Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Система ефективної перевірка цілісності реплікованих даних у хмарі.

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Якимович О.С.
(підпис) (прізвище та ініціали)

Керівник _____ Долінський А.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_8_» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Якимовичу Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система ефективної перевірка цілісності реплікованих даних у хмарі

Керівник роботи Долінський Андрій Михайлович, асистент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 20*26* року № *4/9-239*

2. Термін подання студентом завершеної роботи 22 червня 2026 р.

3. Вихідні дані до роботи журнали синхронізації, час останньої реплікації, кількість копій, статуси успішності

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.
1. Аналіз предметної області та постановка завдання.
2. Перевірка цілісності реплікованих даних у хмарі
3. Ефективна перевірка цілісності реплікованих даних у хмарі з використанням гомоморфного шифрування
4. Безпека життєдіяльності, основи охорони праці
Висновки. Перелік джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	кандидат технічних наук, доцент кафедри МТ Гурик Олег Ярославович		

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	
3.	Виконання дослідження щодо перевірки цілісності	17.02.2026-10.05.2026	
	реплікованих даних у хмарі з використанням		
	гомоморфного шифрування		
4.	Оформлення розділу «Аналіз предметної	11.05.2026-17.05.2026	
	області та постановка завдання»		
5.	Оформлення розділу «Перевірка цілісності	18.05.2026-24.05.2026	
	реплікованих даних у хмарі»		
6.	Оформлення розділу «Ефективна перевірка цілісності	25.05.2026-31.05.2026	
	реплікованих даних у хмарі		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-08.06.2026	
8.	Виконання завдання до підрозділу «Основи охорони	01.06.2026-08.06.2026	
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2026-11.06.2026	
10.	Нормоконтроль	12.06.2026-15.06.2026	
11.	Перевірка на плагіат	16.06.2026	
12.	Попередній захист кваліфікаційної роботи	18.06.2026	
13.	Захист кваліфікаційної роботи	24.06.2026	

Студент _____ Якимович О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Долінський А.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Система ефективної перевірка цілісності реплікованих даних у хмарі // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Якимович Олександр Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2026 // С. 69, рис. – 14, табл. – 1, кресл. – 12, додат. – 0, бібліогр. – 29.

Ключові слова: реплікація, хмарні технології, цілісність даних, алгоритм, безпека.

Було представлено схему перевірки цілісності реплікованих даних у хмарному середовищі. Схема періодично перевіряє правильність та повноту кількох копій даних, що зберігаються в хмарі. Наша схема враховує динамічні операції оновлення даних на копіях даних у процесі перевірки. Усі копії даних можна розшифрувати за допомогою одного ключа розшифрування, що забезпечує безперешкодний доступ для всіх авторизованих користувачів даних. Експериментальні результати з використанням локальних та EC2 хмарних екземплярів показують, що ця схема краща за попередню запропоновану схему з точки зору динамічних операцій з даними, які виконуються набагато швидше. Крім того, ми показали, що наша схема добре працює, коли її розширюють для підтримки кількох версій файлів, де в хмарі зберігаються лише дельти, що економить витрати власника даних на зберігання. Ми вважаємо, що ці результати допоможуть власнику даних домовитися з постачальником хмарних послуг про вартість та гарантії продуктивності, зберігаючи при цьому цілісність даних. Крім того, ці результати нададуть хмарі різні стимули для вжиття відповідних заходів, таких як паралельне виконання обчислень, для забезпечення хорошої продуктивності.

ANNOTATION

Efficient Replicated Data Integrity Verification System in the Cloud // Qualification work of the educational level «Bachelor» // Yakymovych Oleksandr // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2026 // P. 69, fig. – 14, tabl. – 1, chair. – 12, annexes. – 0, references – 29.

Keywords: replication, cloud technologies, data integrity, algorithm, security.

A scheme for verifying the integrity of replicated data in a cloud environment was presented. The scheme periodically verifies the correctness and completeness of multiple copies of data stored in the cloud. Our scheme takes into account dynamic data update operations on the data copies in the verification process. All data copies can be decrypted with a single decryption key, which provides seamless access for all authorized data users. Experimental results using local and EC2 cloud instances show that this scheme is better than the previously proposed scheme in terms of dynamic data operations, which are performed much faster. Furthermore, we have shown that our scheme performs well when extended to support multiple file versions, where only deltas are stored in the cloud, which saves the data owner's storage costs. We believe that these results will help the data owner negotiate with the cloud service provider on cost and performance guarantees while maintaining data integrity. Furthermore, these results will provide various incentives for the cloud to take appropriate measures, such as parallel execution of computations, to ensure good performance.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSP (англ. Cloud Service Providers) – Постачальники хмарних послуг.

DMR-PDP (англ. Dynamic Multi-Replica Provable Data Possession scheme) – Схема динамічного володіння даними з кількома репліками та доказами.

MHT (англ. Merkle Hash Trees,) – Хеш-дерево Меркла.

PDP (англ. Provable Data Possession) – Довідкове володіння даними.

POR (англ. Proof of Retrievability) – Доказ можливості відновлення.

PRF (англ. Pseudo-Random Function) – Псевдовипадкова функція.

SLA (англ. Service Level Agreement) – Угода про рівень обслуговування.

TPA (англ. Third Party Auditor) – Аудитор третьої сторони.

ЗМІСТ

ВСТУП

Актуальність теми. Хмарні обчислення – це нова модель, в якій ресурси обчислювальної інфраструктури надаються як послуга через Інтернет. Власники даних можуть передавати свої дані на аутсорсинг, віддалено зберігаючи їх у хмарі, та користуватися високоякісними послугами на вимогу зі спільного пулу налаштовуваних обчислювальних ресурсів. Використовуючи ці служби зберігання даних, власники даних можуть зняти тягар локального зберігання та обслуговування даних. Однак, оскільки власники даних та хмарні сервери не знаходяться в одному довіреному домені, дані, передані на аутсорсинг, можуть бути під загрозою, оскільки хмарному серверу може бути втрачено повну довіру. Тому цілісність даних має критичне значення в такому сценарії. Хмара повинна дозволяти власникам або довіреній третій стороні перевіряти цілісність свого сховища даних, не вимагаючи локальної копії даних. Власники часто реплікують свої дані на хмарних серверах у кількох центрах обробки даних, щоб забезпечити вищий рівень масштабованості, доступності та довговічності. Коли власники даних просять постачальника хмарних послуг (CSP) реплікувати дані, CSP стягує з них вищу плату за зберігання. Отже, власники даних повинні бути твердо переконані, що CSP зберігає копії даних, узгоджені в контракті на рівень обслуговування, а оновлення даних були правильно виконані на всіх віддалено збережених копіях.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробити схему динамічного багатореплікаційного доказового володіння даними (DMR-PDP), яка запобігає шахрайству з боку CSP, наприклад, шляхом збереження меншої кількості копій, ніж оплачено, та/або підробці даних.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати стан досліджень в галузі перевірки цілісності даних;

– Розробити схему доказового володіння даними для збереження конфіденційності;

– Розробити схему динамічного володіння доказовими даними.

Практичне значення одержаних результатів.

Було розширено схему для підтримки базової системи керування версіями файлів, де поширюється лише різниця між оригінальним файлом та оновленим файлом, а не поширення операцій з міркувань конфіденційності. DMR-PDP також підтримує ефективні динамічні операції, такі як модифікація блоків, вставка та видалення на репліках на хмарних серверах.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ.

1.1 Реплікація даних та проблеми, які при цьому виникають

Хмарні обчислення – це інноваційна та приваблива технологія, за допомогою якої високо масштабовані та технологічно підкріплені сервіси можна легко отримувати через Інтернет. Переваги хмарних обчислень включають самообслуговування на вимогу, повсюдний доступ до мережі, незалежне від місцезнаходження об'єднання ресурсів, ціноутворення на основі використання тощо. Висока гнучкість та економічна економія мотивують як окремих осіб, так і підприємства, наприклад, Amazon, Google, Microsoft, Yahoo та Salesforce, передавати свої дані в хмару. Але побоювання щодо доступності та безпеки даних заважають власникам даних та компаніям скористатися перевагами хмари. Коли користувачі зберігають дані в хмарі, їх головним завданням є те, чи зможе хмара підтримувати цілісність даних і чи можна їх відновити у разі втрати даних або збою сервера. Постачальники хмарних послуг (CSP), щоб заощадити кошти на зберіганні, можуть схильні викидати деякі дані або копії даних, до яких не часто звертаються, або переносити такі дані на пристрої зберігання другого рівня. CSP також можуть приховувати втрату даних через помилки управління, збої обладнання або атаки. Тому критичним питанням зберігання даних у ненадійних CSP є періодична перевірка того, чи підтримують сервери зберігання цілісність даних та чи зберігають дані повністю та правильно, як зазначено в Угоді про рівень обслуговування (SLA).

Реплікація даних – це поширений метод збільшення доступності даних можливості хмарних обчислень. Хмара реплікує дані та стратегічно зберігає їх на кількох серверах, розташованих у різних географічних місцях. Оскільки репліковані дані є копіями, важко перевірити, чи дійсно хмара зберігає кілька копій даних. Хмара може легко обдурити власника, зберігаючи лише одну

копію даних. Таким чином, власник хотів би регулярно перевіряти, чи дійсно хмара має кілька копій даних, як зазначено в угоді про рівень обслуговування.

Загалом, хмара має можливість генерувати кілька реплік, коли власник даних вимагає від CSP довести, що він володіє кількома копіями даних. Також є обґрунтованим припущення, що власник даних може не мати копії даних, що зберігається локально. Отже, головним завданням власника є не лише перевірка цілісності даних, але й їх відновлення, якщо виявлено будь-які видалення/пошкодження даних. Якщо власник під час перевірки виявить втрату даних у будь-якій з реплік у хмарі, він може відновити дані з інших реплік, які зберігаються цілими. Оскільки репліки повинні зберігатися в різних географічних місцях, вважається безпечним, що втрата даних не відбудеться у всіх репліках одночасно.

Власники даних передають свої дані на аутсорсинг у хмару та можуть фізично не мати локальної копії своїх даних. Власники даних повинні мати можливість перевірити цілісність та безпеку даних, що зберігаються у CSP. Хмара повинна дозволяти користувачам або довіреній третій стороні проводити аудит хмарного сховища даних, не вимагаючи локальної копії даних. Ці фактори необхідно враховувати під час розробки схеми перевірки цілісності даних.

- Складність комунікації. Обсяг комунікації між клієнтом і сервером має бути мінімальним.
- Вартість зберігання. Додаткове сховище для клієнта та сервера, необхідне для схеми має бути мінімальним, окрім вихідних даних.
- Відновлення даних. Схема має допомогти у відновленні даних у разі їх втрати.
- Доведена безпека. Схема має бути безпечною.

1.2 Базовий підхід при реалізації перевірки цілісності даних

Один з базових підходів до перевірки цілісності даних полягає в хешуванні всього файлу перед його зберіганням у хмарі. Хеш обчислюється власником даних і зберігається локально. Коли клієнт хоче перевірити, він завантажує весь файл, обчислює хеш і перевіряє, чи збігаються значення. Але ця схема має недоліки, пов'язані з завантаженням і хешуванням усього файлу кожного разу. Ця схема ефективна, якщо розмір файлу менший, і стає накладною для файлів величезного розміру.

У літературі широко обговорюються два типи схем, які пропонують краще рішення, ніж базовий підхід. Це схеми доказового володіння даними (PDP) [1] та схеми доказу можливості відновлення (POR) [4]. Схеми PDP перевіряють лише те, чи дані, що зберігаються в CSP, є цілісними, тоді як схеми POR допомагають власнику даних відновити їх у разі їх втрати.

1.2.1 Гомоморфні перевірювані теги

Атнієзе та ін. [1] запропонували цю схему, за якою власник даних попередньо обробляє файл перед його зберіганням у хмарі. Вони вводять концепцію гомоморфних перевірених тегів (HVT). Генерація тегів базується на схемі підпису RSA. Розглянемо файл F як скінченну впорядковану колекцію з n блоків, $F = (m_1, m_2,.., m_n)$. Отримавши повідомлення m, клієнт обчислює тег T_m. Ці теги будуть зберігатися у хмарі разом із файлом F. Ці гомоморфні перевірені теги діють як метадані перевірки для блоків файлів. Завдяки гомоморфній властивості, теги, обчислені для кількох блоків файлів, можна об'єднати в одне значення. Маючи два значення Tm_i та Tm_j, будь-хто може об'єднати їх у значення Tm_i+m_j, що відповідає сумі повідомлень $m_i + m_j$. Пізніше клієнт може перевірити, чи файл належить серверу, генеруючи випадковий запит до випадково вибраного набору блоків файлів. Використовуючи запитувані блоки

та відповідні їм теги, сервер генерує доказ володіння. Таким чином, клієнт переконаний у володінні даними, без необхідності фактично отримувати блоки файлів. Недоліки цієї схеми полягають у тому, що вона не враховує шифрування даних та реплікацію даних. Ця схема працює лише для статичних файлів.

1.2.2 Схема PDP для збереження конфіденційності

Шах та ін. [11] запропонували протоколи PDP, що зберігають конфіденційність. Використовуючи цю схему, зовнішній сторонній аудитор (TPA) може перевірити цілісність файлів, що зберігаються на віддаленому сервері, не знаючи жодного вмісту файлу. Власник даних спочатку шифрує файл, а потім надсилає зашифрований файл разом із ключем шифрування на віддалений сервер. Крім того, власник даних надсилає зашифрований файл разом із зобов'язанням щодо ключа, яке фіксує значення для ключа, не розкриваючи ключ TPA. Основна мета цієї схеми полягає в тому, щоб гарантувати, що віддалений сервер правильно володіє даними клієнтів разом із ключем шифрування, та запобігти будь-якому витоку інформації до TPA, який відповідає за завдання аудиту. Оскільки ключ має зберігатися в таємниці як від аудитора, так і від хмари, власник даних розміщує зобов'язання щодо ключа gk замість зберігання k на сервері. Аудитор генерує n випадкових чисел та генерує значення MAC для випадкових блоків і зберігає їх. Періодично аудитор надсилає запит серверу, генеруючи випадковий номер блоку, для якого сервер генерує MAC для цього блоку та надсилає його назад аудитору для перевірки. Ця схема має багато недоліків. Кількість разів, коли певний елемент даних може бути перевірений, обмежена та має бути встановлена заздалегідь. TPA повинен перегенерувати новий список хеш-значень, щоб досягти необмеженої кількості аудитів. Відсутність підтримки перевірки без збереження стану, тобто TPA повинен оновлювати свій стан між аудитами, щоб запобігти подвійному

використанню одного й того ж випадкового числа або одного й того ж MAC-адреси.

1.2.3 Схема володіння даними з кількома репліками, що підлягають доведенню

Куртмола [6] запропонував схему багатореплікаційного PDP (MR-PDP), де власник даних може перевірити, чи кілька копій файлу зберігаються постачальником послуг зберігання. Схема MR-PDP є розширенням моделей PDP, запропонованих Атнієзе. Куртмола запропонував створювати окремі репліки/копії файлу даних, спочатку зашифрувавши файл, а потім маскуючи зашифровану версію певною випадковістю, згенерованою за допомогою псевдовипадкової функції (PRF). Різні ключі PRF використовуються для створення кількох копій даних. RSA-підписи використовуються для генерації тегів. RSA-підписи мають голоморфну властивість, коли кілька блоків даних можуть бути перевірені одночасно. Основною перевагою цієї схеми є те, що вона підтверджує цілісність кількох реплік. Недоліками є те, що дані не шифруються, розмір RSA-підписів величезний, вони мають справу зі статичними даними, і авторизовані користувачі повинні знати, яка копія була спеціально отримана від CSP, щоб належним чином розмаскувати її перед розшифруванням.

1.2.4 Динамічне володіння доказовими даними

Ця схема [7] намагається вирішити певні обмеження схеми MRPDP шляхом використання автентифікованого списку пропусків на основі рангу. Ця структура даних використовується для перевірки цілісності даних. Згідно з цією схемою, файл F розбивається на n блоків m_1, m_2,..., m_n. Тег $T(m_i)$ блоку mi зберігається в і-му вузлі нижнього рівня списку пропусків. Блок mi буде зберігатися в іншому місці хмарою. Кожен вузол v списку пропусків зберігає

кількість вузлів нижнього рівня, до яких можна дістатися з v. Це значення називається рангом v і позначається як r(v). На рис. 1.1 показано список пропусків з рангами вузлів. Вставка, видалення або модифікація блоку файлу впливає лише на вузли списку пропусків вздовж шляху пошуку. Ранги вузлів, на які впливає зміна, обчислюються знизу вгору за постійний час.

Верхній лівий вузол списку пропусків називатиметься початковим вузлом. Для вузла v індекси крайнього лівого та крайнього правого вузлів на нижньому рівні досяжний з v, позначається відповідно low(v) та high(v).

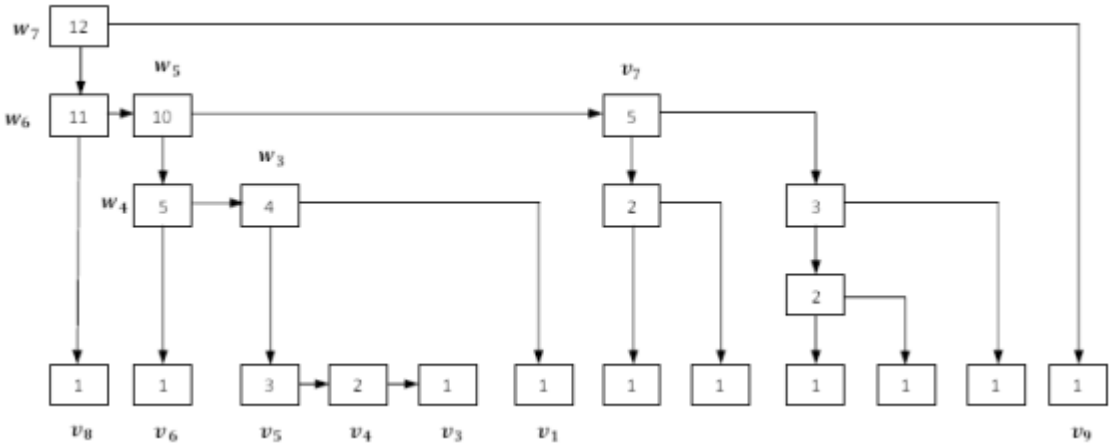


Рисунок 1.1 - Список пропусків на основі рангу

На кожному рівні, залежно від певних умов, кожен вузол також зберігатиме колективне хеш-значення вузлів зліва направо на нижньому рівні таким чином $h(x_1,..., x_k) = h(h(x_1) \parallel ... \parallel h(x_k))$. Де $\parallel$ позначають оператор конкатенації. Ці хеш-записи діють як метадані. Щоразу, коли блок файлу оновлюється або змінюється, хеш-значення вузлів у цьому списку пропусків, з якого доступний оновлений або змінений вузол, перераховується, а шлях, де оновлюються вузли, надсилається користувачеві для перевірки. Схема перевірки цілісності подібна до попередніх схем, але ця схема підтримує динаміку даних та перевіряє операції вставки, оновлення та зміни блоку даних.

1.3 Перевірка динамічних кількох копій даних на хмарних серверах

Ця схема [8] підтримує кілька реплік та шифрування даних. Блоки даних у кожній копії додаються до блоків файлу перед шифруванням. Шифрування AES. Схема використовується для шифрування даних, а підписи BLS використовуються для генерації тегів. Динамічні операції з даними підтримуються за допомогою хеш-дерев Меркла (Merkle Hash Trees, MHT).

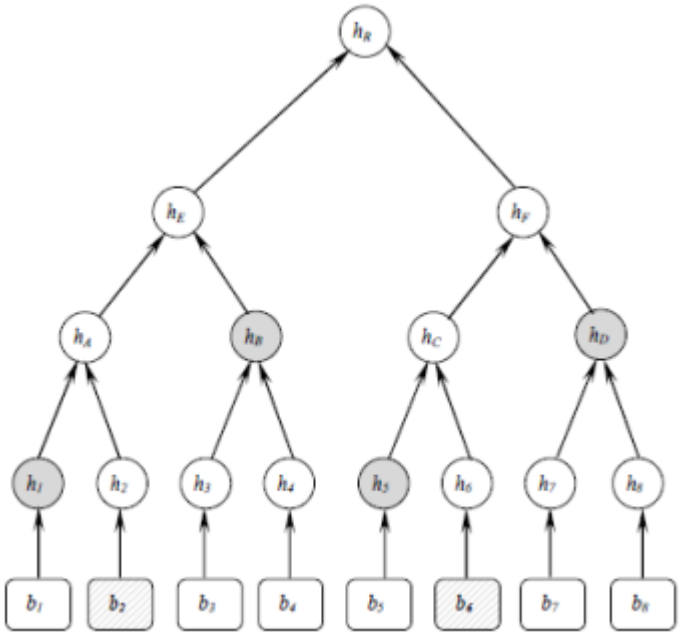


Рисунок 1.2 - Хеш-дерево Меркла

MHT — це бінарна деревоподібна структура, яка використовується для ефективної перевірки цілісності даних. MHT — це дерево хешів, де листя дерева — це хеші блоків даних. На рис. 1.2 $h_A = h(h_1 \parallel h_2), h_B = h(h_3 \parallel h_4)$ і так далі. Нарешті, $h_R = h(h_E \parallel h_F)$ – це хеш кореневого вузла, який використовується для автентифікації цілісності всіх блоків даних. Блоки даних $\{b_1, b_2,..,b_8\}$ зберігаються на віддаленому сервері, і лише автентичне значення h_R зберігається локально на стороні верифікатора. Наприклад, якщо верифікатор запитує перевірку цілісності блоків b_2 та b_6, сервер надішле ці два

блоки разом зі шляхами автентифікації $A_2 = \{h_1, h_B\}$ та $A_6 = \{h_5, h_D\}$, які використовуються для реконструкції кореня МНТ. A_j, шлях автентифікації b_j – це набір вузлів-братів (сірі кола) на шляху від h_j до кореня. У динамічній поведінці даних, що передаються на аутсорсинг, необхідно автентифікувати як значення, так і позиції блоку даних. Це гарантує, що певне значення зберігається на певному кінцевому вузлі. Наприклад, якщо власник даних вимагає вставки нового блоку після позиції j, верифікатор повинен переконатися, що сервер вставив новий блок у запитувану позицію. Для перевірки позицій блоків, кінцеві вузли МНТ обробляються в певній послідовності, наприклад, зліва направо. Отже, хеш будь-якого внутрішнього вузла дорівнює h(лівий дочірній вузол $\|$ правий дочірній вузол), наприклад, $h_A = h(h_1 \| h_2) \neq h(h_2 \| h_1)$. Крім того, шлях автентифікації A_j розглядається як упорядкована множина, і таким чином будь-який кінцевий вузол однозначно визначається шляхом дотримання використаної послідовності побудови кореня МНТ. Для всіх реплік створюється каталог МНТ.

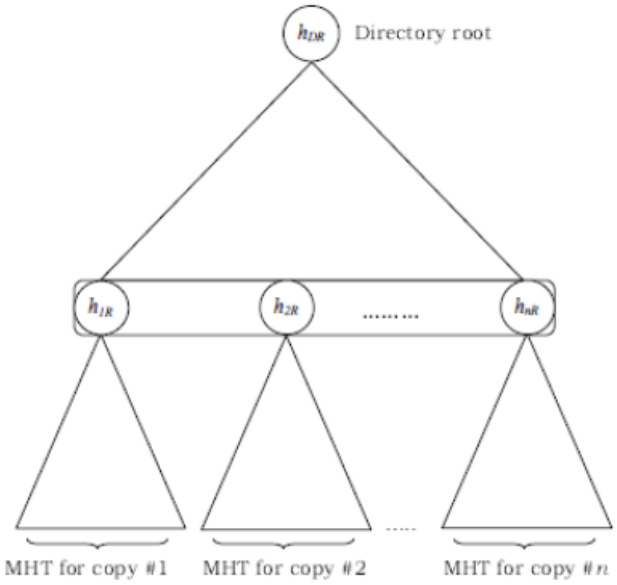


Рисунок 1.3 - Довідник МНТ

Недоліком цієї схеми є те, що кількість вузлів у МНТ залежить від кількості блоків у файлі, а також від кількості копій. Якщо файл має величезну

кількість блоків, то й кількість вузлів у MHT буде величезною. Ця схема створює обчислювальні та комунікаційні витрати для власника даних для генерації та надсилання такої величезної деревоподібної структури. Для кожної перевірки шляхи автентифікації та кілька хеш-значень також надсилаються користувачеві, що створює комунікаційні витрати для власника даних.

1.4 Висновок до першого розділу

Хмарні обчислення забезпечують масштабованість, економічність та гнучкість, але водночас створюють ризики щодо доступності та безпеки даних.

Реплікація даних є ключовим механізмом підвищення надійності та доступності, проте виникає проблема перевірки наявності кількох копій та їхньої цілісності.

Основні вимоги до схем перевірки: мінімізація комунікаційних витрат, низька вартість зберігання, можливість відновлення даних та доведена криптографічна безпека.

Базовий підхід із хешуванням файлу є простим, але неефективним для великих даних. Тому в літературі розроблено більш просунуті схеми:

PDP (доказове володіння даними) — перевірка цілісності без повного завантаження файлу.

POR (доказ відновлюваності) — забезпечує можливість відновлення даних у разі втрати.

Серед PDP-схем виділяються:

Гомоморфні перевірювані теги (Ateniese) — ефективна перевірка без отримання файлу, але лише для статичних даних.

Протоколи збереження конфіденційності (Shah) — дозволяють аудит без розкриття вмісту, але мають обмеження щодо кількості перевірок.

MR-PDP (Kurtmola) — підтверджує наявність кількох копій, але працює зі статичними даними та має великі витрати на RSA-підписи.

Динамічне PDP (Erway) — підтримує оновлення, вставку та видалення блоків завдяки структурі списку пропусків.

Перевірка динамічних копій (Barsoum, Hasan) — поєднує реплікацію, шифрування AES та BLS-підписи, використовуючи хеш-дерева Меркла для динаміки даних.

Недоліки сучасних схем полягають у високих обчислювальних та комунікаційних витратах, особливо при роботі з великими файлами та багатьма копіями.

Таким чином, актуальним завданням є розробка схеми, яка одночасно забезпечує перевірку цілісності, підтримку динаміки даних, роботу з кількома репліками та ефективність у витратах.

РОЗДІЛ 2. ПЕРЕВІРКА ЦІЛІСНОСТІ РЕПЛІКОВАНИХ ДАНИХ У ХМАРІ

2.1 Концепція схеми перевірки цілісності даних

Коли користувачі зберігають дані в хмарі, їх головним завданням є те, чи зможе хмара підтримувати цілісність даних і чи можна буде відновити дані у разі втрати даних або збою сервера. Постачальники хмарних послуг (CSP), щоб заощадити кошти на зберіганні, можуть схильні скидати деякі дані або копії даних, до яких не часто звертаються, або переносити дані на пристрої зберігання другого рівня. CSP також можуть приховувати втрату даних через помилки управління, збої обладнання або атаки. Тому критичним питанням зберігання даних у ненадійних CSP є періодична перевірка того, чи підтримують сервери зберігання цілісність даних; чи зберігають дані повністю та правильно, як зазначено в угоді про рівень обслуговування (SLA).

Реплікація даних – це поширений метод підвищення доступності даних у хмарних обчисленнях. Хмара реплікує дані та стратегічно зберігає їх на кількох серверах, розташованих у різних географічних місцях. Оскільки репліковані копії виглядають абсолютно однаково, важко перевірити, чи дійсно хмара зберігає кілька копій даних. Хмара може легко обдурити власника, зберігаючи лише одну копію даних. Таким чином, власник хотів би регулярно перевіряти, чи дійсно хмара має кілька копій даних, як заявлено в угоді про рівень обслуговування (SLA). Загалом, хмара має можливість генерувати кілька реплік, коли власник даних ставить перед CSP виклик до того, що він має кілька копій даних. Крім того, є обґрунтованим припущення, що власник даних може не мати копії даних, що зберігається локально. Отже, головним завданням власника є не лише перевірка цілісності даних, але й відновлення даних, якщо виявлено будь-які видалення/пошкодження даних. Якщо власник даних під час перевірки за допомогою схеми DMR-PDP виявить втрату даних у будь-якій з реплік у хмарі, він може відновити дані з інших реплік, що зберігаються в

недоторканому вигляді. Оскільки репліки мають зберігатися в різних географічних місцях, можна з упевненістю припустити, що втрата даних не відбудеться на всіх репліках одночасно.

Доказове володіння даними (PDP) [2] – це метод аудиту та перевірки цілісності даних, що зберігаються на віддалених серверах. У типовій моделі PDP власник даних генерує метадані/теги для файлу даних, які згодом будуть використані для перевірки цілісності. Для забезпечення безпеки власник даних шифрує файл і генерує теги на зашифрованому файлі. Власник даних надсилає зашифрований файл і теги до хмари та видаляє локальну копію файлу. Коли власник даних бажає перевірити цілісність даних, він генерує вектор виклику та надсилає його до хмари. Хмара відповідає, обчислюючи відповідь на дані та надсилаючи її верифікатору/власнику даних, щоб довести, що в хмарі зберігається кілька копій файлу даних. Різні варіації схем PDP, такі як [2], [4], [6], [7], [9], [10], [11], [12], [15], були запропоновані за різних криптографічними припущеннями. Але більшість цих схем мають справу лише зі статичними файлами даних і дійсні лише для перевірки однієї копії. Кілька інших схем, таких як [3], [5], [8], [13], [14], забезпечують динамічну масштабованість однієї копії файлу даних для різних застосувань, що означає, що до віддалено збережених даних можуть отримати доступ не лише авторизовані користувачі, але й власник даних може їх оновлювати та масштабувати.

У цьому розділі ми пропонуємо схему, яка дозволяє власнику даних безпечно гарантувати, що CSP зберігає кілька реплік. Простий спосіб зробити репліки унікальними та розрізнювальними – це використання ймовірнісних схем шифрування. Ймовірнісне шифрування створює різні шифротексти щоразу, коли одне й те саме повідомлення шифрується за допомогою одного й того ж ключа. Таким чином, наша схема використовує гомоморфне ймовірнісне шифрування для створення різних реплік/копій файлу даних та підписів BLS [17] для створення постійної кількості метаданих для будь-якої кількості реплік. Ймовірнісне шифрування шифрує всі репліки одним і тим самим

ключем. Таким чином, у нашій схемі власник даних повинен буде поділитися лише одним ключем розшифрування з авторизованими користувачами та не турбуватися про те, що CSP надасть доступ до будь-якої з реплік авторизованим користувачам. Гомоморфна властивість схеми шифрування сприяє ефективному оновленню файлів. Власник даних повинен зашифрувати різницю між оновленим файлом і старим файлом і надіслати її до хмари, яка оновлює всі репліки шляхом виконання гомоморфного додавання копій файлів. Будь-яка автентифікована структура даних, наприклад, хеш-дерева Мерклі та список пропусків, може бути використана з нашою схемою, щоб гарантувати, що хмара використовує правильні блоки файлів для перевірки цілісності даних. Однак, способи ефективного керування автентифікованими структурами даних у хмарі тут не розглядаються.

2.2 Напрацювання інших авторів

Атнієзе та ін. [2] першими визначили модель доказового володіння даними (PDP) для забезпечення володіння файлами на ненадійних сховищах. Вони використовували гомоморфні теги на основі RSA для аудиту даних, переданих на аутсорсинг. Однак динамічне зберігання даних та система з кількома репліками не розглядаються в цій схемі. У своїй наступній роботі [12] вони запропонували динамічну версію, яка підтримує дуже прості блочні операції з обмеженою функціональністю та не підтримує вставку блоків. У [13] Ван та ін. розглянули динамічне зберігання даних у розподіленому сценарії та запропонували протокол запит-відповідь, який може визначати правильність даних, а також знаходити можливі помилки. Подібно до [12], розглядається лише часткова підтримка динамічних операцій з даними. Ервей та ін. [14] розширили модель PDP у [2] для підтримки доказових оновлень збережених файлів даних за допомогою автентифікованих списків пропусків на основі рангу. Однак ефективність їхньої схеми залишається неясною, і ці схеми придатні лише для перевірки однієї копії.

Куртмола та ін. [1] запропонували схему багатореплікаційного PDP (MR-PDP), деВласник даних може перевірити, чи кілька копій файлу зберігаються постачальником послуг зберігання. У їхній схемі окремі репліки створюються шляхом спочатку шифрування даних, а потім їх маскування випадковістю, згенерованою псевдовипадковою функцією (PRF). Рандомізовані дані потім зберігаються на кількох серверах. Схема використовує RSA-сигнатури для створення тегів. Однак їхня схема не враховувала, як авторизовані користувачі даних можуть отримати доступ до копій файлів з хмарних серверів, зазначаючи, що внутрішні операції CSP є непрозорими та не підтримують динамічні операції з даними. Аяд Ф. Барсум та ін. [16] запропонували створювати окремі копії шляхом додавання номера репліки до блоків файлу та шифрування їх за допомогою схеми шифрування, яка має сильну властивість дифузії, наприклад, AES. Їхня схема підтримує динамічні операції з даними, але під час оновлення файлів копії на всіх серверах повинні бути знову зашифровані та оновлені в хмарі. Ця схема ідеально підходить для статичних кількох реплік, але виявляється дорогою в динамічному сценарії.

Підписи BLS використовуються для створення тегів, а автентифіковані структури даних, такі як хеш-дерева Мерклі, використовуються для забезпечення використання правильних блоків файлів під час перевірки. Авторизовані користувачі даних повинні знати випадкові числа з [1] та номер репліки з [16] для генерації оригінального файлу.

2.3 Схема динамічного багатореплікаційного доведення володіння даними (DMR-PDP)

Модель хмарних обчислень, яка розробляється в роботі, складається з трьох основнихкомпоненти, як показано на рис. 2.1:

1. Власник даних, яким може бути фізична особа або організація, яка спочатку володіла конфіденційними даними, що зберігатимуться в хмарі;

2. Постачальник комунікаційних послуг (CSP), який керує хмарними серверами та надає платний простір для зберігання файлів власника на своїй інфраструктурі;

3. Авторизовані користувачі – набір клієнтів власника, які мають право доступу до віддалених даних та обміну деякими ключами з власником даних.

Рисунок 2.1 - Модель зберігання даних хмарних обчислень.

Визначення проблеми та цілі проектування.

Зовсім недавно багато власників даних зняли з себе тягар локального зберігання та обслуговування даних, передаючи свої дані на аутсорсинг постачальнику послуг зв'язку (CSP). CSP виконує завдання реплікації даних, щоб підвищити доступність, довговічність та надійність даних, але клієнти повинні платити за використання інфраструктури зберігання CSP. З іншого боку, клієнти хмарних технологій повинні бути переконані, що (1) CSP дійсно володіє всіма копіями даних, як було домовлено, (2) цілісність цих копій даних зберігається, та (3) клієнти отримують послугу, за яку вони платять. Тому в цьому розділі ми розглядаємо проблему безпечного та ефективного створення кількох реплік файлу даних власника для зберігання на ненадійному CSP, а потім аудит усіх цих копій для перевірки їх повноти та правильності. Наші цілі проектування коротко наведено нижче:

1. Протоколи динамічного багатореплікаційного доказового володіння даними (DMR-PDP) повинні ефективно та безпечно надавати власнику вагомі докази того, що CSP володіє всіма узгодженими копіями даних і що ці копії є неушкодженими.

2. Надання користувачам, уповноваженим власником даних, безперешкодного доступу до копії файлу з CSP.

3. Використання лише одного набору метаданих/тегів для всіх реплік файлів з метою перевірки.

4. Забезпечення підтримки динамічних операцій з даними, що дозволяє власнику даних виконувати операції на рівні блоків над файлами даних, зберігаючи при цьому той самий рівень гарантії правильності даних.

5. Забезпечення як ймовірнісної, так і детерміністичної гарантій перевірки.

У цьому розділі ми детально розглянемо схеми білінійного відображення та шифрування Пайльє, що використовуються в нашій роботі.

1. Припустимо, що F, файл даних, який буде передано на аутсорсинг, складається з послідовності m блоків, тобто $F = \{b_1, b_2,..,b_m\}$.

2. $F_i = \{b_{i1}, b_{i2},\ldots,bim\}$ представляє копію файлу i.

3. Білінійне відображення/спарювання: Нехай G_1, G_2 та G_T – циклічні групи простого порядку *a*. Нехай *u* та *v* – генератори G_1 та G_2 відповідно. Білінійне спаювання це відображення $e : G_1 \times G_2 \rightarrow G_T$ з такими властивостями:

- Білінійний: $e(u_1u_2, v_1) = e(u_1, v_1) \cdot e(u_2, v_1)$, $e(u_1, v_1v_2) = e(u_1, v_1) \cdot e(u_1,v_2)$ $\forall$ $u_1, u_2 \in G_1$ та $v_1, v_2 \in G_2$
- Невироджена: $e(u, v) \neq 1$
- Існує ефективний алгоритм для обчислення *e*
- $e(u_1^x, v_1^y) = e(u_1, v_1)^{xy}$ $\forall$ $u_1 \in G_1$; $v_1 \in G_2$, and x, y $\in Z_a$

4. H(.) — це хеш-функція типу «відображення в точку»: $\{0, 1\}* \rightarrow G_1$.

5. Гомоморфне шифрування: Схема гомоморфного шифрування має наступні властивості зниження.

- $E(m_1 + m_2) = E(m_1) +h E(m_2)$,

де +h — гомоморфна операція додавання.

- $E(k*m) = E(m)^k$.

де E(.) представляє гомоморфну схему шифрування, а m, m_1, m_2 – зашифровані повідомлення, а k – деяке випадкове число.

6. Шифрування Пайльє:Криптосистема Пайє є гомоморфною ймовірнісноюсхема шифрування. Кроки такі.

- Обчисліть N = p * q та λ = НСК (p-1, q-1), де p, q – два простих числа.
- Вибрати випадкове число g таке, що його порядок є кратним N та g $\in Z_N^{2*}$.
- Відкритий ключ — (N, g), а секретний ключ — λ, де N = p*q.
- Шифрований текст для повідомлення m обчислюється як $c = g^m r^N \bmod N^2$, де r – випадкове число, а $r \in Z_N*$, $c \in Z_N^2*$ та $m \in Z_N$.
- Звичайний текст отримується за формулою $m = L(c^\lambda \bmod N^2) * (L(g^\lambda \bmod N^2))^{-1} \bmod N$.

7. Властивості відкритого ключаg у схемі Пайє

- $g \in Z_N^{2*}$.
- Якщо $g = (1 + N) \bmod N^2$, то воно має кілька цікавих властивостей

- Порядок значення (1 + N) дорівнює N.
- $(1 + N)^m \equiv (1 + mN) \bmod N^2$. (1 + mN) можна використовувати безпосередньо замість обчислення $(1 + N)^m$. Це дозволяє уникнути дорогої експоненціальної операції під час шифрування даних.

2.4 Розробка DMR-PDP.

У нашому підході власник даних створює кілька зашифрованих реплік та завантажує їх у хмару. Постачальник послуг зв'язку (CSP) зберігає їх на одному або кількох серверах, розташованих у різних географічних місцях. Власник даних надає ключ розшифрування групі авторизованих користувачів. Для доступудані, авторизований користувач надсилає запит даних до CSP та

отримує копію даних у зашифрованому вигляді, яку можна розшифрувати за допомогою секретного ключа, що надається власником. Запропонована схема складається з семи алгоритмів:KeyGen, Replica-Gen, TagGen, Prove, Verify, PrepareUpdate та ExecUpdate. Огляд комунікації, що використовується в нашій схемі, показано на рисунку 2.

1. (pk, sk) ← KeyGen(). Цей алгоритм виконується власником даних для генерації відкритого ключа *pk* та закритого ключа *sk*. Власник даних генерує три набори ключів.

- Ключі для тегів даних: Цей ключ використовується для створення тегів для даних. Власник даних вибирає білінійне відображення e та вибирає закритий ключ l ∈ Za. Відкритий ключ обчислюється як $y = v^l \in G_2$.

- Ключі для даних: Цей ключ використовується для шифрування даних і, таким чином, створення кількох копій даних. Власник даних вибирає відкриті ключі Пайльє (N, g) з $g = (1 + N) \bmod N^2$ та секретний ключ λ.

- Ключ PRF: Власник даних генерує ключ PRF KeyPRF, який генерує *s* чисел. Ці *s* чисел використовуються для створення *s* копій даних. Кожне число використовується для створення однієї копії даних. Нехай $\{k_1,k_2,..,k_s\} \in Z_N*$ – числа, згенеровані ключем PRF. KeyPRF зберігається конфіденційно власником даних, і звідси номери sякі використовуються для створення кількох копій, невідомі хмарі.

2. $\{F_i\}_{1\le i\le s} \leftarrow \text{ReplicaGen}(s, F)$. Цей алгоритм запускається власником даних. Він приймає кількість репліксі файл *F* як вхідні дані та генерує унікальні *s* диференційовних копій $\{F_i\}_{1\le i\le s}$. Цей алгоритм виконується лише один раз. Унікальні копії кожного блоку файлу *F* створюються шляхом його шифрування за допомогою ймовірнісної схеми шифрування, наприклад, схеми шифрування Пайє. Завдяки ймовірнісному шифруванню, шифрування блоку файлу s разів дає s різних шифртекстів. Для файлу $F = \{b_1, b_2,..,b_m\}$ генерується кілька копій даних за допомогою схеми шифрування Пайє як $F_i = \{(1+N)^{b_1}(k_i r_{i1})^N, (1+N)^{b_2}(k_i r_{i2})^N,..., (1+N)^{b_m}(k_i r_{im})^N\}_{1\le i\le m}$.

Використовуючи властивості Пайє, вищезазначений результат можна отримати $F_i = \{(1+b_1N)(k_ir_{i1})^N, (1+b_2N)(k_ir_{i2})^N,.., (1+b_mN)(k_ir_{im})^N\}_{1\le i\le m}$, де i представляє номер копії файлу, ki представляє числа, згенеровані з ключа PRF KeyPRF, а r_{ij} представляє будь-яке випадкове число, що використовується в схемі шифрування Пайє. k_i множиться на випадкове число r_{ij}, і добуток використовується для шифрування. Наявність k_i в блоці файлу визначає, до якої копії файлу належить блок файлу. Всі ці копії файлів дають оригінальному файлу після розшифрування. Це дозволяє користувачам, уповноваженим власником даних безперешкодно отримати доступ до копії файлу, отриманої від CSP.

3. $\phi \leftarrow \text{TagGen (sk, F)}$. Цей алгоритм запускається власником даних. Він приймає закритий ключ sk та файл F як вхідні дані та виводить теги φ. Ми використовуємо схему підписів BLS для створення тегів на даних. Підписи BLS є короткими та гомоморфними за своєю природою і дозволяють одночасну перевірку даних, що означає, що кілька блоків даних можуть бути перевірені одночасно. У нашій схемі теги генеруються на кожному блоці файлу b_i як $\phi_i = (H(F) \cdot u^{b_iN})^l \in G_1$, де $u \in G_1$, а $H(.) \in G_1$ представляє хеш-значення, яке однозначно представляє файл F. Власник даних надсилає набір тегів $\phi = \{\phi_i\}_{1\le i\le m}$ до хмари.

4. $P \leftarrow \text{Prove (F, } \phi, \textit{challenge})$. Цей алгоритм виконується постачальником послуг зв'язку (CSP). Він приймає копії файлу F, теги φ та вектор викликів, надіслані власником даних, як вхідні дані, та повертає доказ P, який гарантує, що CSP фактично зберігає s копій файлу F, і всі ці копії є цілісними. Власник даних використовує доказ P для перевірки цілісності даних. У цьому алгоритмі є дві фази:

а) Виклик: На цьому етапі власник даних звертається до хмари з проханням перевірити цілісність усіх переданих на аутсорсинг копій. Існує два типи схем перевірки:

- Детермінований – тут для перевірки використовуються всі блоки файлів з усіх копій.

- Імовірнісний – для перевірки використовується лише кілька блоків з усіх копій. Для генерації випадкових індексів у діапазоні від 1 до m використовується ключ псевдовипадкової функції (PRF). Блоки файлу з цих індексів використовуються для перевірки. Під час кожної перевірки перевіряється певний відсоток файлу, і це враховує перевірку всього файлу.

Під час кожного запиту власник даних обирає тип схеми верифікації, яку він бажає використовувати. Якщо власник обирає детерміністичну схему верифікації, він генерує один ключ PRF, Key1. Якщо він обирає ймовірнісну схему, він генерує два ключі PRF, Key1 та Key2. PRF з ключем Key1 генерує c ($1 \leq c \leq m$) випадкових індексів файлів, які вказують блоки файлів, які CSP повинен використовувати для перевірки. PRF з ключем Key2 генерує s випадкових значень, і CSP повинен використовувати кожне з цих випадкових чисел для кожної копії файлу під час обчислення відповіді. Власник даних надсилає згенеровані ключі до CSP.

б) Відповідь: Цей етап виконується CSP, коли від власника даних отримує запит на перевірку цілісності даних. Тут ми показуємо доказ для ймовірнісної схеми перевірки (детерміністична схема перевірки також дотримується тієї ж процедури). CSP отримує від власника даних два ключі PRF, Key1 та Key2. Використовуючи Key1, CSP генерує множину {C} з c ($1 \leq c \leq m$) випадковими індексами файлів ($\{C\} \in \{1, 2,..,m\}$), які вказують на блоки файлу, що CSP має використовувати для перевірки. Використовуючи Key2, CSP генерує 's' випадкових значень $T = \{t_1, t_2,..,t_s\}$. Хмара виконує дві операції. Одну з тегами, а іншу — з блоками файлів.

- Операція з тегами: Хмара множить відповідні теги файлів до файлових індексів, згенерованих ключем PRF Key1.

$$\sigma = \prod_{j \in C} (H(F).u^{b_j N})^l$$

$$= \prod_{j \in C} H(F)^l . \prod_{j \in C} u^{b_j Nl}$$

$$= H(F)^{cl} . u^{Nl \sum_{j \in C} (b_j)}$$

- Операція з блоками файлів: Хмара спочатку бере кожну копію файлу та множить усі блоки файлів, що відповідають індексам файлів, згенерованим ключем PRF Key1. Добуток кожної копії зводиться до степеня випадкового числа, згенерованого для цієї копії ключем PRF Key2. Результат вищезазначеної операції для кожної копії файлу і задається як $(\prod_{j \in C} (1+N)^{b_j} (k_i r_{ij})^N)^{t_i} \bmod N^2$. Потім CSP множиться на результат кожної копії, щоб отримати результат

$$\mu = \prod_{i=1}^{s} (\prod_{j \in C} (1+N)^{b_j} (k_i r_{ij})^N)^{t_i}$$

$$= \prod_{i=1}^{s} (\prod_{j \in C} (1+N)^{b_j t_i} \prod_{j \in C} (k_i r_{ij})^{N t_i})$$

$$= \prod_{i=1}^{s} ((1+N)^{t_i \sum_{j \in C} b_j} \prod_{j \in C} (k_i r_{ij})^{N t_i})$$

$$= (\prod_{i=1}^{s} (1+N)^{t_i \sum_{j \in C} b_j})(\prod_{i=1}^{s} ((k_i)^{c t_i N} \prod_{j \in C} (r_{ij})^{N t_i}))$$

$$= ((1+N)^{\sum_{i=1}^{s} t_i \sum_{j \in C} b_j})(\prod_{i=1}^{s} (k_i)^{c t_i N})(\prod_{i=1}^{s} \prod_{j \in C} (r_{ij})^{N t_i})$$

Використовуючи властивості схеми Пайє, вищезазначене рівняння можна переписати як

$$\mu = (1 + N\sum_{i=1}^{s}(t_i)\sum_{j\in C}(b_j))(\prod_{i=1}^{s}(k_i)^{Nct_i})(\prod_{i=1}^{s}(\prod_{j\in C}(r_{ij})^{t_iN}))$$

CSP надсилає значення σ та μ mod N^2 власнику даних.

5. $\{1, 0\} \leftarrow \text{Verify (pk, P)}$. Цей алгоритм виконується власником даних. Він приймає на вхід відкритий ключ pk та доказ P, повернутий від CSP, і видає 1, якщо цілісність усіх копій файлу правильно перевірена, або 0 в іншому випадку. Після отриманняста значення μ від CSP, власник даних виконує наступне:

- обчислює $\text{v} = (\prod_{i=1}^{s}(\text{k}_i)^{t_icN})$ та $\text{d} = \text{Decrypt}(\mu)/(\sum_{i=1}^{s}\text{k}_i)$. Це може бути розраховане зі значень, згенерованих з KeyPRF та c.

- перевіряє, чи μ mod v ≡ 0. Це гарантує, що хмара використала всі копії файлів під час обчислення відповіді.

- перевіряє, чи $(\text{H(F)}^c\ \text{u}^{dN})^l = \sigma$. Це гарантує, що CSP використав усі блоки файлу під час обчислення відповіді. Якщо варіанти b та c виконуються, це означає, що дані, що зберігаються власником у хмарі, є неушкодженими, і хмара зберегла кілька копій даних, як узгоджено в угоді про рівень обслуговування.

6. $Update \leftarrow \text{PrepareUpdate ()}$. Цей алгоритм запускається власником даних для виконання будь-якої операції з копіями файлів, переданими на аутсорсинг, що зберігаються віддаленим постачальником послуг зв'язку (CSP). Виходом цього алгоритму єОновленнязапит. Власник даних надсилає запит на оновлення до хмари, який матиме форму <Id_F, BlockOp, j, b_i', φ'>, де Id_F – ідентифікатор файлу, BlockOp відповідає операції з блоком, j позначає індекс блоку файлу, b_i' представляє оновлені блоки файлу, а φ' – оновлений тег. BlockOp може бути операцією модифікації, вставки або видалення даних.

Літинг 1.1 - Операція модифікації блоків у схемі DMR-PDP

Owner **CSP**

1. Calculates $\Delta b_j = b_j' - b_j$.
2. Encrypts Δb_j using Paillier encryption.
   $E(\Delta b_j) = (1 + \Delta b_j N)\, r^N$, where r is some random number.
3. Calculates the new file tag for b_j', $\phi' = (H(F)\, u^{b_j' N})^l$.
4. Generates PRF keys Key_1, Key_2 to verify the correctness of modify operation.

$$<Id_F, \text{modify}, j, E(\Delta b_j), \phi'> , Key_1, Key_2 \longrightarrow$$

5. Performs homomorphic addition operation $E(b_j') = E(\Delta b_j) * E(b_j)$ on all the file copies.
6. Deletes the old tag and replaces it with the new tag ϕ'.
7. Calculates a response μ, σ.

$$\mu, \sigma \longleftarrow$$

8. Calculates v and d
9. Verifies if $\mu \bmod v \equiv 0$ and checks if $(H(F)^c\, u^{dN})^l = \sigma$.

7. $(F', \phi') \leftarrow \text{ExecUpdate}\ (F, \phi, Update)$. Цей алгоритм виконується CSP, де вхідними параметрами є копії файлів F, теги φ та запит на оновлення (надісланий від власника). Він виводить оновлену версію всіх копій файлів F' разом з оновленими підписами φ'. Після будь-якої операції з блоком власник даних запускає протокол виклику, щоб переконатися, що хмара виконала операції правильно. Операцією в запиті на оновлення може бути зміна блоку файлу, вставка нового блоку файлу або видалення блоку файлу.

- Модифікація:Модифікація даних є однією з найчастіше використовуваних динамічних операцій. Операція модифікації даних у схемі DMR-PDP показана у лістнгу 1.1.

- Вставка:Під час операції вставки блоку власник вставляє новий блок після позиції j у файлі. Якщо файл F спочатку містив m блоків, то після операції вставки файл матиме m+1 блоків. Операція вставки блоку файлу показана у лістингу 1.2.

- Видалення:Операція видалення блоку є протилежною операції вставки. Коли один блок видаляється, індекси всіх наступних блоків переміщуються на один крок вперед. Щоб видалити певний блок даних у

позиції j з усіх копій, власник надсилає запит на видалення <Id_F , delete, j, null, null> до хмари. Після отримання запиту хмара видаляє тег та блок файлу за індексом j у всіх копіях файлів.

Лістинг 1.2 - Операція вставки блоків у схемі DMR-PDP

Owner **CSP**

1. Encrypts the new file block s times
2. Creates tag ϕ for the new file block
3. Generates PRF keys Key_1, Key_2 to verify the correctness of insert operation.

<Id_F, insert, j, s file blocks, ϕ> , Key_1, Key_2 →

4. Inserts the new file block at location j
5. Stores the new tag ϕ.
6. Calculates a response μ, σ.

← μ, σ

7. Calculates v and d
8. Verifies if $\mu \bmod v \equiv 0$ and checks if $(H(F)^c\ u^{dN})^l = \sigma$.

2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи ми обговорили роботу, пов'язану зі збереженням цілісності реплікованих даних у хмарному середовищі, та представили схему динамічного багатореплікаційного доказового володіння даними (DMR-PDP) для періодичної перевірки правильності та повноти кількох копій даних, що зберігаються в хмарі. Наша схема також підтримує операції динамічного оновлення даних. Усі копії даних можна розшифрувати за допомогою одного ключа розшифрування, що забезпечує безперешкодний доступ до даних для авторизованих користувачів. Цю схему можна розширити для кількох версій, де в хмарі можна зберігати лише дельти, і власник може заощадити на вартості зберігання. Наразі ми впроваджуємо запропоновану схему для її оцінки на реальній хмарній платформі, використовуючи різні показники продуктивності та порівнюючи її з деякими існуючими методами.

РОЗДІЛ 3. ЕФЕКТИВНА ПЕРЕВІРКА ЦІЛІСНОСТІ РЕПЛІКОВАНИХ ДАНИХ У ХМАРІ З ВИКОРИСТАННЯМ ГОМОМОРФНОГО ШИФРУВАННЯ

У цьому розділі ми пропонуємо схему динамічного багатореплікаційного доказового володіння даними (DMR-PDP), яка запобігає шахрайству з боку CSP, наприклад, шляхом збереження меншої кількості копій, ніж оплачено, та/або підробці даних. Крім того, ми також розширюємо схему для підтримки базової системи керування версіями файлів, де поширюється лише різниця між оригінальним та оновленим файлами, а не поширення операцій з міркувань конфіденційності. DMR-PDP також підтримує ефективні динамічні операції, такі як модифікація блоків, вставка та видалення на репліках через хмарні сервери. За допомогою аналізу безпеки та експериментальних результатів ми демонструємо, що запропонована схема є безпечною та працює краще, ніж деякі інші пов'язані ідеї, опубліковані нещодавно.

3.1 Концепція модифікованого алгоритму

Коли користувачі зберігають дані в хмарі, їх головним чином хвилює, чи хмара може підтримувати цілісність даних та чи можна відновити дані у разі втрати даних або збою сервера. Постачальники хмарних послуг (CSP), щоб заощадити кошти на зберіганні, можуть схильні скидати деякі дані або копії даних, до яких не часто звертаються, або переміщувати такі дані на пристрої зберігання другого рівня. CSP також можуть приховувати втрату даних через помилки управління, збої обладнання або атаки. Тому критичним питанням зберігання даних у ненадійних CSP є періодична перевірка того, чи підтримують сервери зберігання цілісність даних та чи зберігають дані повністю та правильно, як зазначено в Угоді про рівень обслуговування (SLA).

Реплікація – це поширений метод підвищення доступності даниху хмарних обчисленнях. Хмара реплікує дані та стратегічно зберігає їх на кількох

серверах, розташованих у різних географічних місцях. Оскільки репліковані дані є копіями, важко перевірити, чи дійсно хмара зберігає кілька копій даних. Хмара може легко обдурити власника, зберігаючи лише одну копію даних. Таким чином, власник хотів би регулярно перевіряти, чи дійсно хмара має кілька копій даних, як заявлено в угоді про рівень обслуговування. Загалом, хмара має можливість генерувати кілька реплік, коли власник даних вимагає від CSP довести, що він володіє кількома копіями даних. Також є обґрунтованим припущення, що власник даних може не мати копії даних, що зберігається локально. Отже, головним завданням власника є не лише перевірка цілісності даних, але й їх відновлення, якщо виявлено будь-які видалення/пошкодження даних. Якщо власник під час перевірки за схемою DMR-PDP виявить втрату даних у будь-якій з реплік у хмарі, він може відновити дані з інших реплік, які зберігаються без змін. Оскільки репліки повинні зберігатися в різних географічних місцях, вважається безпечним, що втрата даних не відбудеться на всіх репліках одночасно.

Доказове володіння даними (PDP) [2] – це метод аудиту та перевірки цілісності даних, що зберігаються на віддалених серверах. У типовій моделі PDP власник даних генерує метадані/теги для файлу даних, які згодом будуть використані для перевірки цілісності. Для забезпечення безпеки власник даних шифрує файл і генерує теги на зашифрованому файлі. Власник даних надсилає зашифрований файл і теги до хмари та видаляє локальну копію файлу. Коли власник даних бажає перевірити цілісність даних, він генерує вектор виклику та надсилає його до хмари. Хмара відповідає, обчислюючи відповідь на дані та надсилаючи її верифікатору/власнику даних, щоб довести, що в хмарі зберігається кілька копій файлу даних. Різні варіації схем PDP, такі як [2], [4], [6], [7], [9], [10], [11], [12], [15], були запропоновані за різних криптографічними припущеннями. Однак більшість цих схем мають справу лише зі статичними файлами даних і дійсні лише для перевірки однієї копії. Кілька інших схем, таких як [3], [5], [8], [13], [14], забезпечують динамічну масштабованість однієї копії файлу даних для різних застосувань, що означає,

що до віддалено збережених даних можуть отримати доступ не лише авторизовані користувачі, але й власник даних може їх оновлювати та масштабувати.

У цьому розділі ми пропонуємо схему, яка дозволяє власнику даних безпечно гарантувати, що CSP зберігає кілька реплік. Простий спосіб створення реплік унікальними та диференційованими завдяки використанню ймовірнісних схем шифрування. Ймовірнісне шифрування створює різні шифротексти щоразу, коли одне й те саме повідомлення шифрується за допомогою одного й того ж ключа. Таким чином, наша схема використовує гомоморфне ймовірнісне шифрування для створення окремих реплік/копій файлу даних та підписи схеми підписів Boneh Lynn Shacham (BLS) [17] для створення постійної кількості метаданих для будь-якої кількості реплік. Ймовірнісне шифрування шифрує всі репліки за допомогою той самий ключ. Таким чином, у нашій схемі власник даних повинен буде поділитися лише одним ключем дешифрування з авторизованими користувачами та не турбуватися про те, що CSP надасть доступ до будь-якої з реплік авторизованим користувачам. Гомоморфна властивість схеми шифрування сприяє ефективному оновленню файлів. Власник даних повинен зашифрувати різницю між оновленим файлом та старим файлом і надіслати її до хмари, яка оновлює всі репліки, виконуючи гомоморфне додавання копій файлів. Будь-яка автентифікована структура даних, така як хеш-дерева Меркла або Skiplist, може бути використана з нашою схемою, щоб гарантувати, що хмара використовує правильні блоки файлів для перевірки цілісності даних. Однак способи ефективного управління автентифікованими структурами даних у хмарі не входять до обсягу цієї статті. Підписи RSA, що використовуються в [1], також можуть бути використані зі схемою DMR-PDP для створення тегів даних. Ми порівнюємо продуктивність підписів BLS та RSA та обговорюємо переваги використання підписів BLS над підписами RSA. Ми визначаємо можливі атаки, які CSP може використовувати для обману власника даних, та надаємо аналіз безпеки запропонованого протоколу від цих атак. Ефективність протоколу була експериментально

перевірена, і результати показують, що наш протокол є ефективнішим, ніж схема в [16].

Ми також розширили схему DMR-PDP для підтримки базового керування версіями файлів системи. Перевага системи версій файлів полягає в тому, що власник даних може переглядати зміни, внесені до файлу, а також отримувати старіші версії файлу. Хмарні обчислювальні рішення, такі як Dropbox, Google Drive, надають базову систему контролю версій файлів на додаток до послуги зберігання даних. Моделі керування версіями файлів клієнт-сервер, такі як SVN, пропонують набагато більше функцій порівняно з тим, що пропонують ці хмарні сервіси пропонують рішення. Усі ці рішення використовують техніку дельта-стиснення, де лише різниця між оригінальним файлом та оновленим файлом поширюється та зберігається на сервері. Різниці записуються в окремі файли, які називаються «дельтами» або «різницями». Щоб отримати будь-яку конкретну версію файлу, зміна або дельта, що зберігається на сервері, об'єднується з базовою версією. Це зменшує пропускну здатність та обсяг пам'яті на сервері. У наших поточних умовах власник даних використовує гомоморфну ймовірнісну схему шифрування для шифрування та генерує кілька зашифрованих реплік файлу. Хмара зберігає репліки на кількох серверах. Завдання полягає в тому, щоб зберігати кілька реплік файлу та підтримувати оновлення файлу як дельти для підтримки базової системи керування версіями файлів, коли файли зашифровані. Щоб вирішити цю проблему, ми пропонуємо систему керування версіями файлів з кількома репліками (MRFVCS) як розширення схеми DMR-PDP, яка підтримує керування версіями зашифрованих файлів, коли дані реплікуються, і власник даних все ще може використовувати схему DMR-PDP для перевірки цілісності даних. Ми реалізували схему MRFVCS, і наші експерименти показують, що вона ефективно підтримує версії та потребує лише невеликого місця для зберігання даних на стороні власника даних.

3.2 Суміжні роботи

Ateniese та ін. [2] першими визначили модель Provable Data Possession (PDP) для забезпечення володіння файлами на ненадійних сховищах. Вони використовували гомоморфні теги на основі RSA для аудиту даних, переданих на аутсорсинг. Однак динамічне сховище даних та система з кількома репліками не враховуються в цій схемі. У своїй наступній роботі [3] та [12] вони запропонували динамічну версію, яка підтримує дуже прості блокові операції з обмеженою функціональністю та не підтримує вставку блоків. У [13] Ван та ін. розглядали динамічне зберігання даних у розподіленому сценарії та запропонували протокол запиту-відповіді, який може визначати правильність даних, а також знаходити можливі помилки. Подібно до [12], розглядається лише часткова підтримка динамічних операцій з даними. Ервей та ін. [5] розширили модель PDP у [2] для підтримки динамічних оновлень збережених файлів даних за допомогою автентифікованих списків пропусків на основі рангу. Однак ефективність їхньої схеми залишається неясною, і ці схеми придатні лише для перевірки однієї копії. Ван та ін. [14] використовують хеш-дерева Меркла (MHT) для перевірки цілісності даних, і їхня схема підтримує динамічні операції з даними, але в їхній схемі дані не шифруються та працюють лише для однієї копії. Хао та ін. [8] запропонували схему, яка підтримує як динамічні операції з даними, так і публічну перевірку. Публічна перевірка дозволяє будь-кому, не обов'язково власнику даних, перевірити цілісність даних, що зберігаються в хмарі, запустивши протокол запиту-відповіді. Їхня схема також не враховує шифрування даних, і їхню схему не можна розширити відповідно до сценарію, коли зберігається кілька копій даних.

Куртмола та ін. [1] запропонували схему багатореплікаційного PDP (MR-PDP), деВласник даних може перевірити, чи кілька копій файлу зберігаються постачальником послуг зберігання. У їхній схемі окремі репліки створюються шляхом спочатку шифрування даних, а потім їх маскування випадковістю, згенерованою псевдовипадковою функцією (PRF). Рандомізовані дані потім

зберігаються на кількох серверах. Схема використовує RSA-підписи для створення тегів. Однак їхня схема не враховувала, як авторизовані користувачі даних можуть отримати доступ до копій файлів з хмарних серверів, зазначаючи, що внутрішні операції CSP є непрозорими та не підтримують динамічні операції з даними. Аяд Ф. Барсум та ін. [16] запропонували створювати окремі копії шляхом додавання номера репліки до блоків файлу та шифрування їх за допомогою схеми шифрування, яка має сильну властивість дифузії, наприклад, AES. Їхня схема підтримує динамічні операції з даними, але під час оновлення файлів копії на всіх серверах повинні бути знову зашифровані та оновлені в хмарі. Ця схема ідеально підходить для статичних кількох реплік, але виявляється дорогою в динамічному сценарії. Підписи BLS використовуються для створення тегів, а автентифіковані структури даних, такі як хеш-дерева Меркла, використовуються для забезпечення використання правильних блоків файлів під час перевірки. Авторизовані користувачі даних повинні знати випадкові числа з [1] та номер репліки з [16] для генерації оригінального файлу.

3.3 Схема динамічного багатореплікаційного доведення володіння даними (DMR-PDP)

Модель хмарних обчислень, що розглядається в цій роботі, складається з трьох основнихкомпоненти, як показано на рис. 3.1:

- власник даних, яким може бути фізична особа або організація, яка спочатку володіла конфіденційними даними, що зберігатимуться в хмарі;

- постачальник комунікаційних послуг (CSP), який керує хмарними серверами та надає платний простір для зберігання файлів власника на своїй інфраструктурі, та (iii) авторизовані користувачі – набір клієнтів власника, які мають право доступу до віддалених даних та обміну деякими ключами з власником даних.

Рисунок 3.1 - Модель зберігання даних хмарних обчислень.

Визначення проблеми та цілі проектування.

Зовсім недавно багато власників даних зняли з себе тягар локального зберігання та обслуговування даних, передаючи свої дані на аутсорсинг постачальнику послуг зв'язку (CSP). CSP виконує завдання реплікації даних, щоб підвищити доступність, довговічність та надійність даних, але клієнти повинні платити за використання інфраструктури зберігання CSP. З іншого боку, клієнти хмара повинні бути переконані, що

- CSP дійсно володіє всіма копіями даних, як було домовлено;

- цілісність цих копій даних зберігається;

- власники можуть оновлювати дані, за які вони платять.

Тому в цьому розділі розглядаємо проблему безпечного та ефективного створення кількох реплік файлу даних власника для зберігання на ненадійному CSP, а потім аудиту всіх цих копій для перевірки їхньої повноти та правильності. Наші цілі проектування коротко викладено нижче:

1. Протоколи динамічного багатореплікаційного доказового володіння даними (DMR-PDP) повинні ефективно та безпечно надавати власнику вагомі докази того, що CSP володіє всіма узгодженими копіями даних і що ці копії є неушкодженими.

2. Надання користувачам, уповноваженим власником даних, безперешкодного доступу до копії файлу з CSP.

3. Використання лише одного набору метаданих/тегів для всіх реплік файлів з метою перевірки.

4. Забезпечення підтримки динамічних операцій з даними, що дозволяє власнику даних виконувати операції на рівні блоків над файлами даних, зберігаючи при цьому той самий рівень гарантії правильності даних.

5. Забезпечення як ймовірнісної, так і детерміністичної гарантій перевірки.

Тут ми детально розглянемо схеми білінійного відображення та шифрування Пайльє, що використовуються в нашій роботі.

1. Припустимо, що F, файл даних, який буде передано на аутсорсинг, складається з послідовності m блоків, тобто $F = \{b_1, b_2,..,b_m\}$, де $b_i \in Z_N$, де Z_N – множина всіх залишків при діленні на N, а N – відкритий ключ у схемі Пайє.

2. Нехай F_i представляє копію файлу i. Отже, $F_i = \{bi_1, bi_2,...,b_{im}\}$, де bij представляє блок файлу b_j копії файлу i.

3. Підписи BLS: Підписи BLS – це короткі гомоморфні підписи, які використовують властивості білінійних пар на певних еліптичних кривих. Ці підписи дозволяють одночасну перевірку даних, коли можна перевірити кілька блоків одночасно.того ж часу.

4. Білінійне відображення/спарювання: Нехай G_1, G_2 та G_T — циклічні групи простого порядку a. Нехай u та v – генератори G_1 та G_2 відповідно. Білінійне сполучення – це відображення e : G_1 x $G_2 \rightarrow G_T$ з такими властивостями:

- Білінійний: $e(u1u2, v1) = e(u_1, v_1) \cdot e(u_2, v_1)$, $e(u_1, v_1v_2) = e(u_1, v_1) \cdot e(u_1,v_2)$ $\forall$ $u_1, u_2 \in G_1$ та $v_1, v_2 \in G_2$.
- Невироджена: $e(u, v) \neq 1$
- Існуєефективний алгоритм для обчислення e
- $e(u_1^x, v_1^y) = e(u_1, v_1)^{xy}$ $\forall$ $u_1 \in G_1$; $v_1 \in G_2$, and x, y $\in Z_a$

5. H(.) — це хеш-функція типу «відображення в точку»: $\{0, 1\}* \rightarrow G_1$.

6. Гомоморфне шифрування: Схема гомоморфного шифрування має наступневластивості зниження.

- $E(m_1 + m_2) = E(m_1) +h E(m_2)$, де +h — гомоморфна операція додавання.
- $E(k*m) = E(m)^k$.

де E(.) представляє гомоморфну схему шифрування, а m, m_1, m_2 – зашифровані повідомлення, а k – деяке випадкове число.

7. Шифрування Пайє: Криптосистема Пайє є гомоморфною ймовірнісноюсхема шифрування. Кроки такі.

- Обчисліть N = p * q та λ = НСК (p-1, q-1), де p, q – два простих числа.
- Вибрати випадкове число g таке, що його порядок є кратним N та g $\in Z_N^{2*}$.
- Відкритий ключ - (N, g), а секретний ключ - λ, де N = p*q.
- Шифрований текст для повідомлення m обчислюється як $c = g^m r^N \bmod N^2$, де r – випадкове число, а $r \in Z_N*$, $c \in Z_N^2*$ та $m \in Z_N$.
- Звичайний текст отримується за формулою $m = L(c^\lambda \bmod N^2) * (L(g^\lambda \bmod N^2))^{-1} \bmod N$.

8. Властивості відкритого ключаg у схемі Пайє

- $g \in Z_N^{2*}$.
- Якщо g = (1 + N) mod N^2, то воно має кілька цікавих властивостей

- Порядок значення (1 + N) дорівнює N.
- $(1 + N)^m \equiv (1 + mN) \bmod N^2$. (1 + mN) можна використовувати безпосередньо замість обчислення $(1 + N)^m$. Це дозволяє уникнути дорогої експоненціальної операції під час шифрування даних.

3.4 Модифікація DMR-PDP.

У нашому підході власник даних створює кілька зашифрованих реплік та завантажує їх у хмару. Постачальник послуг зв'язку (CSP) зберігає їх на одному або кількох серверах, розташованих у різних географічних місцях. Власник даних надає ключ розшифрування групі авторизованих користувачів. Для

доступуДля отримання даних авторизований користувач надсилає запит даних до CSP та отримує копію даних у зашифрованому вигляді, яку можна розшифрувати за допомогою секретного ключа, спільного з власником. Запропонована схема складається з семи алгоритмів: KeyGen, Replica-Gen, TagGen, Prove, Verify, PrepareUpdate та ExecUpdate.

Алгоритм подібний до описаного в розділі 2.4.

Лістинг 3.1 -Операція модифікації блоків у схемі DMR-PDP

Owner — **CSP**

1. Calculates $\Delta b_j = b_j' - b_j$.
2. Encrypts Δb_j using Paillier encryption.
   $E(\Delta b_j) = (1 + \Delta b_j N)\, r^N$, where r is some random number.
3. Calculates the new file tag for b_j', $\phi' = (H(F)\, u^{b_j' N})^l$.
4. Generates PRF keys Key_1, Key_2 to verify the correctness of modify operation.

$<Id_F, \text{modify}, j, E(\Delta b_j), \phi'>$, Key_1, Key_2 →

5. Performs homomorphic addition operation $E(b_j') = E(\Delta b_j) * E(b_j)$ on all the file copies.
6. Deletes the old tag and replaces it with the new tag ϕ'.
7. Calculates a response μ, σ.

← μ, σ

8. Calculates v and d
9. Verifies if $\mu * \text{Inverse}(\prod_{i=1}^{s} (r_i)^{t_i c N}) \bmod v \equiv 0$ and checks if $H(F)^{cl} \cdot u^{(Nd + \sum_{j \in C} a_j) l} = \sigma$.

Схема DMR-PDP добре працює, навіть якщо замість підписів BLS використовуються сигнатури RSA. Повна схема DMR-PDP з використанням сигнатур RSA показана у лістингу 3.3.

Лістинг 3.2 - Операція вставки блоків у схемі DMR-PDP

Owner **CSP**

1. Encrypts the new file block s times
2. Creates tag ϕ for the new file block
3. Generates PRF keys Key_1, Key_2 to verify the correctness of insert operation.

$<\mathrm{Id}_F$, insert, j, s file blocks, $\phi>$, Key_1, Key_2 →

4. Inserts the new file block at location j
5. Stores the new tag ϕ.
6. Calculates a response μ, σ.

← μ, σ

7. Calculates v and d
8. Verifies if μ^* Inverse$(\prod_{i=1}^{s} (\mathrm{r}_i)^{t_i cN})\bmod v \equiv 0$ and checks if $\mathrm{H(F)}^{cl} \cdot \mathrm{u}^{(Nd+\sum_{j\in C}^{i=1} a_j)l} = \sigma$.

3.5 Аналіз безпеки

У цьому розділі ми представляємо формальний аналіз безпеки запропонованого намисхема. Власник даних шифрує файли та зберігає їх у хмарі, яка є ненадійною, тому ми визначаємо хмару як головного супротивника в цій схемі.

Схема безпечна лише тоді, коли вона не дозволяє хмарі обдурити власника даних, видаляючи блоки файлів, і все одно пройти фазу запиту/відповіді, ініційовану власником даних.

- Захист від підробки відповіді противником:На етапі перевірки власник даних надсилає CSP два ключі PRF - Key1, Key2 та параметр 'c', який вказує кількість блоків файлу, які він бажає перевірити. Схема DMR-PDP надає власнику даних гнучкість у надсиланні різних ключів 'c' та PRF на кожному етапі перевірки CSP. Це гарантує, що відповідь, згенерована CSP, не буде однаковою для кожного запиту, надісланого власником даних. Це виключає будь-яку можливість для CSP підробити відповідь, фактично не обчисливши її.

Лістинг 3.3 -Схема DMR-PDP з використанням сигнатур RSA

Preliminaries

1. Data owner selects two prime numbers p, q
2. $N = pq$ is the RSA modulus
3. g is the generator of QR_N (QR_N is the set of quadratic residues modulo N)
4. Publick key is (N, g) and secret key is (p, q)
5. Data owner encrypts the file blocks s times
6. Data owner generates tags T_i for each file block b_i, where $T_i = g^{b_i} \bmod N$
6. The data file and the data tags are sent to the cloud.

Owner **CSP**

1. Generates PRF keys Key_1, Key_2 and sends them to the cloud to verify data integrity

$$\mathrm{Id}_F,\ Key_1,\ Key_2 \longrightarrow$$

2. Calculates μ
3. Calculates response using data tags

$$\sigma = \prod_{i=1}^{s} \mathrm{g}^{b_i} \bmod \mathrm{N}$$

$$\mu,\ \sigma \longleftarrow$$

4. Calculates v and d
5. Verifies if μ * Inverse$(\prod_{i=1}^{s} (\mathrm{r_i})^{t_i cN}) \bmod v \equiv 0$ and checks if $\mathrm{g}^d \bmod \mathrm{N} = \sigma$.

• Захист від видалення блоків файлів з однаковим значенням:Згенеровані теги даних будуть однаковими для подібних блоків файлів. Хоча блоки файлів зашифровані, хмара може ідентифікувати подібні блоки файлів, визначаючи теги з однаковим значенням. Хмара може обдурити користувача, зберігаючи лише один блок і видалення подібних блоків даних. Щоб уникнути цього, схема DMR-PDP рандомізує дані перед побудовою тегів. Дані в тегах додаються до випадкових чисел, згенерованих з ключа PRF Keytag. Отже, навіть якщо значення тегів даних однакові, значення блоків базового файлу даних не будуть однаковими. Для блоків файлу bi = bj

$$Tag(b_i) = (H(F).u^{b_i N + a_i})^l$$

$$Tag(b_j) = (H(F).u^{b_j N + a_j})^l$$

де a_i, a_j – випадкові числа, згенеровані з PRF-ключа Keytag. Хоча блоки даних однакові, згенеровані теги відрізнятимуться за значенням.

3.6 Система керування версіями файлів з кількома репліками

MRFVCS – це розширення схеми DMR-PDP для підтримки базової системи керування версіями файлів. Власник даних шифрує дані, створює кілька реплік та зберігає їх у хмарі. Під час оновлення даних файли даних не оновлюються безпосередньо, а оновлення зберігаються як дельти. За допомогою MRFVCS власник даних все ще може використовувати схему DMR-PDP, щоб перевірити, чи хмара зберігає кілька реплік та дельти неушкодженими.

Будівництво MRFVCS.

Власник даних розділяє файл на кілька файлових блоків та генерує унікальні множинні репліки й теги даних для файлових блоків. Унікальні множинні репліки файлових блоків генеруються за допомогою гомоморфної ймовірнісної схеми шифрування, а теги даних для файлових блоків генеруються за допомогою підписів BLS. Репліки файлових блоків та теги даних для файлових блоків надсилаються до хмари. Ці зашифровані репліки файлових блоків представляють базову версію незашифрованих файлових блоків. Будь-яка модифікація, внесена до останньої версії незашифрованих файлових блоків, призведе до створення нової версії файлових блоків. Нова версія файлових блоків не зберігається безпосередньо в хмарі, а замість цього зберігаються дельти. Дельта розраховується як різниця між нешифрованою новою версією блоку файлу та нешифрованою базовою версією блоку файлу. Коли потрібна певна версія блоків файлу, власник даних запитує хмару на об'єднання блоків дельти з базовою версією блоків файлу, щоб отримати необхідну версію блоків файлу. Власник даних використовує таблицю версій файлу для відстеження версій блоків файлу. Таблиця - це невелика структура даних, що зберігається на стороні верифікатора для перевірки цілісності та

узгодженості всіх файлів та їх версій, що зберігаються CSP. Нові версії блоків файлу генеруються, коли власник даних виконує операцію оновлення блоків файлу. Операція оновлення даних включає вставку нових блоків файлу або зміну чи видалення кількох блоків файлу. Блоки дельти генеруються лише тоді, коли операція оновлення має значення «змінити». Після завершення операції оновлення таблиця версій файлу оновлюється власником даних. Таблиця версій файлу підтримується лише на стороні власника даних. Зберігання таблиці версій файлів лише на стороні власника даних допоможе йому приховати деталі операцій оновлення від CSP. Таблиця версій файлів складається з п'яти стовпців: Номер блоку (BN), Номер дельта-блоку (DBN), Версія файлу (FV), Версія блоку (BV), Операція блоку (BO). BN діє як індексація блоків файлів. Він вказує фізичне положення блоку у файлі даних. DBN є індексацією дельта-блоку. Якщо дельта не існує, значення зберігається як '-'. FV вказує версію всього файлу, а BV вказує версію блоку файлу. BO вказує на операцію, виконану над блоком файлу. Максимальне значення FV вказує на останню версію файлу, а максимальне значення BV для певного BN вказує на останню версію цього конкретного блоку файлу. Якщо в таблиці версій файлу не знайдено запису номера блоку файлу, це означає, що над базовою версією блоку файлу не виконується жодних операцій оновлення, а блок файлу в базовій версії та останній версії файлу однакові. Коли блок файлу з номером блоку B, версією файлу V та версією блоку файлу Y змінюється, власник даних може змінити версію всього файлу на V+1 або зберегти блоки файлу з тією ж версією. Для обох цих варіантів У деяких випадках власник даних додає новий запис до таблиці версій файлу. У першому випадку запис таблиці буде <B, - , V+1, 0, Modify>, а в другому випадку запис таблиці буде <B, -, V, Y+1, Modify>. Запропонована схема складається з семи алгоритмів: Keygen, ReplicaGen, TagGen, Prove, Verify, PrepareUpdate, ExecUpdate, FileVersionRequest, FileVersionDeliver.

Ми реалізували нашу схему та протоколи мовою програмування C. Ми провели кілька експериментів, використовуючи локальні хмарні сервери, а

також хмарні екземпляри EC2 з різними конфігураціями. Ми виміряли час обчислення для різних операцій як для CSP, так і для користувача. Крім того, ми також виміряли затримку з точки зору вартості зв'язку. У цих експериментах враховувалися файли різних розмірів.

Для початку ми провели кілька експериментів на системі з процесором Intel(R) Xeon(R) 2,67 ГГц та 11 ГБ оперативної пам'яті під керуванням CentOS 6.3. У нашій реалізації ми використовуємо бібліотеку PBC версії 0.5.11. Для досягнення 80-бітного параметра безпеки вибрано групу кривих зі 160-бітним порядком групи, а розмір модуля N становить 1024 біти. Ми використовуємо криву Баррето-Неріга (BN) [18], визначену над полем простих чисел GF(p) з |p| = 160 та ступенем вбудовування = 12. Згенеровані теги даних є точками на цій кривій, і точка на цій кривій може бути представлена 160 бітами. Ми використовуємо алгоритм SHA для обчислення хешу файлу, а бібліотека PBC надає функції для представлення хеш-значень як точки на кривій. Власник даних повинен буде зберігати три PRF-ключі розміром 128 біт, один секретний ключ для тегів даних розміром 128 біт та один секретний ключ для шифрування даних розміром 1024 біти. Вартість зв'язку для кожної фази, понесена в цьому протоколі, наведена в таблиці 3.1.

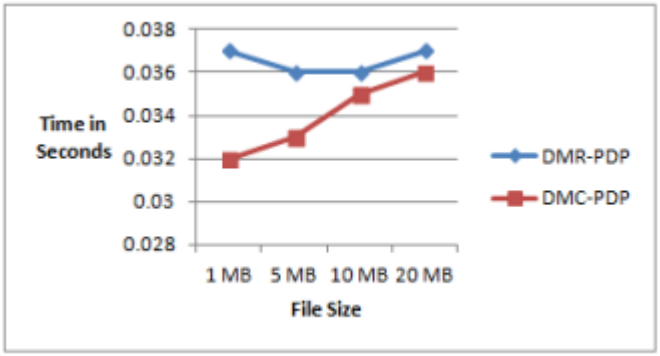
Таблиця 3.1 - Вартість зв'язку DMR-PDP

Фаза	Вартість	Дані	Від	До
Виклик	256 біт	Клавіша 1, Клавіша 2, c	Власник	Хмара
Перевірка	2048 + 160 біт	μ, σ	Хмара	Власник
Оновлення	2048 + 160 біт	bi', φ'	Власник	Хмара

Тут ми порівнюємо продуктивність схеми DMR-PDP, запропонованої в цій статті, зі схемою DMC-PDP, запропонованою в [16]. 1024-бітний модуль, який використовується для шифрування Пайльє в цій статті, порівнянний з точки зору безпеки зі 128-бітним шифруванням AES, що використовується в

[16]. На рис. 3.2 показано час обчислення CSP та User для обох схем з використанням файлів розміром 1, 5, 10 та 20 МБ з 3 репліками. Схема DMR-PDP має менший час обчислення CSP порівняно зі схемою DMC-PDP, тоді як час обчислення User для обох схем відрізняється лише на тисячні частки секунди. Обидві схеми включають лише операцію сполучення на етапі перевірки користувача, а отже, подібний час обчислення User. Продуктивність схеми DMR-PDP краща, ніж у схеми DMC-PDP, і покращується зі збільшенням розміру файлу.

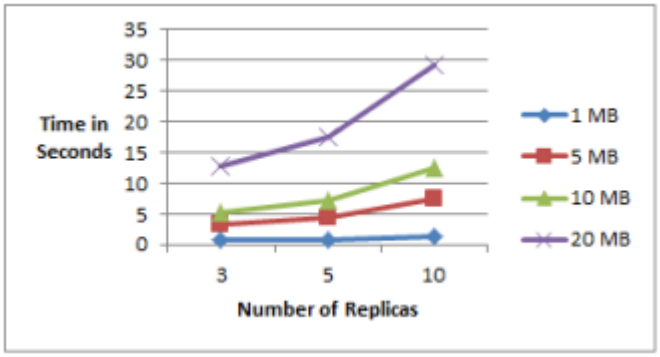
а)



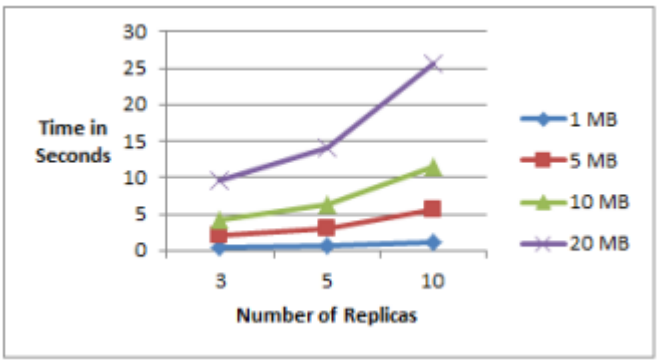
б)

Рисунок 3.2 - Порівняння часу обчислення: а) Час обчислення CSP; б) Час обчислення користувача.

Витрати CSP та користувачів на обчислення, понесені під час фази реагування запропонованої схеми, зображено на рис. 3.3.а та 3.4.а для 1, 5, 10 та 20 МБ файлів з розміром блоку зашифрованого файлу 2 КБ.



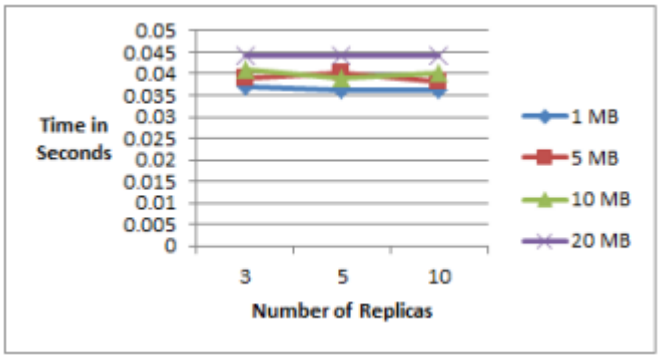
а)



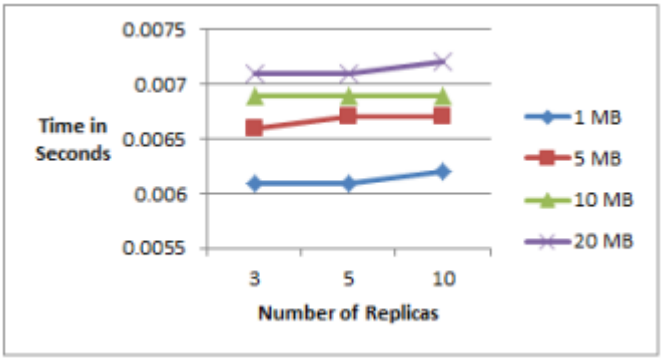
б)

Рисунок 3.3 - Порівняння часу обчислення CSP для кількості реплік налокальні хмарні сервери: а) Час обчислення CSP для сигнатур BLS; б) Час обчислення CSP для RSA-сигнатур

На рис. 3.3.а показано час обчислення в секундах на локальних хмарних серверах для різної кількості реплік. Для схеми DMR-PDP фаза перевірки користувача включає лише одну складну операцію сполучення, а час обчислення користувача не залежить від кількості реплік та розміру файлу, як показано на рис. 3.4.а. У [1] повідомлялося, що якщо на віддаленому сервері відсутня частина даних, то кількість блоків, які необхідно перевірити, щоб виявити неправильну поведінку сервера з високою ймовірністю, є постійною та не залежить від загальної кількості блоків файлу.



а)



б)

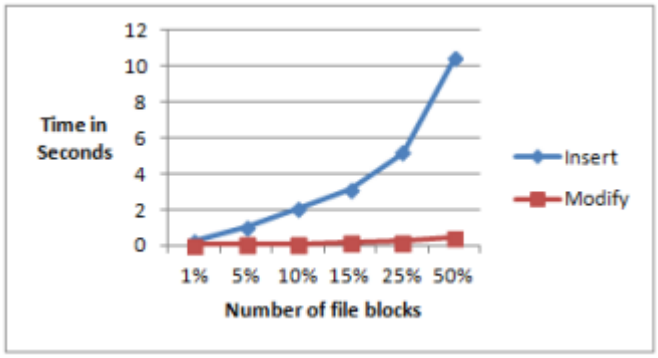
Рисунок 3.4 - Порівняння часу обчислення користувача для кількості реплік налокальні хмарні сервери: а) Час обчислення користувачем сигнатур BLS; б) Час обчислення користувачем RSA-сигнатур.

Наприклад, якщо сервер видаляє 1% файлу даних, верифікатору потрібно перевірити лише c = 460 випадково вибраних блоків файлу, щоб виявити цю неправильну поведінку з ймовірністю більше 99%. Тому в наших експериментах ми використовуємо c = 460 для досягнення високої ймовірності гарантії.

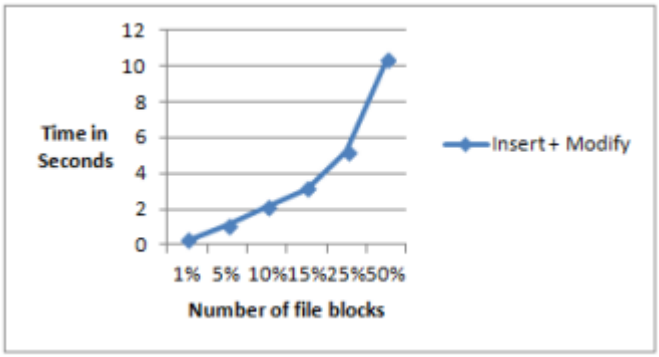
На рис. 3.3.b та 8.b показано час обчислень CSP та користувача на локальних хмарних серверах, коли використовуються RSA-сигнатури. Для кращої безпеки ми використовували N розміром 1024 біти.

З рис. 3.3.a та 3.3.b видно, що час обчислення CSP для підписів BLS та RSA майже однаковий. Рис. 3.4 показує порівняння часу обчислення користувача, коли використовуються підписи BLS та RSA. Оскільки перевірка користувачем підписів RSA включає лише експоненціальні операції та не

передбачає жодних складних операцій сполучення, вона швидша за підписи BLS. Підписи BLS краще використовувати з нашою схемою порівняно з підписами RSA, оскільки підписи BLS коротші. Розмір підписів RSA дорівнює розміру модуля RSA. Оскільки розмір модуля RSA, який ми використовували, становить 1024 біти, підписи RSA також мають розмір 1024 біти, тоді як розмір підписів BLS становить лише 160 бітів. Крім того, конструкція BLS має найкоротший запит та відповідь. Крім того, в нашій схемі блоки даних мають розмір 128 байт, тому розмір тегу також становитиме 128 байт, якщо використовуються підписи RSA. Це збільшить вартість зв'язку. Підписи RSA корисні, якщо розмір блоків даних величезний. [1] використовує підписи RSA, оскільки вони розглядають блоки даних розміром 4 КБ.



а)



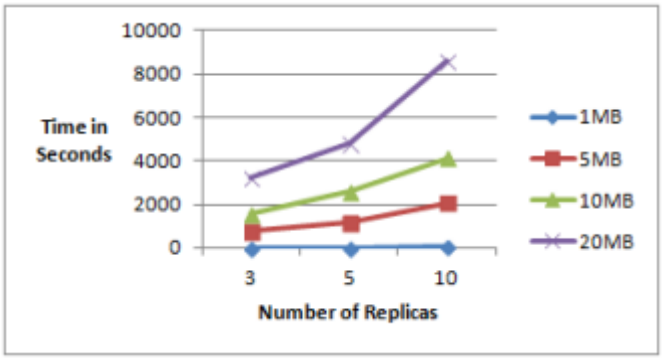
б)

Рисунок 3.5 – Час для операцій вставки та зміни блоків файлів на локальних хмарних серверах: а) Час для операцій вставки та зміни; б) Час для операцій вставки + зміни

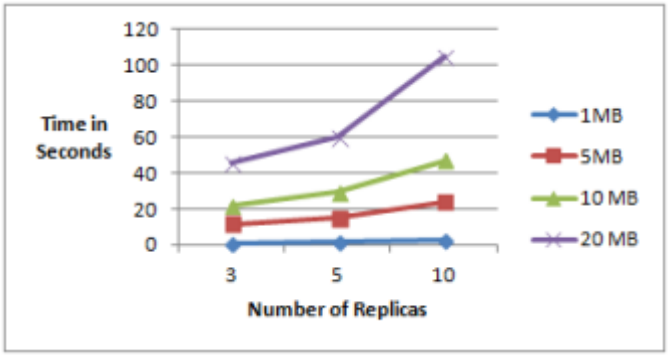
Операції оновлення даних виконуються власником даних одночасно для кількох блоків файлів. Операція оновлення включає операції вставки, зміни та видалення блоків файлів, а також створення нових тегів файлів та їх зберігання в хмарі. Ми провели експерименти з оновлення блоків файлів на файлі розміром 1 МБ з 3 репліками та розміром блоку файлу 128 байт. На рис. 3.5.a показано сумарний час обчислення користувача та CSP для операцій вставки та зміни блоків файлів, що виконуються окремо. Експерименти виконуються шляхом вставки та зміни кількості блоків файлів від 1% до 50%. Наприклад, файл розміром 1 МБ має 8192 блоки файлів. Час обчислення розраховується шляхом вставки від 1% (≈ 82 блоки файлів) до 50% (4096 блоків файлів) нових блоків файлів та зміни від 1% (≈ 82 блоки файлів) до 50% (4096 блоків файлів) від 8192 блоків файлів. На рис. 3.5.b показано час обчислення, коли власник даних виконує операції вставки та зміни певного відсотка блоків файлів. Ми помічаємо, що операції модифікації займають набагато менше часу порівняно з операціями вставки. Час, необхідний для операції модифікації, залежить від часу, необхідного для множення Пайльє двох 256-байтових зашифрованих блоків файлів, тоді як час, необхідний для операції вставки, залежить від часу, необхідного для запису 256-байтових зашифрованих блоків файлів на жорсткий диск. Ми не обчислюємо час, необхідний для операції видалення блоку файлу, оскільки операція видалення не передбачає жодних реальних обчислень користувача та CSP.

Ми також провели наші експерименти на мікро- та великому екземплярі в Амазонії.Хмара EC2. Ми використовували 64-розрядну ОС Ubuntu з 25 ГБ пам'яті для мікро- та великого екземплярів EC2. Мікро-екземпляр має 613 МБ оперативної пам'яті та використовує до 2 обчислювальних блоків EC2, тоді як великий екземпляр має 7,5 ГБ оперативної пам'яті та використовує 4 обчислювальні блоки EC2. На рис. 3.6 показано порівняння часу обчислення CSP у мікро- та великих екземплярах у хмарі EC2. Ми помічаємо, що експерименти виконуються набагато швидше на великому екземплярі EC2 порівняно з мікро-екземпляром. EC2 надає екземпляри з вищою конфігурацією,

ніж великий екземпляр, і власник даних може використовувати їх для підвищення продуктивності.



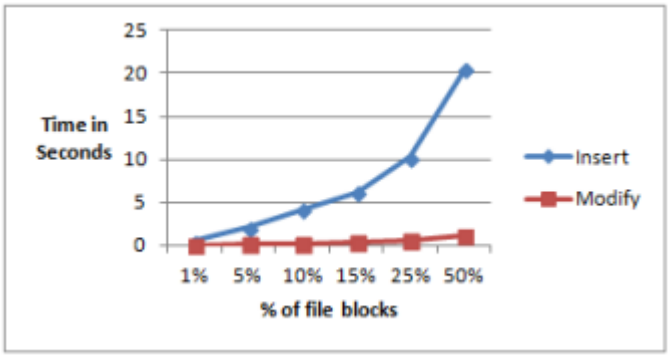
а)



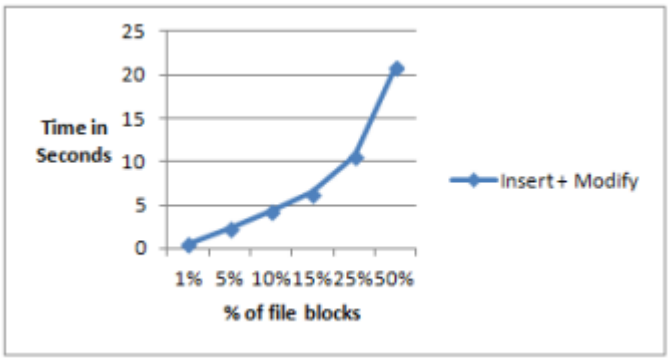
б)

Рисунок 3.6 - Час обчислення CSP для кількості реплік на екземплярах Amazon EC2: а) Час обчислення CSP на мікроекземплярі EC2; б) Час обчислення CSP на великому екземплярі EC2.

Час обчислення користувача становить незалежно від CSP. Ми розрахували це на локальній машині CentOS 6.3, як показано на рис. 3.4.а. На рис. 3.7 та 3.8 показано час обчислення CSP для операцій вставки та зміни на мікро- та великих екземплярах EC2. Виявлено, що продуктивність операції оновлення трохи швидша у великому екземплярі порівняно з мікроекземпляром. Завантаження та вивантаження блоків файлів на мікро- та великі екземпляри EC2 займає майже однаковий час. На рис. 3.9 показано час завантаження та вивантаження блоків файлів на екземпляри EC2.



а)

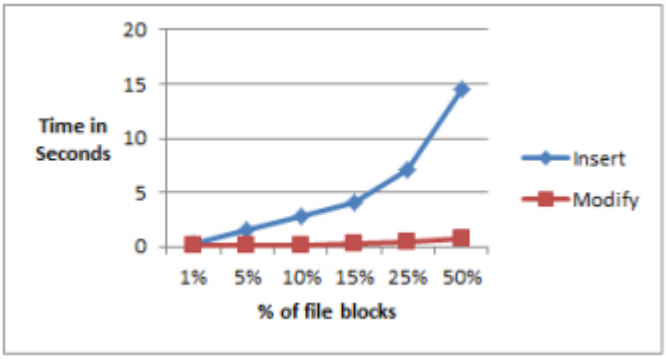


б)

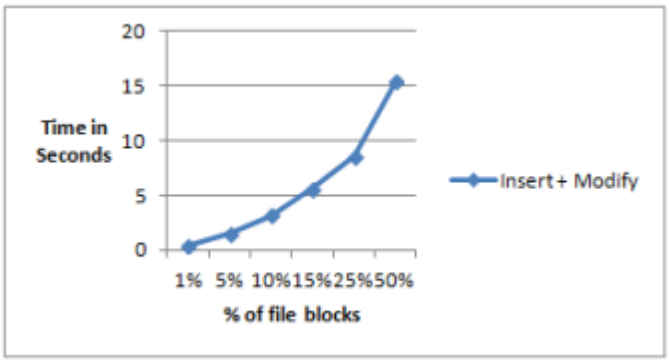
Рисунок 3.7 - Час для операцій вставки та зміни блоку файлу в Amazon EC2 мікроекземпляр: а) Час для операцій вставки та зміни; б) Час для операцій вставки + зміни.

Ми також впровадили базову систему керування версіями файлів «MRFVCS». Власник даних створює таблицю версій файлів для відстеження оновлень даних. Кількість записів у цій таблиці залежить від кількості операцій з динамічними блоками файлів, виконаних над даними. Оновлення файлів зберігаються як дельти у хмарі. Згенеровані дельта-файли мають розмір 128 байт. Щоб отримати певну версію файлу, власник даних надсилає «FileVersionRequest» до хмари з двома параметрами <BN, Ми також впровадили базову систему керування версіями файлів «MRFVCS». Власник даних створює таблицю версій файлів для відстеження оновлень даних. Кількість записів у цій таблиці залежить від кількості операцій з динамічними блоками файлів, виконаних над даними. Оновлення файлів зберігаються як дельти у хмарі. Згенеровані дельта-файли мають розмір 128 байт. Щоб

отримати певну версію файлу, власник даних надсилає «FileVersionRequest» до хмари з двома параметрами <BN, DBN>. Після отримання «FileVersionRequest» хмара виконує алгоритм «FileVersionDeliver». Для FileVersionRequest зі значенням DBN «-» блоки файлів з номером блоку BN безпосередньо доставляються власнику даних і не потребують жодного часу обчислення CSP. Для FileVersionRequest з дійсним значенням DBN хмара шифрує блоки файлів з номером блоку DBN та виконує гомоморфну операцію додавання з блоками файлів з номером блоку BN. Експерименти проводяться на файлі розміром 1 МБ на локальних, мікро- та великих екземплярах Amazon EC2.



а)



б)

Рисунок 3.8 - Час для операцій вставки та зміни блоку файлу в Amazon EC2 великий екземпляр: а) Час для операцій вставки та зміни; б) Час для операцій вставки + зміни.

На рис. 3.10 показано час, який витрачає CSP у різних випадках на виконання алгоритму «FileVersionDeliver», коли власник даних надсилає певну

кількість запитів FileVersionRequests з дійсними значеннями DBN. Ми розглядали FileVersionRequests лише з дійсними значеннями DBN, оскільки для доставки блоків файлів зі значенням DBN «-» не потрібен час обчислення. Кількість FileVersionRequests на рис. 3.0 представлена у відсотках блоків файлів. Наприклад, файл розміром 1 МБ містить 8192 блоки файлів розміром 128 байт. Коли власник даних надсилає 81 запит FileVersionRequests, CSP шифрує 81 дельта-блок (1% від 8192 блоків файлів) та виконує 81 операцію гомоморфного додавання.

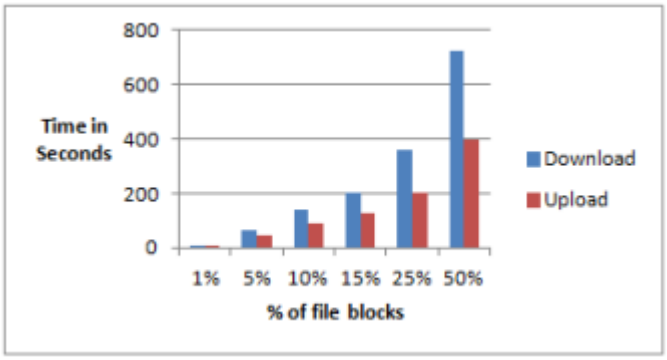


Рисунок 3.9 - Завантаженнята час завантаження до мікро- та великих екземплярів EC2

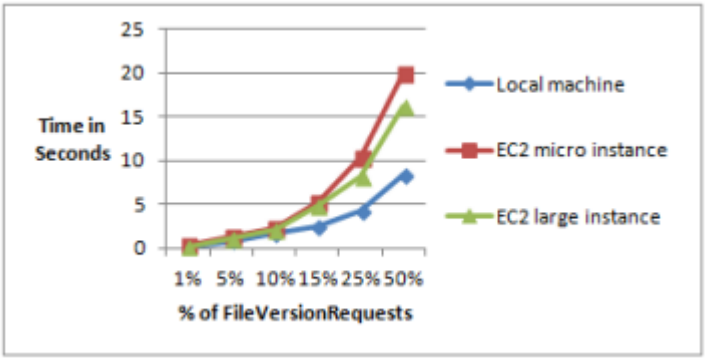


Рисунок 3.10 - Порівняння часу обчислення CSP для алгоритму FileVersionDeliver.

3.7 Висновок до третього розділу

В третьому розділі ми представили схему перевірки цілісності реплікованих даних у хмарному середовищі. Схема під назвою Dynamic Multi-Replica Provable Data Possession scheme (DMR-PDP) періодично перевіряє правильність та повноту кількох копій даних, що зберігаються в хмарі. Наша схема враховує динамічні операції оновлення даних на копіях даних у процесі перевірки. Усі копії даних можна розшифрувати за допомогою одного ключа розшифрування, що забезпечує безперешкодний доступ для всіх авторизованих користувачів даних. Експериментальні результати з використанням локальних та EC2 хмарних екземплярів показують, що ця схема краща за попередню запропоновану схему з точки зору динамічних операцій з даними, які виконуються набагато швидше. Крім того, ми показали, що наша схема добре працює, коли її розширюють для підтримки кількох версій файлів, де в хмарі зберігаються лише дельти, що економить витрати власника даних на зберігання. Ми вважаємо, що ці результати допоможуть власнику даних домовитися з постачальником хмарних послуг про вартість та гарантії продуктивності, зберігаючи при цьому цілісність даних. Крім того, ці результати нададуть хмарі різні стимули для вжиття відповідних заходів, таких як паралельне виконання обчислень, для забезпечення хорошої продуктивності.

РОЗДІЛ 4. Безпека життєдіяльності, основи Охорони праці

4.1 Ергономічні проблеми безпеки життєдіяльності при роботі за комп'ютером

У сучасному світі все більше людей проводять значну частину свого робочого часу за комп'ютером, що призводить до ряду ергономічних проблем, які негативно впливають на безпеку та здоров'я людей. Неправильна позиція тіла, незручне розташування робочого місця, тривале сидіння та напружена робота з клавіатурою та мишею спричиняють м'язове напруження, біль у спині та напругу очей, що викликає дискомфорт та незручності.

Розташування робочого місця відіграє важливу роль у забезпеченні безпеки та здоров'я під час роботи за комп'ютером. Оптимальна висота столу та стільця є ключовим фактором для забезпечення комфорту і підтримки правильної позиції тіла. Стіл належної висоту, щоб лікті могли спокійно розташовуватися на клавіатурі, а стопи - на підлозі. Стілець обладнаний підтримкою для спини та належними регульованими підлокітниками. Клавіатура розташована на рівні ліктів, монітор - належним чином вирівняний перед очима, а миша - в зоні доступу для зап'ястя.

Правильне розташування робочого місця сприяє уникненню незручностей та проблем зі здоров'ям. Ергономічні пристрої підтримки є корисними для забезпечення комфорту та запобігання напрузі м'язів. Використання регульованих підлокітників та підставок для зап'ястя допомагає знизити напругу на м'язах і запобігти незручностям. Оптимальне розташування робочого місця повинне враховувати індивідуальні особливості користувача та забезпечувати комфортні умови для роботи.

Правильна позиція тіла та рухи є ключовими факторами для забезпечення комфорту та запобігання напрузі м'язів та незручностям.

Важливо пам'ятати про правильну позицію спини та уникати підгорблення або надмірного нахилу голови під час тривалої роботи за

комп'ютером. Регулярні перерви для розтяжки та руху є важливими для підтримання здоров'я. Прості фізичні вправи, такі як розтягування шиї, плечей, рук і ніг, допомагають зняти напругу з м'язів та покращити кровообіг. Регулярні перерви дозволяють розслабитися і зберегти енергію для продуктивної роботи. Очі є одними з найбільш вразливих органів під час роботи за комп'ютером. Постійне спрямування погляду на екран призводить до напруження та втому очей.

Для зменшення негативного впливу необхідно використовувати екрани з антиблисковим покриттям, яке знижує відблиск та рефлексію світла. Також важливо налаштовувати яскравість та контрастність екрану для комфортного сприйняття. Щоб зменшити напругу на очі, необхідно робити перерви для відпочинку, фокусуючись на далеких предметах або виконуючи вправи для розслаблення очей.

Використання неправильної клавіатури та миші призводить до м'язового напруження і тунельного синдрому. Важливо використовувати ергономічні клавіатури та миші з додатковою підтримкою для зап'ястя та комфортною формою. Правильна позиція рук та зап'ястя під час роботи з ними має велике значення. Регулярні перерви для розтяжки рук та масажу зап'ястя допомагають уникнути негативних наслідків від тривалого використання клавіатури та миші. Розглядаючи ергономічні проблеми безпеки життєдіяльності при роботі за комп'ютером, варто зазначити, що їх вирішення є ключовим для забезпечення безпеки та здоров'я під час роботи.

Дотримання принципів ергономіки, налагодження робочого місця, правильна позиція тіла та виконання фізичних вправ сприяють покращенню безпеки та створенню здорового робочого середовища. Застосування ергономічних пристроїв підтримки, таких як регульовані підлокітники та підставки для зап'ястя, також сприяє комфорту та попередженню незручностей. Загальною метою є створення безпечного та здорового робочого середовища для людей, які працюють за комп'ютером та виконують дослідження захищеності веб сервісу електронного навчання Atutor, що забезпечує

збереження здоров'я та підвищення продуктивності працівників, сприяє запобіганню травмам та ергономічним проблемам, а також сприяє загальному комфорту та задоволенню від роботи.

Додатково, важливо зазначити, що належна освітленість приміщення також є важливим фактором, який впливає на комфорт та здоров'я під час роботи за комп'ютером. Потрібно забезпечити достатнє природне або штучне освітлення, яке не перевантажує очі. Розміщення робочого місця біля вікна або використання належної освітлювальної техніки допомагає забезпечити оптимальні умови освітлення.

Важливо також уникати відблисків на екрані, розташовуючи монітор під правильним кутом до джерел світла. Безпека та конфіденційність інформації також є важливим аспектом при роботі за комп'ютером. Важливо зберігати конфіденційні дані та захищати їх від несанкціонованого доступу. Використання паролів, шифрування даних та регулярне оновлення програмного забезпечення допомагає забезпечити безпеку інформації.

Крім того, необхідно усвідомлювати потенційні загрози з боку шкідливих програм та фішингових атак і приймати заходи для їх запобігання, такі як використання антивірусного програмного забезпечення та обережне відкривання електронних листів та посилань. Регулярне навчання та свідомість про важливість ергономіки та безпеки при роботі за комп'ютером є важливими.

Працівники повинні мати бути проінформовані про правильні методи роботи, виконання пауз і фізичних вправ, а також процедур безпеки. Організації можуть проводити навчання та інформаційні тренінги, щоб підвищити свідомість та забезпечити правильну поведінку під час роботи за комп'ютером.

Враховуючи всі ці аспекти, створення комфортного, безпечного та здорового робочого середовища є важливим завданням як для працівників, так і для роботодавців. Захист здоров'я та добробуту працівників при роботі за комп'ютером не тільки покращує їхню якість життя, але й сприяє збільшенню

продуктивності та задоволення від роботи, що має позитивний вплив на всю організацію.

4.2 Організація безпечної роботи електроустаткування задіяного при роботі системи електронного навчання

Безпечна робота електроустаткування є ключовим елементом для забезпечення стабільної та безперебійної роботи системи електронного навчання. Це включає в себе правильне проектування, встановлення, експлуатацію та обслуговування електроустаткування. Дотримання стандартів і правил охорони праці допомагає мінімізувати ризики електричних ударів, пожеж та інших небезпек.

Перед встановленням електроустаткування необхідно ретельно оцінити його відповідність вимогам системи електронного навчання. Вибір обладнання повинен базуватися на таких критеріях: надійність та безпека: обладнання має відповідати стандартам якості та безпеки, мати сертифікати відповідності та бути розрахованим на довготривалу експлуатацію; відповідність технічним вимогам: обладнання повинно підтримувати необхідні технічні параметри, такі як напруга, потужність, тип з'єднання тощо;сумісність з іншими компонентами: всі компоненти системи повинні бути сумісні між собою, щоб уникнути збоїв та небезпек при експлуатації/

Під час встановлення необхідно дотримуватися таких заходів безпеки:

правильне заземлення: всі пристрої повинні бути заземлені відповідно до норм, щоб уникнути накопичення статичної електрики та можливості удару струмом; використання захисних пристроїв: встановлення автоматичних вимикачів, пристроїв захисного вимкнення (ПЗВ) та інших захисних засобів є обов’язковим для забезпечення безпеки; професійний монтаж: монтаж повинен виконуватися кваліфікованими спеціалістами з дотриманням всіх норм і правил/

Для забезпечення безпечної експлуатації електроустаткування слід дотримуватися таких рекомендацій: регулярний контроль та технічне обслуговування: обладнання повинно регулярно перевірятися на наявність зношення, перегріву, пошкоджень проводів та інших дефектів. Технічне обслуговування повинно проводитися відповідно до інструкцій виробника; контроль температурного режиму: устаткування не повинно перегріватися. Необхідно забезпечити достатню вентиляцію та уникати розташування пристроїв поблизу джерел тепла; правильне використання: обладнання має використовуватися тільки за призначенням, з дотриманням інструкцій та рекомендацій виробника.

При обслуговуванні та ремонті електроустаткування необхідно дотримуватися таких заходів безпеки: відключення живлення: перед проведенням будь-яких робіт обладнання має бути відключене від джерела живлення; використання засобів індивідуального захисту (ЗІЗ): спеціалісти повинні використовувати відповідні ЗІЗ, такі як діелектричні рукавички, килимки, інструменти з ізоляційними покриттями; дотримання процедур: всі ремонтні роботи повинні проводитися відповідно до технічних інструкцій та нормативних документів;

Управління ризиками та навчання персоналу є невід'ємною частиною забезпечення безпеки електроустаткування: ідентифікація ризиків: постійний аналіз можливих ризиків та їх мінімізація є ключовим аспектом безпеки. Для цього проводяться регулярні оцінки стану обладнання та аналіз умов експлуатації; навчання та інструктажі: персонал повинен проходити регулярні навчання та інструктажі з питань безпечної роботи з електроустаткуванням. Це включає як базові знання, так і спеціальні навички для роботи з конкретними видами обладнання; розробка та впровадження процедур безпеки: необхідно розробити детальні інструкції та процедури з безпеки, які повинні бути доступними для всіх працівників та регулярно оновлюватися. Таким чином, організація безпечної роботи електроустаткування задіяного при роботі системи електронного навчання вимагає комплексного підходу, що включає

вибір надійного обладнання, правильне його встановлення, регулярне технічне обслуговування, навчання персоналу та управління ризиками. Дотримання цих заходів дозволяє забезпечити стабільну та безпечну роботу системи, що є критично важливим для ефективного функціонування освітнього процесу.

4.3 Висновок до четвертого розділу

Ергономічні умови роботи є критично важливими для збереження здоров’я працівників, адже правильна організація робочого місця, регулярні перерви та використання спеціальних пристроїв значно знижують ризики травм і перевтоми.

Безпечна експлуатація електроустаткування потребує комплексного підходу: вибору сертифікованого обладнання, професійного монтажу, регулярного технічного обслуговування та навчання персоналу.

ВИСНОВКИ

Хмарні обчислення забезпечують масштабованість, економічність та гнучкість, але водночас створюють ризики щодо доступності та безпеки даних.

Реплікація даних є ключовим механізмом підвищення надійності та доступності, проте виникає проблема перевірки наявності кількох копій та їхньої цілісності.

Основні вимоги до схем перевірки: мінімізація комунікаційних витрат, низька вартість зберігання, можливість відновлення даних та доведена криптографічна безпека.

Було представлено схему перевірки цілісності реплікованих даних у хмарному середовищі. Схема періодично перевіряє правильність та повноту кількох копій даних, що зберігаються в хмарі. Наша схема враховує динамічні операції оновлення даних на копіях даних у процесі перевірки. Усі копії даних можна розшифрувати за допомогою одного ключа розшифрування, що забезпечує безперешкодний доступ для всіх авторизованих користувачів даних. Експериментальні результати з використанням локальних та EC2 хмарних екземплярів показують, що ця схема краща за попередню запропоновану схему з точки зору динамічних операцій з даними, які виконуються набагато швидше. Крім того, ми показали, що наша схема добре працює, коли її розширюють для підтримки кількох версій файлів, де в хмарі зберігаються лише дельти, що економить витрати власника даних на зберігання. Ми вважаємо, що ці результати допоможуть власнику даних домовитися з постачальником хмарних послуг про вартість та гарантії продуктивності, зберігаючи при цьому цілісність даних. Крім того, ці результати нададуть хмарі різні стимули для вжиття відповідних заходів, таких як паралельне виконання обчислень, для забезпечення хорошої продуктивності.

ПЕРЕЛІК ДЖЕРЕЛ

1 Kurtmola R., Khan O., Burns R., Ateniese G. MR-PDP: Multiple-Replica Provable Data Possession // Proc. 28th IEEE ICDCS. 2008. P. 411–420.

2 Ateniese G., Burns R., Curtmola R., Herring J., Kissner L., Peterson Z., Song D. Provable Data Possession at Untrusted Stores // Proc. 14th ACM Conf. on Computer and Communications Security (CCS'07). New York, USA, 2007. P. 598–609.

3 Ateniese G., Pietro R.D., Mancini L.V., Tsudik G. Scalable and Efficient Provable Data Possession // Proc. SecureComm'08. New York, USA, 2008. P. 1–10.

4 Deswarte Y., Quisquater J.-J., Sadane A. Remote Integrity Checking // Proc. 6th Working Conf. on Integrity and Internal Control in Information Systems (IICIS). 2003. P. 1–11.

5 Erway C., Küpçü A., Papamanthou C., Tamassia R. Dynamic Provable Data Possession // Proc. 16th ACM Conf. on Computer and Communications Security (CCS'09). New York, USA, 2009. P. 213–222.

6 Filho D.L.G., Barreto P.S.L.M. Demonstrating Data Possession and Uncheatable Data Transfer. Cryptology ePrint Archive, Report 2006/150, 2006.

7 Golle P., Jarecki S., Mironov I. Cryptographic Primitives Enforcing Communication and Storage Complexity // Proc. 6th Int. Conf. on Financial Cryptography (FC'02). Berlin, Heidelberg, 2003. P. 120–135.

8 Hao Z., Zhong S., Yu N. A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability // IEEE Trans. on Knowledge and Data Engineering. Vol. 99. No. PrePrints. 2011.

9 Mykletun E., Narasimha M., Tsudik G. Authentication and Integrity in Outsourced Databases // ACM Trans. on Storage. Vol. 2. No. 2. 2006.

10 Sebé F., Domingo-Ferrer J., Martínez-Ballesté A., Deswarte Y., Quisquater J.-J. Efficient Remote Data Possession Checking in Critical Information

Infrastructures // IEEE Trans. on Knowledge and Data Engineering. Vol. 20. No. 8. 2008.

11 Shah M.A., Baker M., Mogul J.C., Swaminathan R. Auditing to Keep Online Storage Services Honest // Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS'07). Berkeley, CA, USA, 2007. P. 1–6.

12 Shah M.A., Swaminathan R., Baker M. Privacy-Preserving Audit and Extraction of Digital Contents. Cryptology ePrint Archive, Report 2008/186, 2008.

13 Wang C., Wang Q., Ren K., Lou W. Ensuring Data Storage Security in Cloud Computing. Cryptology ePrint Archive, Report 2009/081, 2009. URL: http://eprint.iacr.org (дата звернення: 08.06.2013).

14 Wang Q., Wang C., Li J., Ren K., Lou W. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing // Proc. 14th European Symp. on Research in Computer Security (ESORICS'09). Berlin, Heidelberg, 2009. P. 355–370.

15 Tseng K. Publicly Verifiable Remote Data Integrity // Proc. 10th Int. Conf. on Information and Communications Security (ICICS'08). Berlin, Heidelberg: Springer-Verlag, 2008. P. 419–434.

16 Barsoum A.F., Hasan M.A. On Verifying Dynamic Multiple Data Copies over Cloud Servers. Cryptology ePrint Archive, Report 2011/447, 2011. URL: http://eprint.iacr.org (дата звернення: 08.06.2013).

17 Boneh D., Lynn B., Shacham H. Short Signatures from the Weil Pairing // Proc. 7th Int. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT'01). London, UK, 2001. P. 514–532.

18 Barreto P. S. L. M., Naehrig M. Pairing-friendly elliptic curves of prime order // Proc. SAC 2005. Vol. 3897 of LNCS. Berlin, Heidelberg: Springer-Verlag, 2005. P. 319–331.

19 Mukundan R., Madria S., Linderman M. Replicated data integrity verification in cloud // IEEE Data Engineering Bulletin. 2012.

20 Fryz M., Mlynko B. Determination of the characteristic function of discrete-time conditional linear random process and its application // Scientific Journal of TNTU. 2023. Vol. 109, № 1. P. 16–23.

21 M. Fryz and B. Mlynko, “Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal,” in Proceedings of the 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), 2022, vol. 3309, pp. 77–82. Available: https://ceur-ws.org/Vol-3309/short2.pdf

22 Бабак В.П., Куц Ю.В., Мислович М.В., Фриз М.Є., Щербак Л.М. Об’єктно-орієнтована ідентифікація стохастичних шумових сигналів. Київ: Наукова думка, 2024. 240 с. https://doi.org/10.15407/978-966-00-1883-9.

23 V. Babak, A. Zaporozhets, Y. Kuts, M. Fryz, L. Scherbak. Noise signals: Modelling and Analyses. Cham: Springer Nature Switzerland, 2025. 222 p. DOI: https://doi.org/10.1007/978-3-031-71093-3

24 Бабак В.П., Марченко Б.Г., Фриз М.Є. Теорія ймовірностей, випадкові процеси та математична статистика. – К.: Техніка, 2004. – 288 с.

25 M. Fryz, S. Kharchenko, and L. Scherbak, “Ergodicity and Mixing of Conditional Linear Random Processes in the Problems of Information Signal Modelling and Analysis,” in 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2023, 2023, vol. 3628, pp. 306 – 314.

26 Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals // Technol. Audit Prod. Reserv. 2022. Vol. 3, No 2(65). P. 29–32.

27 Шимчук, Г. В., Назаревич, О. Б., Литвиненко, Я. В., Готович, В. А., Никитюк, В. В., & Боднарчук, І. О. (2025). Грід-системи та технології хмарних обчислень. Навчальний посібник для здобувачів освітнього рівня «магістр» спеціальностей: F3 «Комп’ютерні науки», F6 «Інформаційні системи та технології».

28 Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

29 Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.