґрунтується на необхідності вирішення серії задач з пакування ранцю. Незважаючи на це її стійкість до криптоаналітичних атак

незначна. Idempotent Elements є удосконаленим варіантом алгоритму Меркле-Хелмана, але він має ті самі вразливості що й попередник.

Алгоритм шифрування Аді Шаміра [25] дозволяє організувати обмін приватними даними використовуючи відкритий канал зв’язку. Запропонований ним трьох ступеневий протокол шифрування даних не потребує обміну ключами між абонентами мережі. Недоліком вказаного підходу є необхідність передавати секретні дані через мережі кілька разів.

Найбільш відомим і поширеним алгоритмом шифрування з відкритим ключем є RSA [12, 23], в основі якого лежить задача факторизації великих складних чисел. Час необхідний для вирішення вказаної задачі є гарантує обчислювальну стійкість алгоритму для більшості сучасних криптоаналітичних атак. Відомо, що в теперішній час алгоритм набув широкого поширення в банківських комп'ютерних мережах для роботи з віддаленими клієнтами.

Алгоритм шифрування Рабіна [26] є удосконаленою версією алгоритму RSA, яка ґрунтується на вирішенні складної задачі з пошуку квадратних коренів за складним модулем. Особливість вказаного алгоритму полягає у використанні різних типів даних, які можуть бути відновлені користувачем.

Шифрування даних з відкритим ключем Хью Вільямса є удосконаленою версією алгоритму Рабіна [27]. В ньому вирішена проблема неоднозначності відновлюваних даних, що значно вплинуло на розмір шифрограми та виключило необхідність вирішення задачі вирішення квадратичних порівнянь.

Інформація щодо їх практичного застосування відсутня. Перетворення даних з відкритим ключем в алгоритмі Ель-Гамаля [12, 28] є альтернативою RSA, надійність якої ґрунтується на вирішенні складної задачі дискретного логарифму. Вказаний алгоритм є першим, який почав використовуватися для шифрування повідомлень та створення цифрових підписів.

За структурою і математичною базою криптографічного алгоритму Діффі-Хеллмана [29] подібний до алгоритму Ель-Гамаля. Його надійність також ґрунтується неможливості обчислення дискретного логарифму. Відомо, що вказаний алгоритм не може бути використаний для шифрування даних, але

набув практичного застосування у вирішенні завдань з розподілу ключової інформації.

Окремим напрямом розвитку алгоритмів шифрування з відкритим ключем є криптографічні системи на основі еліптичних кривих [30]. Відомо, що надійність вказаних алгоритмів ґрунтується на складності вирішенні задача факторизації та дискретного логарифмування в групі точок еліптичної кривої у кінцевому полі. Вказане зумовлює високу швидкість обробки даних і стійкість до криптоаналітичних атак у порівнянні з іншими алгоритмами. Але вказані переваги можуть бути досягненні лишу у випадку використання ключів меншого розміру та побудови кривих із використанням спеціальних параметрів.

Електронний підпис. В основі сучасних систем електронного цифрового підпису лежить використання криптографічних перетворень [12] та імплементація електронної цифрової мітки у набір даних. Вказане значення дозволяє перевірити автентичність та достовірність отриманого повідомлення.

Вказана технологія дозволяє захистити дані від несанкціонованої зміни та перевірити їх походження. Глобальне впровадження електронного документообігу та поступова відмова від документів на паперових носіях зробили задачу перевірки авторства досить актуальною задачею, яка включає розробку та використання протоколів генерації і автентифікації електронних документів [31, 32].

Використання вказаної технології не потребує від користувачів певних знань та практичних навичок, на відміну від паперового документообігу. Разом з тим, застосування засобів формування цифрового підпису та його подальша імплементація в документ дозволяє попередити можливість його підробки для адресата [12].

Також у попередньому розділі зазначено, що попри свої переваги сучасні системи шифрування не дозволяють перевірити походження перетворених даних. Тому на практиці технології електронного цифрового підпису набули широкого поширення в поєднанні з засобами шифрування даних [25, 26].

Управління ключами. Незважаючи на обчислювальну стійкість сучасних криптографічних алгоритмів, їх слабким місцем є процедури розподілу ключової інформації. Для звичайного користувача вказане є тривіальним питанням, але для багатокористувацьких систем вона є більш складною [12]. Ключова інформація становить множину діючих ключів автоматизованої системи. Недбале ставлення та порушення вимог щодо формування, обробки та зберігання ключів призводить до їх витоку та отримані третіми особами несанкціонованого доступу до ресурсів автоматизованих систем. Однак саме зберігання ключів є найбільш складною задачею, для виконання якої сформовані вимоги [33] обов'язкові для виконання власниками цих систем. Менш складною, але менш важливою, є задача управління ключами, до якої висуваються вимоги з оперативності, точності розподілу та приховування ключів. Для реалізації вказаних вимог використовуються спеціалізовані центри розподілу ключів або прямий обмін ключами. Разом з тим встановлено, що їх виконання створює умови для порушення конфіденційності даних власниками та працівниками автоматизованих систем, а також ускладнене необхідністю автентифікації учасників інформаційного обміну.

Багаторівневе шифрування. Одним із методів підвищення ефективності криптографічних засобів захисту інформації є багаторівневе шифрування [34-40]. Використання даного методу полягає у багатократному шифруванні даних з використання одних і тих самих алгоритмів шифрування та ключів.

Вказане рішення дозволяє збільшити захист вихідного алгоритму до 22N-разів. Визначено декілька варіантів застосування даного рішення: використання різних алгоритмів шифрування та одного ключа шифрування; використання одного алгоритму шифрування та різних ключів шифрування; використання різних алгоритмів шифрування та різних ключа шифрування.

Відомо, що незважаючи на фактичне зростання рівня захисту даних дослідники Меркле і Хеллман знайшли спосіб зламу вказаного рішення за

допомогою MITM-атаки. Разом з тим, рішення отримало практичне застосування в програмних засобах [41, 42].

Метод захисту [35] подібний до попереднього. Визначено, що за допомогою різних ключів стає можливим шифрування кожного блоку з декількох наборів даних одночасно.

Також встановлено, що окремі реалізації даного рішення можуть імітувати ефект від застосування технології заперечуваного шифрування даних, що дозволяє їм розшифровувати різні набори інформації з однієї шифрограми. Однак використання цього підходу накладає обмеження щодо його практичного застосування: шифрування та дешифрування даних з певного набору виконується послідовним дешифруванням усіх попередніх блоків з даними; у разі зміни вмісту одного з блоків даних або недотримання порядку дешифрування, дані які знаходяться на прихованих секторах (замаскованих під псевдовипадкові дані – вільне місце) будуть назавжди втрачені.

Відомо, що раніше [36], Джуліан Ассандж и Роберт Вейнман розробили файлову систему Rubberhose, в якій було реалізовано подібний механізм захисту даних. Вказане рішення було першою системою, яка дозволяла відновлювати різні набори даних з однієї шифрограми без обмежень їх розміру.

Надійність вказаної системи ґрунтувалася на використанні існуючих на той час симетричних алгоритмів шифрування. В основі перетворень даних був поділ дискового простору на частини та їх послідовне шифрування різними ключами, окремих для кожного набору даних, що унеможливлювало виявлення будь-яких інших даних після розшифрування кожного нового розділу. Незважаючи на це, рішення не набуло широкого поширення через неможливість її використання для мережевих завдань. Також на підставі вказаного методу було розроблено кілька рішень призначених для захисту даних у носіях та мобільних пристроях [37-39].

Комутативне шифрування. Використовуючи відкриті джерела встановлено, що іншим перспективним напрямом досліджень криптографії є розробка алгоритмів комутативного шифрування [43, 44]. В його основі лежить

можливість послідовного шифрування вихідних даних М за допомогою ключів К1 і К2. Визначено, що надійність такого рішення ґрунтується на відсутності необхідності створення спільної ключової інформації для організації інформаційного обміну, оскільки учасники мають можливість використовувати особисті секретні ключі для обміну даними. До комутативних алгоритмів відносять: RSA зі спільним модулем, алгоритм Діффі-Хеллмана, гамування, алгоритм Поліга – Хелмана, тощо.

Також відомо, що деякі алгоритми комутативного шифрування набули поширення для вирішення задач з електронного жеребкування, ментального покеру, без ключового шифрування.

Гомоморфне шифрування. Технологія гомоморфного шифрування є окремою задачею математичної криптографії, яка ґрунтується на можливості виконання будь-яких алгебраїчних операцій з конфіденційними даними у шифрованому вигляді [45-47]. В загальному випадку встановлено, що алгоритм шифрування даних E(k,m) відповідає критерію гомоморфності для алгебраїчної операції op лише в тому випадку, коли існує імовірнісний алгоритм M, застосування якої до пари шифрограм E(k,m1) та E(k,m2), при розшифруванні, дає можливість отримати відкритий текст виду m1 op m2. При цьому важливою умовою для використання вказаного механізму є те, що для певного алгоритму шифрування E та алгебраїчної операції з вихідними даними op1 існує така алгебраїчна операція з шифрограмами, в якій після розшифрування шифрограми E(k, m1) op2 E(k, m2) можливо отримати вихідний текст m1 op1 m2.

Вказана особливість робить гомоморфне шифрування досить перспективним напрямом для розробки засобів захисту даних. В теперішній час відомо про існування декількох теоретично обґрунтованих алгоритмів, які можуть бути використані для гомоморфного шифрування даних, зокрема RSA, Ель-Гамаля, Пейє та їх комбінації. Відомо, що у разі створення ефективної реалізації схеми з гомоморфним шифруванням, воно може отримати

поширення для захисту даних у хмарних сховищах, захисту програмного забезпечення (стійка обфускація) та ін.

Заперечуване шифрування. Також встановлено, що окремим напрямом досліджень є розробка алгоритмів заперечуваного шифрування даних, використання яких дозволяє захистити дані від витоку та користувачів від застосування грубої сили (примушування) третіми особами. Оскільки надійність вказаних алгоритмів ґрунтується не лише на обчислювальній, а також на алгоритмічній стійкості, то їх практичне застосування дозволить захистити дані від атак спрямованих на отримання ключової інформації безпосередньо від користувачів інформаційних систем.

Так в 1984 році Шаффі Голдвассер и Синтія Мікалі представили роботу, в якій був продемонстрований прототип алгоритму імовірнісного шифрування даних [48]. Їх розробка дозволяла виконувати розшифровку кількох імовірних варіантів даних з однієї шифрограми. Однак створена ними система не набула практичного застосування, оскільки шифрограми генеровані їх системою значно перевищували розміри вхідних даних.

В тому ж році група дослідників, до якої увійшли Рейн Канетті, Синтія Дворк и Монті Наор та Рафаїл Островський [49], представили алгоритм заперечуваного шифрування даних з публічним ключем. Їх розробка передбачала шифрування окремих бітів даних за допомогою випадкових значень. Однак їх розробка вказана обставина не дозволяла виконувати обробку наборів даних великого розміру, а також захищала лише джерело даних, в окремих випадках. Враховуючи це їх алгоритм не був застосований на практиці.

Сюзана Ряжкова запропонувала схему шифрування даних [50], яка ґрунтувалася на перетвореннях протоколу шифрування RSA. Її розробка передбачала обмін даними через мережу та в подальшому повинна була використовуватися в системах електронного голосування. Однак з зв’язку з високою надмірністю даних (105 біт службових даних на 1 біт інформації), які

необхідно було передавати по відкритих каналах, вказаний алгоритм не був застосований на практиці.

У 2009 році індійський вчений Хамада Ібрахім використовуючи особливості бітових алгоритмів заперечуваного шифрування [51, 52] та степеневих порівнянь створив новий алгоритм заперечуваного шифрування, в повній мірі захищав чутливі дані від витоку [53]. Але його бітовий алгоритм шифрування потребував значних витрат системних ресурсів для перетворення даних. Відомості щодо його практичного застосування не були виявлені у відкритих джерелах.

Від так він запропонував інший алгоритм, який ґрунтувався на використанні особливостей протоколу Mediated RSA PKI [54]. Вказаний протокол дозволяє розділяти секретний ключ таким чином, що навіть виток однієї з його частин не дає можливості отримати доступ до захищених даних та в повній мірі заперечити факт їх існування. Для відновлення даних користувачем обов’язковою умовою є отримання другої частини ключа від довіреної сторони, обмін даними між якими виконувався за допомогою протоколу 1 n OT на основі протоколу RSA [55]. Однак під час дослідження алгоритму виявилося, що він не достатньо опрацьований в частині щодо заперечування даних і має обмеження щодо їх розміру. Таким чином, вказаний алгоритм не набув практичного застосування.

Того ж року китайські дослідники Jiang Qing Wang та Bo Meng використовуючи вищевказані алгоритми та схему заперечуваного шифрування запропоновану Михайлом Клоновскі [56], розробили протокол електронного голосування BCP [57]. Вказаний протокол ґрунтувався на двох схемах генерації ключів та використовував схему перетворень даних Ель-Гамаля. Вказаний протокол не мав вад із продуктивністю та був застосований на практиці. Од дослідження протоколу BCP виявило можливі проблеми з безпекою даних, тому їх розробка не набула широкого поширення в більшій частині мереж.

Узагальнюючи дані в [51-53] Баракат Т.М. запропонував рішення, яке ґрунтується на використанні бітового шифрування даних [58]. Відмінність

запропонованого підходу полягає в усуненні необхідності масштабного використання квадратичних порівнянь в процедурах перетворення даних.

Результативна швидкість даного рішення склала 2,15-3,68 Кбіт/с. В 2013 році Молдовян М., Морозова О. та Мондікова Я. опублікували алгоритм заперечуваного шифрування даних на базі розподілу секретних параметрів [59]. Згідно з їх твердженнями надійність алгоритму ґрунтується на специфіці розподілу ключів шифрування та використанні існуючих симетричних алгоритмів. Тому швидкість алгоритму повинна була досягати 10-1000 Кбіт/с, у порівнянні з швидкістю вихідного алгоритм у 1-1000 Мбіт/с. Однак інформації щодо умов проведення експериментальних досліджень з використанням вказаних алгоритмів, за яких можливо отримати вказані швидкості перетворення даних відсутні.

Під час своїх досліджень в 2014 році Молдовян розробив алгоритм заперечуваного шифрування даних з публічним ключем [60], в основі якого лежить використання розширеної криптографічної схеми Рабіна. Проте вказаний алгоритм досить схожий на алгоритм імовірнісного шифрування даних Шаффі Голдвассера та Синтії Мікалі [48]. Також встановлено, що особливість схеми перетворень вказаного алгоритму дозволяє збільшувати кількість наборів даних, які в подальшому можуть бути розшифровані користувачем [61]. При цьому швидкість роботи алгоритму суттєво зменшується, оскільки його виконання потребує більш значних обчислювальних ресурсів. Інформація щодо практичної реалізації або дослідження його швидкодії у відкритих джерелах відсутні.

Того ж року він провів дослідження щодо побудови швидкісного алгоритму заперечуваного шифрування на основі існуючих блочних шифрів [62]. Ідея його розробки полягала в одночасному шифруванні фіктивного та секретного повідомлень за допомогою одного випадкового параметра (ключового) та двох різних ключів. Використання вказаного алгоритму повинно забезпечити швидкість перетворення даних близьку до існуючих симетричних алгоритмів (близько 2,6-3,9 Мбіт/с). Однак в роботі наведено

лише оцінку швидкості шифрування даних, яка складає до 10 Кбіт/с. Вказані результати не такі перспективні, як у симетричних алгоритмів. Відомості щодо практичного застосування вказаного алгоритму відсутні.

Також відомо, що в 2016 року Микола Молдовян продемонстрував потоковий алгоритм заперечуваного шифрування [63], який дозволяв виконувати потокову обробку даних та був орієнтований на виконання мережевих задач. Його перетворення ґрунтуються на використанні пари ключів для шифрування відкритого (фіктивного) та закритого повідомлень з використанням симетричного алгоритму. При застосуванні примушування, користувач системи має можливість використати ключ для розшифрування фіктивного повідомлення, передати його третій стороні та ввести її в оману автентичністю даних. Дані щодо його експериментальної оцінки відсутні.

Вже у 2018 році була опублікована його праця щодо можливості заперечуваного шифрування даних за допомогою степеневих порівнянь [64]. Основною ідеєю його підходу було використання порівнянь N-ї степені для шифрування даних та відновлення N-ї кількості вхідних даних. Вказана робота є досить перспективною з точки зору інформаційної безпеки. Разом з тим в одній із своїх праць він наголошував, що збільшення кількості шифрованих даних неминуче призводить до падіння швидкодії вихідного алгоритму заперечуваного шифрування даних. Експериментальне дослідження швидкодії вказаного алгоритму не проводилося.

Таким чином, алгоритми заперечуваного шифрування досить ефективні для захисту даних як від криптоаналітичних атак, так і від атак на основі грубої сили. Той факт, що вони ґрунтуються на використанні існуючих криптографічних схем перетворення даних дозволяє застосовувати їх для виконання більшості існуючих задач в галу лузі інформаційної безпеки, а також можуть досить легко інтегруватися з іншими перспективними напрямами криптографії [65]. Незважаючи на це, відомості щодо їх експериментальних досліджень та практичного застосування у відкритих джерелах відсутні (окрім [34-42, 58, 65]). Можна зробити припущення, що причинами підбої ситуації є

важкі математичні обчислення, які лежать в їх основі та ґрунтуються переважно на використанні криптографічних систем з відкритим ключем. Тому подальше дослідження та удосконалення вказаного класу алгоритмів залишається предметом досліджень вчених і криптографів у різних країнах світу [66-69].

Окрім того, в теперішній час проводяться дослідження щодо створення алгоритмів шифрування, які можливо застосовувати у пост квантову епоху галузі інформаційної безпеки [70, 71]. Однак вказаний напрям досліджень виходить за межі питань, розглянутих у даному рукописі.

2.2 **Методи підвищення продуктивності обчислень**

Нижче наведено перелік методів та технологій прискорення даних, які можуть бути використані для досягнення мети поставленої на початку роботи.

Елементи блокового шифрування даних. Сучасні симетричні алгоритми шифрування забезпечують високий рівень обчислювальної стійкості та швидкості перетворення даних (від 2,6-3,9 Мбіт/с та вище). Досягнення останнього показника виконується за допомогою використання простих базових механізмів перетворення даних: оператори перестановки (P-boxes), оператори підстановки (S-boxes), сумування за модулем 2 (XOR), циклічний зсув, заміна, розкладання та об'єднання блоків [3, 5, 12]. Їх коректне застосування дозволяє виконувати вимоги з розсіювання інформації та перемішування інформації, які висуваються до шифрів даного класу. Крім того, комбінування вказаних елементів дозволяє збільшити стійкість шифрів до атак [22] з мінімальним навантаженням на обчислювальну систему. На відміну від P-boxes, S-boxes, XOR, циклічного зсуву та заміни, розкладання та об'єднання блоків є універсальними механізмами для обробки масивів даних будь-якого розміру. Застосування елементів розкладання та об'єднання інформації дозволяє зменшити кінцевий розмір блоків з даними і збільшити швидкість їх перетворення. Окрім того, за вказаних обставин з’являється можливість

використання інших елементів блокового шифрування (незалежно від операцій, які лежать в основі алгоритму перетворення даних).

Паралельні обчислення. Під паралельними обчисленнями розуміють таку організацію процесу обчислень, при якому вирішення завдання зводиться до виконання декількох окремих операцій в єдиний проміжок часу за допомогою кількох обчислювальних пристроїв в рамках однієї обчислювальної системи [76]. Встановлено, що можливість використання паралельних обчислень є доцільною лише у випадку коли ефективність обчислень після їх впровадження становить E 0,5 [77]. При цьому максимально можливе прискорення обчислень може складати від 2 до 10 разів. Разом з тим, у разі виникнення проблем з ефективністю технології паралельних обчислень, використання методів [78, 79] дозволяє виконати аналіз методики обчислень та знайти місце падіння продуктивності.

Реалізація паралельних обчислень виконується в 2 способи: чистий паралелізм [80] і конвеєризація [81]. Враховуючи кількість блоків з даними та лінійність процедур, які реалізують обчислення складних задач в алгоритмах заперечуваного шифрування, декомпозиція процедур на мікрооперації не принесе необхідного рівня прискорення обчислень.

Також, зазначається, що для реалізації паралельної обробки даних використовують методи [76]: розділення та склеювання даних, виділення незалежних елементів даних, попередньої обробки даних. Метод розділення та склеювання даних передбачає поділ масиву даних на частини з наступної обробкою в паралельному режимі (кількість частин відповідає кількості обчислювачів системи [82, 83]). При цьому скорочення часу обчислень можливо досягти лише за виконання однієї з умов: результат обробки частини даних є відомий на час початку обчислень, розмір блоку даних є незначним по відношенню до загального розміру масиву даних. Метод виділення незалежних елементів даних не потребує поділу даних на частини, оскільки по завершенню їх обробки результати кожного потоку об’єднуються у кінцеве значення [76].

Метод попередньої обробки даних ґрунтується на підготовці масиву інформації до вигляду, який найбільше підходить для обробки у паралельному режимі. При цьому операції поділяються на послідовні та паралельні. У разі наявності значної кількості послідовних операцій, час необхідний для їх обробки Тр → ∞, як у монопоточному режимі, або може бути гіршим [84].

Кодування даних. Використання методів кодування даних – це ефективний спосіб збереження масивів з даними значного розміру. При цьому основними критеріями ефективності методів кодування є високий коефіцієнт компресії та відсутність втрат інформації, при відновленні даних [85-87].

Разом з тим використання технології кодування даних не усіх випадках є ефективним у зв'язку з: інформація зазнала попереднього стиснення [88], використанням даних спеціального вигляду [89], стиснення бінарних файлів [90]. Тому визначення ефективності кодування даних запропоновано використовувати підхід щодо прогнозування імовірного рівня компресії [91].

Відомо, що в теперішній час існує різноманітна кількість стандартизованих форматів даних (текстові, графічні аудіальні, відео, тощо), кожний з яких дозволяє забезпечити визначений рівень компресії вихідних даних.

Виділяють 2 типи алгоритмів кодування: з втратами (Layer-1, Layer-2 і Layer-3 (MP3), MPEG-1, MPEG-2 і MPEG-4, JPEG, JPEG-2000, wavelet-перетворення, фрактальне шифрування, тощо) та без втрат (LZ-алгоритми, RLE, BWT, алгоритми унарного, арифметичного, імовірнісного кодування та алгоритми на основі методу Хаффмана) [90]. Аналіз особливостей роботи та призначень вказаних алгоритмів дозволив видіти алгоритми кодування Deflate [92] і LZMA2 [93], які в теперішній час є досить ефективними та універсальними рішеннями для стиснення даних з можливістю гнучкого налаштування параметрів компресії [91].

Також встановлено, що використання алгоритмів кодування має кілька практичних особливостей, зокрема можливість автоматичного відновлення пошкоджених даних (у разі наявності незначної кількості помилок) [87] і

забезпечення додаткового захисту вихідної інформації (забезпечує повне знищення інформаційних моделей і шаблонів для попередження можливості аналізу даних до їх повного відтворення) [94].

Розподілені обчислення. Розподіленою системою є система, яка дозволяє виконувати обчислення незалежно від географічного розміщення вузлів [95]. Реалізація подібної системи досить складна, аніж її практичне застосування. Необхідність її застосування визначається наступними умовами: дані фізично знаходяться в іншій мережі або мають географічне положення відмінне від поточної системи, швидкість обчислень неможливо збільшити засобами локальних вузлів [96].

Кожна розподілена система використовує обмін даними для комунікації вузлів. В його основі лежить модель «клієнт-сервер» (синхронна або асинхронна в залежності від складності системи). Таким чином створено дві найбільш поширені моделі розподілу обчислень: з поділом програмної логіки між вузлами, організації багатоланкової архітектури «в ширину» (дозволяє попередити прямий доступ до даних) [96].

Попередження відмови розподілених систем, зростання ефективності розподілу та виконання обчислень досягається шляхом балансування навантаження на канальному, мережевому, транспортному та прикладному рівнях [97-99] та реалізовані у відповідних алгоритмах і протоколах: вирівнювання за DNS, NLB-кластер, IP-адреса, Google Compute Engine та Elastic Load Balancer, Nginx, HAProxy, LVS, Google Compute Engine, Azure LoadMaster, Elastic Load Balancer, CloudStack, Least Connections, Sticky Sessions, Round Robin, Weighted Round Robin, Monitoring Load Balancing та Throttled Load Balancing [97].

За результатами аналізу способів захисту даних у сучасних програмних засобах, можна зробити висновок, що найбільш поширеним класом алгоритмів перетворення даних є криптографічні алгоритми захисту інформації. Однак, незважаючи на їх перевагу та значну теоретичну базу, лише окремі криптографічні алгоритми набули практичного застосування. Визначено,

причинами подібної ситуації стали: низька швидкість перетворення даних, відсутність ефективних реалізацій і вразливості в алгоритмічній структурі. Визначено, що окрім деяких з існуючих, низька перспективних алгоритмів шифрування даних, зокрема заперечуваного шифрування, не знайшла практичного застосування в сучасних автоматизованих системах у зв'язку з низькою швидкістю перетворення даних і відсутністю їх ефективних реалізацій. Для вирішення вказаної проблеми узагальнено схему перетворення даних, яка використовується в більшості алгоритмів заперечуваного шифрування.

Також виконано огляд рішень, які можуть бути використані для розв'язання проблеми практичного застосування алгоритмів заперечуваного шифрування:

1) встановлено що для збільшення розміру перетворення даних та швидкості їх обробки доцільним є створення блокової моделі перетворення даних, яка реалізує механізми симетричних криптографічних систем незалежно від типу використаних обчислювальних алгоритмів;

2) визначено, що більша кількість процедур перетворення даних, які вирішують складні обчислювальні задачі лінійні, тому при побудові системи з паралельними обчисленнями доцільним є використання чистого паралелізму та методу декомпозиції даних;

3) вивчення алгоритмів кодування даних дозволило обрати методи і гнучкі алгоритми стиснення даних, які в можуть бути використані для побудови механізму ефективного управління розміром вихідних даних;

4) огляд методів і вимог до побудови систем розподілених обчислень дозволив встановити, що їх використання для вирішення проблеми з низькою продуктивністю алгоритмів заперечуваного шифрування залежатиме від обчислювальної потужності вузлів системи та втрат часу на обмін даними між вузлами (ефект їх використання є неоднозначним для досягнення кінцевої мети даної роботи).

РОЗДІЛ 2.
БЛОКОВА МОДЕЛЬ ЗАПЕРЕЧУВАНОГО ШИФРУВАННЯ

2.1 Розробка структури моделі перетворення даних

Алгоритми заперечуваного шифрування даних забезпечують перетворення вихідних даних в криптограму (шифрограму). Значення подібних шифрограм подібні до тих, які були отримані за допомогою алгоритмів імовірнісного шифрування даних [48].

Інша та найбільш важлива особливість вказаних алгоритмів – це можливість отримання кількох варіантів вихідних даних з однієї шифрограми, на етапі дешифрування [60, 61]. Подібний підхід до захисту даних є ефективним особливо проти атак, які ґрунтуються на застосуванні примусу до учасників інформаційного обміну з боку третіх осіб (зловмисників).

Окрім неоднозначності відновлюваних даних, стійкість існуючих алгоритмів заперечуваного шифрування ґрунтується на обчислювальній стійкості [3-49]. Але вирішення складних задач обмежує можливість практичного застосування вказаних алгоритмів, одним з яких є складність їх реалізації та застосування для роботи з великими об’ємами даних (наприклад, в складі «big data» систем). Розмір вихідних даних суттєво обмежений довжиною ключа шифрування.

Для вирішення вказаної проблеми в [62] запропоновано використати підхід на основі блочних шифрів, в основі якого лежить наступне:

1) використання двох різних ключів дешифрування даних Т К (закритих) та М К (відкритих);

2) генерації випадкових значень Ri : r1, r2 , r3 , , rn , які задовольняють наступній системі виразів (2.3).

Вище вказаний спосіб заперечуваного шифрування даних навіть з використанням блочних алгоритмів не є досить ефективним, оскільки вибір випадкових значень Ri : r1, r2 , r3 , , rn передбачає їх прямий перебір,

складність якого еквівалентна складності задачі дискретного логарифмування. Крім того, запропонований спосіб шифрування даних передбачає використання двох різних ключів (Т К та М К) для відновлення відкритих і закритих даних. Це суттєво впливає на час роботи кінцевого алгоритму.

З метою збереження гарантованого рівня захисту даних алгоритмами заперечуваного шифрування розроблено універсальну модель перетворення даних (Рисунок 2.1). В наведеній моделі виділяють функціональні модулі прямої/зворотної обробки та перетворення даних.

Модулі прямої та зворотної обробки даних призначені для трансформування даних у форм, яка найбільш підходить для їх перетворення в подальшому. Вказані модулі не маю стандартизованих вимог щодо складу та технологій обробки даних. Їх основне завдання мінімізувати кількість важких операцій, які виконуються в модулях перетворення даних.

Модулі прямого та зворотного перетворення даних є алгоритмами заперечуваного шифрування. Їх єдиним призначенням є перетворення даних з одного вигляду в інший за найменший проміжок часу. Використання попередньої оброки даних виключає необхідність виконання будь-яких процедур, окрім обчислень.

Такий підхід до побудови системи перетворення даних дозволяє:

1) виключити можливість появи додаткових вразливостей в структурі
2) алгоритмів заперечуваного шифрування;
3) зробити модель перетворення даних універсальною та дозволяє використання різних алгоритмів заперечуваного шифрування.
4) зробити модель незалежною від кількості обчислювальних ресурсів (не виключаючи необхідність виконання мінімальних вимог).

В криптографії існує велика кількість методів і підходів для підвищення рівня захищеності даних, які перетворюються одним з існуючих алгоритмів шифрування. Найпростішими з них є режими шифрування даних, які підвищення стійкість шифрів до криптоаналітичних атак [12]. Їх використання

дозволяє перетворити вихідні дані у шифрограму та однозначно відновити вихідні дані з шифрограми по завершенню їх обробки.

З метою оцінки можливості використання різних режимів шифрування у подальших дослідженнях і їх вплив на рівень захисту алгоритмів заперечуваного шифрування виконано огляд популярних режимів шифрування даних: режим електронної кодової книги (ECB), режим зчеплення блоків даних і шифрограми (CBC), режим гамування зі зворотнім зв'язком (CFB), режим потокового шифрування (OFB) та режим шифрування із використанням лічильника (CTR).

Часто їх використання можна зустріти саме в традиційних блочних шифрах. Однак алгоритми заперечуваного шифрування мають іншу структуру та механізми перетворення даних. Від так з'явилася необхідність адаптувати вихідні режими шифрування до використання з механізмами заперечуваного шифрування.

Режим шифрування електронної цифрової книги найпростіший. Його використання забезпечує найнижчий рівень захисту даних у порівнянні з іншими режимами. Його практичне застосування в алгоритмах заперечуваного шифрування зводиться до послідовної обробки блоків з даними, що, в теорії, призведе до появи умов для реалізації зворотних алгоритмічних атак. Вказаний режим може бути описаний виразом.

Його адаптація до використання в алгоритмах заперечуваного шифрування не викликала складностей. Разом з тим, встановлено, що його використання не захищає дані від втрати або підміни. Тому його використання не забезпечує додаткового захисту даних від існуючих криптографічних атак.

На відміну від нього, використання режиму CBC забезпечує більш високий рівень захисту та стійкість алгоритмів заперечуваного шифрування до більшості атак. В його основі лежить такий порядок застосування процедур перетворення даних, при якому результат шифрування поточного блоку залежить від результатів шифрування попереднього блоку. Це створює тісний зв'язок між блоками шифрограми та не дозволяє виконати їх аналіз і підміну.

Математичне обґрунтування даного режиму для алгоритмів заперечуваного шифрування описане виразом.

Згідно з вищевказаним виразом, практична реалізація даного режиму потребує формування вектору ініціалізації, використання якого збільшує псевдовипадковість розміщення бітів у кожному блоці шифрограми. Крім того використання даного режиму дозволяє захисти блоки від підміни. Однак її реалізація не дозволяє реалізувати обробку даних в паралельному режимі, що з огляду на складність та кількість обчислень є досить важливою обставиною, при використанні алгоритмів заперечуваного шифрування.

Режим CFB має структуру подібну до CBC-режиму, а також наслідує деякі з його властивостей, що забезпечує шифрам стійкість до більшості атак. В основі його перетворень лежить застосування блочних шифрів таким чином, що процедури шифрування та дешифрування застосовуються не до вихідних даних, а по відношенню до вектору ініціалізації і блоків самої шифрограми. Гамування з використанням блоків даних в даному випадку додає псевдовипадковості набору бітів шифрограми. Вираз, яким можуть бути описані вищевказані перетворення (2.6):

Фактично виконується захист не вихідних даних, але шифрограми. Це забезпечує захист блоків шифрограми від поширення помилок, у випадку їх виникнення під час обробки даних, та створює досить тісний блоками шифрограми, що стає причиною неможливості багатопотокової обробки даних (за виключенням окремих файлів і їх фрагментів). Вказане призводить до отримання вихідних даних шляхом зламу гами, яка залежить від різниці між блоками шифрограми. Тому використання даного режиму не лише не забезпечує збільшення рівня захищеності даних, але й не забезпечує мінімальний захист вихідних даних від існуючих криптографічних атак.

Режим OFB є еквівалентом потокового шифру з синхронізацією перетворення даних. На відміну від попередників, його використання передбачає багаторівневе перетворення даних з використанням вихідних алгоритмів шифрування. Ключем в даному випадку є вектор ініціалізації (ключ

шифрування даних), від так зникає необхідність у генерації стійкого ключа згідно вимог асиметричних схем. Його математичне обґрунтування базується на використанні лише однієї функції шифрування, що випливає з виразу (2.7):

Метод подвійного шифрування блоків та їх подальше гамування з іншими даними забезпечує досить надійний захист від деяких способів криптоаналізу. Також він дозволяє використовувати апаратну реалізацію CBC-режиму, що безперечно впливає на продуктивність його роботи. Однак, він виключає необхідність дешифрування даних і зламу ключа шифрування, що практично знищить обчислювальну та гарантовану стійкість алгоритмів заперечуваного шифрування.

Режим CTR є аналогом OFB-режиму. Однак на відміну від нього він ґрунтується на генерації набору додаткових ключових параметрів, які після попереднього шифрування гамуються з блоками даних. Математичний вираз, який дозволяє реалізувати вказані перетворення, наведений нижче (2.8):

Безумовними перевагами даного алгоритму є використання однієї функції шифрування та можливість паралельної організації обчислень. Разом з тим, створення додаткових ключових параметрів та їх шифрування з використанням односторонньої функції призводить до можливості підміни даних і їх подальшого аналізу з використанням засобів криптоаналізу. Тим не менш, вищевказаний режим є більш поширеним, оскільки в ньому відсутні вразливості ECB-режиму.

Для перевірки роботоспроможності, ефективності та впливу на захист вихідних алгоритмів заперечуваного шифрування проведено дослідження вищевказаних адаптованих моделей, результати яких викладені в табл. 2.1 та на круговій діаграмі (Рисунок 2.7).

Результати вищевказаних тестів показали, що всі режими шифрування можуть бути використані сумісно з алгоритмами заперечуваного шифрування, оскільки алгоритми заперечуваного шифрування функціонально ґрунтуються на існуючих алгоритмах перетворення даних. Разом з тим, встановлено, що використання режимів CFB, OFB і CTR не передбачає шифрування вихідних

даних, лише додаткових ключових параметрів [100]. Від так захист даних зводиться до звичайного гамування, яке має досить високі показники продуктивності, оскільки зникає необхідність використання важких обчислень. Їх використання можливе лише за рахунок модифікації вихідних алгоритмів шифрування, тобто зміни їх структури та механізмів обчислень. Вказане може призвести до появи додаткових вразливостей у вказаних алгоритмах, тому використання даних режимів не є доцільним на даному етапі.

Таким чином, саме режими ECB і CBC дозволяють в повній мірі реалізувати процедури прямого та зворотного перетворення даних. Однак, з огляду на кількість даних, які необхідно обробити за допомогою механізмів заперечуваного шифрування, в теперішній час пріоритетним є підвищення продуктивності вихідних алгоритмів шифрування. Від так для досягнення кращих показників продуктивності оптимальним рішенням є використання режиму ECB.

Вирівнювання різних типів даних. Схеми заперечуваного шифрування даних, які дозволяють реалізуватися неоднозначне відновлення завчасно підготовлених даних, передбачають одночасне завантаження та обробку файлів із даними. При цьому початковий розмір файлу з публічними даними позначається, як M F , розмір файлу з секретними даними T F .

Обґрунтування залежності блоку даних від ключа шифрування. У вихідних варіантах алгоритмів заперечуваного шифрування з неоднозначним дешифруванням даних існує функціональна залежність між розмірами блоку шифрограми та ключа, яка може бути описана виразом – S S C u K . При цьому u залежить від кількості даних, які необхідно вмістити в шифрограму або її частину.

Крім того, для використання вказаних алгоритмів на практиці необхідно реалізувати механізм автоматичного дешифрування інформаційної складової з шифрограми, без залучення користувача. Одним із можливих способів вирішення вказаної задачі є включення додаткової ключової інформації до складу шифрограми.

Для недопущення появи додаткових ключових елементів, які в подальшому також потребуватимуть захисту, запропоновано механізм імплементації додаткових бітів МК (контрольна мітка) до складу шифрограми. При цьому кінцевий розмір блоку даних, який буде перетворений в блок шифрограми, може бути оцінений виразом (2.10).

Надмірність даних з'являється в алгоритмах шифрування в двох випадках: використання сміттєвих обчислень та неефективній побудові процедур обробки даних. Інша ситуація спостерігається в алгоритмах заперечуваного шифрування. В зв'язку з шифруванням кількох наборів даних і можливістю їх неоднозначного дешифрування в подальшому, з'являється деяка надмірність витрат пам'яті для шифрування і дешифрування даних. Вона може бути виражена за допомогою виразів (2.11) і (2.12).

В загальному випадку розмір шифрограми зростає в 2 рази через необхідність шифрування декількох наборів даних. Разом в тим оцінка (2.11) є приблизною та не враховує особливостей практичної реалізації обчислень на відміну від виразу (2.12).

При практичному застосуванні розробленої моделі необхідно звернути увагу на секретність моделі, автентифікацію користувачів.

Секретність моделі. Згідно основних вимог [12], які висуваються до криптографічних засобів шифрування даних, алгоритми шифрування повинні бути публічні та прості в реалізації. Перше повинно забезпечити можливість їх безперешкодного вивчення фахівцями з криптології та інших галузей для оцінки їх надійності. Останнє повинне забезпечити умови для їх практичного використання і поширення в інших галузях.

Схеми заперечуваного шифрування мають механізми захисту і для даних, і для їх користувачів. В останньому випадку захист користувача забезпечується шляхом передачі зловмиснику ключа шифрування зловмисникові. При цьому допускається, що зловмисник знає лише алгоритмічні особливості дешифрування публічних даних, не секретних.

Однак, інформація про можливість додаткового дешифрування секретних даних з тієї самої шифрограми та алгоритм цієї процедури, зводять стійкість захисту до часу необхідного для дешифрування даних.

Враховуючи вищевказане основним з аспектів криптографічної стійкості алгоритмів заперечуваного дешифрування є обмеження доступу до інформації\ щодо структури та особливостей реалізації.

Автентифікація користувачів. Іншим важлим і складним для реалізації алгоритмів заперечуваного шифрування є процедура автентифікація користувачів. Її основним призначенням є автоматичний вибір типу відновлюваних з шифрограми. Реалізація вказаного механізму є досить критичною, при значній кількості даних, які відновлюються шифром.

Для забезпечення належного рівня захисту даних процедура має бути односторонньою, а також виключати участь користувачів у прийнятті рішень із визначення типу даних, які будуть відновлені алгоритмом шифрування.

Для її практичної реалізації рекомендовані наступні шляхи: мережевий, апаратний або програмний. Мережевий метод автентифікації користувачів є найпростішим, але потребує постійного підключення до мережі. За рахунок цього процедура прийняття рішення зводиться до автентифікації користувача на сервері (за особистими атрибутами доступу або іншими алгоритмами, які ґрунтуються переважно на нечіткій логіці). Подібний підхід повинен попередити втручання зловмисника та користувачів у процедуру автентифікації блоків з відновленими даними.

Локальним способами реалізації процедури автентифікації є застосування програмного або апаратного рішень:

1) Автентифікація за допомогою програмного забезпечення потребує генерації додаткового ключа авторизації (наприклад, для двох факторної автентифікації) користувача в системі. Вказаний ключ повинен обов’язково зберігатися на зовнішньому носії інформації, щоб у разі необхідності він міг бути знищеним. При цьому необхідно передбачити можливість його

відновлення з резервних копії, інакше користувач назавжди втратить доступ до даних.

2) Перевірка на базі апаратних рішень є більш простою. Її надійність ґрунтується на забороні модернізації існуючого апаратного забезпечення (наприклад, на внесення змін в електричну схему материнської плати комп'ютера) або на використанні зовнішніх апаратних пристроїв. Подібний спосіб повинен попередити спроби дешифрування закритих даних на інших пристроях та їх емуляторах (наприклад, у разі проведення криміналістичної експертизи носія без згоди власника).

2.2 Прикладна модель заперечуваного шифрування

Для демонстрації роботоспроможності запропонованої вище моделі створено прикладну модель, яка реалізує механізм заперечуваного шифрування даних з відкритим ключем на основі розширеної криптографічної схеми Рабіна [60].

Враховуючи недоліки вказаного способу заперечуваного шифрування даних на основі блочних шифрів [62], які підтверджуються експериментами, проведено аналіз вказаного способу заперечуваного шифрування даних та запропоновано інший підхід до вирішення цієї проблеми.

З цією метою розроблено та реалізовано наступні алгоритми, які пройшли апробацію в ході серії експериментів з використанням реальних даних на стендовому обладнанні:

1) вирівнювання наборів з даними;

2) розділення наборів даних на блоки шифрування/дешифрування;

3) автоматична ідентифікація відновлюваних даних;

4) шифрування та дешифрування (відкритих і закритих) даних.

З урахуванням особливостей структури вихідного алгоритму, при внесенні змін автори розділили його на 4 окремих частини: генерація ключів

(алгоритм 1), шифрування даних (алгоритм 2) дешифрування відкритих даних (алгоритми 3), дешифрування секретних даних (алгоритми 4).

Алгоритм генерації ключів (алгоритм 1) залежить виключно від алгоритму заперечуваного шифрування даних, який обирається користувачем.

Приклад опису даного блоку нижче за розділом.

Структурна схема алгоритму 2 приведена на Рисунок 2.8. Вона складається з функціональних блоків обробки даних, імплементації контрольних міток в блоки з вихідними даними, перетворення даних згідно з обраним алгоритмом заперечуваного шифрування (включає блоки перестановки та перетворення даних).

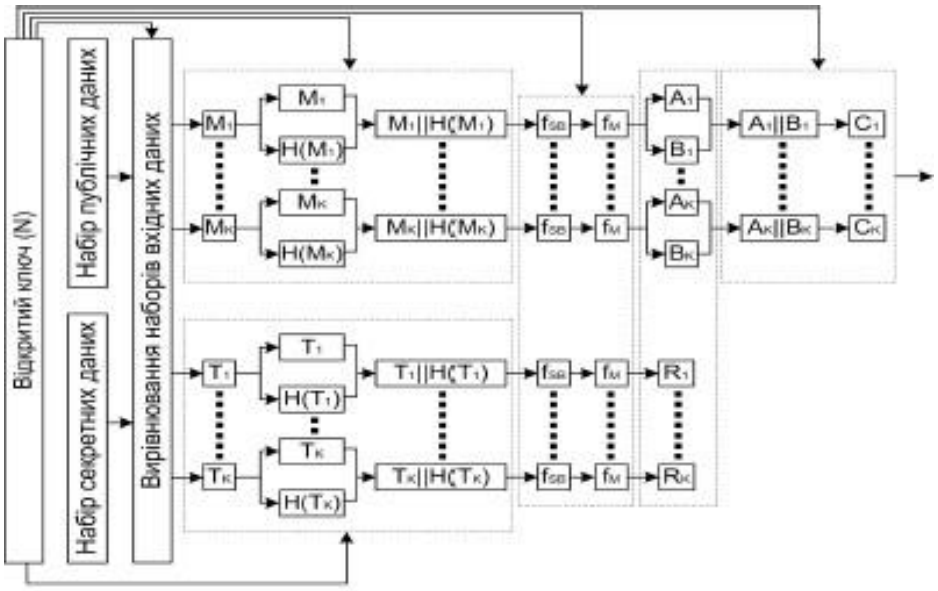


Рисунок 2.8 – Схема блокового шифрування даних (алгоритм 2)

На вищевказаній схемі блок обробки даних виконує завдання із завантаження даних у програму. Крім того, до переліку вирішуваних ним завдань входять субблоки підготовки даних, визначення розміру блоку шифрограми, розділення файлів з даними на блоки та перестановки байтів (аналог P-boxes).

Блок імплементації контрольних міток призначений для встановлення достовірності блоків з даними на етапі їх дешифрування та включає субблоки

формування контрольних міток і імплементації контрольних міток у блок з даних.

Блок перетворення (шифрування) даних залежить від алгоритму заперечуваного шифрування, який використовується користувачем. Приклад опису даного блоку нижче за розділом.

Схематичний опис алгоритму дешифрування публічних даних наведений на Рисунок 2.9 та складається з функціональних блоків обробки даних, дешифрування відкритих даних та аналізу даних:

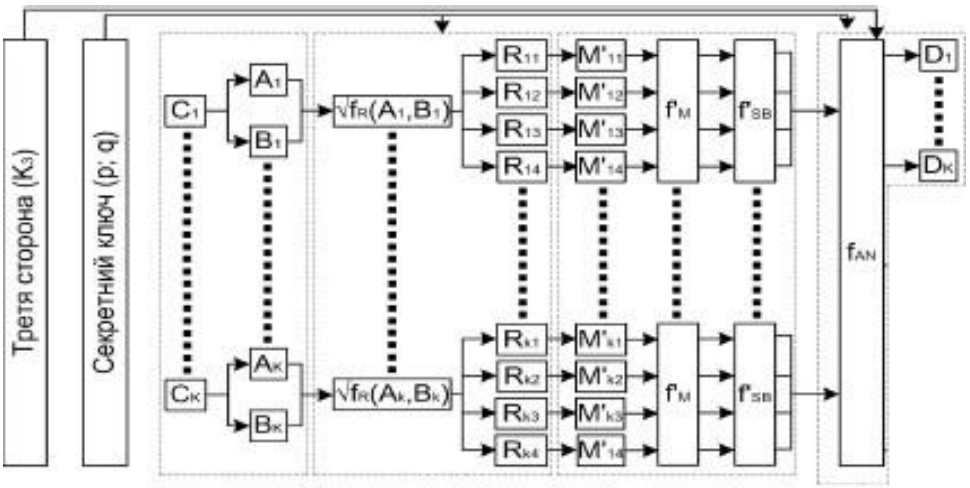


Рисунок 2.9 – Схема дешифрування відкритих даних (алгоритм 3)

На вищевказаній схемі блок обробки даних виконує завдання із завантаження шифрограми у програму та збереження отриманого результату. Крім того, до переліку вирішуваних ним завдань входить обчислення розміру блоку шифрограми та її розділення на частини.

Блок перетворення (дешифрування) даних залежить від алгоритму заперечуваного шифрування, який використовується користувачем. Блок аналізу даних призначений для виділення інформаційної складової із дешифрованих наборів даних, а також для вилучення контрольних міток, автентифікації дешифрованих блоків, зворотної перестановки байтів (аналог Pboxes).

Формальний опис алгоритму дешифрування закритих даних наведений на Рисунок 2.10 та включає блоки з обробки даних, дешифрування відкритих даних, дешифрування закритих даних та аналізу даних:

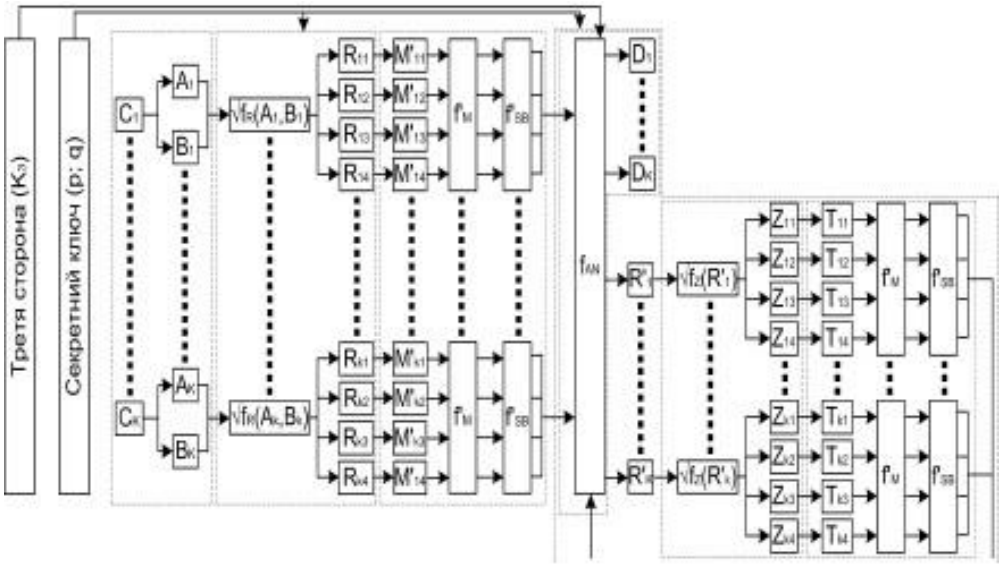


Рисунок 2.10 – Схема дешифрування закритих даних (алгоритм 4)

Згідно з вищевказаною схемою блок обробки даних виконує завдання із завантаження шифрограми у програму та збереження отриманого результату. До переліку вирішуваних ним завдань входить обчислення розміру блоку шифрограми та її розділення на блоки.

Блоки перетворення (дешифрування) відкритих і закритих даних залежать від алгоритмів заперечуваного шифрування, які використовується користувачем.

Блок аналізу даних призначений для виділення інформаційної складової з дешифрованих наборів даних. Він вирішує завдання з вилучення контрольних міток, автентифікації дешифрованих даних і зворотної перестановки байтів (аналог P-boxes).

Розділення даних на блоки в даній моделі може бути здійснення в двох режимах: з буферизацією та без буферизації.

Останній спосіб полягає в одночасному завантаженні файлів з даними до оперативної пам’яті робочої станції, в якій вони будуть зберігатися до завершення роботи програмного забезпечення. Вказаний спосіб може стати

причиною появи значних витоків пам’яті, нестабільної роботи робочої станції та вказаного програмного забезпечення в цілому. Від так перший спосіб поділу даних на блоки є більш оптимальним і включає наступний порядок дій:

1) Зчитати блоки даних розміром, згідно виразу (2.5).

2) Додати захисну мітку розміру

3) Обчислити хеш-образ блоку за допомогою однієї із доступних функцій хешування (md5, sha та ін) для формування контрольної мітки.

4) Додати контрольну мітку розміру

5) Виконати перестановку байтів в блоці. Для перестановки можуть бути використані статичні або динамічні P-boxes (крок виконується для посилення захисту даних, при бажанні користувача).

6) Повторювати кроки 1-5 доки не будуть оброблені усі блоки.

Розділення шифрограми на блоки подібне до поділу на блоки для шифрування. Однак є значно простішим, оскільки блоки даних мають статичний розмір, який залежить від розмір ключа шифрування. Від так вищевказана процедура декомпозиції даних може бути спрощена.

Процедура автоматичної ідентифікації відновлюваних даних є ядром вказаного алгоритму [60]. У разі виникнення будь-яких помилок під час її виконання, користувач може назавжди втратити доступ як до відкритих, так і до закритих даних. Вона включає наступний порядок дій:

1) Виділити з блоку даних контрольну мітку розміром S H -байт.

2) Обчислити хеш-образ блоку використовуючи заздалегідь визначену функцію хешування даних (md5, sha та ін).

3) Порівняти значення контрольної мітки (з кроку 1) та отриманого хеш-образу. У разі збігу перейти до відновлення наступних блоків, інакше перевірити інший елемент набору. Якщо перевірка усіх елементів набору не булла успішною, то припинити роботу алгоритму.

Для перевірки робото спроможності моделі блочного шифрування, оцінки її характеристик і можливості її використання на практиці, авторами була проведена серія експериментів.

Проведення експериментів з оцінки ефективності виконання та пошуку найбільш оптимальних параметрів в алгоритмах 2-4 передбачають наступний порядок дій: вибір вхідних параметрів та областей їх значень, виконання алгоритмів 2-4, вимірювання часу виконання алгоритмів 2-4, вимірювання витрат пам’яті на виконання алгоритмів 2-4, аналіз отриманих результатів та пошук найбільш оптимальних значень для кожного з алгоритмів і зміна параметрів та підходів до реалізації алгоритмів.

З метою дослідження роботоспроможності розробленого алгоритму та встановлення його оптимальних параметрів, при яких його робота найбільш ефективна, авторами проведено оцінки додаткових параметрів алгоритму (наприклад, кількість блоків, швидкість шифрування-дешифрування тощо) та проведено серії експериментів з оцінки часу і витрат пам’яті на виконання алгоритмів 2-4.

Також для перевірки рівня залежності показників ефективності моделі від технічних характеристик стендового обладнання (апаратних і програмних) проведено дослідження на двох різних стендах:

1) x86-based PC, ЦП x64 Family 16 Model 6 Stepping 3 AuthenticAMD ~1679 МГц, ОЗП 3067МБ, ОС Microsoft Windows 7 Ultimate 6.1.7601 Service Pack 1 збірка 7601, IDLE Python v3.6.2, бібліотеки line_profiler та mem_profiler, MS Excel 2003;

2) x64-based PC, ЦП Intel64 Family 6 Model 142 Stepping 10 Genuine Intel ~1801 МГц, ОЗП 8 ГБ, ОС Windows 10 Pro 10.0.18362 – Multiprocessor Free, IDLE Python v3.7, бібліотека line_profiler, MS Excel 2007.

З метою повної та усесторонньої оцінки ефективності роботи запропонованих алгоритмів автори відкинув оцінку алгоритму 1 та зосередився на алгоритмах 2-4.

Для проведення експериментів в тестовому середовищі 1 встановлено наступні обмеження:

1) формати вхідних даних: текстові (*.dat, *.txt, *.doc, *.xls), графічні (*.pdf, *.jpg, *.gif, *.png, *.bmp), медіа (*.avi, *.3gp, *.mp3) та інші (*.exe, *.dll, *.zip);

2) розмір вхідних даних: 1КБ ÷ 100 МБ;

3) розмір ключа шифрування: 128, 256, 512, 1024, 2048 та 4096 біт;

4) розмір контрольної мітки: 1 ÷ 2 байти;

5) розмір захисної мітки: 3 байти.

Для проведення експериментів в тестовому середовищі 2 встановлено наступні обмеження:

1) формати вхідних даних: текстові *.exe, *.zip;

2) розмір вхідних даних: 1КБ ÷ 20 МБ;

3) розмір ключа шифрування: 1024 (з точки зору продуктивності) та 8192 (з точки зору безпеки) біти;

4) розмір хешу даних: 1 ÷ 2 байти;

5) формат захисної мітки довільного змісту (наприклад, «!block!») та розміру від 3 байт.

За результатами експериментів в тестовому середовищі 1 були отримані дані, які описують роботу та ефективність запропонованих алгоритмів, та наведені на Рисунок 12 – 16.

Так, одною з основних цілей експериментів було встановлення залежності часу виконання алгоритмів 2-4 від розміру вхідних даних і криптограми для оцінки можливості застосування вихідного алгоритму для блочного шифрування даних та реалізації на основі сучасних програмно-апаратних рішень (рис. 2.12 і 2.13):

Іншим ключовим питанням було встановлення залежності часу виконання алгоритмів 2-4 від розміру ключа шифрування, оскільки в запропонованій версії вихідного алгоритму розмір ключа безпосередньо впливає як на кількість ітерацій в алгоритмах, так і на витрати пам’яті для їх виконання (рис. 2.13 і 2.14).

Вище вказані результати доводять можливість практичної реалізації ідеї блочного шифрування даних з використанням асиметричних схем перетворення даних, а також дають перспективи для їх подальшого дослідження.

В результаті проведених експериментів встановлено, що в порівнянні показниками продуктивності блочних алгоритмів шифрування [58, 62], запропонований алгоритм значно програє в продуктивності (приблизно до 2 разів, в залежності від розміру використаних ключів). Але так само, за певних умов, запропонований алгоритм демонструє кращі показники продуктивності у порівнянні з подібними алгоритмами.

Після проведеного аналізу моделі заперечуваного шифрування даних встановлено, що суттєвий вплив на ефективність роботи та можливість його застосування створюють витрати часу та пам’яті, які необхідні для його виконання, розміри вихідних даних та ключа, а також ступінь декомпозиції вихідних даних на блоки.

При узагальненому аналізі роботи алгоритму 2 встановлено, що суттєвий вплив на розмір кінцевої криптограми створює різниця в розмірах вихідних файлів з відкритими та закритими даними. Таким чином, при різниці в ~30%, розмір вихідної криптограми збільшується в 2-2,5 рази порівняно з теоретичними розрахунками (2.16).

Вказане пояснюється неможливістю підбору вихідних даних з однаковими розмірами та необхідністю їх особливістю алгоритму в частині вирівнювання файлів на етапі їх шифрування. Подібні коливання спостерігаються, при виконанні алгоритмах 2-4, оскільки під час експериментів виконувалися наступні умови (2.17).

При аналізі алгоритму 2 встановлено, що за будь-яких обставин його часові характеристики в більшій степені рівномірні та носять експоненціальний характер, що робить його придатним для роботи на будь-яких ЕОМ. Також, при подальшому аналізі показників, встановлено, що час виконання алгоритму 2 спадає на деяких проміжках часу, оскільки між вхідними даними незначна. При

тестуванні алгоритму 2 з вхідними даними розміром 1 МБ та більше, час шифрування вихідних даних збільшується до 10 разів.

Крім того, час шифрування відкритих та закритих даних в значній мірі залежить від розміру ключа шифрування. При оцінці впливу розмірів ключа на виконання алгоритму 2 встановлено, що найбільший час виконання алгоритму спостерігається, при використанні 128-бітового ключа, та середні, при використанні 512-2048-бітових ключів. Таким чином, експериментально встановлено, що найбільш оптимальний час шифрування даних забезпечується, при використанні ключа в 1024 біти.

Цілком інша ситуація спостерігається при виконанні алгоритмів 3 і 4, оскільки різниця в часі їх виконання досягає 150-200%. Так, при розмірі криптограми від 3 МБ, спостерігається збільшення часу на виконання алгоритмів 3 і 4, а при 11 МБ та більше спостерігається незначне спадання часу виконання алгоритму 3 та його значне зростання для алгоритму 4. Вказане є наслідком подвоєння кількості обчислень в алгоритмі 4 порівняно з алгоритмом 3.

Також час виконання алгоритмів 3 і 4 так само залежить від розміру ключа, як і алгоритм 2. Так при детальному аналізі роботи алгоритмів 3 і 4 встановлено, що середні показники часу їх виконання спостерігаються, при використанні 128, 256, 512 та 1024-бітових ключів, та гірші, при 2048 і 4096 бітового ключах. Проте порівняно з алгоритмом 2 присутні деякі розбіжності, оскільки:

1) в алгоритмі 2 найбільш оптимальний розмір ключа становить 1024 біт;

2) в алгоритмі 3 найбільш оптимальний розмір ключа становить 512 біт;

3) в алгоритмі 4 найбільш оптимальний розмір ключа становить 256 біт.

Враховуючи вищевказане, з метою пошуку причин збільшення часу виконання алгоритмів 2-4 та їх усунення, було проведено оцінку витрат пам’яті, які відведені на їх виконання. Так в ході експериментів з алгоритмом 2 встановлено, що витрати пам’яті на шифрування вхідних даних зростають пропорційно розміру вхідних даних та є значними, при використанні ключів до

512 біт включно. Вказане пояснюється тим, що при використанні ключів малого розміру кількість блоків даних зростає, що призводить до збільшення кількості обчислень з ними. Інше спостерігається, при аналізі алгоритмів 3 і 4.

Так при дешифруванні відкритих та закритих даних витрати пам’яті значно збільшуються порівняно з алгоритмом 2, але в межах алгоритмів 3 і 4 є майже однакові для ключів 512-2048 біт. Вказане є наслідком вирівнювання вихідних даних в алгоритмі 2.

Таким чином, запропонована ідея блочного шифрування даних для реалізації механізмів заперечуваного шифрування, в якому не використовуються існуючі блочні алгоритми шифрування, лише їх елементи, в порівнянні з іншими рішеннями [58, 62] забезпечує збільшення швидкості шифрування даних до 20 разів та дешифрування даних до 5 разів (залежно від розміру ключа).

Порівняння продуктивності запропонованого авторами алгоритму з першочерговою ідеєю та блочним алгоритмом шифрування даних [62] наведені в табл. 2.4 та на рис. 2.16:

Чисельні результати, які були отримані в ході експериментів, свідчать про те, що попри структурні особливості на кінцеву продуктивність запропонованої моделі суттєвий вплив здійснюють характеристики апаратного та програмного забезпечення стенду.

Згідно з отриманими результатами швидкість роботи моделі в середовищі 2 зросла до 7-10 разів. Вказане пояснюється:

1) збільшенням кількості обчислювальних ядер в 2 рази;

2) використання технології Hyper-Threading збільшило кількість потоків обробки даних в 2 рази;

3) збільшення розміру оперативної пам’яті з 3 до 8 ГБ.

РОЗДІЛ 3.
МОДЕЛІ ЗАПЕРЕЧУВАНОГО ШИФРУВАННЯ ДАНИХ

3.1 Модель заперечуваного шифрування з кодуванням даних

Алгоритми заперечуваного шифрування мають досить низьку продуктивність. Причиною цього є необхідність вирішення великої кількості важких задач за обмежений проміжок часу. Експерименти проведені у попередніх розділах підтверджують гіпотезу щодо можливості управління продуктивністю перетворення даних за рахунок маніпулювання характеристиками інформації, зокрема її розміром.

Враховуючи дані експериментів з запропонованими у попередніх розділах моделей виникла необхідні модифікувати вихідну та підхідні від неї моделі перетворення даних. Внаслідок цього вирази (2.1), (3.1) та (3.4) зазнали змін (4.1) – (4.3):

$$\begin{cases} Y = f_1(f_C(X)) \\ X' = f_D(f_2(Y)) \end{cases} \quad (4.1)$$

$$\begin{cases} Y = f_1(f_C(X_{i,1})) + f_1(f_C(X_{i+1,1})) + f_1(f_C(X_{i+2,1})) + \ldots + f_1\left(f_C\left(X_{k-1,\frac{1}{k}}\right)\right) \\ X' = f_D(f_2(Y_{i,1})) + f_D(f_2(Y_{i+1,2})) + f_D(f_2(Y_{i+2,3})) + \ldots + f_D\left(f_2\left(Y_{k-1,\frac{1}{k}}\right)\right) \end{cases} \quad (4.2)$$

$$\begin{cases} FOD_i = \left\{X_{1\ldots k,1\ldots n};\ Y_{1\ldots k,1\ldots n}\right\} \\ Y = f_1(f_C(FOD_1)) \| f_1(f_C(FOD_2)) \| \ldots \| f_1(f_C(FOD_k)) \\ X' = f_D(f_2(FOD_1)) \| f_D(f_2(FOD_2)) \| \ldots \| f_D(f_2(FOD_k)) \end{cases} \quad (4.3)$$

де f_C – алгоритм кодування даних; f_D – алгоритм декодування даних.

Модифікація вихідної та інших моделей перетворення даних призвело до появи в модулях прямої та зворотної обробки даних блоків, які відповідають за кодування даних.

Основним завданням вказаних модулів є оцінка доцільності кодування, пряме та зворотне кодування даних з використанням існуючих алгоритмів кодування даних.

Такий підхід до побудови системи перетворення даних дозволяє: виключити можливість появи додаткових вразливостей в структурі алгоритмів заперечуваного шифрування:

1) гнучко управляти розміром вихідних даних;

2) використовувати кодування, як додатковий (не основний) інструмент обробки даних;

3) збільшувати рівень захищеності даних за рахунок використання додаткових перетворень за незначний проміжок часу.

За результатами вивчення методів оцінки ефективності алгоритмів кодування інформації виявлено спосіб математичного прогнозування імовірного рівня компресії даних [91]. Однак в ході чисельних експериментів вказаного критерію встановлено, що оцінка теоретичного та фактичного рівнів компресії даних відрізняються. При цьому, відмінність їх значень становить близько 2,5 раз.

Таким чином, кінцевий алгоритм прогнозування теоретичного рівня компресії файлів з даними включатиме наступні кроки:

1) обчислити теоретичний рівень ентропії вихідних файлів з даними H(M);

2) обрати значення середньої довжини кодового слова Lcp, відповідно до характеристик використаного алгоритму кодування та налаштувань програмного середовища;

3) обчислити значення коефіцієнту компресії Kk (4.4).

При цьому, доцільність використання алгоритмів кодування існує лише за умови, якщо $Kk \rightarrow 1$ і час відведений на стиснення даних $T \rightarrow 0$.

Окрім того, в п.п 1.3.3 зазначено, що вказаний спосіб оцінки рівня компресії даних не може бути застосований до файлів, які зазнали

попереднього стиснення (наприклад, графічні файли формату GIF, JPG, тощо) або шифрування.

Реалізація моделі з попереднім кодуванням даних має деякі особливості. До них можна віднести:

1) реалізацію ефективних алгоритмів кодування даних з можливістю їх гнучкого налаштування;

2) зменшення часу відведеного на кодування вихідних даних за рахунок використання апаратного забезпечення з оптимальними параметрами.

Використання адаптованих алгоритмів кодування даних і програмного забезпечення розробленого на їх основі дозволяє наблизити коефіцієнт компресії даних до розрахункового (4.4).

Разом з тим використання алгоритмів кодування даних призводить появи додаткових витрат часу на компресію вихідних даних. Оскільки швидкість компресії так само, які і продуктивність роботи решти програмного забезпечення залежить від обчислювальних потужностей робочого місця. Разом з тим користувач може скоротити час необхідний системі для компресії вихідних даних за рахунок того, що буде виконувати їх обробку без попереднього вилучення з архіву. Вказаний підхід призводить до необхідності використання додаткового програного забезпечення, яке дозволяє здійснювати доступ до файлів без необхідності їх попереднього вилучення з архіву.

Особливості використання розроблених моделей шифрування даних такі самі, як і в розділах 2 та 3. Крім того, наслідком використання вищевказаних алгоритмів кодування даних і програмного забезпечення на їх основі є поява вразливості безпеки, яка полягає у формуванні тимчасових файлів з даними під час доступу до них користувача.

Небезпека вказаної вразливості ґрунтується на неконтрольованому поширенні даних і їх фрагментів, внаслідок їх обробки без їх попереднього вилучення з архівного контейнеру. Часто вказана вразливість призводить до появи копій або фрагментів файлів з даними у директорії з тимчасовими файлами на системному накопичувачеві інформації.

З метою попередження витоку даних та отримання системної інформації про використані алгоритми обробки і захисту даних, рекомендовано:

1) реалізувати додаткові алгоритми очищення вмісту тимчасових директорій або виконати налаштування системного програмного забезпечення для виконання вказаного завдання;

2) виконувати вилучення даних з архівного контейнеру (потребує витрат часу на повторну компресію даних).

Згідно з інформацією, яка викладена в розділах 1 і 2 встановлено, що одним з напрямів підвищення продуктивності алгоритмів заперечуваного шифрування є використання алгоритмів кодування даних. Використовуючи вказане твердження виконано побудову моделі заперечуваного шифрування, яка використовує алгоритми кодування для зменшення кількості блоків з даними BQ → 0. При цьому виділяють наступні блоки обробки даних: підготовка (кодування) даних; шифрування та дешифрування (відкритих і

закритих) даних. Схема, яка реалізує вказані перетворення наведена на рис. 4.1:

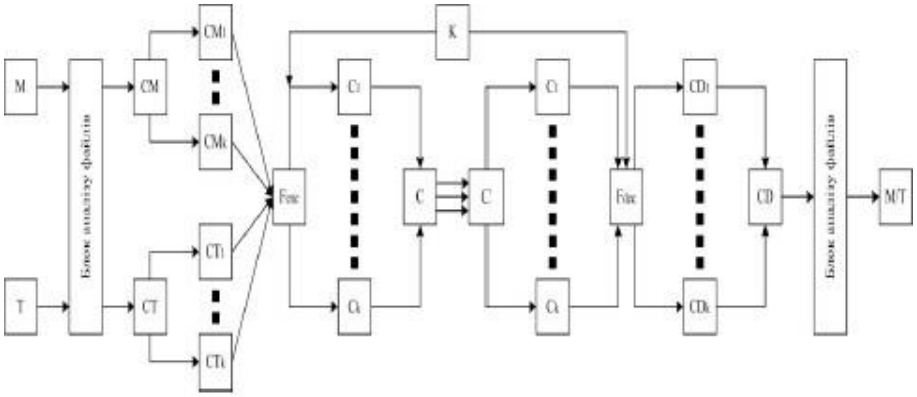


Рисунок 4.1 – Схема моделі заперечуваного шифрування

На вищевказаній схемі шифрування блок підготовки даних виконує попередню обробку даних забезпечуючи приведення даних до найбільш придатного для шифрування вигляду. Враховуючи запропоновані у попередніх розділах рішення вказаний блок включає наступні субблоки: обробки вихідних даних і результатів обчислень.

Субблок обробки вихідних даних виконує приведення відкритих і закритих даних до вигляду, який найбільш придатним для подальшого перетворення у програмі. Він включає наступні алгоритми: оцінка можливого рівня стиснення вихідних даних, вирівнювання вихідних файлів відносно одне одного, вирівнювання вихідних файлів відносно ключа шифрування, декомпозиції вихідних даних на фрагменти та окремі блоки.

Субблок обробки результатів обчислень виконує їх перетворення в дані, які зберігаються у вигляді шифрограми або дешифрованих даних.

В загальному випадку алгоритм підготовки даних до шифрування передбачає виконання наступного порядку дій:

1) Виконати перевірку формату вихідних даних. У разі якщо він відноситься до групи форматів, які використовують алгоритми кодування на етапі формування даних, перейти до кроку 2. Інакше алгоритм повинен завершити свою роботу.

2) Обрати ефективний алгоритм кодування даних, який забезпечить виконання умови.

3) Використовуючи алгоритм прогнозування імовірного рівня компресії даних Kk (4.4), виконати оцінку доцільності використання алгоритмів кодування для зменшення розміру файлів з даними. У разі отримання позитивної оцінки Kk → 1 виконати процедуру компресії файлів використовуючи параметри, які були використані під час оцінки прогнозованого рівня компресії даних, інакше перейти до шифрування даних.

Результатом виконання даного алгоритму є зменшення кількості блоків обробки даних і як наслідок зростання продуктивності шифрування. Вказане прискорення може бути оцінене виразом (4.5):

Алгоритм відновлення даних на етапі їх дешифрування ґрунтується на зворотному перетворенні (декодуванні) даних. Внаслідок цього відновлюється початковий набір даних відповідного розміру. Він передбачає виконання наступного порядку дій:

1) Виконати аналіз службових даних шифрограми для встановлення необхідності їх декомпресії після дешифрування даних.

2) Отримати інформацію щодо вихідного алгоритму кодування та його налаштувань, які були використані під час їх шифрування вихідних даних.

3) Використовуючи гнучке програмне забезпечення та відповідні налаштування виконати процедуру декомпресії файлів.

Результатом виконання даного алгоритму є отримання файлу з вихідними (відкритими або закритими) даними. Реалізація ефективних процедур шифрування та дешифрування даних, які ґрунтуються на використанні механізмів заперечуваного шифрування були описані в попередніх розділах.

Вищевказані перетворення не змінюють базовий алгоритм, але їх використання, в теорії, повинне збільшити швидкість роботи моделі в - 1k K - разів.

Для перевірки ефективності запропонованого підходу та його впливу на кінцеву продуктивність алгоритмів заперечуваного шифрування даних було проведено обчислювальні експерименти: з прогнозування рівня компресії вихідних даних, оцінки залежності рівня компресії даних від налаштувань алгоритмів кодування даних і оцінки продуктивності процедур шифрування та дешифрування від досягнутого рівня компресії вихідних даних.

Для проведення експериментів було використано стендове обладнання з наступними характеристиками: ЦП x64-based PC, Intel64 Family 6 Model 142 Stepping 10 Genuine Intel ~1801 МГц, 8 ГБ RAM, HDD 500 ГБ, ОС Windows 10 Pro (10.0.18362 – Multiprocessor Free, IDLE Python v3.7, бібліотека line_profiler, MS Excel 2007, 7-Zip архіватор.

Для проведення експериментів з використанням запропонованої моделі та можливості відтворення їх результатів і їх подальшого використання у дослідженнях алгоритмів заперечуваного шифрування встановлені наступні вихідні дані:

1) розмір вхідних даних 1КБ-1,5ГБ;

2) формати тестових файлів: BMP, PNG, GIF, JPG, TXT, DOCX, PDF, EXE

3) розмір ключа шифрування: 1024 (з точки зору продуктивності) та 8192 (з точки зору безпеки) біти;

4) розмір хешу даних: 1 ÷ 2 байти;

5) формат захисної мітки довільного змісту (наприклад, «!block!») та розміру від 3 байт.

6) методи кодування даних: deflate, lzma2.

Основна ідея вищевказаної моделі ґрунтується на можливості отримання приросту швидкодії алгоритмів заперечуваного шифрування внаслідок попередньої компресії тестових даних.

Так під час дослідження обраних алгоритмів кодування було визначено максимальний рівень компресії тестових файлів з використання виразу (4.4). Для отримання вказаних показників стиснення даних були виконані відповідні налаштування програмного забезпечення, зокрема була досліджена залежність рівня компресії даних від довжини кодового слова Lcp .

Однак отримані під час проведення експериментів показники компресії відрізняються від теоретичних. Їх аналіз показав, що причиною таких отримання подібного результату стало те, що вираз (4.4) враховує залежність k K лише від Lcp .

Також під час експериментів була перевірена залежність часу k T та коефіцієнту компресії k K від Lcp . Як наслідок встановлено, що зміни значення Lcp незначною мірою впливає на k K . Середня різниця між значеннями k K при зміні Lcp становить 2-4% (табл. 4.5, 4.6). При цьому спостерігається зворотній ефект на показник k T . Експериментальним шляхом встановлено, що значення k T , при Lcp (табл. 4.3, 4.4). Окрім того, виявлено, що використання методу кодування LZMA2 дає в 3 рази кращі часові характеристики під час компресії тестових файлів, аніж Deflate.

Під час проведення порівняльного аналізу швидкодії тестових моделей обробки даних були перевірені залежність кількості блоків, часу виконання

процедур перетворення даних у поєднанні з алгоритмами кодування даних і залежність вказаних результатів від використання ключів різного розміру S K .

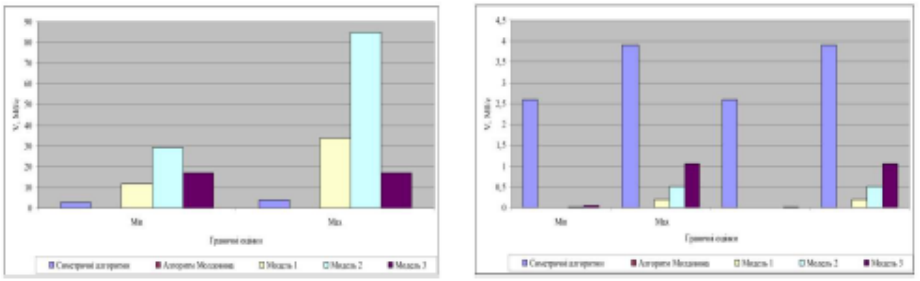
В результаті експериментів було отримане незначне прискорення в роботі процедур перетворення даних. Вказані результати є наслідком впливу коефіцієнту компресії k K на розмір тестових даних, що призвело до скорочення блоків з даними BQ у тестових файлах до 16 разів. При цьому суттєвого прискорення зазнала процедура дешифрування закритих даних. В результаті поєднання алгоритму компресії даних LZMA2 та моделі 3 призвело до зменшення часу необхідного на обробку даних до 2,5-3 разів. Таким чином, кінцева швидкість перетворення даних склала близько 1,06 Мб/с.

На відміну від вищевказаних результатів, використання в досліджуваних моделях ключа розміром 8192 біт не призвело до очікуваного результату. Під час вивчення причин виникнення даної ситуації встановлено, що в даному випадку кількість блоків з даними також зазнала скорочення до 16 разів. Разом з тим ключ більшого розміру повинен був призвести їх до подальшого зменшення внаслідок зростання розміру блоків. Однак використання 8192-бітних значень потребує більш потужного апаратного забезпечення і розробки ефективних протоколів їх програмної обробки. Незважаючи на те, що програмне середовище розробки дозволяє виконувати подібні обчислення, при використанні інших мов програмування вказане дослідження мало б значні ускладнення.

Тим не менш, отримані показники продуктивності є досить перспективними для практичного застосування. З метою отримання їх об’єктивної оцінки було виконано їх порівняння з даними швидкодії окремих блочних алгоритмів шифрування та алгоритмами заперечуваного шифрування даних створеними на їх основі. Результат їх порівняння викладений в табл. Результати порівняння викладені в табл. 4.19 (рис. 4.3):

Порівняння результатів тестування з компресією

Процедури	Тип даних	Показники	Оцінки продуктивності, Мб/с				
			Симетричні алгоритми	Алгоритм Молдовяна	Розроблені моделі		
					Модель 1	Модель 2	Модель 3
Шифрування	Відкриті	Min	2,602	0,001	11,54	29,24	16,87
		Max	3,904	0,010	34,03	85,08	16,87
	Закриті	Min	2,602	0,001	11,54	29,24	16,87
		Max	3,904	0,010	34,03	85,08	16,87
Дешифрування	Відкриті	Min	2,602	-	0,007	0,015	0,042
		Max	3,904	-	0,206	0,515	1,06
	Закриті	Min	2,602	-	0,002	0,005	0,012
		Max	3,904	-	0,206	0,515	1,06



а б

Рисунок 4.3 – Порівняння результатів продуктивності
запропонованих моделей з існуючими алгоритмами шифрування

Вищевказані дані порівняння швидкодії блочних адаптивних моделей заперечуваного шифрування у поєднанні з алгоритмами кодування даних призвели до 30% від швидкодії симетричних алгоритмів. Незважаючи на це, вищевказані показники продуктивності не враховують час компресії і декомпресії даних, лише витрати на перетворення даних. Тому кінцевий час перетворення даних може відрізнятися від вказаного. Разом з тим, при використанні ефективних алгоритмів кодування – 0 k T .

3.2 Модель розподіленого заперечуваного шифрування

При моделювання механізмів заперечуваного шифрування даних на базі розподіленої системи були досліджені наступні питання: реалізація моделі мережевого обміну даними, вибір оптимальної моделі обчислень, реалізація основних і допоміжних алгоритмів обчислень, особливості реалізації і практичного застосування моделі. Для перевірки ефективності запропонованих рішень проведено експериментальне дослідження моделі.

В основі будь-якої розподіленої (мережевої) системи обробки даних лежить використання моделі «клієнт-сервер» та її варіацій. В загальному випадку вказана модель є дволанковою (рис. 5.1), до її складу входять клієнт і сервер:

У вищенаведеній схемі клієнт формує запити на обробку даних і передає їх на сервер разом з усією необхідною для їх виконання інформацією. В свою чергу сервер виконує отримання та обробку запитів, обробку даних і відправку результатів обчислень на клієнт.

Алгоритм функціонування клієнта досить простий. Він передбачає виконання наступних дій:

1) Відправка запиту на з’єднання з сервером.

2) Отримання відповіді від серверу.

3) Відправка запиту на виконання завдання сервером.

4) Отримання запиту від серверу з переліком даних, які необхідні йому для розв’язання завдання.

5) Відправка даних на сервер.

6) Отримання результатів обчислень.

7) Відправка запиту на завершення з’єднання з сервером.

8) Закриття з’єднання на стороні клієнту.

На відміну від клієнта, сервер потребує більш широкого функціоналу для виконання різноманітних запитів з обробки даних. При цьому функціонал може поповнюватися шляхом підключення відповідних модулів. Алгоритм функціонування серверу включає наступні кроки:

1) Отримання запиту на з’єднання з клієнтом.

2) Відправка відповіді на клієнт.

3) Отримання відповіді на клієнт щодо можливості виконання завдання сервером.

4) Відправка на клієнт запиту з переліком даних, які необхідні для розв’язання завдання.

5) Отримання даних від клієнта.

6) Виконання обчислень.

7) Відправка результатів обчислень на клієнт.

8) Отримання від клієнту запиту на завершення з’єднання.

9) Відправка клієнту підтвердження на завершення з’єднання.

10) Закриття з’єднання на стороні серверу.

Кроки 1-5, 7-10 є стандартними для серверів обробки запитів. Однак на виконувані обчислення (крок 10) залежать виключно від обраного алгоритму перетворення даних.

В контексті вищевказаних алгоритмів була визначена можлива структура програмного забезпечення, яке розміщується на мережевих вузлах.

Вищевказана структурна схема є досить узагальненою. Вона включає наступні субблоки: мережевий агент, ядро, база даних, блок обчислень та інтерфейс користувача (рис. 5.3):

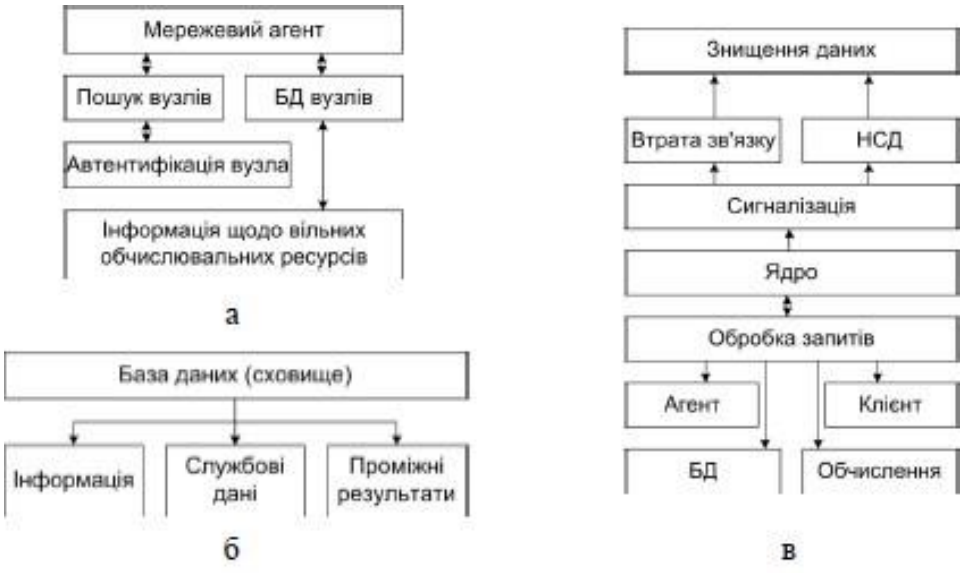


Рисунок 5.3 – Структура мережевого агенту, бази даних, ядра програмного забезпечення на вузлах

Інтерфейс користувача є стандартним модулем взаємодії між користувачем і програмним забезпеченням. Його використання в даній моделі обчислень призначене для отримання завдань і відповідного набору даних від користувача для їх наступного виконання. Він не має окремих функціональних блоків.

Мережевий агент – це універсальний програмний модуль. Його використання дозволяє забезпечити безперервний канал зв’язку між клієнтом і сервером, а також реалізує наступні процедури: пошук вузлів, оновлення бази вузлів, автентифікація вузлів, визначення вільних обчислювальних ресурсів, підключення до серверу, закриття з’єднання з сервером, обмін інформацією між клієнтом і сервером.

Програмний модуль, який реалізує функції ядра, призначений для управління запитами та обчислювальним процесом на вузлах. Він реалізує процедури: знищення тимчасових даних і результатів обчислень, моніторинг активності мережевого зв’язку, реагування на несанкціонований доступ до обчислювального вузла, формування та обробки запитів, виконання обчислень та формування їх результатів.

База даних – це накопичувач інформації, який використовується вузлами для зберігання відкритих, закритих, службових, тимчасових даних і ключової інформації. В ньому немає окремих процедур, оскільки його використання здійснюється ядром і обчислювальним блоком.

Обчислювальний блок (рис. 5.4) реалізує перетворення, які виконуються над даними. Зокрема в ньому реалізовані наступні процедури: шифрування даних, дешифрування відкритих (публічних) і закритих (секретних) даних, монопотокова та багатопотокова обробка даних, кодування (компресія) даних, мінімізація обчислень за рахунок декомпозиції даних:

Вищевказана модель є основою розподіленої системи обчислень. Разом з тим незалежно від реалізації її практичне застосування передбачає втрати часу на обмін даними між вузлами. Він більшості випадків вони залежать від технологій побудови мережі та відповідного обладнання. Тому під час вирішення завдання з визначення структури розподіленої системи було створено та досліджено декілька моделей: клієнт-клієнт, клієнт-сервер та клієнт-сервер-клієнт.

Система обробки даних, в основі якої лежить модель «клієнт-клієнт», є найпростішою для організації обчислень. Вона подібна до моделі «клієнт-

сервер», але на відміну від неї не передбачає використання виділеного серверу для обробки запитів і подальшої обробки даних. Для вказаних цілей використовується один з мережевих вузлів, який ініціює обробку даних (рис. 5.5). Для коректного функціонування такої системи необхідно розмістити пакети з модулями вихідного програмного забезпечення на кожному вузлі в мережі.

Під час обробки запитів головний вузол виконує поділ даних і завдань на їх обробку між доступними на момент обробки вузлами (обчислювачами). Після того як вузли завершують виконання обчислень вони повертають результати на головний клієнт для формування вихідного файлу з даними (шифрограмою). Реалізація вищевказаної схеми обчислень неминуче призводить до появи додаткових витрат часу на обмін даними між вузлами мережі, який впливає на сумарний час обробки даних кожним вузлом в цілому (5.1).

Незважаючи на те, що вказаний підхід призводить до дублювання програмного коду та можливості доступу до нього третіх осіб (у разі несанкціонованого доступу на вузлах), його перевагами подібного рішення є можливість виконання програмного коду без необхідності його передачі по мережі та можливість ініціювання обробки даних на будь-якому вузлі, як локально (автономно, так і за умови наявності в мережі вільних вузлів з відповідним програмним забезпечення. Окрім того, відмова від використання виділеного серверу дозволяє зменшити витрати часу на обмін даними між головним вузлом та обчислювачами, а також знижує імовірність моніторингу мережевого трафіку зловмисниками.

До недоліків подібного рішення обмеження можливостей масштабування системи (до 28 3 вузлів), стійкість системи до відмов і можливість хибного спрацювання сигналізації (у разі вимкненні вузлів), що призводить до падіння продуктивності всієї системи.

Система обробки даних на основі моделі «клієнт-сервер» є класичною схемою обробки даних [95], яка використовується для розв’язання завдань з

використанням ресурсів мережі. В її основі лежить централізована обробка запитів від інших вузлів, їх виконання на виділеному сервері та повернення результатів обчислень відповідному вузлу. Разом з тим в контексті вищевказаних завдань сервер або група серверів повинні мати значні обчислювальні ресурси для виконання поставлених завдань, в іншому він використовується як балансувальник навантаження. Окрім обробки запитів від вузлів, він виконує централізований розподіл завдань між рештою мережевих вузлів і контроль за їх виконанням (рис. 5.6):

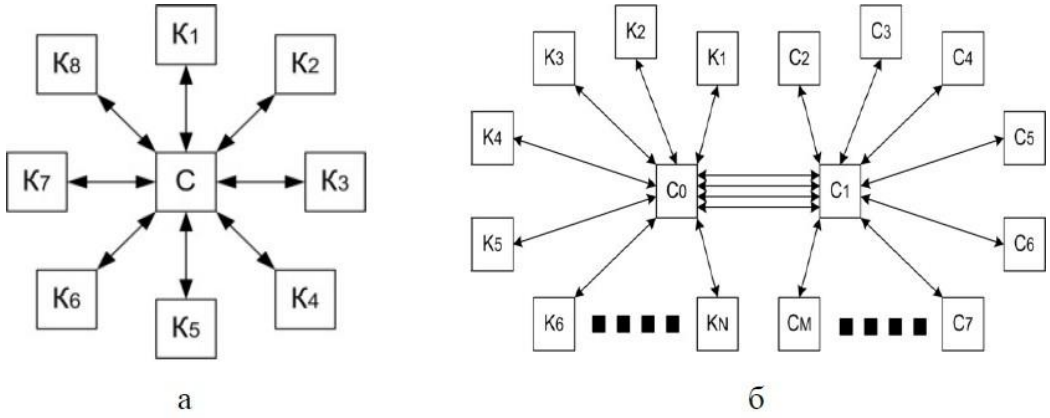


Рисунок 5.6 – Структурна схема моделі клієнт-сервер

Подібно до попередньої моделі вказане рішення також дозволяє будь-якому з вузлів ініціювати перетворення даних. Для цього вузол повинен передати на сервер інформацію щодо необхідних перетворень та деякі вихідні дані (файли з даними або шифрограмою і відповідний ключ). При цьому можуть бути виділені 3 режими роботи вказаної системи:

1) сервер (С) виконує роль обчислювача, з достатньою кількістю ресурсів для виконання будь-яких запитів від вузлів;

2) сервер (С) не має достатньої кількістю ресурсів для виконання будь-яких запитів від вузлів, тому він працює в режимі балансувальника навантаження між головним вузлом і рештою обчислювачів;

3) сервер (0С) працює в режимі балансувальника навантаження між групою серверів з потужними обчислювальними ресурсами для виконання запитів від вузлів.

У разі наявності достатньої кількості ресурсів сервер виконує процедури обробки даних самостійно і повертає клієнтам результати обчислень у вигляді файлів з даними (шифрограмою). В іншому випадку виконуючи обробку запиту від вузлів сервер проводить пошук вільних у мережі вузлів, які відповідають технічним характеристикам необхідним для виконання перетворень і мають вільні обчислювальні ресурси і, в якості балансувальника навантаження, виконує розподіл завдань з обробки даних між доступними на поточний момент вузлами (відправляє обчислювачам дані з описом необхідних перетворень).

Після завершення обчислень вузли повертають результати обчислень на сервер, де відбувається їх трансформація у файл з даними (шифрограмою). Вказаний файл повертається вузлу, який ініціював перетворення даних.

З метою централізації обчислень, в останній схемі (рис. 5.6) допускається використання групи серверів 1...MC з потужними апаратним забезпеченням, основним призначенням яких буде виконання важких обчислень замість клієнтів.

До переваг даного рішення можна віднести можливість виконання програмного коду без його розміщення на мережевих вузлах, що підвищує захист програмного забезпечення від несанкціонованого доступу, при цьому кожний вузол мережі має можливість ініціювати обробку даних (режими 1 і 3).

Недолік режиму 2, як і в попередній моделі, хоч і дозволяє автономну обробку даних на вузлах, але збільшує ризики доступу до програмного коду, його подальшого аналізу та модифікації зловмисниками. Подібно до попередньої моделі, у вказаній реалізації зростають витрати часу на обмін даними між вузлами і серверами (5.2).

Незважаючи на це в даному рішенні також присутній недолік пов'язаний з масштабуванням системи (до 28-2 серверів). При цьому збільшує збільшується стійкість системи до відмов шляхом проведення замкнутого циклу обчислень. У випадку несанкціонованого доступу до одного з серверів, він може бути вимкнутим, а його завдання будуть розділені між рештою (лише для режиму 3).

Дана модель обчислень на відміну від попередніх призначена для масштабування обробки даних (PCL). Вона не потребує, щоб вузли знаходилися в одній мережі. Натомість вона передбачає кооперацію між серверами різних мереж з метою отримання доступу до їх обчислювальних ресурсів. Такий підхід до організації обчислень дозволяє нескінченно нарощувати потужність системи (рис. 5.7).

Стійкість системи до відмови забезпечується централізованим розподілом навантаження з використанням кластеру серверів, які обробляють запити від вузлів всієї системи та виконують рівномірний розподіл завдань між вузлами в залежності від їх технічних характеристик. Однак значне розгалуження обчислень потребує більших витрат часу на обмін даними між вузлами і серверами, ускладнює процедуру поділу та моніторингу виконання завдань, а також створює передумови для появи проблем пов’язані з безпекою обчислень (поверхня атаки зловмисників розширюється пропорційно N -мереж).

Остаточний час виконання розподілених системою завдань може бути оцінений виразом (5.3).

Незважаючи на зростання кінцевого часу обробки даних, продуктивність системи може досягати максимального значення. Але реалізація подібного рішення досить складна в зв’язку з неоднорідністю апаратного та програмного забезпечення вузлів, а також мережевих технологій використаних для обміну даними між ними.

Таким чином, вищевказані моделі обробки даних мають свою переваги та недоліки. З метою вибору найбільш оптимальної моделі обчислень виконано їх порівняння. Основними критеріями були простота реалізації, мінімальні втрати часу на передачу та захист даних. За результатами оцінки вищевказаних моделей встановлено:

1) найбільш оптимальною з точки зору простоти реалізації і мінімальних витрат часу на обмін даних є модель «клієнт-клієнт»;

2) кращі показники продуктивності можуть бути досягнуті в мультисерверній моделі і моделі клієнт-сервер-клієнт за рахунок масштабування обчислень;

3) використання мультисерверної і клієнт-сервер-клієнт моделей повинне забезпечити захист даних і результатів обчислень на кінцевих вузлах;

4) на відміну від останніх, перша модель не потребує значних витрат на апаратне і програмне забезпечення та може бути реалізована у будь-якій локальній обчислювальній мережі.

Для реалізації перетворень даних, які покладені в основу запропонованих моделей виконано розробку та реалізацію наступних алгоритмів: пошук, ідентифікація та автентифікація вузлів, декомпозиція та відновлення файлів, кодування та декодування файлів, поділ даних на блоки та відновлення даних, автентифікація блоків, захист тимчасових і ключових даних, шифрування і дешифрування даних.

Ідентифікація вузлів – це одна з ключових процедур безпеки в системі. Її використання дозволяє попередити підміну вузлів зловмисниками та проведення MITM-атаки на систему. Для гарантованої ідентифікації вузлів необхідно встановити їх походження на етапі імплементації системи у мережу.

Вона передбачає наступну послідовність дій:

1) Налаштувати статичну маршрутизацію в мережі, для уникнення можливості підключення сторонніх обчислювальних пристроїв.

2) Визначити параметри апаратного забезпечення вузла, зокрема параметри материнської плати, накопичувачів, мережевого адаптера, тощо.

3) Сформувати відповідний токен для вузла.

4) Зберегти токен в базу даних.

5) Повторювати кроки 2-4 доки не будуть перевірені всі вузли мережі.

У разі використання моделі «клієнт-клієнт» алгоритм формування токену може бути змінений і бути неявним, для попередження їх підміни зловмисниками.

Автентифікація вузлів також є важливою процедурою системи безпеки. На підставі її рішення той чи інший вузол може бути включеним до переліку DEPCL -вузлів і його використанням в подальших обчисленнях. Він подібний до процедури ідентифікації та потребує виконання наступних кроків:

1) Визначити параметри апаратного забезпечення вузла, зокрема параметри материнської плати, накопичувачів, мережевого адаптера, тощо.

2) Сформувати відповідний токен для вузла.

3) Порівняти отриманий токен з переліком токенів у базі даних.

4) Додати вузол до переліку DEPCL -вузлів, у разі збігу з переліком вузлів, інакше перейти до аналізу наступного вузла.

Декомпозиція даних на блоки здійснюється лише обчислювачами в процесі перетворення даних. Вказана процедура є однією з базових у вихідній моделі обчислень та передбачає виконання наступних кроків:

1) Визначити розмір блоку з даними S B та підготувати дані до декомпозиції.

2) Завантажити з файлу S B -байт даних. У разі поділу на блоки шифрограми вказаний перейти до кроку 7.

3) Додати захисну мітку МК в блок з вихідними даними.

4) Обчислити хеш-образ блоку з вихідними даними використовуючи заздалегідь погоджену функцію хешування даних.

5) Додати в блок з вихідними даними контрольну мітку H .

6) Виконати перестановку байтів в блоці з даними використовуючи P-boxes (у разі необхідності).

7) Повторювати кроки 2-6 до завершення потоків з даними.

Процедура автоматичної ідентифікації відновлюваних даних є одним із механізмів безпеки в даній моделі. Будь-які помилки або спроби відновлення даних неавторизованим користувачем або використання для вказаних цілей неавторизованого робочого місця неминуче призведуть або до аварійного завершення обробки даних і їх втрати, або до відновлення лише відкритих даних. В якості рішення щодо відновлюваного типу даних разом з блоками

даних і ключем на вузли відправляється токен авторизації, згідно з яким виконується відновлення вихідних даних. Однак автономно вказана процедура включає наступний порядок дій:

1) Зчитати блок даних Bi з системного буферу.

2) Вилучити з Bi -блоку контрольну мітку розміром SH -байт.

3) Обчислити хеш-образ Bi -блоку використовуючи за допомогою визначеної функції хешування даних.

4) Порівняти значення контрольної мітки (з кроку 2) та отриманого хеш-образу. У разі їх подібності перейти до кроку 1, інакше припинити обробку даних.

Також під час реалізації системи фрагменти програмного коду були розміщені на кожному з тестових вузлів, тобто була реалізована модель «клієнт-клієнт». Незважаючи на те, що подібний підхід дозволяє використовувати систему в автономному та мережевому режимах, але застосування саме цієї моделі на практиці має ряд недоліків пов’язаних з оновленням програмного коду та оновлення бази даних авторизованих вузлів.

Вказані недоліки створюють передумови для нестабільної роботи коду в разі його зміни на деяких вузлах, а також можливості втрати обчислювальних потужностей системи за рахунок зменшення кількості доступних авторизованих вузлів-обчислювачів.

Також подібні проблеми можуть виникати за умови використання моделей «клієнт-сервер» (у режимі 2) та клієнт-сервер-клієнт.

Окрім вищевказаного недоліку, розміщення програмного коду на обчислювальних вузлах створює передумови для його витоку на стороні клієнта. Внаслідок цього зловмисник матиме можливість виконати його аналіз і адаптувати методи крипто аналізу для проведення атаки на вихідні алгоритми шифрування. Таким чином, при застосуванні примусу до користувача, останній не матиме можливості надати зловмиснику відкриті дані, замість закритих, оскільки перший знатиме про особливості захисту вказаної системи.

З метою попередження настання вищевказаних негативних наслідків, при розміщенні програмного коду на обчислювальних вузлах необхідно використовувати методи його захисту. Однак вказаних методів небагато [102]:

1) використання електронних ключів активації програмного забезпечення;

2) активація програмного забезпечення через мережу Інтернет;

3) використання засобів обфускації програмного коду;

4) алгоритми гомоморфного шифрування [46].

Серед вищевказаних засобів протидії витоку програмного коду найбільш ефективним, в контексті даної системи, є останній. Його надійність ґрунтується на заплутуванні вихідного програмного коду з використанням деяких алгоритмів перетворення. Подібний підхід не дозволяє забезпечити повний захист коду, лише ускладнює його розуміння, можливість модифікації і відтворення зловмисником.

Під час проведення нижче вказаних експериментів використовувалася технологія сокетів для обміну даними між вузлами. Для її стабільного використання під час створення каналу передачі даних їй необхідно передати наступні параметри: IP адресу та порт адресанта, а також протокол за допомогою якого дані будуть передачі по мережі.

В кожній операційній системі за замовчування функціонує штатне програмне забезпечення, яке відповідає за захист робочої станції від несанкціонованого доступу з мережі – мережевий екран. Він має стандартні налаштування, які дозволяють попередити проникнення на вузол і використання більшості типів шкідливого програмного забезпечення. Також для забезпечення стабільної роботи програмного забезпечення, яке використовується користувачем і потребує доступу до мережі, мережевий екран створює окремі налаштування (правила). Однак вищевказана система не є комерційним рішення і не має механізмів взаємодії з мережевим екраном. В зв’язку з цим після розміщення програмного коду на тестових вузлах необхідно виконати налаштування мережевого екрану, зокрема:

1) перевірити перелік відкритих портів, для отримання інформації які з портів можуть бути використані системою;

2) виконати відповідні налаштування мережевого екрану.

3) Однак налаштування мережевого екрану відрізняються залежно від операційної системи користувача. В зв’язку з тим, що на тестових клієнтах використовується ОС Windows 7/10, то для налаштування їх мережевих екранів використовуються наступні кроки:

4) Виконати запуск системної консолі «CMD.EXE» з правами адміністратора системи.

5) Отримати інформацію щодо поточних налаштувань мережевого екрану за допомогою команди «netsh advfirewall firewall show rule name=all».

6) Обрати вільний порт «PORT» для обміну даними та створити відповідні правила для мережевого екрану за допомогою команд: «netsh advfirewall firewall add rule name="DDS_IN" protocol=TCP localport=«PORT» action=allow dir=IN» та «netsh advfirewall firewall add rule name="DDS_ OUT" protocol=TCP localport=«PORT» action=allow dir= OUT».

Також необхідно виконати відповідні налаштування доступності мережевих вузлів для того, щоб вони мали можливість отримати доступ один до одного. Для цього необхідно виконати наступні кроки:

1) Перейти в панель управління операційної системи: меню «Пуск» > кнопка «Панель управління».

2) Вибрати пункт «Центр управління мережею та спільним доступом».

3) Вибрати пункт «Додаткові налаштування спільного доступу».

4) Для профілю «Гостьової і загальнодоступної мережі» ввімкнути правило «Дозволити виявлення в мережі».

5) Після виконання вказаних налаштувань програмний код повинен без проблем працювати на мережевих вузлах.

Для автоматичного використання мережевих вузлів в обчисленнях без участі користувача необхідно додати скрипт програмного модулю «Агент» до переліку програмного забезпечення, запуск якого здійснюється при

завантаженні операційної системи. Вказане завдання потребує виконання наступних кроків:

1) Створити виконуваний файл BAT-формату, який містить 3 команд: @echo off, start agent.py, exit.

2) Зберегти вказаний файл в директорію з програмним кодом.

3) Відкрити «Планувальник завдань» та створити завдання щодо запуску вищевказаного файлу. При цьому необхідно вказати опцію «Run with highest privileges».

4) На вкладці «Action» обрати необхідний файл.

5) Вказати планувальнику необхідність запуску даного файлу, при кожному запуску системи.

6) Зберегти налаштування.

Вищевказані рішення не змінюють базовий алгоритм. Теоретично, їх використання повинне збільшити швидкість роботи вихідної моделі пропорційно кількості задіяних в обчисленнях вузлів. З метою перевірки цієї гіпотези були проведені відповідні обчислювальні експерименти.

Технічним обмеженням вказаного експерименту є наявні обчислювальні ресурси. В ролі головного вузла використано реальне робоче місце, а в ролі обчислювальних вузлів 2 віртуальні машини. Таким чином, експерименти були проведені лише з моделлю «клієнт-клієнт».

При проведенні експериментів з використанням запропонованої моделі були використані наступні вихідні дані:

1) розмір вхідних даних 1КБ-1,5ГБ;

2) формати тестових файлів: EXE, PDF, XML, DOCX;

3) розмір ключа шифрування: 1024 (з точки зору продуктивності) та 8192 (з точки зору безпеки) біти;

4) розмір хешу даних: 1 ÷ 2 байти;

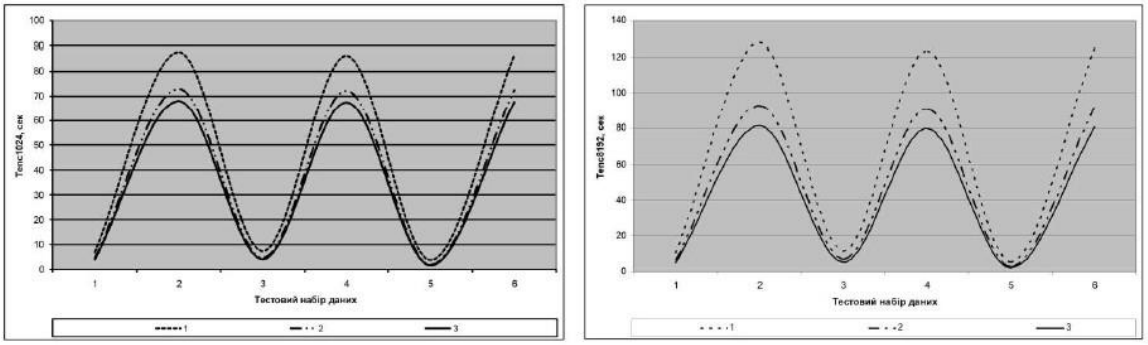
5) формат захисної мітки довільного змісту (наприклад, «!block!») та розміру від 3 байт.

6) методи кодування даних: deflate, lzma.

В результаті обчислювальних експериментів були отримані дані щодо швидкості роботи вихідних алгоритмів шифрування в наслідок поділу завдань з обробки даних між групою тестових вузлів (табл. 5.1-5.3 та рис. 5.9-5.11):

Результати обчислень шифрограми

Експ-т	ТК	TE_{1024}			TE_{8192}		
		1	2	3	1	2	3
1	2,32	6,86	4,59	3,83	10,14	6,23	4,93
2	58,15	87,68	72,92	67,99	127,68	92,92	81,33
3	2,32	7,38	4,85	4,00	11,56	6,94	5,39
4	58,15	86,15	72,15	67,49	123, 5	90,82	79,93
5	0,42	3,75	2,09	1,53	5,28	2,85	2,04
6	58,15	86,67	72,41	67,66	124,91	91,53	80,41



а б

Рисунок 5.9 – Візуальний аналіз результатів

В основі запропонованого рішення лежить поділ черги завдань щодо перетворення даних між групою тестових вузлів. Його застосування теоретично повинне прискорити виконання процедур перетворення згідно з виразом Кп PCL . Вказана умова є показником ефективності використання вузлів. Разом з тим аналіз результатів, зокрема часу виконання процедури шифрування даних, демонструє невиконання вищевказаної умови.

Максимальний рівень прискорень обчислень досягає лише 50% від теоретичного. Вказане свідчить про низьку ефективність від застосування запропонованого в даному розділі підходу для вирішення поставленого завдання.

Аналіз причин виникнення даної ситуації продемонстрував, що в основі процедури шифрування даних лежить використання незначної кількості простих математичних операцій. Використання потужностей тестового стенду дозволяє їх виконання одному потоку за найменший час. Таким чином

утворюється ситуація, яка є подібною до ситуації у розділі 3, коли здійснювалася спроба прискорити виконання паралельних обчислень за рахунок попередньої обробки даних.

Інша ситуація спостерігається, при дешифруванні даних. Зокрема затримки часу пов’язані з використанням малоефективних обчислювальних алгоритмів уповільнює обчислювальний процес, зменшує конкуренцію завдань за обчислювальні ресурси та вирівнює чергу поділяючи її завдання рівномірно між усіма вузлами.

Також аналіз графіків (рис. 5.9-5.11) продемонстрував залежність рівня прискорення від розміру черги завдань, яка формується при обробці даних в системі. Вказане пояснюється тим, що для попередньої обробки даних і їх подальшої відправки на обчислювальні вузли, витрачається додатковий системний час. Таким чином, при 0 S F , час на їх обробку та передачу по мережі зростає T f та може перевищувати час необхідний для перетворення даних f TE TPD TSDT, , .

Незважаючи на виявлені недоліки, вищевказані показники демонструють кращі показники продуктивності у порівнянні з попередніми моделями. Однак порівнюючи результати експериментів з отриманими у попередніх розділах можна зробити висновок, що найкращі показники продуктивності досягаються лише в разі використання паралельних обчислень без додаткової обробки даних. Таким чином, використання попередньої обробки даних є найбільш ефективним для використання в процедурах дешифрування даних, як най вимогливіших до обчислювальних ресурсів.

Подібно до попередніх розділів, з метою перевірки якості отриманих результатів виконано їх порівняння з результатами досліджень, які були проведені іншими дослідниками даної галузі. Результати порівняння та їх оцінка приведені в табл. 5.4 (рис. 5.12).

Вищевказані оцінки продуктивності виконання процедур перетворення даних свідчать про отримання перспективних показників швидкодії вихідних алгоритмів заперечуваного шифрування не лише для подальших досліджень, а

також для вирішення практичних завдань в галузі інформаційної безпеки. Разом з тим запропоновані рішення використовують майже всі відомі способи програмного прискорення обчислень, на відміну від подібних рішень інших авторів. Однак порівнюючи дані вихідної моделі обчислень з даними інших авторів можна дійти висновку, що при використанні близького по конфігураціям стендового обладнання, запропоноване рішення є більш продуктивним в залежності від розміру ключа шифрування.

ВИСНОВКИ

Основним завданням магістерської роботи є вирішення проблем з практичним застосування перспективних алгоритмів захисту даних, в основі яких лежить використання механізмів заперечуваного шифрування даних. Для цього виконано інформаційний огляд існуючих алгоритмів заперечуваного шифрування в контексті програмних засобів захисту, виявлено проблеми та підходи для їх можливого вирішення. В результаті проведеного дослідження отримано наступні наукові та практичні результати:

1. Розроблено модель, яка дозволяє реалізувати операції заперечуваного шифрування даних на основі існуючих елементів симетричної криптографії.

2. Удосконалено блокову модель обчислень у частині кодування інформації, що дозволило скоротити кількість блоків з вихідними даними та відповідний час на перетворення окремих типів даних до $\sqrt{K}$-раз.

3. Удосконалено блокову модель заперечуваного шифрування у частині використання в паралельних комп’ютерних системах зі спільною пам’яттю, що призвело до збільшення швидкості перетворення даних до 8 раз.

4. Удосконалено блокову модель перетворення даних у частині використання в паралельних комп’ютерних системах з роздільною пам’яттю (обчислювальних кластерах) та статичним балансуванням навантаження, що дозволило збільшити швидкість обробки даних великого обсягу до 1,5рази/вузол.

5. Створено прототипи програмних модулів, які реалізують перетворення закладені в запропоновані моделі обчислень.

Отримані результати демонструють перспективність запропонованого рішення та можливість його застосування для вирішення подібних задач в інших напрямах і галузях знань.

Створена інструментальна система спрямована на те, щоб дати дослідникові можливість вирішувати проблем з продуктивністю обчислень без необхідності внесення змін у вихідні обчислювальні алгоритми.

Програмні модулі розроблені на мові програмування Python досить просто розуміти, удосконалювати та використовувати, навіть дослідникам з низькими навичками програмування.

Таким чином, мету роботи, яка полягає в розробці способів ефективної організації обчислень, які лежать в основі існуючих алгоритмів заперечуваного шифрування даних, досягнуто в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Кузнецов О. О., Євсеєв С. П., Король О. Г. Стеганографія: навч. посіб. Харків: ХНЕУ, 2011. 232 с.
2. Навроцький Д. О. Методи комп'ютерної стенографії. Вісник Національного технічного університету України "КПІ". 2007. № 35. С. 105-108.
3. Стасюк О. І., Гнатюк С. О., Довгич Н. І., Літош М. С. Сучасні стеганографічні методи захисту інформації. НАУ "Захист інформації". 2011. № 1. С. 1-7.
4. Advanced encryption standart (AES). FIPS 197. [Чинний від 2001-11-26]. Гейтерсберг: NIST, 2001. 51 с.
5. AES (Advanced Encryption Standard). URL: https://ru.bmstu.wiki/AES_ (Advanced_Encryption_Standard).
6. Alvila M. A performance evaluation of post-quantum cryptography in the signal protocol: Master's thesis: LITH-ISY-EX--19/5211--SE / Linköping, 2019. 50 p.
7. Anderson R., Biham E., Knudsen L. (2000) Serpent and Smartcards. Proceedings of the CARDIS'98: 3th International Conference on Smart Card Research and Applications in Belgium, (Louvain-la-Neuve, September 14-16, 2000), pp. 246-253.
8. Barakat T. M. A New Sender-Side Public-Key Deniable Encryption Scheme with Fast Decryption. KSII Transactions on Internet and information systems. 2014. Vol. 8, No. 9. pp. 3231-3249.
9. Canetti R., Dwork C., Naor M., Ostrovsky R. (1997) Deniable Encryption: Lecture Notes in Computer Science. Advances in Cryptology – CRYPTO '97, (Santa Barbara, August 17-21, 1997), PP. 90-104.
10. Caro A. D., Iovino V., O'Neill A. (2016) Deniable Functional Encryption. Public-Key Cryptography. PKC 2016 Proceedings in Taipei, (Taipei, March 6-9, 2016), pp. 196-222.
11. Chen B. Deniable encryption storage for mobile devices. TechTalks. 2017. Vol. 49. PP. 1-4.
12. Chou T., Orlandi C. (2015) The Simplest Protocol for Oblivious Transfer.

Proceedings of LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, (Guadalajara, August 23-26, 2015), pp. 40-58.

13. Diffie W., Hellman M. E. New Directions in Cryptography. IEEE Transactions on Information Theory. 1976, V. 22, No. 6. pp. 44-54.
14. Durgesh P., Rajawat A. S. Efficient Throttled Load Balancing Algorithm in Cloud Environment. International Journal of Modern Trends in Engineering and Research. 2015. Vol. 2 (3). PP. 463-480.
15. ElGamal T. On Computing Logarithms Over Finite Fields: Lecture Notes in Computer Science. Proceedings of CRYPTO'85 – Advances in Cryptology in Santa Barbara, (Santa Barbara, August 18-22, 1985), pp. 396-402.
16. FreeOTFE. URL: https://wikiprograms.org/freeotfe.
17. GIF. URL: https://uk.wikipedia.org/wiki/GIF.
18. Goldwasser S., Micali C. Probabilistic encryption. Journal of Computer and System Sciences. 1984. Vol. 28. PP. 277-299.
19. Hong X., Wang B. A Non-interactive Deniable Authentication Scheme in the Standard Model. Journal of Electrical and Electronic Engineering. 2017. Vol. 5, No. 2. pp. 80-85.
20. Howlader J., Basu S. Sender-Side Public Key Deniable Encryption Scheme. ARTCom 2009: International Conference on Advances in Recent Technologies in Communication and Computing in Kottayam, (Kerala, October 27- 28, 2009), PP. 9-13.
21. Howlader J., Deepa N. Sender-Side Public Key Deniable Encryption Scheme. International Journal of Recent Trends in Engineering. 2009. Vol. 1, No. 1. PP. 668-671.
22. Ibrahim H. A Method for Obtaining Deniable Public-Key Encryption. International Journal of Network Security. 2009. Vol. 8, No.1. pp. 1–9.
23. Ibrahim H. Receiver-Deniable Public-Key Encryption. International Journal of Network Security. 2009. Vol. 8, No. 2. pp. 159-165.
24. Kedarnath J. B., Nur A. T. Relationship Between Entropy and Test Data Compression. IEEE Transactions on computer-aided design of integrated circuits and systems. 2007. Vol. 26, No. 2. PP. 386-395.

25. Klonowski M., Kubiak P. and Kutyłowsk M. (2008) Practical Deniable Encryption. SOFSEM 2008 Proceedings: 34th Conference on Current Trends in Theory and Practice of Computer Science in Nový Smokovec, (Nový Smokovec January 19-25, 2008), pp. 599-609.
26. Krishnamurthy G. N., Ramaswamy Dr. V. Encryption Quality Analysis and Security Evaluation of CAST-128 Algorithm and its Modified Version using Digital Images. International Journal of Network Security & Its Applications (IJNSA). 2009. Vol. 1, No. 1. P. 28-33.
27. LZMA. URL: https://ru.wikipedia.org/wiki/LZMA.
28. Menezes A. J., Oorschot P. V., Vanstone S. A. Handbook of Applied Cryptography: CRC Press, 1996. 816 p.
29. Meng B., Wang JQ. A Receiver Deniable Encryption Scheme. ISIP'09 Processing: International Symposium on Information in Huangshan (Huangshan, August 21-23, 2009), pp. 254-257.
30. Moldovyan N. A., Moldovyan A. A., Moldovyan D. N., Shcherbacov V. A. Stream Deniable-Encryption Algorithms. Computer Science Journal of Moldova. 2016. Vol. 24, No. 1. PP. 68-82.
31. Moldovyan N. A., Moldovyan A. A., Shcherbacov V. A. Generating Cubic Equations as a Method for Public Encryption. Buletinul academiei de stiinte. 2015. Vol. 3, No. 79. PP. 60-71.
32. Moldovyan N. A., Moldovyan A. A., Shcherbacov V. A. Post-quantum Nokey Protocol. Buletinul academiei de stiinte. 2017. Vol. 3, No. 85. pp. 115-119.
33. Moldovyan N. A., Moldovyan A. A., Shcherbacov V. A. Provably senderdeniable encryption scheme. Computer Science Journal of Moldova. 2015. Vol. 23, No. 1 (67). pp. 62-71.
34. Moldovyan N. A., Shcherbacov A. V., Eremeev A. E. Deniable-encryption protocols based on commutative ciphers. Quasigroups and Related Systems. 2017. Vol. 25. pp. 95-108.

35. Moriai S., Yiqun L. Y. Cryptanalysis of Twofish (II). URL: https://www.schneier.com/wp-content/uploads/2015/01/twofish-analysis-shiho.pdf
36. Prohibiting RC4 Cipher Suites. URL: https://tools.ietf.org/html/rfc7465.
37. Rabin M. O. Digital Signatures and Public-Key Functions as Intractable as Factorization: Technical Report. Cambridge: MIT Laboratory for Computer Science, 1979. 212 p.
38. Rives R. L., Shamir A., Adleman L. M. A Method tor Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM. 1978. V. 21, No. 2. PP. 120-126.
39. Rjazhkova Z. Electronic Voting Schemes: Master thesis / Physics and Informatics Comenius University. Bratislava, 2002. 64 p.
40. Rubberhose filesystem. URL: https://web.archive.org/web/20110628084231/http://iq.org/~proff/rubberhose.org (Access: 01.08.2020).
41. Salomon D. Bryant D., Motta G. Handbook of Data Compression. London: Springer, 2010. 1361 p.
42. Skillen A. Deniable Storage Encryption for Mobile Devices: A Thesis for the Degree of Master of Applied Science in Information Systems Security. Concordia University. Montr´eal, 2013. 135 p.
43. Skillen A., Mannan M. On Implementing Deniable Storage Encryption for Mobile Devices. NDSS Symposium, (San Diego, February 24-27, 2013), PP. 1-17.
44. Stallings W., Brown L. Computer security: principles and practice. Third edition. New South Wales, 2003. 838 p.
45. The RSA Algorithm. URL: https://www.researchgate.net/publication/338623532_The_RSA_Algorithm/link/5e204028299bf1e1fab53e7f/download.
46. Triple DES. URL: http://kriptografea.narod.ru/TDES.html.
47. Triple DES. URL: https://ru.wikipedia.org/wiki/Triple_DES.

48. Vanhoef M., Piessens F. (2015) All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS. 24th USENIX Security Symposium in USA (Washington, August 12–14, 2015), pp. 97-112.
49. Williams H. C. A Modification of the RSA Public-Key Encryption Procedure. IEEE Transactions on Information Theory. 1980. V. 26, No. 6. pp. 726-729.