Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія перевірки цілісності даних в хмарному середовищі»

Виконав: студент 2-го курсу, групи 2КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Борка М.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. КН
_____ Барабан С.В.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., доц. каф. САІТ
_____ Варчук І.В.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«08.12» 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ

Завідувач кафедри КН
д.т.н., проф. Яровий А.А.

(підпис)

"29" 08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Борці Миколі Юрійовичу

1 Тема роботи: «Інформаційна технологія перевірки цілісності даних в хмарному середовищі».
Керівник роботи: Барабан Сергій Володимирович, к.т.н., доцент.
затверджені наказом вищого навчального закладу «18» 09 2023 року № 247

2 Строк подання студентом роботи 13.11.2023

3 Вихідні дані до роботи: розрядність геш-значень файлів – 256 біт; спосіб захисту файлів – цифровий підпис; кількість файлів для організації захисту – до 100шт; мова програмування - об'єктно-орієнтована.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, Аналіз сучасних технологій перевірки цілісності даних; розробка інформаційної технології перевірки цілісності даних в хмарному середовищі; програмна реалізація інформаційної технології перевірки цілісності даних в хмарному середовищі; економічна частина, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму перевірки цілісності даних в хмарному середовищі; структура інформаційної технології перевірки цілісності даних в хмарному середовищі; приклад роботи програмного засобу перевірки цілісності даних.

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Барабан С.В., к.т.н., доцент каф. КН	29.08.23	01.11.23
4	Ратушняк О. Г., к.т.н, доц. каф. ЕПВМ	24.10.23	24.10.23

7 Дата видачі завдання 29.08.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних технологій перевірки цілісності даних. Постановка задач дослідження	01.09.23-07.09.23	
2	Розробка інформаційної технології перевірки цілісності даних в хмарному середовищі	08.09.23-24.09.23	
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка ефективності	25.09.23-15.10.23	
4	Підготовка економічної частини	16.10.23-20.10.23	
5	Апробація та/або впровадження результатів дослідження	21.10.23-05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	06.11.23-10.11.23	

Студент _____ Борка М.Ю.
(підпис)

Керівник роботи _____ Барабан С.В.
(підпис)

АНОТАЦІЯ

УДК 621.374.411

Борка М.Ю. Інформаційна технологія перевірки цілісності даних в хмарному середовищі. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 97с.

На укр. мові. Бібліогр.: 20 назв; рис.: 19; табл. 12.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології перевірки цілісності даних в хмарному середовищі.

В роботі проведено дослідження основних методів гешування та цифрового підпису файлів, розроблено інформаційну технологію перевірки цілісності даних в хмарному середовищі, здійснено програму реалізацію інформаційної технології перевірки цілісності даних з використанням сучасних програмних засобів і технологій програмування.

Згідно проведених досліджень рівень комерційного потенціалу нової розробки становить 33 бали, що, свідчить про комерційну важливість проведення даних досліджень. Термін окупності становить 1,8 р., що менше 3-х років, а це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Ключові слова: інформаційна технологія, захист файлів, цифровий підпис, гешування, SHA-3-256.

ABSTRACT

Borka M.Y. Information technology for checking data integrity in the cloud environment. Master's thesis on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 97 p.

In Ukrainian speech Bibliography: 20 titles; Fig.: 19; table 12.

This master's thesis is devoted to the development of information technology for data integrity verification in the cloud environment.

In the work, the main methods of hashing and digital signature of files were researched, the information technology of checking the integrity of data in the cloud environment was developed, the program was implemented to implement the information technology of checking the integrity of data using modern software tools and programming technologies.

According to the conducted studies, the level of commercial potential of the new development is 33 points, which indicates the commercial importance of conducting these studies. The payback period is 1.8 years, which is less than 3 years, and this indicates the commercial attractiveness of the scientific and technical development and may encourage a potential investor to finance the implementation of this development and its introduction to the market.

Keywords: information technology, file protection, digital signature, hashing, SHA-3-256.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Технології захисту програмного забезпечення постійно розвиваються та еволюціонують. З появою нових і недосліджених способів захисту програмного забезпечення з'являються і нові методи обходу систем захисту. Через велику кількість незаконно поширеного програмного забезпечення є постійна потреба у створенні нових технологій і методів захисту програмного забезпечення.

За одностайними прогнозами провідних консалтингових компаній світу, швидке вдосконалення та поширення хмарних технологій зараз є одним з тих ключових трендів, що в найближчі 5-8 років помітно вплинуть на глобальний розвиток не лише IT-індустрії, але й бізнесу, фінансів, державного управління, медицини, освіти і багатьох інших сфер людського життя. В умовах випереджаючого розвитку ІКТ і чергового спаду світової економіки технологія, яка, наприклад, дозволяє організаціям та іншим суб'єктам відмовитись від значних витрат на власну IT-інфраструктуру на користь отримання всіх необхідних IT-ресурсів онлайн, розглядається як перспективний та рентабельний модернізаційний вибір, оптимальна інвестиція в майбутнє. Достатньо сказати, що, за підрахунками авторитетної International Data Corporation (IDC), вже у 2017 році до 70 % всіх даних людства зберігатиметься у хмарах. У найрозвиненіших регіонах світу вже прийняті стратегічні рішення та плани дій щодо системного та комплексного розвитку хмарних сервісів, розгорнута відповідна робота.

Одним із найновіших методів захисту програмного забезпечення є використання хмарних технологій. Завдяки хмарним технологіям дані для перевірки містяться не на дисках або у файлах на комп'ютері, а на сервері, що не дає можливість зловмисникам переглядати дані локально.

Актуальність теми магістерського дослідження не викликає сумніву, оскільки на сьогоднішній день є велика кількість різноманітних вірусів, які дописують свій код у різні файли або видають себе за ці самі файли, перезаписуючи їх. У такому випадку для перевірки цілісності та істинності файлу пропонується перевірити геш-значення

файлу за допомогою алгоритму гешування SHA-3. Якщо користувач бажає завантажити файл з інтернету і разом з цим файлом міститься інформація про розмір геш-значення, то після завантаження користувач може звірити геш-значення завантаженого файлу та геш-значення файлу, вказаного в Інтернеті.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення якості зберігання даних в хмарному середовищі.

Для досягнення поставленої мети слід розв'язати такі завдання:

- розглянути та проаналізувати існуючі програмні реалізації забезпечення цілісності;
- розробити структуру та алгоритм роботи інформаційної технології перевірки цілісності даних в хмарному середовищі;
- виконати програмну реалізацію запропонованої інформаційної технології перевірки цілісності даних в хмарному середовищі;
- провести тестування програмного продукту та виконати аналіз отриманих результатів;
- здійснити економічні розрахунки доцільності розробки нової інформаційної технології.

Об'єкт дослідження – процес перевірки цілісності даних в хмарному середовищі.

Предмет дослідження – програмні засоби перевірки цілісності даних в хмарному середовищі.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрових файлів, теорія штучних нейронних мереж для реалізації інформаційної технології перевірки цілісності даних в хмарному середовищі, методи

математичної статистики для обрахунків результатів отриманих за допомогою програмного засобу, програмування на мовах високого рівня.

Наукова новизна одержаних результатів полягає в наступному:

Удосконалено інформаційну технологію перевірки цілісності даних, що відрізняється від відомих використанням хмарних технологій та покращеною інформаційною моделлю гешування файлів, що забезпечує підвищення якості зберігання даних в хмарному середовищі.

Практичне значення одержаних результатів:

1. Розроблено інформаційну модель перевірки цілісності даних в хмарному середовищі.
2. Розроблено програмне забезпечення перевірки цілісності даних в хмарному середовищі.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У опублікованій роботі автору належить: створення структури інформаційної технології перевірки цілісності даних у хмарному середовищі [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» (м. Вінниця, Україна, 2023 р.).

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тезу доповіді на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ПЕРЕВІРКИ ЦІЛІСНОСТІ ДАНИХ

1.1 Стан і перспективи розвитку хмарних технологій у світі

Питання спрощення обігу і зберігання інформації стає дедалі актуальним. А з огляду на значну кількість незручностей під час користування, скажімо, паперовим документообігом, які мають не лише українці, популярності набирають різноманітні технологічні рішення, покликані оптимізувати процеси з обробки і збереження інформації [2].

Чи не найвдалішим рішенням цієї проблеми є використання хмарних технологій – сервер або мережу, на яких зберігаються дані та програми і є доступними за умови підключення до Інтернету. Також завдяки хмарним технологіям можна використовувати програми без установлення, вести значно ефективніше управління, наприклад, підприємством, завдяки тому, що вся управлінська та облікова інформація буде централізованою. Простим прикладом хмарних технологій є сервіси електронної пошти, наприклад, Gmail, коли користувачу потрібно лише підключення до Інтернету, щоб надіслати лист із прикріпленими файлами будь-якого розміру. Безкоштовні хмарні технології надає і компанія Microsoft [3].

Таким чином, наприклад, муніципалітети Данії застосовують Google Apps (пакет хмарних сервісів і додатків для спільної роботи, які компанія Google надає за підпискою) для роботи в школах та муніципалітетах. А Великобританія всю інфраструктуру обробки зберігає в Government Cloud (з англ. «урядова хмара»). Як наслідок, державні витрати в IT-секторі скоротилися на 50%. Також у державних цілях застосовують хмарні технології Сінгапур, Італія, Молдова, Бельгія, Австрія, Словенія та Португалія [4].

За підрахунками експертів «Ініціативи довіри хмарним технологіям», загальний економічний ефект від використання нової системи зберігання інформації у п’яти найбільших європейських економіках становитиме 763 мільярди євро.

До того ж, використання хмарних технологій набуває дедалі більшої популярності й в Україні. Згідно з даними спільного дослідження компаній DeNovo

та IDC, обсяг продажів хмарних послуг Iaas/Saas 2014 року зріс порівняно з 2013 роком майже удвічі — до 8,5 мільйона доларів. Зростання спостерігалося і попереднього року. Тоді ринок хмарних сервісів збільшився у 2,8 разу і склав 5,78 мільйона доларів.

Відтак, нещодавно «ПриватБанк» і Poster запустили першу в Україні хмарну касу для кафе, магазинів, ресторанів з можливістю приймати банківські картки. Використання нової технології, за словами ініціаторів проекту, надасть можливість підприємствам харчування і торгівлі не лише відмовитися від громіздких касових апаратів на користь сучасних планшетів або смартфонів, а й отримувати оплату за допомогою банківських карток.

За даними опитування RightScale 93% компаній так чи інакше використовують хмари (рис. 1.1). При цьому, чим вище рівень розвитку хмарної стратегії, тим краще показники ефективності: приріст виручки, скорочення термінів конфігурування сервісів, зменшення IT-витрат і тощо.

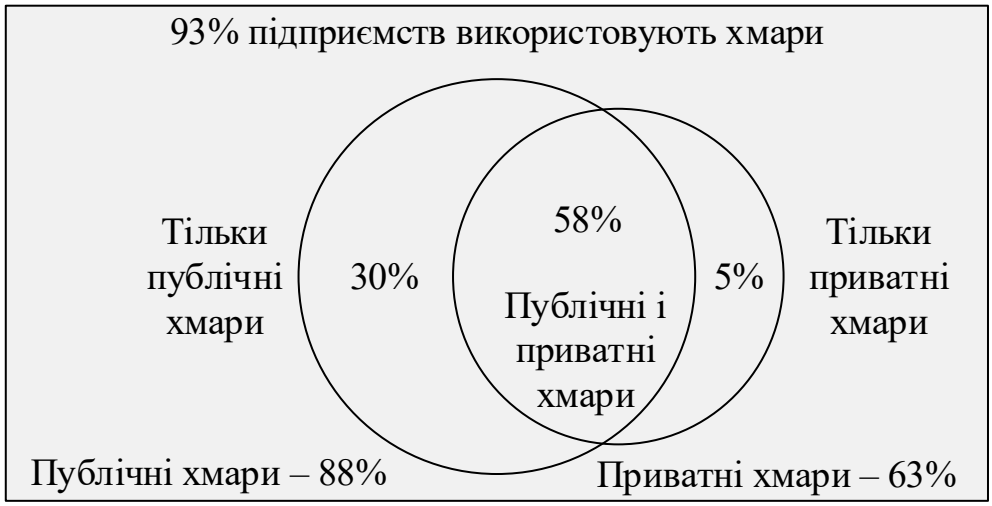
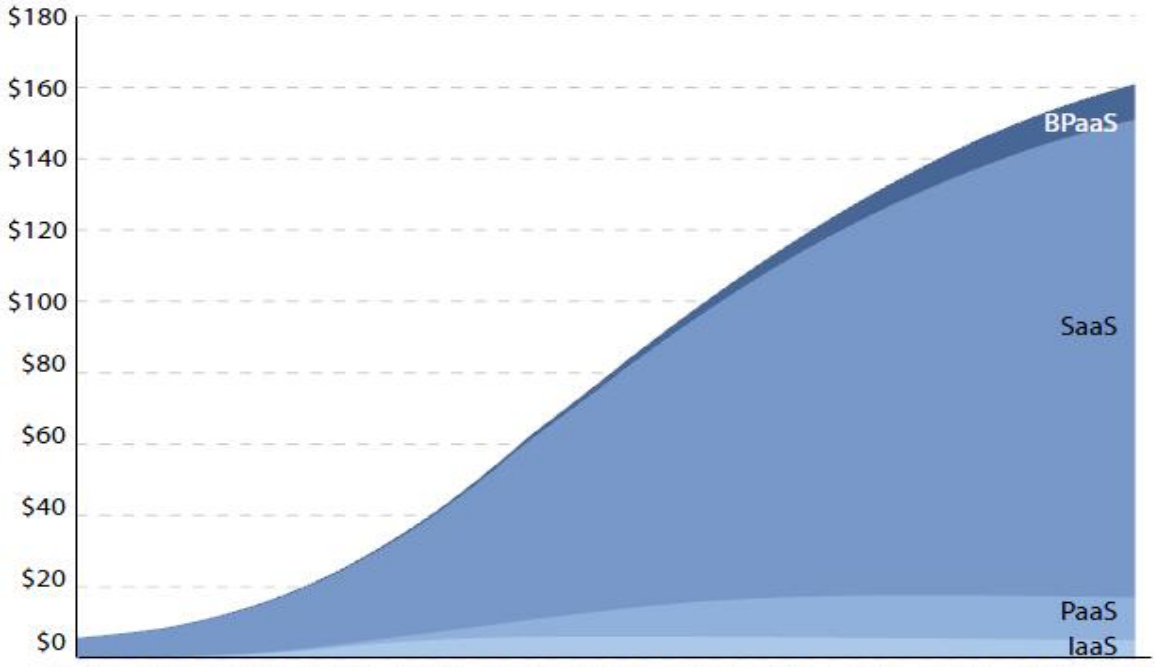


Рисунок 1.1 – Дані щодо використання хмарних технологій

Серед європейських компаній показник використання хмарних сервісів досяг 79% В Україні кількість компаній, які використовують хмарні технології, досягла 48%. Про це йдеться в повідомленні дослідницької компанії IDC. За даними аналітиків, впровадження хмарних сервісів для основних бізнес-додатків за 2016 рік збільшилася з 22% до 42%, для ERP-додатки та бухгалтерського обліку з 22 до 35%, для CRM-рішень з 15 до 31%. У той же час серед європейських компаній показник

використання хмарних сервісів ще вище. У минулому році 79% європейських компаній впроваджували хмарні технології, ще 13% планують зробити це найближчим часом. За прогнозом IDC, до 2018 року інвестиції в IT-інфраструктуру 40% європейських компаній будуть сфокусовані на хмарних рішеннях. Український ринок хмарних сервісів у 2017 році зріс на 20%.

Аналітична компанія Forrester Research опублікувала прогноз розвитку ринку публічних хмарних обчислень до 2020 р. (рис. 1.2) [5]. На рисунку наведено назви сервісів, які існують на ринку публічних хмарних обчислень. Присутні наступні сервіси: IaaS – інфраструктура як сервіс, PaaS – платформа як сервіс, SaaS – програмне забезпечення як сервіс, BPaaS – бізнес платформа як сервіс.



	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
BPaaS ($)	0.15	0.23	0.35	0.53	0.80	1.26	1.95	2.93	4.28	6.00	7.66	9.08	10.02
SaaS ($)	5.56	8.09	13.40	21.21	33.09	47.22	63.19	78.43	92.75	105.49	116.39	125.52	132.57
PaaS ($)	0.05	0.12	0.31	0.82	2.08	4.38	7.39	9.80	11.26	11.94	12.15	12.10	11.91

Рисунок 1.2 – Прогноз зростання обсягу ринку cloud computing за сегментами

Переважна більшість компаній обирає використання хмарних технологій як стратегічний крок у своєму розвитку. Хоча не всі підприємства тут же повністю переходять до хмарної інфраструктури, статистика вказує на тенденцію до розширення використання хмарних середовищ. Сучасні компанії нерідко

впроваджують гібридні рішення, одночасно використовуючи як приватні, так і публічні хмарні платформи в залежності від конкретних завдань та вимог.

Цей підхід дозволяє підприємствам максимально використовувати переваги хмарної технології, забезпечуючи гнучкість, масштабованість та високий рівень безпеки. Приватні хмарні рішення можуть бути використані для чутливих даних та задач, де потрібно більше контролю, в той час як публічні хмарні середовища можуть забезпечувати доступність та обчислювальні ресурси на вимогу.

Цей стратегічний підхід дає компаніям можливість оптимізувати свою інфраструктуру, знижуючи витрати та підвищуючи загальну ефективність бізнес-процесів. У контексті стрімкого технологічного розвитку використання хмарних технологій стає необхідністю для компаній, які бажають залишатися конкурентоспроможними та готовими до майбутніх викликів ринку.

1.2 Вибір моделі хмарних технологій

Хмарні технології перетворюють спосіб, яким бізнеси взаємодіють з персоналом, особливо в контексті підключення віддалених та сезонних працівників. Забезпечуючи онлайн доступ до обчислювальних ресурсів через Інтернет або локальну мережу, хмарні технології стають ефективним інструментом для оптимізації кадрового управління.

Збільшення кількості персоналу більше не повинно бути обмежене фізичним місцезнаходженням, оскільки хмарні обчислення дозволяють підключати співробітників з будь-якого місця, де є доступ до Інтернету. Керівники мають можливість не лише ефективно управляти великим обсягом працівників через хмарний сервіс, але і відключати неактивних користувачів, що сприяє економіці ресурсів та підвищує загальну продуктивність.

Хмарні технології представляють собою широкий спектр концепцій та послуг, з якими пов'язані хмарні сервіси. Однак у центрі уваги стоїть ідея хмарних обчислень, де обчислювальні ресурси надаються користувачеві через мережу, а комп'ютер користувача виступає терміналом для віддаленого доступу до виділених ресурсів.

Інтеграція хмарних технологій у бізнес-процеси створює потужні обчислювальні хмари, де автоматизоване розподілення навантаження між комп'ютерами гарантує максимальне та оптимальне використання обчислювальних ресурсів. Ця ефективна управлінська стратегія дозволяє підприємствам з легкістю масштабувати свої операції, забезпечуючи при цьому виняткову гнучкість у формуванні та керуванні робочими групами працівників, які працюють в різних частинах світу.

Цей підхід до використання хмарних технологій відкриває нові перспективи для глобального бізнесу, де віртуальні команди можуть ефективно співпрацювати навіть з великими географічними відстанями. Забезпечуючи найсучасніші засоби зв'язку та спільної роботи, хмарні технології дозволяють бізнесам створювати інноваційні стратегії та швидко реагувати на зміни в умовах ринку.

Крім того, це сприяє збільшенню продуктивності працівників, оскільки вони можуть мати доступ до необхідних ресурсів та інструментів з будь-якого місця планети. Інтеграція хмарних технологій у бізнес-середовище стає ключовим фактором для досягнення конкурентної переваги в епоху швидких змін та технологічного прогресу.

Хмарні обчислення представляють собою ефективну модель доступу до колективно використовуваних обчислювальних ресурсів у режимі «на вимогу». Користувач має можливість оперативно отримати доступ до широкого спектру налаштовуваних параметрів, таких як мережі, сервери, сховища даних, додатки і/або сервіси, і використовувати їх для вирішення своїх завдань. Однією з ключових переваг цієї моделі є можливість користувача оперативно масштабувати ресурси за потреби та вивільняти їх, зменшуючи взаємодії з постачальником послуги або власні управлінські зусилля [7].

Модель хмарних обчислень включає в себе три моделі обслуговування і чотири моделі розгортання:

- приватна хмара (Private cloud): Ця інфраструктура призначена для використання лише однією організацією або обмеженим колом користувачів. Вона забезпечує високий рівень контролю та конфіденційності даних;

- публічна хмара (Public cloud). Цей тип інфраструктури призначений для вільного використання широким колом користувачів. Ресурси в цьому випадку надаються через інтернет, і користувачі можуть звертатися до них за необхідності;

- громадська хмара (Community cloud). Ця модель передбачає використання інфраструктури спільною групою людей чи організацій, які мають спільні цілі та вимоги. Це дозволяє забезпечити спільні ресурси та взаємодію в специфічному спільнотовому контексті;

- гібридна хмара (Hybrid cloud). Це поєднання двох або більше інфраструктур, які взаємодіють між собою за допомогою технологій передачі даних. Гібридні хмари дозволяють оптимально використовувати ресурси приватних і публічних хмар в залежності від конкретних вимог і завдань.

Ця модель хмарних обчислень визначається різноманіттям своїх режимів обслуговування та можливістю адаптації до різних потреб користувачів, надаючи їм гнучкість та ефективність при використанні обчислювальних ресурсів.

Також існує три моделі обслуговування хмар [7]:

- програмне забезпечення хмари у якості сервісу (англ. Software as a Service, SaaS) – забезпечує мережевий доступ до комерційного програмного забезпечення через браузер або спеціалізовані засоби з будь-якого комп'ютера без необхідності зберігання даних на цьому ж комп'ютері. Дана модель прискорює розробку та впровадження програмного забезпечення, спрощує реалізацію та оновлення;

- платформа хмари у якості сервісу (англ. Platform as a Service, PaaS) – надає можливість розробнику використовувати операційні системи та засоби тестування, створювати додатки на пропонованій інфраструктурі і використовувати обчислювані ресурси хмарної інфраструктури. Головними компонентами PaaS є обчислювальна платформа та сервіс;

- інфраструктура як сервіс (англ. Infrastructure as a Service, IaaS) – інтеграція сервісів, таких як віртуальні сервери, сховища даних і бази даних, в рамках єдиної платформи для розробки і виконання додатків. Користувач може керувати операційними системами, додатками та має контроль над мережевими компонентами.

Класифікація моделей розгортання та моделей обслуговування хмар наведено на рис. 1.3.

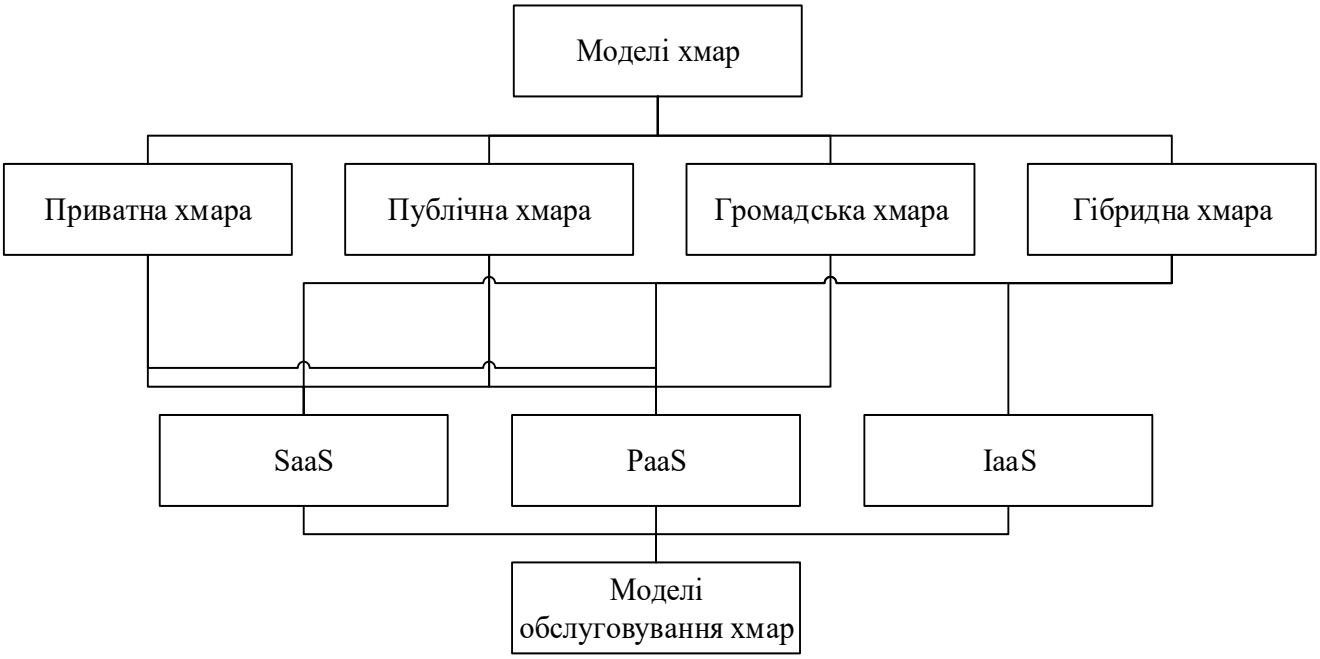


Рисунок 1.3 – Класифікація моделей хмарних технологій

При використанні хмарних технологій споживачі можуть знизити витрати на побудову ЦОД, закупку мережевого та серверного обладнання, апаратного та програмного забезпечення щодо підвищення надійності та працездатності – так як ці витрати бере на себе постачальник хмарних послуг. У разі використання хмарних технологій витрати користувача здійснюються лише на оплату послух хмарних провайдерів [5].

Деякі аналітики, як наприклад, Марк Андерсон, керівник IT-видання Strategic News Service, вважав, що через велику кількість користувачів сервісів, які використовують хмарні обчислення (наприклад, Amazon або Flickr), росте вартість помилок та витоків інформації з подібних ресурсів [6]. Наприклад, у 2009 році сервіс Magnolia (зберігання закладок) втратив абсолютно всі свої дані.

1.3 Проблеми інформаційної безпеки в хмарах

Останнім часом хмарні технології отримали широке розповсюдження та знайшли застосування у різних сферах, таких як синхронізація даних, розподілені обчислення, збереження та передача файлів, інше. Використання хмарних обчислень передбачає, що програмне забезпечення надається користувачеві у вигляді Інтернет-сервісу. Таким чином, термін "хмара" визначає спосіб організації комп'ютерних обчислень, де дані користувача постійно знаходяться на серверах хмарного сервісу.

Під час використання хмарних обчислень, користувач має доступ до власної інформації в хмарі, але зазвичай не має контролю або можливості управління інфраструктурою, де зберігаються й обробляються його дані. Це може включати серверне збереження, обчислення та інші ресурси, які надаються хмарним провайдером.

Однак використання хмарних технологій має свої недоліки. Один з головних недоліків полягає у тому, що приватна інформація користувача, зберігається на серверах хмарного провайдера, тим самим стає доступною для третьої сторони. Це може породжувати питання щодо конфіденційності та безпеки даних, особливо в разі, коли користувач не має повного контролю над інфраструктурою. Крім того, дані можуть бути вразливими під час їх передачі по каналах зв'язку, що також варто враховувати при роботі з хмарними сервісами.

Недоліками хмарних технологій є неповна захищеність та недостатнє забезпечення конфіденційності інформації, зокрема:

- конфіденційність зберігання даних користувача - однією з основних аспектів є гарантування того, що дані користувача зберігаються безпечно і не можуть бути переглянуті або змінені іншими людьми, включаючи операторів хмарних платформ;

- конфіденційність інформації під час перегляду або виконання інших операцій - важливо, щоб під час виконання різних операцій, таких як перегляд чи обробка даних, забезпечувалася конфіденційність інформації, і вона не була доступною для неповноважених користувачів;

- конфіденційність даних під час їх передачі - під час передачі даних між користувачем і хмарною інфраструктурою, а також в межах самої інфраструктури, важливо, щоб дані залишались конфіденційними і захищеними від несанкціонованого доступу чи перехоплення.

Для забезпечення доступу користувачів до своїх даних необхідна процедура однозначної ідентифікації. Користувачі повинні мати можливість самостійно отримувати доступ до своєї інформації та/або давати дозвіл іншим користувачам на авторизований доступ до своїх даних. Важливо враховувати ці аспекти при розгляді конфіденційності та безпеки інформації в хмарних технологіях.

У реальній практиці міграції в хмарові сервіси необхідно здійснювати баланс між централізованими заходами забезпечення інформаційної безпеки, за які відповідає постачальник інфраструктурних послуг, та локальними заходами, які забезпечує сам клієнт. Одним із ключових аспектів є визначення, хто та яким чином контролює ресурси в хмарі, що визначається структурою та політикою самої хмарової організації.

При наданні IaaS-послуг провайдер не може впливати на дії клієнта, пов'язані з установкою та налаштуванням додаткових програмних компонентів з урахуванням вимог безпеки. В разі PaaS-послуг провайдер не може гарантувати правильність розробки програмного забезпечення клієнтами на хмарній платформі. Щодо SaaS-послуг, провайдер не може контролювати правильність організації доступу на клієнтській стороні. Основне завдання провайдера полягає в створенні базового захищеного середовища, де дані різних клієнтів ізольовані один від одного, а також у забезпеченні контролю над діями своїх системних адміністраторів.

Цей підхід сприяє створенню ефективного та безпечного хмарового середовища, де враховуються вимоги різних клієнтів, але при цьому забезпечується загальна ізоляція та контроль захисту інформації. Важливою частиною цього процесу є взаємодія між постачальником та користувачем для досягнення оптимальної рівноваги між забезпеченням безпеки та гнучкістю використання хмарових послуг.

Таким чином, ефективне рішення щодо забезпечення інформаційної безпеки хмарної інфраструкури повинно включати:

- закритий доступ до даних – необхідно забезпечити надійне управління ключами шифрування;

- політики доступу – тільки авторизовані користувачі повинні мати доступ до конфіденційної інформації;

- інтелектуальна система, яка повинна збирати інформацію для аналізу поведінки користувачів і оповіщати у разі виявлення підозрілої активності.

Забезпечення інформаційної безпеки в хмарі – не тривіальна задача, однак, при відповідному під ході можливий ідеальний баланс всіх переваг хмарної моделі і високого рівня захисту, безпеки та доступності даних та інформаційних систем (рис. 1.4).

З боку користувача найбільш надійною гарантією контролю за даними є шифрування [8]. В випадку передачі в систему ключа дешифрування є впевненість, що сторонні користувачі того ж сервісу не отримають доступу до важливих даних. Сам процес передачі ключа шифрування і взаємної автентифікації користувача і серверів хмари також повинен бути побудований на основі криптографії з відкритим ключем.

Однак відсутність закінчених та ефективних рішень, які гарантували би криптографічний захист даних при обробці їх в хмарі, пов’язана з наступними проблемами:

- шифрування, вбудоване в хмарну інфраструктуру, сильно гальмує роботу хмарних додатків;

- дешифрування віртуальної машини перед її запуском може сильно уповільнити як запуск програми, так і її роботу;

- для перенесення віртуального середовища з одного вузла кластера на другий також потрібно виконати цикл шифрування/дешифрування;

- затримки, що виникають при цьому, можуть істотно сповільнити роботу хмарного додатку;

- процедура взаємної автентифікації користувача і хмари також має певні проблеми, як за швидкістю (асиметрична криптографія досить ресурсомістка), так і

за логікою (який саме сервер автентифікувати, якщо їх може бути декілька, і вони передають дані з одного вузла на інший).

Рисунок 1.4 – Вимоги до забезпечення і проблеми інформаційної безпеки хмарної інфраструктури

Проблеми, пов'язані з автентифікацією, стають неодноразовими викликами для безпеки в хмарних середовищах. Однак існують ефективні методи вирішення цих питань, і одним з таких рішень є використання PKI-інфраструктури, протоколу SSL та сертифікатів. Ці засоби дозволяють ефективно керувати процесом автентифікації, забезпечуючи безпеку обміну даними між користувачами та хмарними ресурсами.

При адаптації подібної інфраструктури до хмарних умов слід враховувати особливості цього середовища та впроваджувати відповідні механізми. Важливо розробляти рішення, які враховують гнучкість та динамічність хмарних обчислень.

Одним із важливих аспектів захисту даних в хмарних середовищах є ефективне шифрування. Важливо враховувати, що шифрування всього віртуального середовища може бути неефективним. Замість цього, здебільшого застосовується шифрування конкретних даних, обсяг яких зазвичай значно менший. Це дозволяє забезпечити високий рівень захисту для конфіденційних інформаційних одиниць, не втрачаючи при цьому відправних обчислювальних ресурсів.

Хоча це може впливати на продуктивність системи зберігання, воно забезпечує надійний захист конфіденційності інформації в хмарних обчисленнях [7]. Такий підхід враховує баланс між безпекою та ефективністю, який є важливим у сучасному бізнес-середовищі.

1.4 Висновки до розділу 1

В даному розділі проаналізовано предметну область перевірки цілісності файлів за допомогою хмарних технологій, зазначено актуальність дослідження, визначено основні проблеми при захисті файлів від несанкціонованого доступу. Проаналізовано проблеми безпеки в хмарному середовищі. Обгрунтовано вибір моделі хмарного середовища та використання гешування файлів, як основу для їх безпеки в хмарах.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПЕРЕВІРКИ ЦІЛІСНОСТІ ДАНИХ В ХМАРНОМУ СЕРЕДОВИЩІ

2.1 Використання гешування як елемента безпеки в хмарних технологіях

Із підвищенням інтенсвності використання хмарних обчислень постає необхідність у збільшенні рівня безпеки та захисту інформації в хмарних середовищах. Запити на надійність та конфіденційність стають дедалі більшими, оскільки обсяги генерованих і зберіганих даних у хмарних сховищах зростають щороку, апроксимуючи збільшення на приблизно 60%. Це призводить до актуалізації вимог до захисту цих даних та забезпечення їх постійного доступу. Зростання обсягів інформації робить її економічним активом високої цінності.

Одним з ключових аспектів в сфері хмарних обчислень є проблема автентифікації для доступу до хмарних сховищ. Користувачам хмарних платформ необхідно пройти процедуру автентифікації для доступу до своїх даних. Зазвичай ця процедура включає в себе введення ім'я користувача та пароля, які вказує власник зберігаємої інформації. Однак для забезпечення вищого рівня безпеки і надійності цього методу, важливим є введення додаткового етапу – одноразового пароля. Цей одноразовий пароль має бути введений в спеціально відведене поле для підтвердження раніше введених даних, роблячи процес автентифікації більш надійним та захищеним від несанкціонованого доступу.

Гешування (англ. Hashing) – перетворення масиву вхідних даних довільної довжини у бітовий рядок чітко фіксованої довжини, який виконується заданим алгоритмом. Функція, яка реалізує цей алгоритм і виконує такі перетворення, називається «геш-функцією» а результат такого перетворення називається «геш-значенням» [9].

Гешування застосовується у наступних випадках:

- при побудові асоціативних масивів;
- при пошуку дублікатів в серіях наборів даних;
- при побудові унікальних ідентифікаторів для наборів даних;
- при обчисленні контрольних сум (геш-значень);
- при збереженні паролів в системах захисту у вигляді геш-значення;

- при виробленні електронного підпису.

Геш-значення використовується для перевірки цілісності та істинності даних. Геш-значення, як правило, записується у шістнадцятковому вигляді.

Геш-функції представляють собою спеціальні функції, призначені для "стиснення" довільного повідомлення або набору даних, які, як правило, подаються у двійковому алфавіті, до фіксованої бітової комбінації фіксованої довжини [9]. Геш-функції використовуються в різноманітних областях, включаючи проведення статистичних експериментів, тестування логічних пристроїв, а також в побудові алгоритмів для швидкого пошуку і перевірки цілісності записів в базах даних.

Однією з основних вимог до геш-функцій є рівномірність розподілу їх значень при випадковому виборі значень аргументу. Це означає, що кожне можливе вхідне значення має максимально рівну ймовірність отримання певного вихідного значення, що робить геш-функції ефективними і надійними у великому спектрі застосувань.

Криптографічного геш-функцією називається будь-яка геш-функція, яка має достатню криптостійкість, іншими словами задовольняє ряду вимог, які є специфічними для додатків, які використовують криптографічні методи захисту. У криптографії такі геш-функції застосовують для розв’язання наступних задач:

- побудова системи контролю цілісності даних під час їх передавання або зберігання;

- автентифікація джерела даних.

Виділяють два основних види криптографічних геш-функцій – ключові і безключові. Ключові геш-функції називають кодами автентифікації повідомлень. Вони дають можливість без додаткових засобів гарантувати як правильність джерела даних, так і цілісність даних в системах з довірливими один одному користувачами.

Безключові геш-функції називаються кодами виявлення помилок. Вони дають можливість за допомогою додаткових можливостей (наприклад, шифрування) гарантувати цілісність даних. Ці геш-функції можуть застосовуватися в системах як з довірливими, так і не довірливими один одному користувачами.

До ключових функцій гешування ставляться такі вимоги:

- неможливість фабрикації;

- неможливість модифікації.

Перша вимога означає високу складність підбору повідомлення з правильним значенням згортки. Друге – високу складність підбору для заданого повідомлення з відомим значенням згортки іншого повідомлення з правильним значенням згортки.

До безключової функцій висувають наступні вимоги:

- односпрямованість;

- стійкість до колізій;

- стійкість до знаходження другого прообразу.

Під односпрямованістю розуміють високу складність знаходження повідомлення по заданому значенню згортки. Слід зауважити, що на даний момент немає використовуваних геш-функцій з доведеною односпрямованістю.

Під стійкістю до колізій розуміють складність знаходження пари повідомлень з однаковими значеннями згортки. Зазвичай саме знаходження способу побудови колізій криптоаналітиків служить першим сигналом старіння алгоритму і необхідності його швидкої заміни.

Під стійкістю до знаходження другого прообразу розуміють складність знаходження другого повідомлення з тим же значенням згортки для заданого повідомлення з відомим значенням згортки.

Існує два методи гешування даних: статичне та динамічне. При статичному гешуванні даних геш-значення задається безпосередньо при створенні файлу. При динамічному гешуванні геш-значення має вигляд послідовності бітів, і (англ. progressive digitization) цієї послідовності. Динамічна геш-функція виробляє значення в широкому діапазоні, а саме b-бітові виконавчі цілі числа, де b зазвичай дорівнює 32.

Серед найпопулярніших геш-алгоритмів такі [9]:

Циклічний контроль надмірності (англ. Cyclic redundancy check, CRC) – алгоритм знаходження контрольної суми для перевірки цілісності даних, заснований на певних математичних властивостях циклічного коду. На рис. 2.1 зображено схему формування контрольної суми CRC-8, де D0, .., D8 – регістри [9].

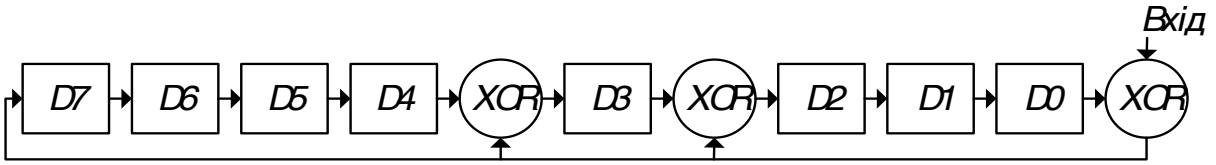


Рисунок 2.1 – Схема формування контрольної суми CRC-8

Алгоритми лінійки SHA (англ. Secure Hash Algorithm) – алгоритм криптографічного гешування. З вхідного повідомлення даний алгоритм генерує 160 бітне геш-значення. Існує 3 версії даного алгоритму: SHA-1, SHA-2 та SHA-3, який в свою чергу, містить в собі наступні алгоритми: SHA-224, SHA-256, SHA-384 та SHA-512. Принципи, покладені в основу SHA-1 аналогічні тим, що використовувались Рональдом Рівестом при проектуванні MD4. На рисунку 2.2 зображено схему одної ітерації алгоритму SHA-2 [10].

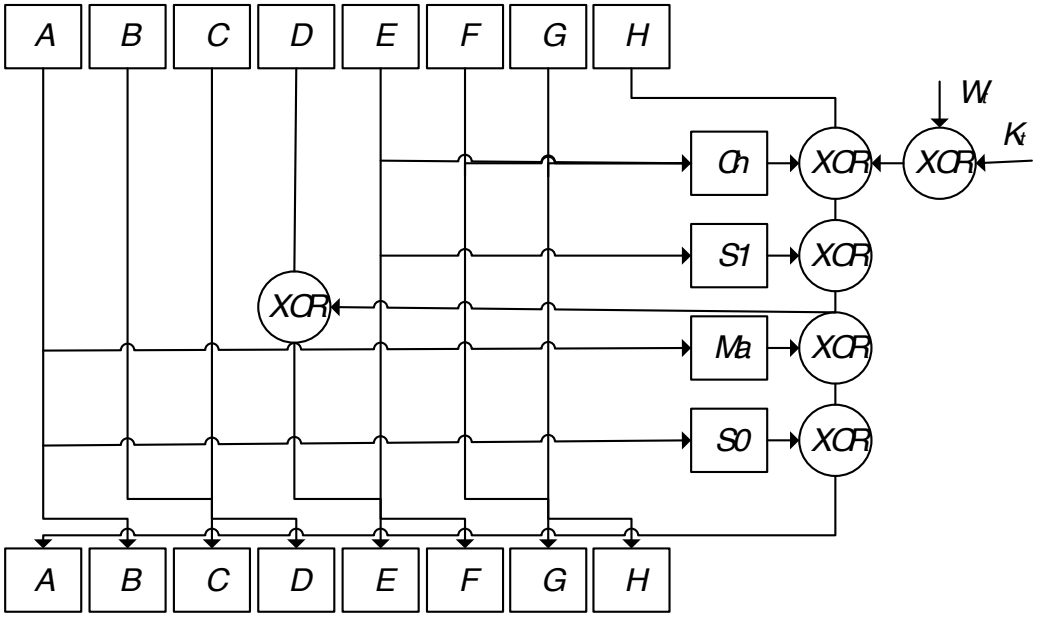


Рисунок 2.2 – Схема одної ітерації алгоритму SHA-2

Алгоритми MD2/4/5/6 (англ. Message Digest) – 128 бітний алгоритм гешування, розроблений професором Масачусетського технологічного університету Рональдом Рівестом у 1991 році [9]. На рис. 2.3 зображено схему роботи алгоритму MD5 [11].

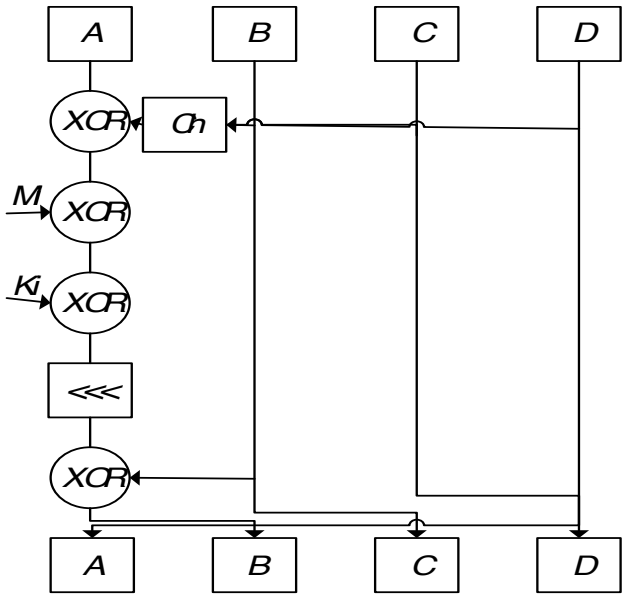


Рисунок 2.3 – Схема роботи алгоритму MD5

Отже, розглянуто декілька алгоритмів гешування: CRC-8, SHA-2 та MD5, а також схеми роботи даних алгоритмів, але, оскільки при розробці програмного засобу буде використано гешування за алгоритм SHA-3, алгоритм цього методу розглянемо детальніше.

Для реалізації програмного засобу у магістерській кваліфікаційній роботі обрано алгоритм гешування SHA-3, оскільки даний алгоритм гешування є надійним і стійким до зламу, а популярні алгоритми SHA-2 та MD5 мають проблеми з надійністю, хоча і працюють швидше за алгоритм SHA-3.

Геш-функції сімейства SHA-3 побудовані на основі структури Меркле – Дамгарда [10].

SHA-3 (Secure Hashing Algorithm-3) представляє собою важливий криптографічний алгоритм гешування, який був розроблений Національним інститутом стандартів і технологій США (NIST) і вперше опублікований у 2015 році. Цей алгоритм входить в сімейство алгоритмів SHA, які використовуються для створення фіксованих хеш-значень з випадкових наборів даних.

Задачею SHA-3 є створення унікального, незмінного та фіксованого хеш-коду для будь-якого вхідного набору даних. Цей хеш-код служить ідентифікатором для визначення цих даних, і його основне використання полягає в забезпеченні цілісності даних та валідації цифрових підписів. Отже, SHA-3 виконує важливу роль у

забезпеченні безпеки та відслідковуваності інформації в різних областях, включаючи кібербезпеку та захист від фальсифікації даних.

Враховуючи постійні виклики в галузі кібербезпеки, SHA-3 виявляється ефективним і надійним інструментом для гешування, забезпечуючи високий рівень стійкості до криптоаналізу та забезпечуючи велику безпеку даних. Опублікований у 2015 році, алгоритм враховує останні досягнення в галузі криптографії, щоб забезпечити високий стандарт захисту від вразливостей та атак з боку несанкціонованих осіб.

Отримане значення порівнюється з дублікатами вихідних даних, витягти які неможливо. Основна сфера застосування алгоритму – використання в різних додатках або сервісах, пов'язаних із захистом інформації, де функція і набула широкого поширення (рис. 2.4).

Рисунок 2.4 – Принцип роботи алгоритму SHA-3

Основний принцип роботи алгоритму SHA-3 включає в себе кілька етапів:

- ініціалізація. Алгоритм SHA-3 використовує певний розмір внутрішнього стану (байтовий буфер), який ініціалізується певним чином перед обробкою кожного блоку вхідних даних;

- поділ даних на блоки. Вхідні дані розділяються на блоки фіксованої довжини перед їхнім подальшим обчисленням;

- додавання даних до буфера. Кожен блок вхідних даних додається до внутрішнього буфера.

- обробка буфера через кубітний мережевий конструктор (Sponge Construction). SHA-3 використовує концепцію «кубітного мережевого конструктора» для обробки даних. Це особлива конструкція, яка використовує фазу вбирання (absorption phase) та фазу видавлювання (squeezing phase). В фазі вбирання, алгоритм поглиблює дані у свій внутрішній буфер. В фазі видавлювання, генерується геш-код;

- отримання геш-коду: Після завершення обробки всіх блоків даних та виконання конструкції Sponge, отримується фінальний геш-код фіксованої довжини, який і вважається вихідним значенням алгоритму SHA-3.

Принциповою особливістю SHA-3 є використання конструкції Sponge, яка визначається матрицею кубітів, де кубіти використовуються для вивчення вхідних даних та генерації хеш-коду. Це забезпечує стійкість алгоритму до різноманітних криптоаналітичних атак та забезпечує високий рівень безпеки в галузі гешування даних.

Далі оригінал тексту після доповнення розбивається на блоки, кожен блок – на 16 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64 ітераціями. На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням геш-функції. Оскільки ініціалізація внутрішнього стану проводиться за результатом обробки попереднього блоку, то немає можливості обробляти блоки паралельно.

2.2 Використання цифрового підпису для перевірки цілісності даних

Електронний підпис - це електронні дані, які в цифровій формі приєднуються до інших електронних даних або логічно пов'язані з ними з метою ідентифікації особи або сутності, яка є власником цих даних [12].

Електронний підпис представляє собою набір електронних даних, які в цифровій формі приєднуються до інших електронних інформаційних блоків або логічно пов'язаних наборів даних. Цей інтегрований електронний знак відіграє важливу роль у сфері ідентифікації та автентифікації в Інтернет-середовищі та електронних комунікаціях.

Основна мета електронного підпису полягає в забезпеченні впізнаваності та вірогідності електронних даних, а також у підтвердженні їх походження та невідмінності під час передачі чи зберігання. Цей механізм використовується для захисту від несанкціонованого доступу, фальсифікації чи модифікації електронної інформації, забезпечуючи надійність та конфіденційність обміну даними в електронному середовищі.

Ідентифікація користувача, що володіє електронним підписом, стає ефективним засобом впізнавання особи чи організації, яка здійснює передачу чи створення електронних даних. Такий метод стає невід'ємною складовою електронної безпеки, дозволяючи впевнено взаємодіяти в електронному середовищі та забезпечувати надійний захист інформації в цифровій формі.

Електронний цифровий підпис (ЕЦП) представляє собою специфічний вид електронного підпису, що виникає в результаті криптографічного перетворення конкретного набору електронних даних. Цей підпис приєднується до вказаного набору даних або логічно з ним поєднується з метою підтвердження цілісності та ідентифікації особи, яка його створила. Процес створення ЕЦП включає в себе використання особистого ключа, а перевірка виконується за допомогою відкритого ключа [13].

Електронний цифровий підпис виявляє широкий спектр застосувань і може бути використаний як юридично визнаний аналог власноручного підпису. Це особливо актуально для юридичних та фізичних осіб, які прагнуть надати електронним документам юридичну силу. ЕЦП гарантує юридичну важливість електронного документа, еквівалентну значенню паперового документа з власноручним підписом повноважної особи та використанням печатки.

Електронний цифровий підпис (ЕЦП) обладнаний всіма ключовими властивостями, які характерні для власноручного підпису [14]:

- засвідчення джерела - ЕЦП підтверджує, що отриманий документ дійсно надійшов від особи, яка ним підписала. Це гарантує визнання авторства та ідентифікацію підписувача;

- цілісність і захист від спотворення - ЕЦП надійно захищає від будь-яких спроб спотворення чи виправлень підписаного документа. Це забезпечує невідмінність та непорушеність інформації, яка міститься у документі;

- неможливість відмови від зобов'язань - особа, яка створила ЕЦП, не може відмовитися від відповідальності за зобов'язання, що виникли в результаті підписання електронного документа. Це забезпечує надійність та важливість підпису в юридичному контексті.

Створення електронного цифрового підпису включає в себе використання криптографічних методів для забезпечення унікальності, цілісності та автентичності даних. Нижче розглянуті основні етапи і методи створення електронного цифрового підпису:

- Створення пари ключів: перший крок полягає в створенні пари криптографічних ключів – приватного і публічного. Приватний ключ залишається секретним і використовується для підписування даних, тоді як публічний ключ розповсюджується для перевірки підпису;

- Вибір геш-функції: важливим етапом є вибір геш-функції, яка використовується для створення унікального «відбитку» вхідних даних. Цей відбиток використовується в подальших операціях для забезпечення цілісності ідентифікованих даних;

- гешування даних: вхідні дані гешуються за допомогою вибраної геш-функції. Отриманий геш-ключ представляє собою фіксовану довжину і унікально ідентифікує конкретні дані.

- підписування приватним ключем: приватний ключ використовується для створення електронного підпису шляхом зашифрування хеш-значення від хешування даних. Цей підпис є унікальним для конкретних даних і приватного ключа, забезпечуючи невідмінність та автентичність;

- публікація публічного ключа: публічний ключ розповсюджується відомим чином, таким чином, щоб будь-хто міг використовувати його для перевірки електронного підпису;

- перевірка підпису: отримувач може перевірити електронний підпис за допомогою публічного ключа. Він розшифровує підпис і порівнює отриманий хеш-значок з тим, який сам отримав в результаті хешування отриманих даних. Якщо хеш-значки співпадають, підпис вважається перевіреним.

Цей процес забезпечує високий рівень безпеки електронних підписів, оскільки приватний ключ залишається тільки у власника електронного підпису, тоді як публічний ключ розповсюджується для загального використання.

Для створення цифрового підпису у даній роботі буде використано утиліту GnuPG.

GnuPG (GNU Privacy Guard) - це безкоштовне і відкрите програмне забезпечення для шифрування та створення цифрових підписів. Воно реалізоване як вільна заміна комерційній програмі PGP (Pretty Good Privacy). GnuPG розроблена за ліцензією GNU General Public License (GPL) і надає можливість шифрування та підписування електронної кореспонденції та файлів.

Основні можливості та функції GnuPG:

- шифрування файлів. GnuPG дозволяє користувачам шифрувати файли, щоб забезпечити їх конфіденційність. Тільки особа, яка має відповідний приватний ключ, зможе розшифрувати дані;

- цифрові підписи - утиліта дозволяє створювати цифрові підписи для файлів або повідомлень. Це забезпечує можливість перевірки автентичності та цілісності даних;

- симетричне та асиметричне шифрування. GnuPG підтримує як симетричне (використовуючи один ключ для шифрування і розшифрування) так і асиметричне (використовуючи пари ключів, приватний та публічний) шифрування;

- підтримка різних алгоритмів. GnuPG підтримує різні криптографічні алгоритми, включаючи RSA, DSA, ElGamal, AES, IDEA, Twofish та інші;

- підтримка сертифікатів OpenPGP: GnuPG використовує стандарт OpenPGP для форматування ключів та обміну даними з іншими програмами, що підтримують цей стандарт;

- інтеграція з різними програмами - утиліта може використовуватися для інтеграції з різними електронними поштовими клієнтами та іншими програмами для забезпечення шифрування та підписування даних;

- підтримка роботи в командному рядку та графічному інтерфейсі. GnuPG може використовуватися через командний рядок або мати графічний інтерфейс для зручності користувача.

GnuPG використовується як в Linux-середовищі, так і в інших операційних системах, і є важливим інструментом для забезпечення конфіденційності та безпеки в електронній комунікації.

GnuPG служить для створення цифрових підписів і шифрування даних. Наприклад, питання ідентифікації листів завжди є актуальною. За допомогою GnuPG можна «вкласти» в лист електронний підпис і, таким чином, одержувач визначить справжність відправника і приналежність йому цього листа. Процес роботи GnuPG наступний: за найскладнішими алгоритмами шифрування прихована проста логіка: використовується пара ключів, один з яких є приватним, а другий – публічним. Файл другого містить публічний ключ і підписи ваших респондентів. Виходить, що після доставки підписаного листа одержувач порівнює публічні ключі, і таким чином ідентифікує відправника.

Особливості GnuPG:

- повноцінна альтернатива PGP;
- не використовує патентовані алгоритми;
- поширюється під ліцензією GPL;
- повна реалізація OpenPGP (RFC4880);
- розшифрування і аутентифікація повідомлень, створених за допомогою PGP5, PGP6 і PGP7;
- підтримка електронного підпису за допомогою алгоритмів ElGamal, DSA, RSA і геш-функцій MD5, SHA-1, RIPE-MD-160 і TIGER;
- робота з асиметричним шифруванням ElGamal і RSA (довжина ключа від 1024 до 4096 біт);

- підтримка блокових алгоритмів симетричного шифрування AES, 3DES, Blowfish, Twofish, CAST5, а також IDEA за допомогою модуля;
- легка реалізація нових алгоритмів за допомогою додаткових модулів;
- підтримка прострочених ключів і підписів;
- вбудована підтримка HKP-серверів ключів.

Як вже було зазначено, GnuPG розроблений відповідно до стандарту OpenPGP, а це значить, що підписи і зашифровані дані, створені іншими програмами, сумісними з OpenPGP, працюватимуть з GnuPG.

Використання різних криптографічних алгоритмів, таких як симетричні шифри, шифрування з відкритим ключем і змішані алгоритми, дозволяє надійно захищати секретні дані і передавати їх. Довжини ключа в 1024 або 2048 біт досить, щоб не турбуватися про злам зашифрованої інформації [15].

Цифровий підпис засвідчує творця і дату створення документа. Якщо документ буде якимось чином змінено, то перевірка цифрового підпису буде невдалою. Цифровий підпис може використовуватися в тих же цілях, що і звичайна підпис. Вихідні тексти GnuPG, наприклад, підписані, і Ви можете переконатися, що вони дійшли до Вас незміненими.

Створення і перевірка підписів відрізняється від шифрування та розшифрування. При підписі документа використовується закритий ключ підписувача, а перевіряється підпис з використанням його відкритого ключа. Наприклад, Alice використовує свій секретний ключ, щоб підписати свою нову статтю в журнал [18]. Редактор, отримавши лист, використовує відкритий ключ Alice, щоб перевірити, що лист дійсно від Alice і не було змінено за час передачі.

2.3 Використання хмарного провайдера AWS для збереження інформації

Amazon Web Services (AWS) - хмарна платформа, яка надає користувачам передплатні сервіси. Перелік послуг включає як інфраструктурні рішення (наприклад, сервери і СХД), і готові платформи (бази даних, середовище розробки та інших.).

AWS з'явилася на ринку в 2006 році і на сьогоднішній день займає друге місце за потужностями, що надаються.

В AWS безпека – головний пріоритет, а безпека у хмарі – це загальна відповідальність AWS та клієнта. Більшість клієнтів, включаючи постачальників фінансових та медичних послуг, а також урядові установи, довіряють AWS обробку самої конфіденційної інформації. Ви можете покращити свою відповідність основним вимогам безпеки, конфіденційності та відповідності за допомогою наших комплексних послуг, будь то через Amazon GuardDuty або AWS Nitro System, базову платформу для наших інстансів EC2. AWS створили сервіс Nitro System для забезпечення конфіденційності робочих навантажень без доступу до оператора. Завдяки Nitro System ніяка система або людина не здатні отримати доступ до серверів EC2, виконати читання даних із пам'яті інстансів EC2 або отримати доступ до будь-яких даних у сховищі інстансів та зашифрованих томів EBS. Крім того, такі сервіси як AWS CloudHSM та AWS Key Management Service дозволяють безпечно створювати ключі шифрування та керувати ними, а AWS Config та AWS CloudTrail надають можливість здійснювати моніторинг та вести журнали, щоб відповідати вимогам та проводити аудити.

Проаналізуємо можливості цих сервісів більш детально.

Сервіс Elastic Compute Cloud (EC2) надає передплатникам віртуальні серверні платформи, системи зберігання даних та балансувальник навантаження. Користувачі можуть вибирати як заздалегідь налаштований сервер з попередньо встановленою операційною системою, так і зібрати його самостійно. Сервіс також дає можливість створювати образи або використовувати власну ОС. Для забезпечення безпеки передплатники можуть розмежовувати доступ до серверів EC2 за IP-адресами.

Вартість послуги оплачується за кожну годину, а деякі опції – на щомісячній основі. Якщо клієнт планує використовувати EC2, рекомендується скористатися функцією резервації. Передплатник оплачує відразу 3 місяці роботи, і підсумкова вартість стає нижчою у півтора рази.

Amazon Simple Storage Service (Amazon S3) – це сервіс зберігання об'єктів, що пропонує найкращі в галузі показники продуктивності, масштабованості, доступності

та безпеки даних. Клієнти будь-якої величини та з будь-якої промислової галузі можуть зберігати та захищати необхідний обсяг даних для практично будь-якого прикладу використання. Наприклад, для озер даних, хмарних додатків та мобільних додатків. Вигідні класи сховища та прості у використанні інструменти адміністрування дозволяють оптимізувати витрати, організувати дані та точно налаштувати обмеження доступу відповідно до потреб бізнесу чи законодавчих вимог.

Сервіс Simple Storage (S3) надає передплатникам розподілений дисковий простір із максимальним об'ємом 5 Тб. S3 зберігає файли в так званих бакетах, які розташовані на різних майданчиках Amazon.

Сервіс веде постійний лог всіх операцій на сховищі та зберігає інформацію у спеціальному бакеті. Це дозволяє клієнтам відразу отримувати докладну інформацію про останні операції. Відмінна риса сервісу - заміна протоколу http на BitTorrent для завантаження файлів.

Передплатники можуть самостійно вибирати безпекові політики, які застосовуються до бакетів. Це дозволяє розмежувати права доступу у користувачів, зробити хмару повністю приватною тощо. Оплата послуги стягується щомісяця. Вартість залежить від обсягу дискового простору, кількості запитів та вихідного трафіку.

На рис. 2.5 показано, як переміщати дані в Amazon S3, керувати збереженими даними в Amazon S3 та аналізувати дані за допомогою інших служб. Три розділи відображаються зліва направо.

Перший розділ містить ілюстрацію бази даних, сервера та документа. Він називається "Переміщення даних". Можна переміщуйвати дані на Amazon S3, де б вони не знаходилися - у хмарі або в додатках, а також локальні дані. Значки поряд вказують на різні типи даних: дані аналітики, файли журналів, дані програм, відео та зображення, а також дані резервного копіювання та архівації.

Другий розділ називається Amazon S3. Він призначений для об'єктного сховища, яке розраховане на зберігання та вилучення будь-яких обсягів даних з будь-якої точки мережі. Можна створити кошик, вкащати регіон, елементи контролю

доступу та параметри управління. І завантажуйвати будь-який обсяг даних. У другому розділі також є значки, що відображають можливості Amazon S3. Серед них: контроль доступу до даних, оптимізація витрат за допомогою класів сховищ, реплікація даних у будь-який регіон, доступ з локальної мережі або VPC, захист даних та прозорість зберігання даних.

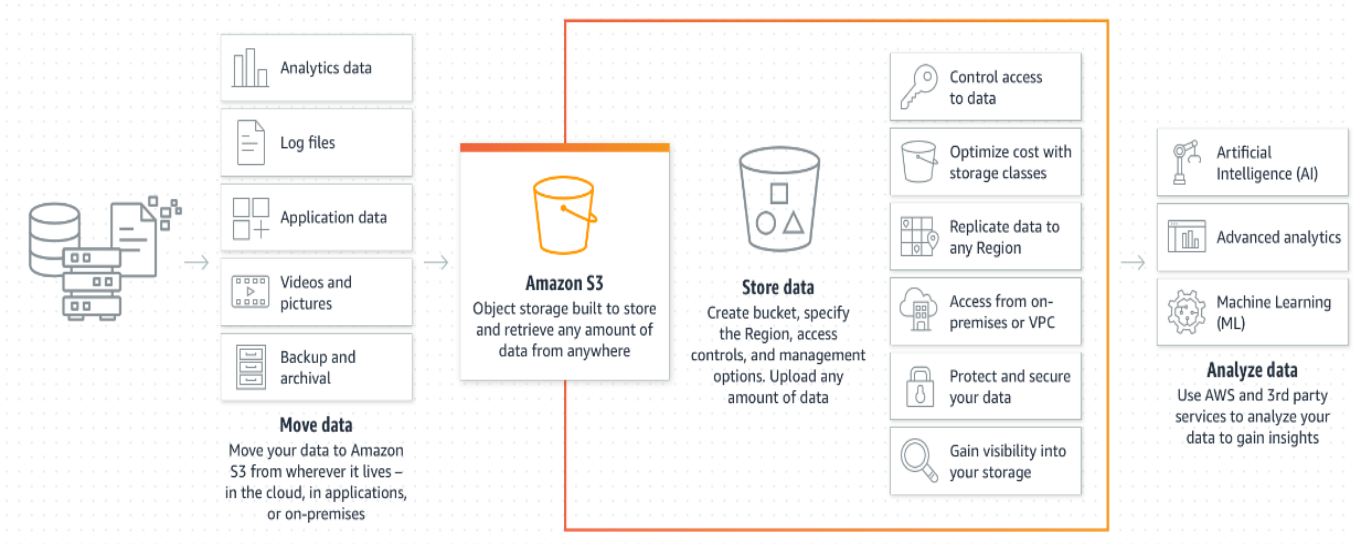


Рисунок 2.5 – Принцип роботи сервісу Amazon S3

Третій розділ називається "Аналіз даних". Можна використовувати AWS та інші сервіси для аналізу даних, щоб отримати корисну аналітичну інформацію. Це можна зробити за допомогою штучного інтелекту (ШІ), розширеної аналітики та машинного навчання (ML).

Сервіс Relational Database Service надає користувачам віртуальну базу даних, розташовану на виділеному сервері. При цьому платформа налаштована та оптимізована під обрану БД. Мінімальний розмір дискового простору під базу даних – 5 Тб.

Доступ налаштовується залежно від побажань користувачів та політик безпеки. Наприклад, замовники можуть дозволити підключення лише з певних IP-адрес (підмереж) або для тих груп безпеки, які вказані в сервісі EC2.

Передплатникам доступні функції миттєвих знімків (SnapShot), реплікація (асинхронна та синхронна), резервне збереження даних в автоматичному режимі та метрокластер.

Оплата проводиться аналогічно до сервісу EC2 — щогодини. Якщо оплачувати три і більше місяців наперед, то підсумкова вартість знижується у півтора рази. Додаткові функції оплачуються окремо.

Сервіс Route 53 дає користувачам можливість піднімати DNS-сервер у хмарі Amazon. Послуга легко інтегрується з іншими AWS.

Оплата стягується за кількість запитів до сервера, але є безкоштовний ліміт.

Amazon Web Services надає користувачам більшість хмарних рішень за доступною ціною. Надійність і стійкість до відмов гарантується на рівні «шості дев'яток» (99,9999%). Клієнт може гнучко налаштувати потрібний список доступних сервісів, вибравши тільки необхідні. Кожна послуга має власний персональний ліміт (Free Tier), що суттєво знижує вартість сервісу, що надається.

Сервіс керування ключами AWS (KMS) забезпечує централізоване керування криптографічними ключами, що використовуються для захисту даних. Цей сервіс інтегрований з іншими сервісами AWS, що спрощує шифрування даних, що зберігаються в цих сервісах, та керування доступом до ключів для їх дешифрування. Завдяки інтеграції AWS KMS з AWS CloudTrail існує можливість перевірити, хто використовував ключі, які саме, коли і для яких ресурсів. AWS KMS дозволяє розробникам без особливих труднощів додавати шифрування або цифрові підписи до коду своїх програм, як безпосередньо, так і через AWS SDK. AWS Шифрування SDK підтримує AWS KMS як постачальника ключів для розробників, які використовують у своїх додатках локальне шифрування та (або) розшифрування даних.

AWS KMS надає централізоване керування життєвим циклом та дозволами для ключів. Ви зможете створювати нові ключі у будь-який зручний момент, а також окремо управляти правами на створення та використання цих ключів. Крім використання ключів, згенерованих AWS KMS, можна також імпортувати ключі з власної інфраструктури керування ключами, використовувати ключі, що зберігаються у кластері AWS CloudHSM, або ключі, що зберігаються за межами AWS

у вашому зовнішньому менеджері ключів. Можна вибрати автоматичну щорічну ротацію кореневих ключів, створених у AWS KMS, без повторного шифрування раніше зашифрованих даних. Сервіс автоматично підтримує доступність старих версій кореневого ключа для розшифрування раніше зашифрованих даних. Керувати кореневими ключами та виконувати аудит їх використання можна за допомогою Консолі керування AWS, AWS SDK або інтерфейсу командного рядка (CLI) AWS.

На рис. 2.6 показано ключові можливості сервісу керування ключами AWS (AWS KMS) та доступні варіанти інтеграції з іншими сервісами AWS.

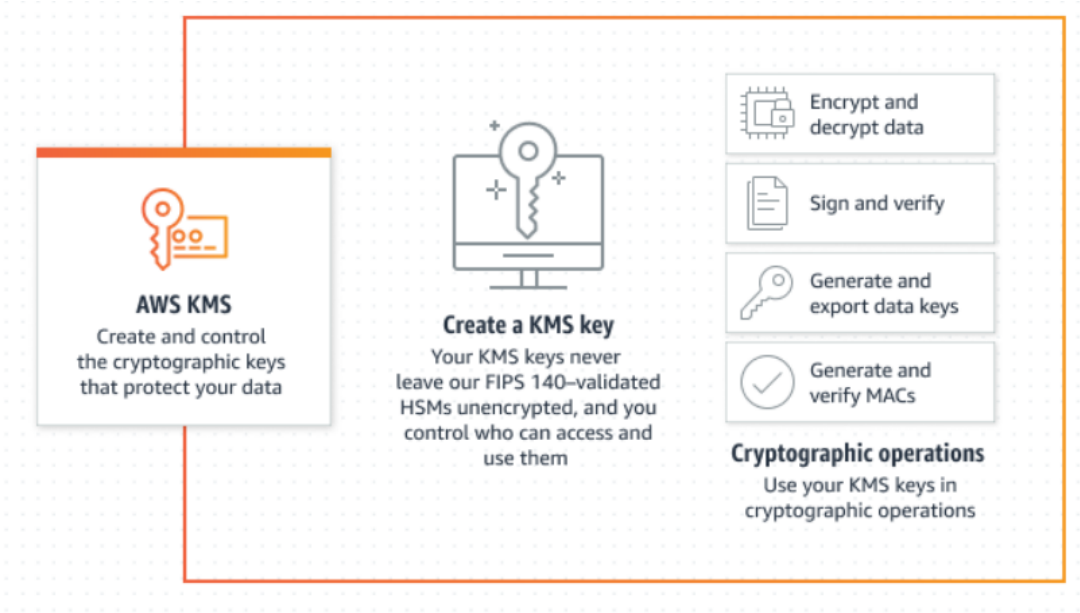


Рисунок 2.6 – Принцип роботи сервісу AWS Key Management Service

Amazon Web Services (AWS) пропонує ряд інструментів та сервісів для керування ключами, які забезпечують безпеку вашої інфраструктури та даних. Одним з ключових сервісів для цієї мети є AWS Key Management Service (KMS).

Основні можливості:

- створення та управління ключами. AWS KMS дозволяє створювати та управляти ключами шифрування. Ви можете використовувати ключі, створені в самому сервісі, або імпортувати власні ключі для використання в AWS;

- шифрування та дешифрування даних. AWS KMS дозволяє легко шифрувати та дешифрувати дані. Це особливо важливо для захисту конфіденційної інформації в системах зберігання та передачі даних;

- інтеграція з іншими AWS Сервісами. AWS KMS інтегрований з іншими сервісами AWS, такими як Amazon S3, Amazon RDS, Amazon EBS та інші. Це дозволяє автоматизувати процеси шифрування та розшифрування для різноманітних ресурсів;

- керування доступом до ключів. AWS KMS дозволяє налаштовувати політики доступу до ключів, визначаючи, хто та як може використовувати ключі. Це забезпечує гнучкість та контроль над використанням криптографічних ключів;

- аудит та журналювання. AWS KMS забезпечує можливості аудиту та журналювання, що дозволяє вам відстежувати використання ключів та виявляти події, пов'язані з їхнім використанням;

- інтеграція з Сервісом AWS CloudTrail. AWS Key Management Service може інтегруватися з AWS CloudTrail для ведення журналу подій та відслідковування всіх дій, пов'язаних із керуванням ключами;

- регіональна та глобальна реплікація ключів. Можнае створювати та управляти ключами в різних регіонах, що дозволяє підтримувати високу доступність та відновлюваність систем.

AWS KMS є важливим компонентом для забезпечення безпеки в хмарному середовищі та використовується для різноманітних сценаріїв, включаючи захист даних, відповідність правилам та нормативам, а також керування ключами для різних сервісів AWS.

2.4 Структура інформаційної технології перевірки цілісності даних у хмарному середовищі

За результатами здійсненого аналізу інформаційних джерел можна зробити висновок, що одним з недоліків хмарних технологій є недостатня захищеність і

недостатнє забезпечення конфіденційності інформації в хмарі. Основними аспектами таких проблем є такі:

- необхідність конфіденційності зберігання даних користувача, оскільки дані не можуть бути переглянуті або змінені іншими людьми;
- необхідність збереження конфіденційність інформації під час перегляду або виконання інших операцій;
- неможливість показу та модифікації даних іншими людьми під час їх виконання (завантаження в системну пам'ять);
- необхідність конфіденційності під час передачі даних.

Для доступу користувачів до своїх даних необхідна процедура однозначної ідентифікації. Користувачі можуть отримати доступ до своєї інформації самі та/або дозволити авторизацію інших користувачів для доступу до своїх даних.

Отже, однією з найважливіших проблем при використанні хмарних технологій є забезпечення цілісності та істинності даних в середовищі, де інформація часто пересувається між різними платформами та системами. Це стає особливо актуальним у зв'язку з розповсюдженням розподілених обчислень та зберігання даних в хмарних сервісах.

Для вирішення цієї проблеми широко використовуються криптографічні алгоритми, зокрема алгоритми гешування. Головна мета використання таких алгоритмів полягає в тому, щоб забезпечити стійкість до змін та невідомість при передачі даних. Процес гешування конвертує вхідні дані будь-якої довжини в фіксований хеш-код фіксованої довжини, що служить унікальним ідентифікатором для цих даних.

Використання алгоритмів гешування не лише дозволяє впевнено визначити цілісність даних, але і забезпечує захист від навмисних змін чи корупції інформації. Додатково, вони застосовуються для валідації цифрових підписів та підтвердження автентичності даних під час їхнього переміщення в хмарних середовищах.

Застосування алгоритмів гешування стає ключовим елементом стратегій кібербезпеки в хмарних обчисленнях, забезпечуючи надійний механізм для

збереження цілісності даних та підвищення загальної безпеки в цьому електронному середовищі [16].

Окрім цього, важливим є унеможливлювання проведення несанкціонованого дослідження вмісту даних користувача, що може бути досягнене шляхом шифрування, оскільки шифрування – це один з найбільш стійких способів захисту інформації [17].

З огляду на це, структура інформаційної технології перевірки цілісності даних в хмарному середовищі, яка реалізується у роботі, представлена на рис. 2.6.

Рисунок 2.6 – Структура інформаційної технології перевірки цілісності даних в хмарному середовищі

2.4 Висновки до розділу 2

У даному розділі було проаналізовано методи гешування та цифрового підпису у хмарних технологіях, розроблено структуру захисту даних у хмарі за рахунок використання алгоритму гешування SHA-3 та цифрового підпису.

У створюваній інформаційній технології файли зберігатимуться на сервері, відомості про геш-значення файлів містяться у текстових файлах, а ключі цифрового підпису зберігаються у недоступному для користувачі місці.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІНОЇ ТЕХНОЛОГІЇ ПЕРЕВІРКИ ЦІЛІСНОСТІ ДАНИХ В ХМАРНОМУ СЕРЕДОВИЩІ

3.1 Принцип роботи програмного забезпечення перевірки цілісності даних

В магістерській кваліфікаційній роботі необхідно розробити інформаційну технологію перевірки цілісноіті даних в хмарному середовищі та здійснити її програмну реалізацію у вигляді додатоку, який дозволить здійснювати перевірку істинності файлів за допомогою цифрового підпису та цілісності даних за допомогою алгоритму гешування SHA-3. Додаток складається з клієнтської частини, яка містить графічний інтерфейс користувача та взаємодіє з сервером за допомогою протоколу SSH. Сервер та клієнт будуть взаємодіяти між собою за допомогою запитів та відповідей (рис. 3.1).

Рисунок 3.1 – Вигляд взаємодії між сервером та комп’ютером

Доступ до сервера можна отримати з різних комп’ютерів, які мають доступ до мережі Інтернет та незаблокований порт 22 для передачі даних на сервер. Перевагою використання хмарної технології є те, що можна одним натисканням миші відновити усі файли, які містяться на сервері, а також отримати доступ до даних з будь-якого комп’ютера.

Сервер очікує на підключення користувача за допомогою додатку з графічним інтерфейсом, після чого виконує дії, які задає йому користувач. На сервері створюється скрін, в якому запускається bash-скрипт, який здійснює обрахування геш-значень файлів та підписує їх за допомогою секретного ключа користувача.

На рис. 3.2 наведено складові, які використовуються на серверній та клієнтській частинах інформаційної технології перевірки цілісності даних в хмарному середовищі.

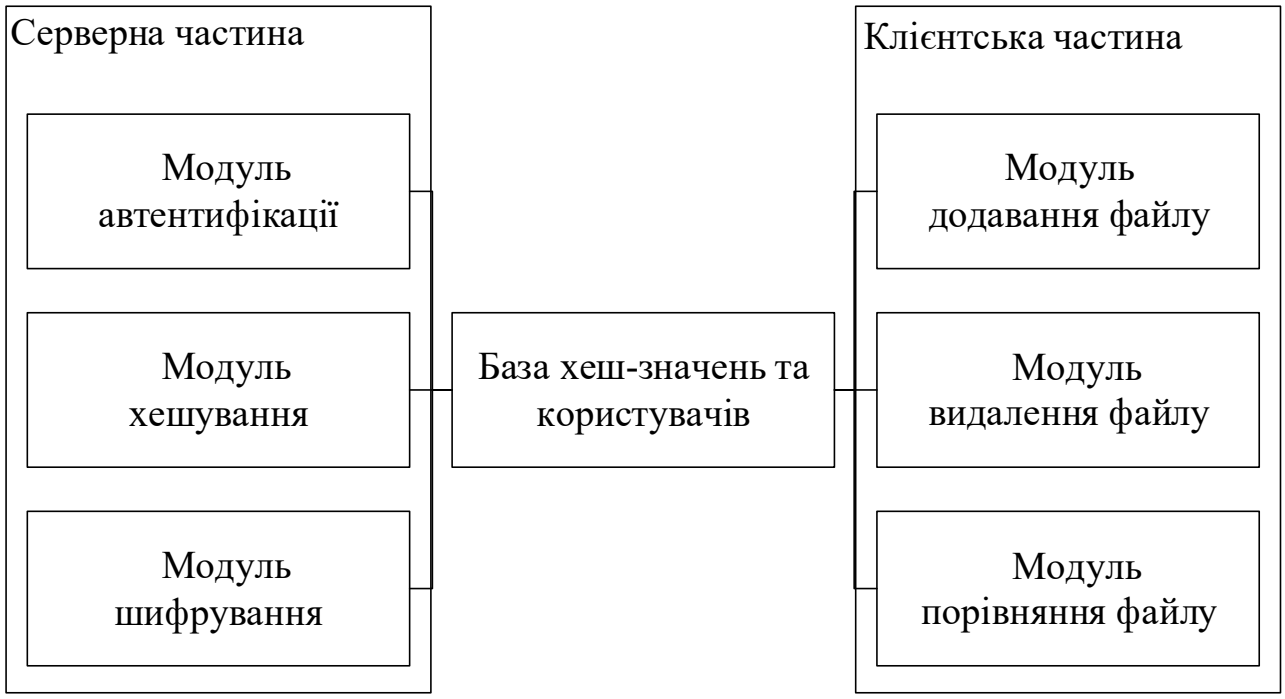


Рисунок 3.2 – Складові серверної та клієнтської частин інформаційної технології перевірки цілісності даних в хмарному середовищі

Принцип роботи програмного додатку наступний: користувач створює обліковий запис, ввівши логін та пароль, які будуть в подальшому використовуватись для автентифікації та підключення до сервера.

Користувач має можливість додавати та відновлювати файли на сервері, а також переглядати список файлів через браузер або графічний інтерфейс додатку.

Якщо користувач обирає опцію додавання файлу на сервер – цей файл копіюється з комп’ютера користувача на сервер за допомогою протоколу SSH, де з використанням bash-скрипту здійснюється обрахування геш-значення цього файлу для подальшої перевірки цілісності файлу, а для перевірки істинності файлу – підписується за допомогою цифрового підпису. Так як для цифрового підпису

використовується приватний та публічний ключі – розшифрувати файл зможе лише довірена особа, яка має публічний ключ.

Для обрахування геш-значення використовується утиліта sha3sum з параметром 256 для створення геш-значення певної фіксованої довжини 256 біт або 32 байти.

Для створення цифрового підпису використовується утиліта GnuPG. Спершу необхідно створити секретний закритий ключ, після чого буде створено публічний ключ. Це надає можливість за допомогою спеціальної команди підписувати файли цифровим ключем задля підтвердження істинності та достовірності цього файлу. З використанням публічного ключа користувач має можливість розшифрувати підписаний файл і упевнитись у його достовірності та істинності.

Якщо геш-значення файлу на комп’ютері не співпадає з геш-значенням файлу в хмарі – користувач має можливість відновити цілісний та істинний файл, який зберігається на сервері.

На рис. 3.3 зображено алгоритм автентифікації користувача.

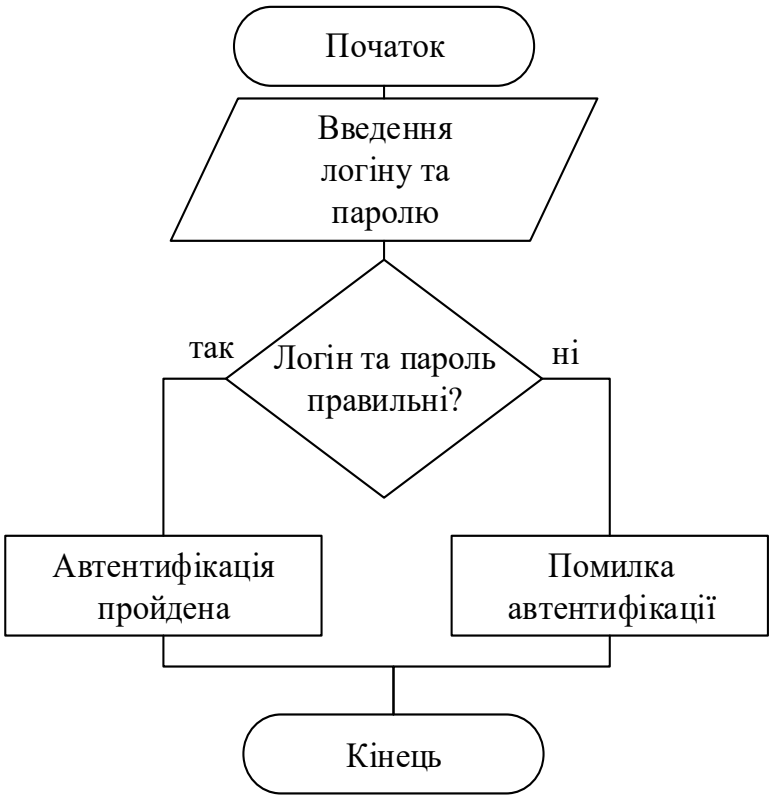


Рисунок 3.3 – Алгоритм автентифікації користувача

3.2 Обґрунтування вибору засобів програмної реалізації інформаційної технології перевірки цілісності даних в хмарному середовищі

Програма буде розроблена як самостійний додаток повного функціонування, який складається з клієнтської частини та взаємодіє з сервером за допомогою використання протоколу SSH.

Протокол SSH (Secure Shell) є криптографічним мережевим протоколом, призначеним для безпечного управління мережевими пристроями та обміну даними через незахищені мережі. Введений як заміна небезпечних протоколів, таких як Telnet, SSH надає шифровану та автентифіковану з'єднаність між двома пристроями, такими як комп'ютери, сервери, маршрутизатори, і т.д.

Основні характеристики та функції протоколу SSH:

- шифрування з'єднання. SSH шифрує весь обмін даними між клієнтом та сервером, що робить його відомим як безпечний тунель. Це захищає інформацію від несанкціонованого доступу під час передачі через небезпечні мережі, такі як Інтернет;

- аутентифікація користувача. SSH використовує різні методи аутентифікації, включаючи паролі, ключі, а також інші методи, такі як аутентифікація на основі клієнтських сертифікатів або інтеграція з системами одноразових паролів;

- сесії та тунелювання. SSH може встановлювати не лише інтерактивні сесії для введення команд в терміналі, але і створювати тунелі для захищеного передавання інших протоколів, таких як FTP чи VNC;

- можливості перенаправлення портів. SSH дозволяє перенаправляти порти між локальним та віддаленим пристроями, що використовується, наприклад, для захищеного доступу до служб, які раніше були недоступні через відкриті мережі;

- сумісність з публічними ключами. Однією з ключових функцій SSH є використання пар ключів (приватний та публічний), що робить процес аутентифікації більш безпечним і зручним;

- підтримка різних версій. SSH має кілька версій, таких як SSH-1 і SSH-2, але зазвичай використовується SSH-2, яка включає більше функцій та поліпшену безпеку.

Використання протоколу SSH стало стандартним для віддаленого адміністрування серверів та інших мережевих пристроїв, а також для безпечного передавання файлів і встановлення захищених мережевих тунелів. Велика частина Інтернет-сервісів та систем використовують SSH для захищеного обміну даними і забезпечення конфіденційності та цілісності інформації.

Реалізація поставленої задачі проводитиметься за допомогою мови програмування Java та bash-скриптів.

Мова Java – об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems. Дата офіційного релізу – травень 1995 року.

У мови Java є багато переваг:

Дизайнери платформи Java вірили у важливість мереж. У Java неймовірно легко працювати з ресурсами по мережі і створити мережеві додатки, використовуючи клієнт/сервер або багаторівневі архітектури. Це означає, що Java програмісти мають серйозні переваги в формуванні мережевої економіки.

Java мова динамічна і розширювальна. Java код організована в модульних об’єктно-орієнтованих одиницях, званими класами. Класи зберігаються в окремих файлах і завантажуються в Java тільки в разі потреби. Це означає, що додаток може вирішити, як він працює, які класи він повинен і може завантажувати їх, коли це потрібно йому. Це також означає, що програма може динамічно розширювати себе шляхом завантаження класів, необхідні для розширення його функціональних можливостей.

Мережа орієнтованих конструкцій платформи Java означає, що Java додаток може динамічно розширити себе шляхом завантаження нових класів по мережі. Додаток, який використовує ці функції перестає бути монолітним блоком коду. Замість цього, він стає взаємодіючим набором незалежних компонентів програмного забезпечення. Таким чином, Java дозволяє нові і потужні функції для проектування і розробки додатків.

Мова Java і платформа Java є єдиною і широко використовуваною мовою програмування, яка є інтернаціональною. У той час як більшість мов програмування використовують 8-бітові символи, які представляють тільки алфавіти англійською та

західноєвропейських мов, Java використовує 16-бітові символи Unicode, які представляють фонетичні алфавіти і ідеографічні набори символів усього світу. Особливості інтернаціоналізації Java-не обмежуються тільки символьним уявленням низького рівня. Особливості пронизують платформу Java, що робить його легшим у написанні багатомовних програм, ніж у будь-якому іншому середовищі.

Остаточною, і, можливо, найбільш важливою причиною, щоб використовувати Java в тому, що програмістам подобається це. Java є елегантною мовою в поєднанні з потужним і добре розробленим набором API. Програмісти насолоджуються програмуванням в Java і, як правило, вражені тим, як швидко вони можуть отримати результати з ним. Дослідження показали, що перехід на Java підвищує ефективність роботи програміста. Оскільки Java є простою і елегантною мовою з добре продуманою, інтуїтивно зрозумілим набором API, програмістам краще писати код з меншою кількістю помилок, ніж для інших платформ, що також скорочує час розробки.

На серверній частині для автоматизації роботи використовуються bash-скрипти.

Bash – одна з найбільш популярних сучасних різновидів командної оболонки UNIX. Особливо популярна в середовищі Linux, де вона часто використовується в якості попередньо встановленої командної оболонки.

Bash-скрипти – це послідовність команд, написаних у мові програмування Bash (Bourne Again SHell), яка є командним інтерпретатором для багатьох операційних систем, зокрема, для багатьох варіантів Linux і Unix. Сценарії Bash використовуються для автоматизації рутинних завдань, написання складних команд або виконання послідовностей операцій.

Розглянемо основні характеристики та можливості bash-скриптів.

Синтаксис. Bash-скрипти пишуться за допомогою текстового редактора, такого як nano або vim, і мають розширення .sh. Скрипти починаються з рядка #!/bin/bash, який вказує операційній системі використовувати Bash для виконання команд:

bashCopy code

#!/bin/bash # Тут пишуться команди скрипту

Змінні. Змінні використовуються для зберігання значень та мають префікс $:

bashCopy code

name="John" echo "Hello, $name!"

Умовні оператори. Bash підтримує умовні оператори, такі як if, else і elif, які дозволяють виконувати частини коду в залежності від умов:

bashCopy code

if [$age -ge 18]; then echo "You are an adult." else echo "You are a minor." fi

Цикли. Bash підтримує цикли, такі як for і while, для повторення виконання певних блоків коду.

Функції. Можна визначати функції для організації коду та викликати їх при потребі.

Аргументи командного рядка. Скрипти можуть приймати аргументи від користувача при виклику.

Пайплайни і перенаправлення. Можна використовувати пайплайни та перенаправлення для обробки виводу команд та введення/виведення файлів.

Bash-скрипти використовуються для автоматизації завдань системного адміністрування, створення скриптів командного рядка та інших завдань, що спрощують та автоматизують взаємодію користувача з операційною системою.

3.3 Розробка серверної та клієнтської частини програмного забезпечення перевірки цілісності даних

Для передачі даних використовується протокол TCP/IP. Для встановлення з’єднання між сервером та додатком використовується протокол SSH.

Серверна частина після запуску очікує підключення користувача. При підключені користувача створюється новий потік, в якому відбувається діалог між користувачем і сервером. Клієнт надсилає запити серверу, а він відправляє відповіді

клієнту, такі як: оновлення даних, додавання файлу та перевірка геш-значення (рис. 3.4).

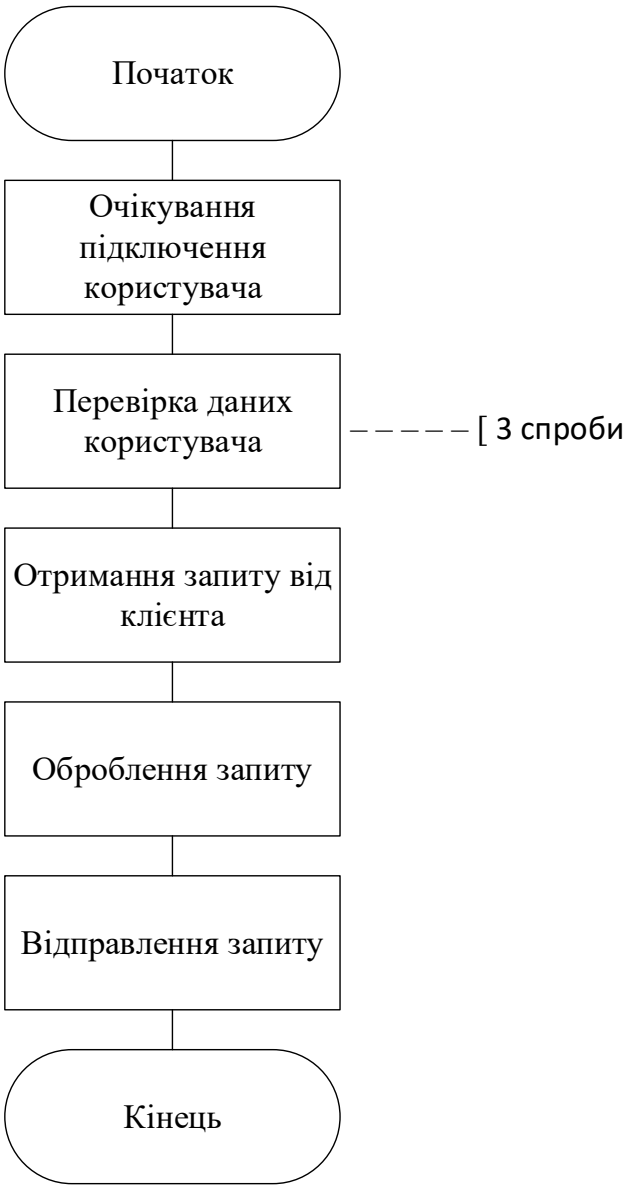


Рисунок 3.4 – Алгоритм роботи серверної частини

Після трьох невдалих спроб авторизації з’єднання з сервером буде завершено.

Користувач через додаток з графічним інтерфейсом відправляє запити серверу за допомогою елементів керування та складається з наступних функцій (рис. 3.5):

Рисунок 3.5 – Алгоритм роботи клієнтської частини

Розглянемо основні класи клієнтської частини (рис. 3.6).

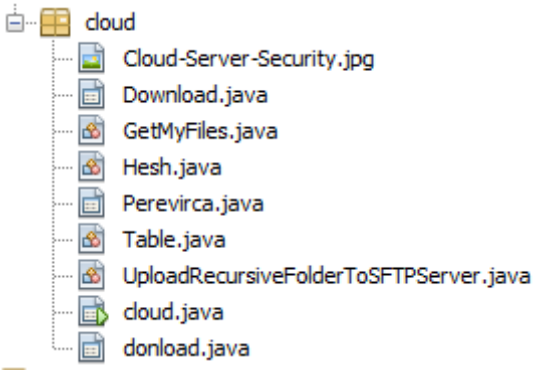


Рисунок 3.6 – Перелік файлів проекту

Клас GetMyFiles відповідає за завантаження файлів з серверу на комп’ютер за допомогою протоколу SSH. Завантаження файлів відбувається у функції get(). Для завантаження файлів з серверу користувачу необхідно ввести логін та пароль для підключення до серверу. При успішному введенні даних для входу – користувач буде підключений до серверу. Далі користувач має можливість перевірити список файлів на сервері. Ці дії здійснюються у класі Table. Це відбувається за рахунок зчитування списку файлів з спеціального файлу на сервері, у який здійснюється запис усіх файлів, що завантажуються на сервер. При вводі назви файлу, який міститься на сервері – користувач завантажить цей фай на комп’ютер в спеціалізовану папку.

Так як обрахування геш-значень здійснюється на сервері, клас Hesh необхідний для того, щоб порівняти значення файлу на сервері з обраним файлом користувачем. Перевірка здійснюється за допомогою функції get, яка зчитує вміст файлів з геш-значенням файлів. Якщо геш-значення файлу на сервері співпадатиме або не співпадатиме з геш-значенням файлу на компютері – користувач отримає відповідне повідомлення.

Клас UploadRecursiveFolderToSFTPServer відповідає за завантаження файлів з комп’ютеру на сервер за допомогою протоколу SSH. Завантаження файлів відбувається у функції recursiveFolderUpload(). Для завантаження файлів на сервер користувачу необхідно ввести логін та пароль для підключення до серверу. При успішному введенні даних для входу – користувач буде підключений до серверу, а файл завантажено у спеціалізовану папку на сервері.

На серверній частині у фоновому режимі працюють два bash-скрипти. Перший аналізує вміст папки, в яку користувач завантажує файли за допомогою класу UploadRecursiveFolderToSFTPServer, де здійснюється обрахування геш-значення файлу за допомогою утиліти sha3sum та здійснюється підпис файлу за допомогою утиліти GnuPG, що дозволить забезпечити істинність файлу. Другий bash-скрипт відповідає за обрахунок геш-значення файлу з комп’ютера, коли користувач бажає порівняти геш-значення файлу з комп’ютера з файлом на сервері.

3.4 Розробка інтерфейсу клієнтської частини

Клієнт має графічний інтерфейс для зручності роботи та взаємодії з сервером (рис. 3.7).

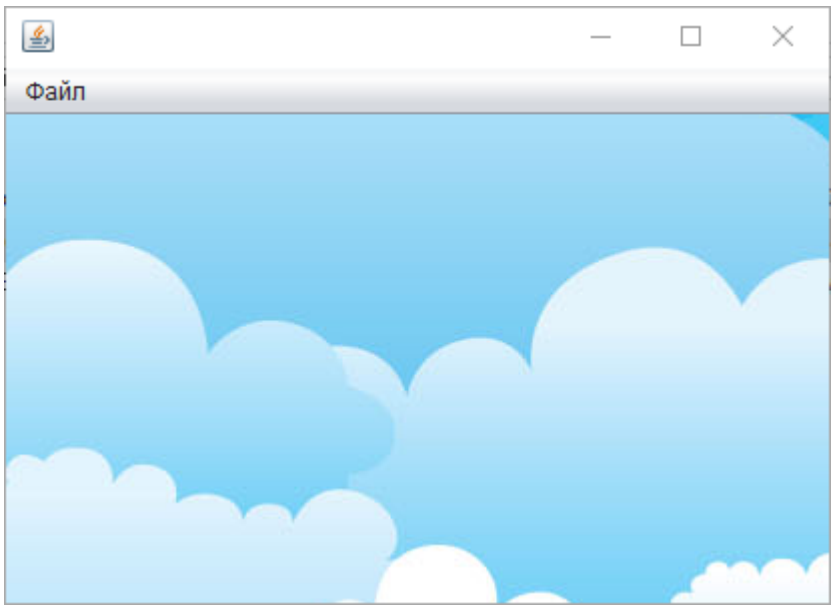


Рисунок 3.7 – Вигляд головного вікна

Користувач надсилає запити серверу за допомогою взаємодії з елементами керування у вікні. Сервер відправляє відповіді клієнту та при необхідності показує список файлів, які були завантажені на сервер для подальшої перевірки на істинність та цілісність. Для авторизації клієнт надсилає клієнту логін та пароль, а при успішній авторизації має можливість переглядати відомості про файли та їхню істинність і цілісність, за допомогою геш-значень. Якщо користувач бажає додати файл на сервер, йому необхідно обрати опцію в меню «Додати файл» (рис. 3.8), обрати файл та після чого файл буде завантажено на сервер через протокол SSH та збережено у спеціальній папці хмари.

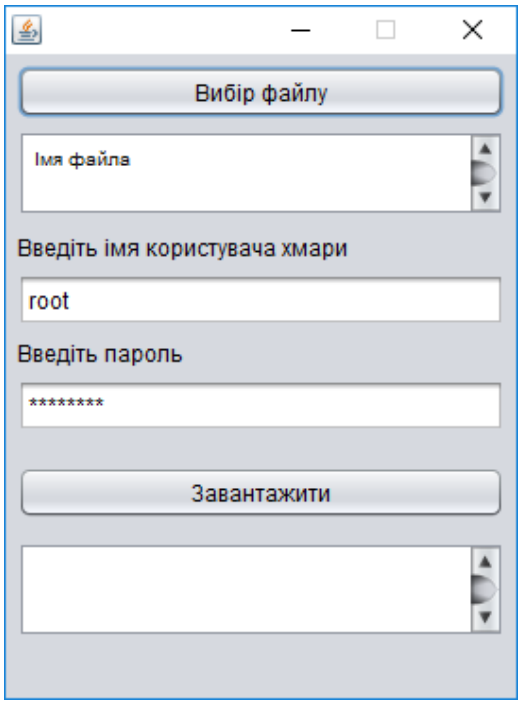


Рисунок 3.8 – Вигляд вікна додавання файлу на сервер

Якщо користувач бажає відновити файл з серверу, йому необхідно обрати опцію в меню «Відновити файл» (рис. 3.9), обрати файл та після чого файл буде завантажено на комп'ютер через протокол SSH та збережено у спеціальній папці на комп'ютері.

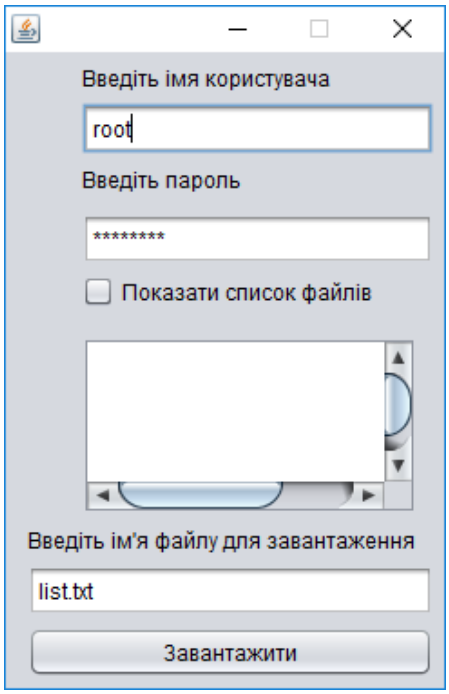


Рисунок 3.9 – Вигляд вікна відновлення файлу з серверу

Якщо користувач бажає перевірити геш-значення файлу на компютері з геш-значенням файлу на сервері, йому необхідно обрати опцію в меню «Перевірити геш-значення» (рис. 3.10), обрати файл та після чого буде здійснено порівняння геш-значень обраних файлів, а результат виведено у діалогове вікно.

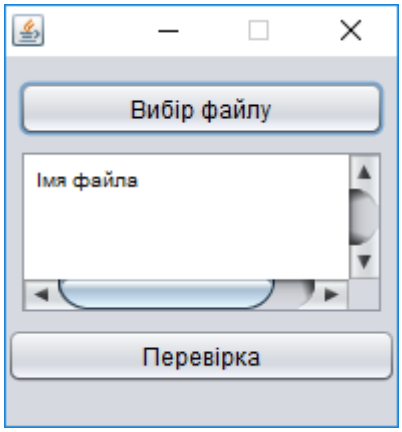


Рисунок 3.10 – Вигляд вікна перевірки геш-значення

Для зручності використання додатку, користувач може взаємодіяти з елементами меню за допомогою “гарячих” клавіш:

– «Додати файл» – Ctrl + A
– «Відновити файл» – Ctrl + R
– «Перевірити геш-значення» – Ctrl + H

3.5 Процес створення секретного ключа для цифрового підпису даних

Для створення цифрового підпису задля перевірки істинності файлів на сервері використовується утиліта GnuPG. Для підпису файлів необхідно згенерувати приватний ключ. Виконуємо команду $ gpg –gen-key (рис 3.11).

```
root@server:~# gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 3
```

Рисунок 3.11 – Виконання команди $ gpg –gen-key

Після виконання цієї команди пропонується обрати алгоритм шифрування. Для створення цифрового підпису достатньо обрати алгоритм DSA (sign only).

Далі необхідно обрати довжину ключа, задавши будь-яку довжину від 1024 до 3072 бітів та вказати термін дії ключа. Для терміну дії ключа є наступні опції:

0 – необмежений термін дії ключа;

<n> – термін дії ключа n днів;

<n>w – термін дії ключа n тижнів;

<n>m – термін дії ключа n місяців;

<n>y – термін дії ключа n років.

Після цього з’являється інформація про термін дії ключа (рис 3.12).

```
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 6m
Key expires at Mon 10 Dec 2018 12:01:55 AM EET
Is this correct? (y/N) y
```

Рисунок 3.12 – Інформація про термін дії ключа

Далі необхідно ввести дані користувача: імя та прізвище, електронну адресу – це відомості, які показують ким файл було підписано (рис. 3.13).

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Mykola Borka
Email address: finage20@gmail.com
Comment:
You selected this USER-ID:
    "Mykola Borka <finage20@gmail.com>"
```

Рисунок 3.13 – Відомості про власника ключа

Наступний крок – введення паролю. Пароль використовується для розшифрування підписаного файлу (рис 3.14).

```
You need a Passphrase to protect your secret key.

gpg: gpg-agent is not available in this session
Enter passphrase:
```

Рисунок 3.14 – Введення паролю для ключа

Після введення паролю починається процедура створення випадкових байтів (рис. 3.15).

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: WARNING: some OpenPGP programs can't handle a DSA key with this digest size
.++++++++++++++++.+++++..++++++++++...++++++++++.....+++++..++++++++++.+++++.+++++.
++++.....................<.+++++...>..+++++.................................
+++++
```

Рисунок 3.15 – Процес генерації ключа

Після завершення процесу генерації випадкових байтів ключ створено. Також з’явиться інформація з параметрами ключа (рис 3.16)

```
gpg: next trustdb check due at 2018-12-09
pub   2048D/9F33FF56 2018-06-12 [expires: 2018-12-09]
      Key fingerprint = 3824 92C7 8E05 5BDD 0D68  5670 038F 568D 9F33 FF56
uid                  Mykola Borka <finage20@gmail.com>

Note that this key cannot be used for encryption.  You may want to use
the command "--edit-key" to generate a subkey for this purpose.
```

Рисунок 3.16 – Параметри ключа

Надалі цей ключ використовується для підпису та розшифрування файлів. При розшифруванні підписаного файлу обов'язковою умовою є введення паролю, який був введений при генерації ключа.

Отже, таким чином було розроблено програмний засіб. Тепер необхідно перевірити його на правильність роботи та відсутність помилок.

3.6 Тестування програмного забезпечення перевірки цілісності даних в хмарному середовищі

Для початку роботи необхідно підключитись до сервера за допомогою протоколу SSH та запустити виконання bash-скриптів у фоновому режимі (рис. 3.17).

```
root@server:/miracle# ./s &
[1] 5914
root@server:/miracle# Scanning folder each 10 seconds
Thu Jun 14 13:28:19

^C
root@server:/miracle# ./z &
[2] 5922
root@server:/miracle# Scanning folder each 10 seconds
Thu Jun 14 13:28:23
```

Рисунок 3.17 – Вікно запуску bash-скриптів у фоновому режимі

Далі користувачеві необхідно запустити додаток з графічним інтерфейсом для зручної взаємодії з серверною частиною (рис. 3.18). У меню вікна містяться 3 кнопки, при взаємодії з якими з'являється нове вікно. Також для зручності взаємодії з додатком, користувач має можливість викликати необхідне йому вікно за допомогою "гарячих" клавіш.

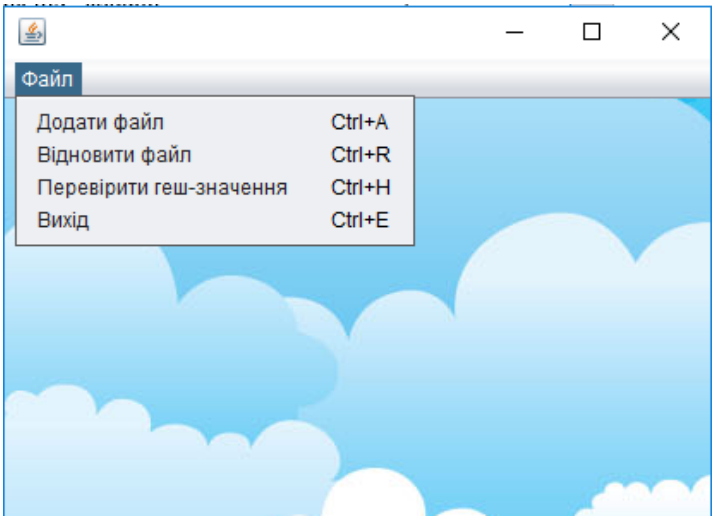


Рисунок 3.18 – Головне вікно додатку

Якщо користувач натисне кнопку в меню «Додати файл» – відкриється нове вікно з можливістю вибору та завантаження файлу на сервер. Обираємо файл на комп'ютері натиснувши кнопку Вибір файлу та завантажуємо його на сервер натиснувши кнопку Завантажити (рис. 3.19).

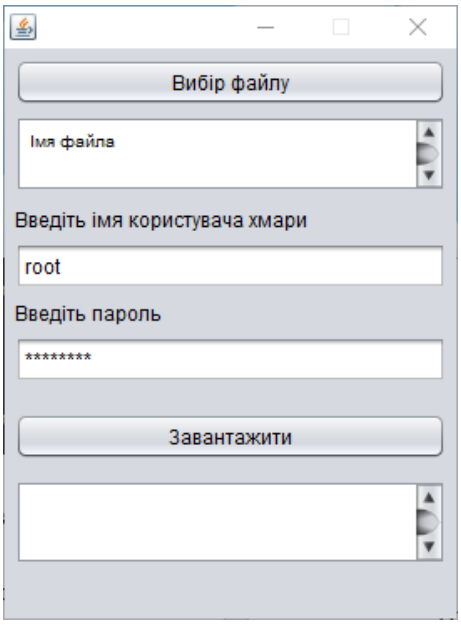


Рисунок 3.19 – Вікно додавання файлу на сервер

Після успішного завантаження файлу на сервер знизу у вікні з'явиться відповідне повідомлення.

Якщо користувач натисне кнопку в меню «Відновити файл» – відкриється нове вікно з можливістю перегляду файлів на сервері та завантаження файлу з серверу на комп’ютер (рис. 3.20).

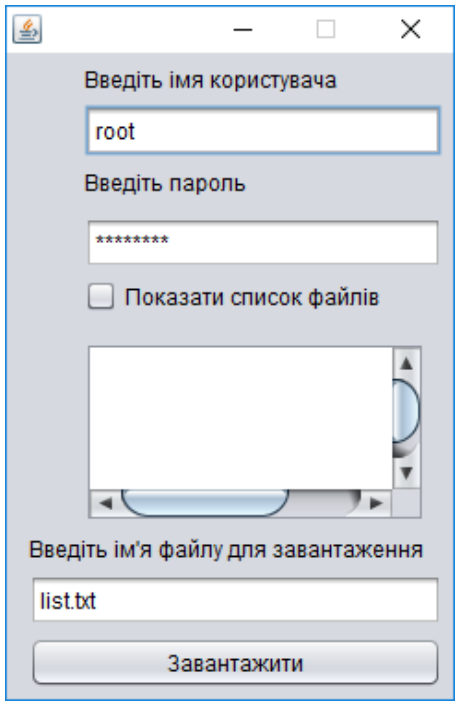


Рисунок 3.20 – Вікно завантаження файлу з сервера

Для цього необхідно ввести його назву та натиснути кнопку Завантажити, після чого файл буде завантажено на комп’ютер у спеціалізовану папку і з’явиться відповідне вікно з повідомленням про успішне завантаження файлу з серверу на комп’ютер (рис 3.21).

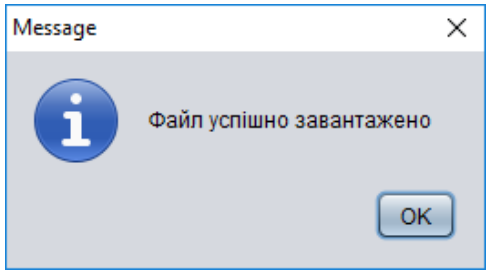


Рисунок 3.21 – Вікно з повідомленням про успішне завантаження файлу

Якщо користувач натисне кнопку в меню «Перевірити геш-значення» – відкриється нове вікно з можливістю перевірки геш-значення обраного файлу на

комп’ютері з файлом на сервері. Обираємо файл на комп’ютері натиснувши кнопку Вибір файлу та натискаємо кнопку Перевірка (рис. 3.22).

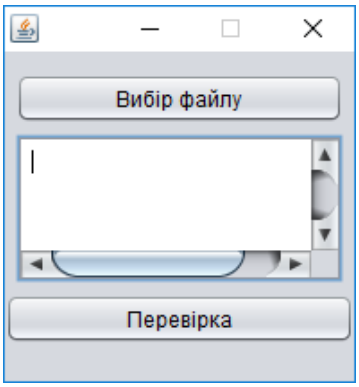


Рисунок 3.22 – Вікно перевірки геш-значення

Якщо геш-значення співпадають – користувач отримає відповідне повідомлення (рис. 3.23)

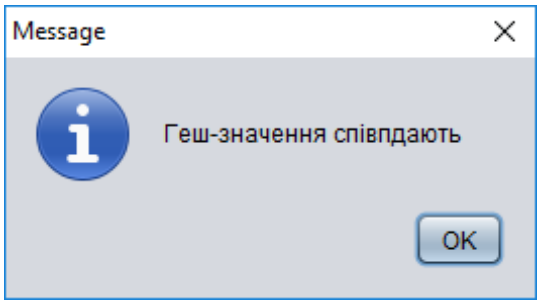


Рисунок 3.23 – Повідомлення про збіг геш-значень обох файлів

Отже, програмний додаток працює вірно, під час тестування помилок не було виявлено.

Також було проведено тестування серед користувачів в галузі ІТ. Для тестування на достовірність правдивості відповідей було обрано десять спеціалістів, що працюють в галузі інформаційних технологій. Оцінка роботи виставлялась від 1 до 10, в залежності від того була дана програма корисною чи ні. Результати тестування наведені в табл. 3.1

Таблиця 3.1 – Результати тестування інформаційної технології перевірки цілісності даних в хмарному середовищі

Спеціаліст	1	2	3	4	5	6	7	8	9	10
Оцінка	10	7	10	9	9	8	7	9	10	8

Проаналізувавши дані, маємо загальну оцінку 87 із 100. Отже, можна дійти висновку що якість роботи інформаційної технології перевірки цілісності даних в хмарному середовищі складає 87%. Це свідчить про досить високий рівень підвищення якості процесу перевірки цілісності даних у хмарному середовищі.

3.7 Висновки до розділу 3

Для реалізації програмного забезпечення використано мову програмування Java та bash-скрипти. Програму розроблено як самостійний додаток повного функціонування, який складається з клієнтської частини та взаємодіє з сервером за допомогою використання протоколу SSH.

Після тестування розробленого програмного забезпечення помилок у роботі не виявлено. З’єднання між клієнтом та сервером працює у нормальному режимі, перевірка геш-значень вірна, шифрування та розшифрування файлів здійснюється без помилок, файли відновлюються та видаляються також без помилок. Якість роботи інформаційної технології перевірки цілісності даних в хмарному середовищі складає 87%. Це свідчить про досить високий рівень підвищення якості процесу перевірки цілісності даних у хмарному середовищі.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки важливо, щоб вона відповідала сучасним вимогам науково-технічного прогресу та враховувала економічні аспекти. Оцінка економічної ефективності результатів науково-дослідної роботи є необхідною частиною цього процесу.

Дослідження, представлене в магістерській роботі, що присвячена розробці та вивченню "Інформаційної технології перевірки цілісності даних в хмарному середовищі", відноситься до науково-технічних робіт, спрямованих на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі самої роботи). Це визначається як комерціалізація науково-технічної розробки, і цей напрямок розглядається як пріоритетний, оскільки результати можуть бути корисними для різних зацікавлених сторін, приносячи економічні вигоди. Однак для успішного втілення цього процесу важливо знайти потенційного інвестора, зацікавленого в реалізації проекту, та переконати його в обґрунтованості вкладання інвестицій у дану розробку.

Для цього визначені наступні етапи виконання робіт:

1. Проведено комерційний аудит науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.

2. Розраховані витрати на реалізацію науково-технічної розробки.

3. Проведено розрахунок економічної ефективності науково-технічної розробки у разі її впровадження та комерціалізації потенційним інвестором, а також обґрунтовано економічну доцільність комерціалізації для інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі"

є розширення функціональних можливостей програмного забезпечення для захисту файлів.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [20].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1.

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів: Мальований Ф.Ю (Senior Systems Engineer, EPAM), Климчук О.Д. (DevOps Engineer, Avenga), Лахман І.В. (Senior Systems Engineer, EPAM).

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Мальований Ф.Ю	Климчук О.Д.	Лахман І.В.
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	4	4	4
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	4	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	2	3	3
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	2	2	2
Сума балів	$СБ_1=31$	$СБ_2=34$	$СБ_3=34$
Середньоарифметична сума балів $СБ_c$	$\overline{СБ}=\frac{\sum_1^3 СБ_i}{3}=\frac{31+34+34}{3}=33$		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [20].

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" становить 33 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки вище середнього, що свідчить про комерційну важливість проведення даних досліджень.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Технологія, яка є результатом магістерської кваліфікаційної роботи може бути інтегрована в програмне забезпечення для перевірки цілісності даних з використанням хмарних технологій. В даному випадку, використані хмарні технології Amazon Web Services (AWS). Використовувати цю технологію можуть компанії або звичайні користувачі.

Прямих аналогів даній технології немає, так як можуть бути застосовані різноманітні способи перевірки цілісності даних та різний спосіб розповсюдження технології (безкоштовно, корпоративна ліцензія, тощо). В якості порівняння, можна використати корпоративне рішення Oracle Data Integrator.

Основними недоліками аналога є: корпоративний спосіб розповсюдження; потреба знань використання цієї технології, які можна здобути після проходження спеціального курсу; складність експлуатації.

У розробці дана проблема вирішується: простотою використання технології для виконання робіт; безкоштовним способом розповсюдження.

4.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему "Інформаційна технологія перевірки цілісності даних в хмарному середовищі", під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та

додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [20]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17000 \cdot 5 / 21 = 3864 \; грн.$$

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17000	772,7	5	3864
Інженер-програміст	12000	545,5	35	19091
Всього				22955

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [20];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6500{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8$ *грн.*

$З_{р1} = 65{,}8 \cdot 2 = 131{,}6$ *грн.*

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1. Планування	2	1	65,8	131,6
2. Впровадження	9	3	88,8	799,5
3. Налагоджувальні	2	5	111,9	223,7
4. Випробувальні	4	3	88,8	355,3
5. Розширення	4	4	59,8	239,3
Всього				1749,5

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (22955+ 1749,5) · 11 / 100% = 2717,44 грн.

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (22955+ 1749,5+2717,44) · 22 / 100% = 6032,72 грн.

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які

придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі".

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \qquad (4.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А4	180	1	180
Ручка	20	1	20
Диск оптичний OPTIMA CD	14	2	28
Flesh-пам'ять 64	400	1	400
Всього			628
З врахуванням коефіцієнта транспортування			690,8

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" відсутні.

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в роботі відсутні.

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \,, \tag{4.7}$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1{,}10…1{,}12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 100 \cdot 1 \cdot 1{,}11 = 110$ грн.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \tag{4.8}$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$А_{обл} = (75000 \cdot 1) / (2 \cdot 12) = 3125$ *грн.*

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Macbook Pro 14	75000	2	1	3125,00
Приміщення лабораторії	230000	20	1	958,33
Всього				4083,33

Витрати на силову електроенергію ($В_е$) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot К_{вni}}{\eta_i}, \quad (4.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 7,5$ грн;

$К_{вni}$ – коефіцієнт, що враховує використання потужності, $К_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,25 \cdot 290,0 \cdot 7,5 \cdot 0,5 / 0,8 = 339,84$ *грн.*

До статті «Службові відрядження» дослідної роботи на тему "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.10)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (22955+ 1749,5) · 20 / 100% = 4940,8 грн.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{H_{ів}}{100\%}, \quad (4.11)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (22955+ 1749,5) · 50 / 100% = 12352,01 грн.

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (4.12)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (22955+ 1749,5) · 100 / 100% = 24704,02 грн.

Витрати на проведення науково-дослідної роботи на тему "Інформаційна технологія перевірки цілісності даних в хмарному середовищі". розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.13)$$

$В_{заг}$=22955+1749,5+2717,44+690,8+110+*3125*+4083,33+*339,84*+4940,8+ +12352,01+ 24704,02 = 80675 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.14)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 80675 / 0,9 = 89638,89 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N– кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 400,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 50,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [20]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.15)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda=0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ=40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta=18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(1·50+400·1700)·0,83·0,4·(1-0,18/100%)=130690,81 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(1·50+400·(1700+1200))·0,83·0,4·(1-0,18/100%)=222978,58 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(1·50+400·(1700+1200+900))·0,83·0,4·(1-0,18/100%)=292163,32 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.16}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 130690,81 /(1+0,18)^1+222978,58/(1+0,18)^2+292163,32/(1+0,18)^3=$$
$$= 433617,64 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.17}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 89638,89 грн.

$PV = k_{інв} \cdot ЗВ$= 2 · 89638,89 = 179277,79 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.14}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 433617,64грн;

PV – теперішня вартість початкових інвестицій179277,79 грн.

$E_{абс} = ПП - PV$= 433617,64- 179277,79 = 254339,85 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.15}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_{в} = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+254339,85 /179277,79)^{1/3} - 1 = 0,57.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.16}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,57 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.17}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,57 = 1,8 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.4 Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі" становить 33 бали, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 1,8 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою "Інформаційна технологія перевірки цілісності даних в хмарному середовищі"

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію перевірки цілісності даних в хмарному середовищі.

В першому розділі проаналізовано предметну область перевірки цілісності файлів за допомогою хмарних технологій, зазначено актуальність дослідження, визначено основні проблеми при захисті файлів від несанкціонованого доступу. Проаналізовано проблеми безпеки в хмарному середовищі. Обгрунтовано вибір моделі хмарного середовища та використання гешування файлів, як основу для їх безпеки в хмарах.

У другому розділі було проаналізовано методи гешування та цифрового підпису у хмарних технологіях, розроблено структуру захисту даних у хмарі за рахунок використання алгоритму гешування SHA-3 та цифрового підпису.

У створюваній інформаційній технології файли зберігатимуться на сервері, відомості про геш-значення файлів містяться у текстових файлах, а ключі цифрового підпису зберігаються у недоступному для користувачі місці.

У трерьому розділі здійснено прогрману реалізацію інформаційної технології перевірки цілісності даних в хмарному середовищі.

Цілісність даних користувача забезпечується за рахунок перевірки геш-значень файлів. При додаванні файлів на сервер створюється геш-значення файлу, яке використовується для подальшої перевірки файлів на комп’ютері. Для реалізації гешування здійснено аналіз методів гешування: CRC-8, SHA-2, MD5 і побудовано їх алгоритми.

Істинність даних користувача забезпечується за рахунок цифрового підпису файлів, для чого розроблено структуру захисту даних у хмарі за рахунок використання алгоритму гешування SHA-3 та цифрового підпису.

Захист від несанкціонованого дослідження досягається за рахунок необхідності авторизації – неавторизовані особи не мають доступу до серверу.

Для реалізації програмного забезпечення використано мову програмування Java та bash-скрипти. Програму розроблено як самостійний додаток повного

функціонування, який складається з клієнтської частини та взаємодіє з сервером за допомогою використання протоколу SSH.

Після тестування розробленого програмного забезпечення помилок у роботі не виявлено. З’єднання між клієнтом та сервером працює у нормальному режимі, перевірка геш-значень вірна, шифрування та розшифрування файлів здійснюється без помилок, файли відновлюються та видаляються також без помилок. Якість роботи інформаційної технології перевірки цілісності даних в хмарному середовищі складає 87%. Це свідчить про досить високий рівень підвищення якості процесу перевірки цілісності даних у хмарному середовищі.

У четвертому розділі було виконано розрахунок загальних витрат на завершення науково-технічної роботи та оформлення її результатів, які складають 89638,89 гривень. Також розраховано період окупності витрат, який склав 1,8 року. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналоги і є високо конкурентоспроможним.

Отже, всі поставленні завдання виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С.В. Барабан, М.Ю. Борка. Структура інформаційної технології перевірки цілісності даних в хмарному середовищі. Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» − [Електронний ресурс]. – https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19691
2. Про інформацію [Електронний ресурс]. – Режим доступу: URL: http://zakon2.rada.gov.ua/laws/show/2657-12/. – Назва з екрану.
3. What is Azure? [Електронний ресурс]. – Режим доступу: URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/. – Назва з екрану.
4. Symantec: Protecting a Cloudier Future. Market Report. November 2012. [Електронний ресурс]. – Режим доступу: URL: http://www.symantec.com/content/ en/us/enterprise/white_papers/esg-protecting-a-cloudier-future.en-us.pdf/. – Назва з екрану.
5. Symantec. Avoiding the Hidden Costs of the Cloud. 15.12.2013. [Електронний ресурс]. – Режим доступу: URL: http://www.symantec.com/connect/blogs/ avoiding-hidden-costs-cloud/. – Назва з екрану.
6. Хмарні технології в Україні: високий старт. [Електронний ресурс]. – Режим доступу: URL: https://day.kyiv.ua/uk/article/cuspilstvo/. – Назва з екрану.
7. Каплун В.А., Дудатьєв А.В., Семеренко В.П., Захист програмного забезпечення, частина 1 – Вінниця, ВНТУ, 2005 – 140 с.
8. Каплун В.А., Дмитришин О.В., Баришев Ю.В. Захист програмного забезпечення, частина 2 – Вінниця, ВНТУ, 2014 – 105 с.
9. Захист програмних засобів від дослідження [Електронний ресурс]. – Режим доступу: URL: http://sumk.ulstu.ru/docs/mszki/Zavgorodnii/ 8.2.3.html/. – Назва з екрану.
10. Основні логічні операції. [Електронний ресурс]. – Режим доступу: URL: http://cppstudio.com/uk/post/500/. – Назва з екрану.

11. Хостинг сервіс, [Електронний ресурс]. Режим доступу: – https://vercel.com/dashboard
12. UML діаграма послідовності (sequence diagram) [Електронний ресурс] – Режим доступу: http://flash.retejo.info/cxefpagxo/uml/diagrama-poslidovnosti
13. Hash Algorithm Comparison: MD5, SHA-1, SHA-2 & SHA-3 [Електронний ресурс] – режим доступу: https://codesigningstore.com/hash-algorithm-comparison
14. Загальна характеристика UML [Електронний ресурс] – Режим доступу: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/
15. Копитко М. Ф., Іванків К.С. Основи програмування мовою Java: тексти лекцій. Львів: Видавничий центр ЛНУ ім. Івана Франка, 2002. 83
16. Каліниченко А. В. Порівняння інтегрованих середовищ розробки додатків JAVA із відкритим кодом: ECLIPSE та INTELLIJ IDE. Foss Lviv 2013
17. Що таке електронний цифровий підпис (ЕЦП)? Урядовий портал. URL: https://www.kmu.gov.ua/usi-pitannya-po-e-poslugam/sho-tak-elektronnijcifrovij-pidpis-ecp.
18. Поліщук В. В. Програмні технології захисту інформації : конспект лекцій. Ужгород : УжНУ, 2018. 80 с.
19. Лагун А. Е. Криптографічні системи та протоколи : нав. посібник. Львів : Видавництво Львівської політехніки, 2013. 96 с.
20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія перевірки цілісності даних в хмарному середовищі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 87,9% Схожість 12,1%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Борка М.Ю.

Керівник роботи Барабан С.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

parameters:

Клас GetMyFiles

```
package cloud;
import static cloud.UploadRecursiveFolderToSFTPServer.channel;
import static cloud.UploadRecursiveFolderToSFTPServer.session;
import com.jcraft.jsch.ChannelSftp;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.SftpException;
import java.util.Properties;
public class GetMyFiles {
 static Properties props;
 public void startFTP(String name, String pasword,String nameFile){
 String SFTPHOST = "194.28.87.227";
        int SFTPPORT = 22;
        String SFTPUSER = name;
        String SFTPPASS = pasword;
        try {
            JSch jsch = new JSch();
            session = jsch.getSession(SFTPUSER, SFTPHOST, SFTPPORT);
            session.setPassword(SFTPPASS);
            java.util.Properties config = new java.util.Properties();
            config.put("StrictHostKeyChecking", "no");
           session.setConfig(config);
            session.connect();
            channel = session.openChannel("sftp");
            channel.connect();
            ChannelSftp sftpChannel = (ChannelSftp) channel;
            sftpChannel.get("/miracle/cloud/"+nameFile+"/"+nameFile, "/File/"+nameFile);
            sftpChannel.exit();
            session.disconnect();
```

```
        } catch (JSchException e) {
            e.printStackTrace();
        } catch (SftpException e) {
            e.printStackTrace();
        }
    }
}
```

Клас Hesh

```
package cloud;
import static cloud.UploadRecursiveFolderToSFTPServer.channel;
import static cloud.UploadRecursiveFolderToSFTPServer.session;
import com.jcraft.jsch.ChannelSftp;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.SftpException;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import javax.swing.JOptionPane;
public class Hesh {
 static Properties props;
 public void startFTP(String name, String pasword,String nameFile, String put) throws IOException
{
 String SFTPHOST = "194.28.87.227";
        int SFTPPORT = 22;
        String SFTPUSER = name;
        String SFTPPASS = pasword;
        try {
            JSch jsch = new JSch();
            session = jsch.getSession(SFTPUSER, SFTPHOST, SFTPPORT);
            session.setPassword(SFTPPASS);
            java.util.Properties config = new java.util.Properties();
            config.put("StrictHostKeyChecking", "no");
```

```
                session.setConfig(config);
                session.connect();
                channel = session.openChannel("sftp");
                channel.connect();
                ChannelSftp sftpChannel = (ChannelSftp) channel;
                        sftpChannel.get(put+nameFile+"/"+"hash_"+nameFile,
"F:/File"+put+"hash_"+nameFile);
                sftpChannel.exit();
                session.disconnect();
            } catch (JSchException e) {
                e.printStackTrace();
            } catch (SftpException e) {
                e.printStackTrace();
            }
        }
         public String read(String nameFile,String put)throws FileNotFoundException, IOException{
            donload iio=new donload();
           BufferedReader inputStream = null;
            String str=new String();
            try {
                inputStream = new BufferedReader(new FileReader("F:/File/"+put+"hash_"+nameFile));
                 String l;
                    l = inputStream.readLine();
                    str=str+l;
            } finally {
            }
            return str;
         }
        }
```

Клас Table

```
package cloud;
import static cloud.UploadRecursiveFolderToSFTPServer.channel;
import static cloud.UploadRecursiveFolderToSFTPServer.session;
import com.jcraft.jsch.ChannelSftp;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.SftpException;
```

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
public class Table {
  public String startFTP(String name, String pasword){
   String SFTPHOST = "194.28.87.227";
      int SFTPPORT = 22;
      String SFTPUSER = name;
      String SFTPPASS = pasword;
      try {
         JSch jsch = new JSch();
         session = jsch.getSession(SFTPUSER, SFTPHOST, SFTPPORT);
         session.setPassword(SFTPPASS);
         java.util.Properties config = new java.util.Properties();
         config.put("StrictHostKeyChecking", "no");
         session.setConfig(config);
         session.connect();
         channel = session.openChannel("sftp");
         channel.connect();
         ChannelSftp sftpChannel = (ChannelSftp) channel;
         sftpChannel.get("/miracle/cloud_list.txt", "F:/File/cloud_list.txt");
         sftpChannel.exit();
         session.disconnect();
      } catch (JSchException e) {
         e.printStackTrace();
      } catch (SftpException e) {
         e.printStackTrace();
      }
     String str= read();
      return str;
}
  public String read(){
      donload iio=new donload();
    BufferedReader inputStream = null;
      String str=new String();
      try {
         try {
            inputStream = new BufferedReader(new FileReader("F:/File/cloud_list.txt"));
         } catch (FileNotFoundException ex) {
            Logger.getLogger(Table.class.getName()).log(Level.SEVERE, null, ex);
         }
          String l;
         try {
            //тепер читаємо дані цілими рядками
            while ((l = inputStream.readLine()) != null) {
```

```
                    str=str+l+"\n";
                }
            } catch (IOException ex) {
                Logger.getLogger(Table.class.getName()).log(Level.SEVERE, null, ex);
            }
        } finally { //дії коли не знайдено файли
        }
        return str;
    }
}
```

Клас UploadRecursiveFolderToSFTPServer

```
package cloud;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import com.jcraft.jsch.Channel;
import com.jcraft.jsch.ChannelSftp;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.Session;
import com.jcraft.jsch.SftpATTRS;
import com.jcraft.jsch.SftpException;
import java.io.File;
public class UploadRecursiveFolderToSFTPServer {
    static ChannelSftp channelSftp = null;
    static Session session = null;
    static Channel channel = null;
    static String PATHSEPARATOR = "/";
    public UploadRecursiveFolderToSFTPServer(String name,String Password,File file,String
SFTPWORKINGDIR1) {
        String SFTPHOST = "194.28.87.227";
        int SFTPPORT = 22;
        String SFTPUSER = name;
        String SFTPPASS = Password;
        String SFTPWORKINGDIR = SFTPWORKINGDIR1;
        String LOCALDIRECTORY = ""+file;
        try {
            JSch jsch = new JSch();
            session = jsch.getSession(SFTPUSER, SFTPHOST, SFTPPORT);
            session.setPassword(SFTPPASS);
            java.util.Properties config = new java.util.Properties();
            config.put("StrictHostKeyChecking", "no");
            session.setConfig(config);
            session.connect();
            channel = session.openChannel("sftp");
            channel.connect();
            channelSftp = (ChannelSftp) channel;
            channelSftp.cd(SFTPWORKINGDIR);
            recursiveFolderUpload(LOCALDIRECTORY,SFTPWORKINGDIR);
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            if (channelSftp != null)
                channelSftp.disconnect();
```

```
            if (channel != null)
                channel.disconnect();
            if (session != null)
                session.disconnect();
        }
    }
    private static void recursiveFolderUpload(String sourcePath, String destinationPath) throws SftpException, FileNotFoundException {
        File sourceFile = new File(sourcePath);
        if (sourceFile.isFile()) {
            channelSftp.cd(destinationPath);
            if (!sourceFile.getName().startsWith("."))
                channelSftp.put(new FileInputStream(sourceFile), sourceFile.getName(), ChannelSftp.OVERWRITE);
        } else {
            System.out.println("inside else " + sourceFile.getName());
            File[] files = sourceFile.listFiles();
            if (files != null && !sourceFile.getName().startsWith(".")) {
                channelSftp.cd(destinationPath);
                SftpATTRS attrs = null;
                try {
                    attrs = channelSftp.stat(destinationPath + "/" + sourceFile.getName());
                } catch (Exception e) {
                    System.out.println(destinationPath + "/" + sourceFile.getName() + " not found");
                }
                if (attrs != null) {
                    System.out.println("Directory exists IsDir=" + attrs.isDir());
                } else {
                    System.out.println("Creating dir " + sourceFile.getName());
                    channelSftp.mkdir(sourceFile.getName());
                }
                for (File f: files) {
                    recursiveFolderUpload(f.getAbsolutePath(), destinationPath + "/" + sourceFile.getName());
                }
            }
        }
    }
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕВІРКИ ЦІЛІСНОСТІ ДАНИХ В ХМАРНОМУ СЕРЕДОВИЩІ

Виконав: студент 2 курсу, групи 2КН-22м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Борка М.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Барабан С.В.
(прізвище та ініціали)

«07» 12 2023 р.

Вінниця ВНТУ – 2023 рік

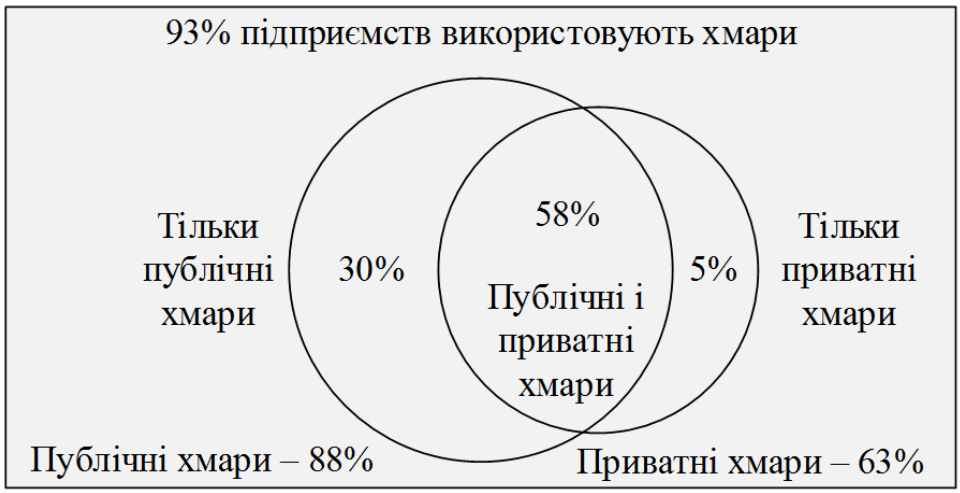


Рисунок В.1 – Дані щодо використання хмарних технологій

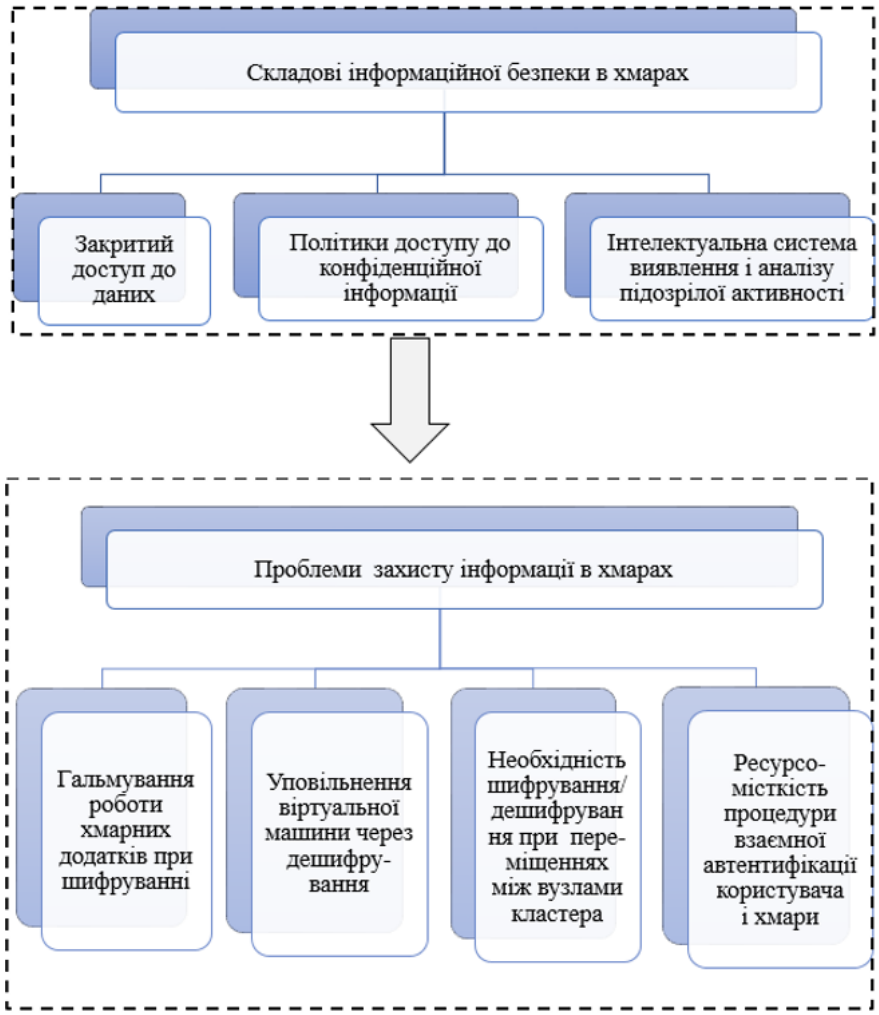


Рисунок В.2 – Вимоги до забезпечення і проблеми інформаційної без¬пеки хмарної інфраструктури

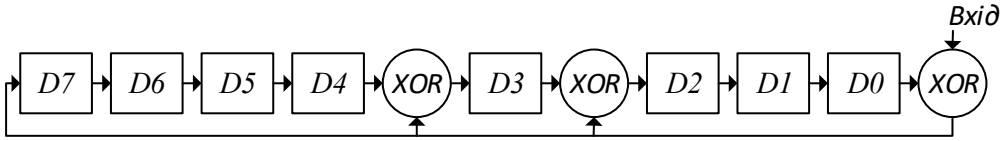


Рисунок В.3 – Формування контрольної суми CRC-8

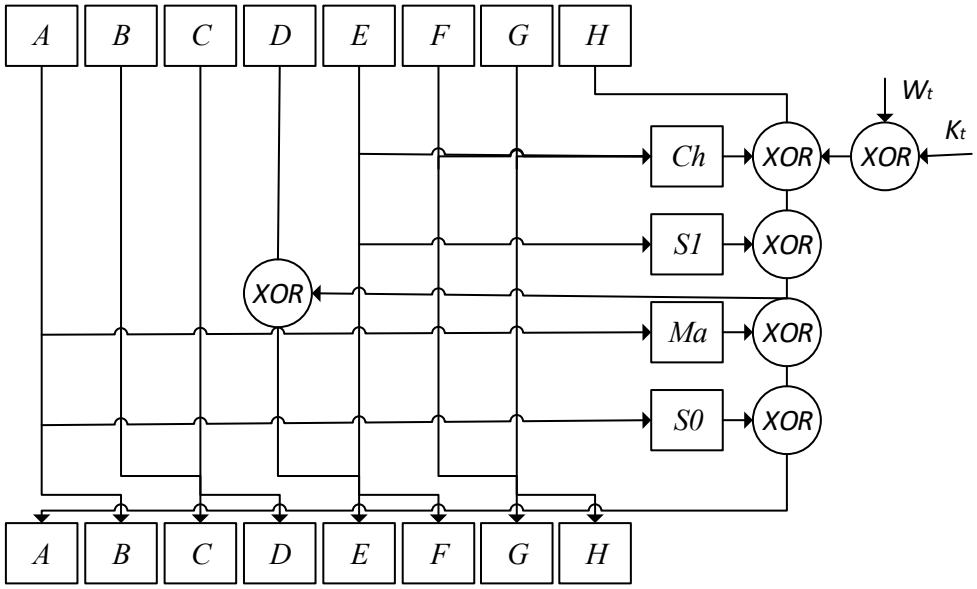


Рисунок В.4 – Алгоритм SHA-2

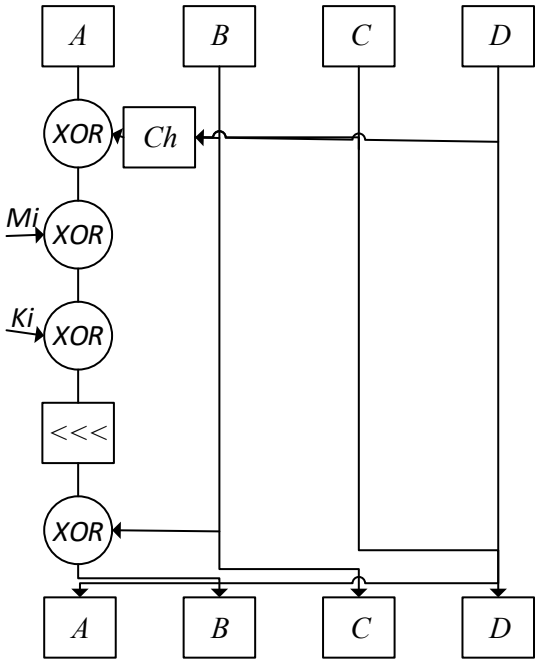


Рисунок В.5 – Алгоритм MD5

Рисунок В.6 – Алгоритм гешування SHA-3

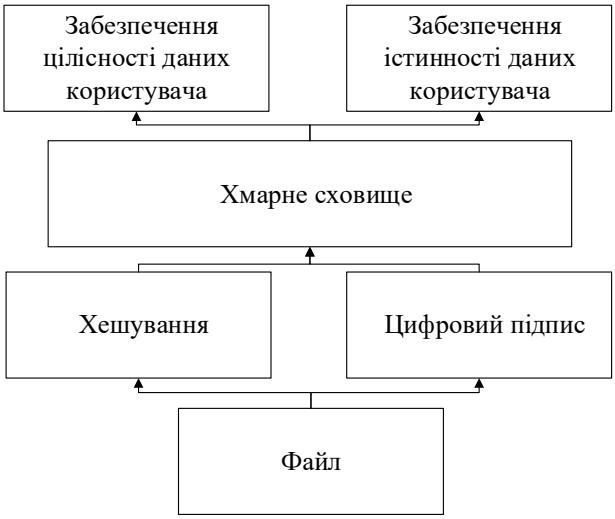


Рисунок В.7 – Структура інформаційної технології перевірки цілісності даних в хмарному середовищі

Рисунок В.8 – Складові серверної та клієнтської частин інформаційної технології перевірки цілісності даних в хмарному середовищі

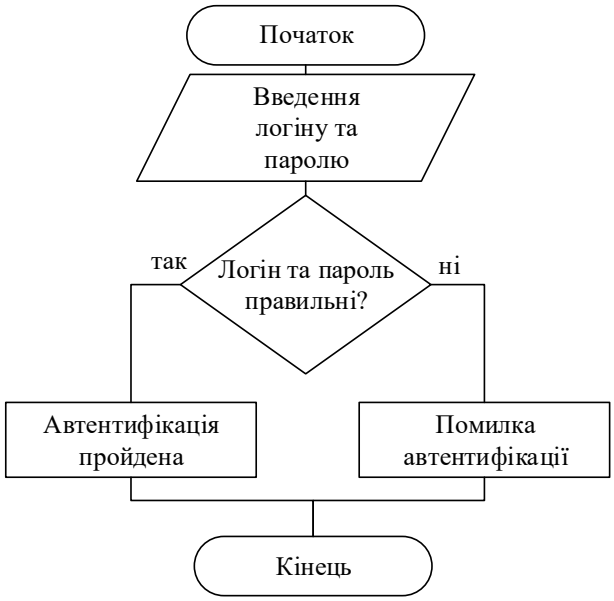


Рисунок В.9 – Алгоритм автентифікації користувача

Рисунок В.10 – Алгоритм функціонування серверної частини

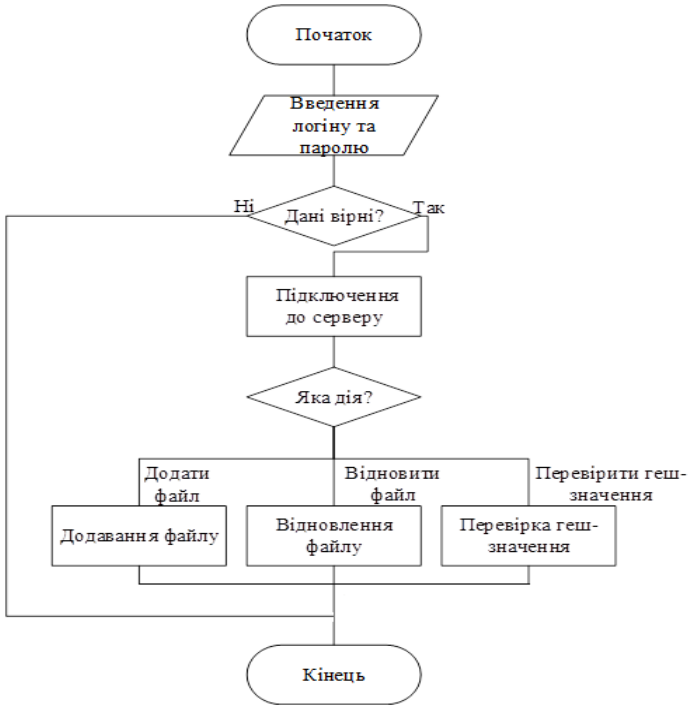


Рисунок В.11 – Алгоритм функціонування клієнтської частини

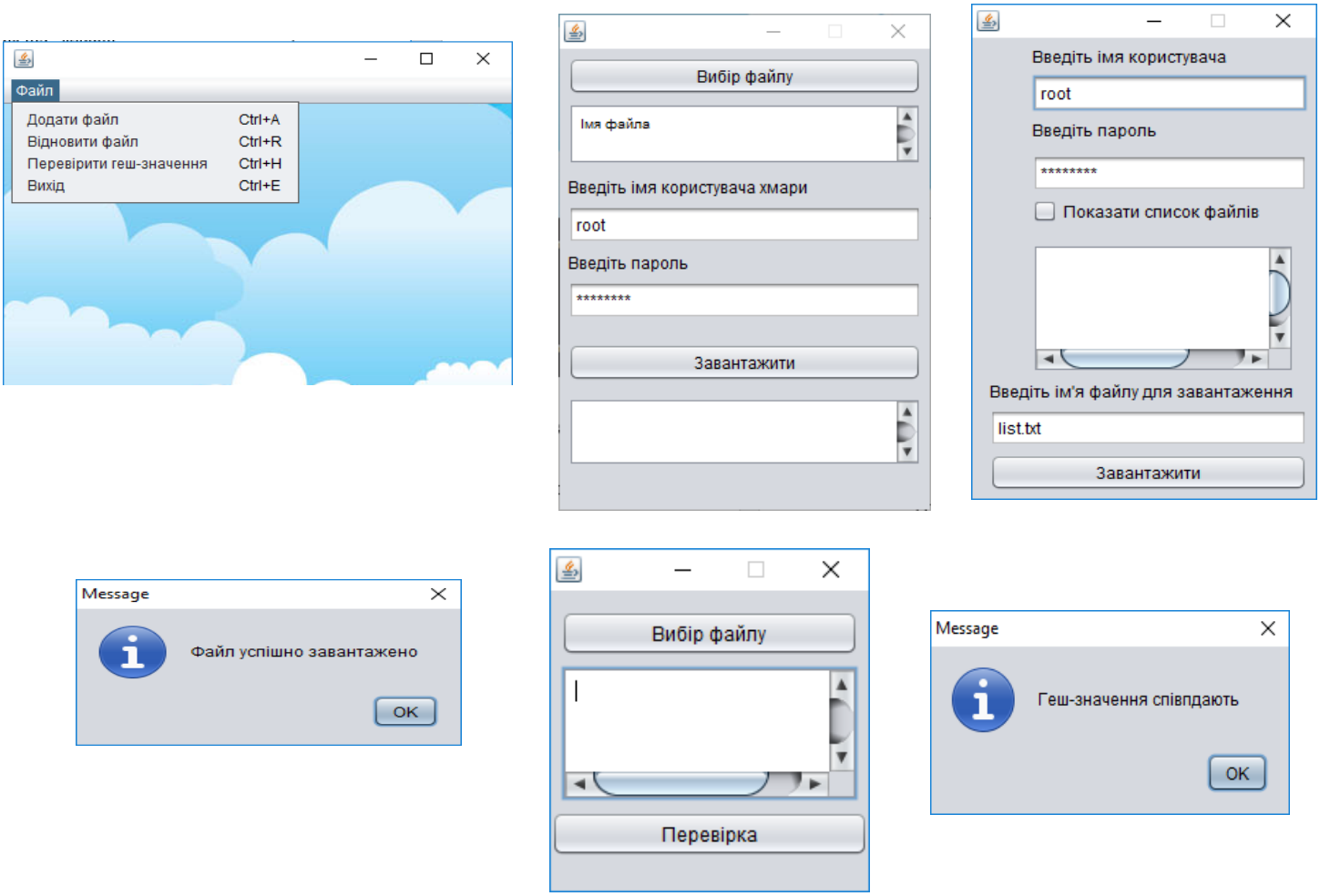


Рисунок В.12 – Графічний інтерфейс програмного забезпечення перевірки цілісності даних в хмарному середовищі

Додаток Г (довідниковий)

Інструкція користувача

Користувачеві необхідно запустити додаток з графічним інтерфейсом для зручної взаємодії з серверною частиною (рис. Г.1). У меню вікна містяться 3 кнопки, при взаємодії з якими з’являється нове вікно. Також для зручності взаємодії з додатком, користувач має можливість викликати необхідне йому вікно за допомогою “гарячих” клавіш.

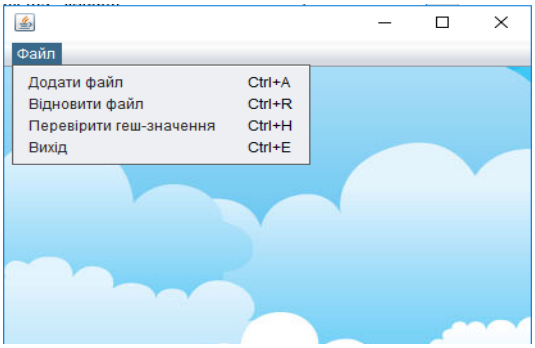


Рисунок Г.1 – Головне вікно додатку

Якщо користувач натисне кнопку в меню «Додати файл» – відкриється нове вікно з можливістю вибору та завантаження файлу на сервер. Обираємо файл на комп’ютері натиснувши кнопку Вибір файлу та завантажуємо його на сервер натиснувши кнопку Завантажити (рис. Г.2).

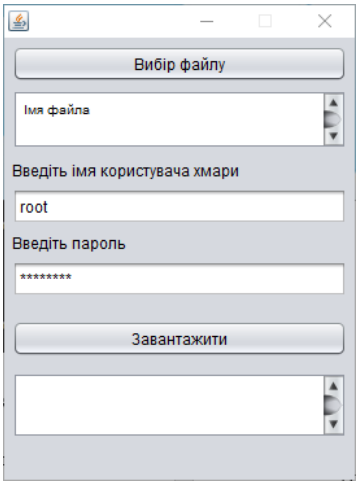


Рисунок Г.2 – Вікно додавання файлу на сервер

Після успішного завантаження файлу на сервер знизу у вікні з’явиться відповідне повідомлення.

Якщо користувач натисне кнопку в меню «Відновити файл» – відкриється нове вікно з можливістю перегляду файлів на сервері та завантаження файлу з серверу на комп’ютер (рис. Г.3).

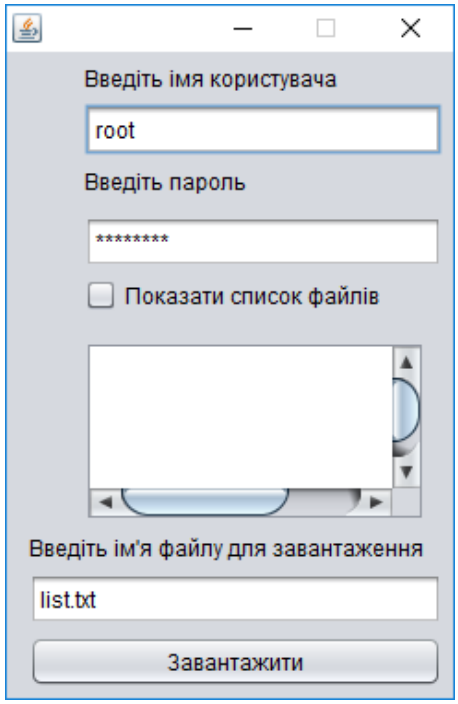


Рисунок Г.3 – Вікно завантаження файлу з сервера

Для цього необхідно ввести його назву та натиснути кнопку Завантажити, після чого файл буде завантажено на комп’ютер у спеціалізовану папку і з’явиться відповідне вікно з повідомленням про успішне завантаження файлу з серверу на комп’ютер (рис Г.4).

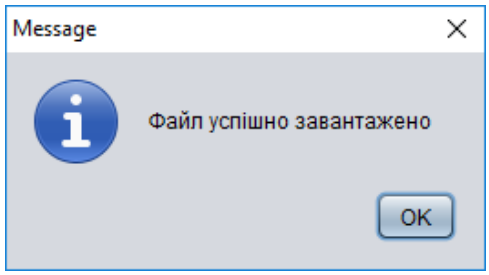


Рисунок Г.4 – Вікно з повідомленням про успішне завантаження файлу

Якщо користувач натисне кнопку в меню «Перевірити геш-значення» – відкриється нове вікно з можливістю перевірки геш-значення обраного файлу на комп’ютері з файлом на сервері. Обираємо файл на комп’ютері натиснувши кнопку Вибір файлу та натискаємо кнопку Перевірка (рис. Г.5).

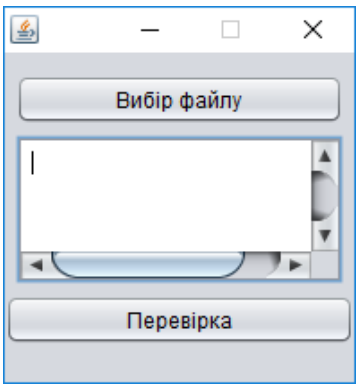


Рисунок Г.5 – Вікно перевірки геш-значення

Якщо геш-значення співпадають – користувач отримає відповідне повідомлення (рис. Г.6)

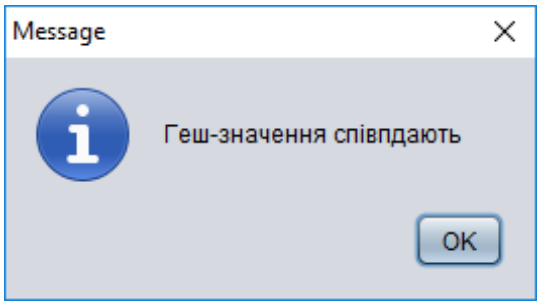


Рисунок Г.6 – Повідомлення про збіг геш-значень обох файлів