НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.9

«До захисту допущено»
В.о. завідувача кафедри
_____ О.І. Ролік
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Інформаційно-орієнтований підхід забезпечення безпеки даних в хмарному середовищі»

Виконав:
студент ІІ курсу, групи ІА-61м
Пирожков Олексій Юрійович _____

Керівник:
Доцент, к.т.н., с.н.с.
Савчук О.В. _____

Рецензент:
Доц. каф. АПЕПС, к.т.н.
Михайлова І. Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.Ф. Теленик

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Пирожкову Олексію Юрійовичу

1. Тема дисертації «Інформаційно-орієнтований підхід забезпечення безпеки даних в хмарному середовищі», науковий керівник дисертації Савчук Олена Володимирівна, доцент, к.т.н., с.н.с., затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – дані в хмарному середовищі.

4. Предмет дослідження – інформаційно-орієнтована безпека даних в хмарному середовищі.

5. Перелік завдань, які потрібно розробити: дослідження традиційних методів забезпечення безпеки даних в хмарному середовищі; дослідження існуючих концепцій в області інформаційно-орієнтованої безпеки; розроблення концептуальної основи підходу до забезпечення безпеки даних; розроблення моделі інформаційно-орієнтованої безпеки даних; розроблення прикладних модулів на основі моделі; написання пояснювальної записки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: базова архітектура для збереження конфіденційності даних в хмарі; основа ОБІ; процедура контролю доступа; процедура безпечного пошуку; результати створення структури ОБІ підходу; експериментальне середовище для запропонованого рішення.

7. Перелік публікацій: 1) Безпека даних в хмарних середовищах: матеріали V міжнар. наук.-практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 28 с. 2) Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень [Текст] / Пирожков О., Савчук О. // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 32-36.

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Дослідження традиційних методів забезпечення безпеки даних в хмарному середовищі	15.03.18-31.03.18	
2	Дослідження існуючих концепцій і результатів в області інформаційно-орієнтованої безпеки	01.04.18-10.04.18	
3	Розроблення концептуальної основи підходу до забезпечення безпеки даних	10.04.18-15.04.18	
4	Розроблення моделі інформаційно-орієнтованої безпеки даних	15.04.18-28.04.18	
5	Розроблення прикладних модулів на основі моделі	29.04.18-03.05.18	
6	Написання пояснювальної записки	04.05.18-14.05.18	

Студент О.Ю. Пирожков

Науковий керівник дисертації О.В. Савчук

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня «магістр» на тему «Інформаційно-орієнтований підхід забезпечення безпеки даних в хмарному середовищі» містить: 151 с., 25 рис., 31 таб., 4 додатки, 27 джерел.

Об’єктом дослідження є дані, що знаходяться в інфраструктурі хмарного середовища та потребують належного рівня безпеки та конфіденційності, що є пріоритетною потребою клієнтів які обслуговуються в публічних хмарах.

Метою дослідження є підвищення ефективності безпеки даних в хмарному середовищі.

Методи дослідження. Емпіричний аналіз існуючих традиційних підходів безпеки хмарних обчислень. Емпіричний аналіз концептуальних складових ІОБ. Синтез концептуальної основи ІОБ. Формалізація концептуальної основи за результатами дослідження.

В експериментальному середовищі розроблене програмне рішення дозволило переконатися у практичній важливості результатів дослідження, а саме у працездатності концепції та у високій ефективності алгоритмів шифрування.

Отримані у ході дослідження програмні модулі, що підтримують функції зашифрованого пошуку та контролю доступу можуть бути використані як ключові криптографічні елементи комерційного клієнт-серверного рішення.

Результати роботи оприлюднені на V Міжнародній науково-практичної конференції Winter InfoCom Advanced Solutions 2017, в науково-практичному виданні «Інфокомунікаційні системи та технології» № 2(2) 2018 р.

ІНФОРМАЦІЙНО-ОРІЄНТОВАНА БЕЗПЕКА ДАНИХ, БЕЗПЕКА ДАНИХ У ХМАРНОМУ СЕРЕДОВИЩІ, ІНФОРМАЦІЙНО-ЦЕНТРОВАНА БЕЗПЕКА ДАНИХ, БЕЗПЕКА ОРІЄНТОВАНА НА ДАНІ.

ABSTRACT

Master’s thesis " Information-oriented approach to data security in cloud environments " consists of: 151 pages, 25 figures, 31 tables, 4 appendices, 27 sources.

The subject of the study is data located in the cloud infrastructure that requires the appropriate level of security and privacy, which is a priority need of clients served in public clouds.

The purpose of the study is to enhance data security level in cloud environment using information-oriented approach.

Research methods. An empirical analysis of existing traditional cloud computing security approaches. Empirical analysis of the conceptual components of the IOS. Synthesis of the conceptual framework of the IOS. Formalization of the conceptual basis of the research results.

Software solution developed in the experimental environment made it possible to verify the practical importance of the results of the research namely, the efficiency of the concept and the high efficiency of the encryption algorithms.

Obtained program modules that support ciphered search and access control can be used as key cryptographic elements of a commercial client-server solution.

The results of the work were announced at the Winter InfoCom Advanced Solutions 2017 Conference, in the scientific journal Infocommunication Systems and Technologies No. 2 (2), 2018.

INFORMATION-ORIENTED SECURITY, DATA SECURITY IN CLOUD ENVIRONMENT, INFORMATION-CENTRIC SECURITY, DATA-ORIENTED SECURITY

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІОБ – інформаційно-орієнтована безпека

ОБІ – орієнтований на безпеку інформації

IaaS – Infrastructure as a Service

PaaS – Platform as a Service

SaaS – Software as a Service

ПХС – провайдер хмарних сервісів

ТС – Trusted Computing

РАЕ – розширений алгоритм Евкліда

КТЗ – китайська теорема про залишки

VPC – Virtual Private Cloud

VPN – Virtual Private Network

SLA – Service Layer Agreement

ВСТУП

Актуальність дослідження новітніх методів забезпечення безпеки в хмарному середовищі зумовлено тим, що при розповсюдженні цієї технології в корпоративному сегменті у споживачів виникає низка перешкод, що насамперед включає проблеми безпеки та конфіденційності. На додаток до традиційних ризиків безпеки, що виникають в обчислювальних системах, підключених до Інтернету, хмарні системи мають специфічні проблеми безпеки та конфіденційності через віртуалізацію хмар та характер своєї багаторівневої природи.

Метою роботи є підвищення ефективності безпеки даних в хмарному середовищі.

Об’єктом дослідження є дані, що розташовані в хмарному середовищі.

Предметом дослідження є безпека даних в хмарному середовищі.

Під час дослідження, що спрямоване на досягнення цієї мети були виконані наступні задачі:

– дослідження традиційних методів забезпечення безпеки в хмарному середовищі;

– дослідження існуючих концепцій ОБІ, визначення характеристик та критеріїв підходу для досягнення максимальної ефективності;

– розроблення концептуальної основи інформаційно-орієнтованого підходу;

– дослідження та вибір алгоритмів шифрування даних та підтримки цілісності;

– дослідження та вибір алгоритмів забезпечення контролю доступу та перевірки аутентифікації;

– дослідження та розроблення складової частини інформаційно-орієнтовного підходу, що забезпечує безпечний пошук у зашифрованих даних;

– тестування клієнт-серверної моделі, що імітує хмарне середовище;

Методи дослідження. Емпіричний аналіз існуючих традиційних підходів безпеки хмарних обчислень. Емпіричний аналіз концептуальних складових ІОБ.

Синтез концептуальної основи ІОБ. Формалізація концептуальної основи за результатами дослідження.

Наукова новизна результуючого рішення забезпечує безпеку хмарних обчислень на рівні даних з високою ефективністю завдяки ефективному алгоритму, що дає змогу не покладатися на провайдера послуг в цьому питанні.

Відомі раніше дослідження зосереджені на підвищенні безпеки на рівні додатків, операційних систем, віртуальних машин або на апаратному рівні. Такі рішення, як правило, не пропонують комплексне рішення, а також проходять під контролем постачальника хмар. У контрасті, інформаційно-орієнтована безпека - це новий підхід, що спрямований на надання повного контролю над забезпеченням безпеки власникам даних, які повністю відповідальні за це упродовж усього життєвого циклу даних у хмарі.

Практична значимість отриманих результатів полягає у тому, що рішення можна використовувати у якості модулів при комерційній реалізації системи, що використовує ОБІ підхід. Отримане рішення слугуватиме ключовими криптографічними елементами ОБІ системи. Прикладами таких систем можуть бути клієнт-серверні рішення.

Апробація результатів роботи

Результати дослідження оприлюднені на конференції Winter InfoCom Advanced Solutions 2017, в науково-практичному виданні «Інфокомунікаційні системи та технології» № 2(2) 2018 р.

Публікації

Безпека даних в хмарних середовищах: матеріали V міжнар. наук.-практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 28 с.

Пирожков О. Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень [Текст] / Пирожков О., Савчук О. // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 32-36

1 ХМАРНІ ОБЧИСЛЕННЯ ТА БЕЗПЕКА

У цьому розділі обговорюються фундаментальні основи технологій хмарних обчислень та проблеми, пов’язані з безпекою, головним чином конфіденційністю, у послугах хмарних обчислень. Розуміння концепції хмарних обчислень та яким чином ця концепція впроваджується в різні моделі надання послуг та моделі розгортання допомагає визначити проблеми безпеки, що стоять перед цією новою технологією.

1.1 Визначення та характеристика хмарних обчислень

Існує ряд визначень терміну «хмарні обчислення» наданих в літературі [1]. Фактично, сам термін відноситься до концепції нової технології, яка все ще змінюється і розвивається. Незважаючи на те, що запропоновано кілька визначень, які відображають суттєві характеристики цієї нової обчислювальної парадигми, кожне визначення зосереджується на певних особливостях та характеристиках. Вакеро, Родеро-Меріно та співавтори зазначають, що набір мінімальних функцій, які включено в більшість визначень, включає віртуалізацію, модель оплати за використання утиліти та масштабованість, причому віртуалізація розглядається як основна технологія хмарних обчислень. Тим не менш, існує безліч інших функцій та характеристик, таких як забезпечення, простоти використання, а також передача даних та процесів аутсорсингу, що, на мою думку, має дати більш вичерпне визначення. Хоча 16-а версія визначення хмарних обчислень, опублікована Національним інститутом стандартів і технологій США (NIST) широко прийнята та добре продумана, визначення хмарних обчислень, запропонована в , є більш комплексним, оскільки воно охоплює більше аспектів, таких як простоти використання, і це визначення полягає в наступному:

Хмари - це великий басейн легких для використання і доступних віртуалізованих ресурсів (таких як апаратні засоби, платформи розробки та/або послуги). Ці ресурси можуть бути динамічно реконфігуровані, щоб пристосуватися до змінного навантаження (масштабу), що також дозволяє оптимально використовувати ресурси. Цей басейн ресурсів, як правило, експлуатується за моделлю оплати за використання, в якій гарантії запропоновані провайдером інфраструктури за допомогою індивідуальних угод про рівень послуг (SLA).

Формулювання визначення може бути іншим підходом для кращого розуміння основних елементів хмарних систем з технічної точки зору. Наприклад, початкове формулювання визначення, запропоноване в, сприймає хмарну модель як рівняння, яке відображає зв’язок між основними компонентами хмари, такими як центри обробки даних, апаратне обладнання машин та локалізації, які використовуються для побудови хмари.

Визначення хмарних обчислень повинне визначати фундаментальні характеристики, які роблять обчислювальні послуги цінними та помітними. NIST зазначив більшість характеристик, які широко використовуються серед спільноти і такі характеристики включають [2]:

– Замовлення самообслуговування. Хмарні обчислювальні ресурси (наприклад, процесор, накопичувач, програмне забезпечення) надаються за потребою та заплановано, не вимагаючи людської взаємодії з провайдером послуг. Ключовими моментами цієї функції є економія часу, економічність, зручність використання та обсяг наданих послуг.

– Широкий доступ по мережі. Хмарні послуги доступні через широко доступну мережу, головним чином Інтернет, який використовує стандартні протоколи та механізми для підтримки різних типів пристроїв і платформ, наприклад, смартфонів, тонких клієнтів та КПК.

– Поєднання ресурсів. Фізичні хмарні ресурси, засновані на технології віртуалізації, розподіляються серед хмарних клієнтів, залежно від потреб

споживання. Клієнти не усвідомлюють фізичних обмежень ресурсів, оскільки вони віртуально забезпечуються та автоматично вилучаються відповідно до попиту.

– Розрахункові послуги. Оскільки послуга надається згідно з бізнес-моделлю «оплати за використання», використання послуг та ресурсів можна виміряти та автоматично виставляти рахунок за кожний конкретний сеанс клієнта.

– Швидка гнучкість. Забезпечення та вилучення хмарних послуг та ресурсів здійснюється швидко та гнучко для кожного хмарного клієнта в режимі реального часу.

– Незалежність від місцевості. Хмарні ресурси можуть бути фізично розташовані в будь-якій географічній локації, якщо доступні можливості зв’язку. Хмарні клієнти також можуть володіти доступом до послуги з будь-якої точки за тими самими умовами.

1.2 Хмарні обчислення: труднощі та проблеми

У послугах хмарних обчислень клієнти стурбовані переміщенням своїх конфіденційних даних і додатків зі своїх приватних обчислювальних середовищ в хмарну середовище, яка спільно використовується різними клієнтами і яке зазвичай доступне через загальнодоступну мережу. Опитування, проведене у вересні 2009 року Міжнародною корпорацією даних (IDC), показує кілька проблем, які стосуються клієнтами хмарних обчислень, як показано на рисунку 1.1 [5]. Безпека була ранжована як найвища проблема.

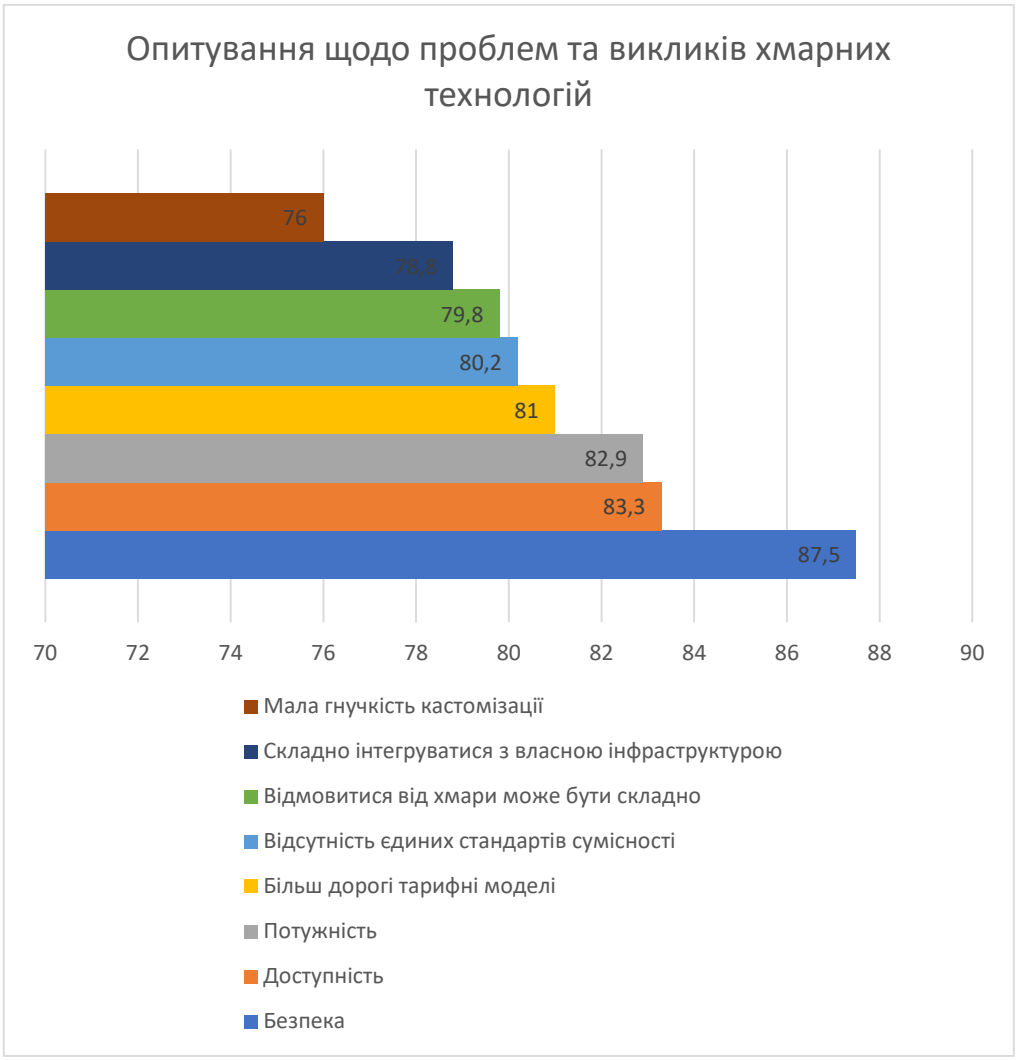

Рисунок 1.1 – Ранжування проблем клієнтами хмарних послуг [5]

Проблеми безпеки в хмарних обчисленнях зазвичай пов’язані з основними технологічними компонентами, на які покладаються хмарні обчислення. Цими компонентами є:

– Веб-додатки та служби є найбільш часто використовуваними технологіями для доступу до хмарних обчислень.
– Віртуалізація є основною технологією надання хмарних обчислень. Обидві SaaS і PaaS засновані на віртуалізації інфраструктури, що надається на рівні IaaS.
– Криптографічні методи в даний час є найбільш поширеними методами для досягнення задовільного рівня вимог безпеки для хмарних обчислень.

Отже, будь-які відомі вразливості вищезазначених трьох основних технологічних компонентів можна розглядати як вразливості хмарних обчислювальних систем. Наприклад, протокол HTTP, який використовується в веб-технологіях, піддається перехопленню сеансів та атакам сеансового рівня. Тому хмарні обчислювальні системи вразливі для такого роду атак і повинні подолати цю слабкість. Віртуалізація - ще одна вразлива проблема. Зловмисник може прорватися через віртуальний поділ і отримати доступ до даних і ресурсів або пасивно, спостерігаючи за даними, або активно, змінюючи дані і конфігурації. Крім того, існують різні інші можливі вразливості, які можуть бути присутніми в хмарних обчислювальних системах, і вони пов’язані з його інфраструктурою та середовищем. Оскільки хмарні послуги зазвичай надаються через Інтернет, все очікувані проблеми, пов’язані з Інтернетом, також пов’язані з хмарними обчисленнями. Вразливості в операційних системах і інших програмах, реалізованих програмно і встановлених в хмарну інфраструктуру, також можна розглядати як такі, що пов’язані з вразливостями хмарних обчислень.

У цьому пункті розглядаються деякі конкретні вразливості і проблеми безпеки, пов’язані з природою хмарних обчислень:

Несанкціонований доступ до інтерфейсу управління: в хмарних обчисленнях інтерфейси управління зазвичай доступні через загальнодоступні мережі для авторизованих клієнтів і можливих неавторизованих зловмисників, тоді як звичайні центри обробки даних зазвичай доступні тільки авторизованим адміністраторам безпосередньо або через приватні мережі. Більш того, доступ до управління зазвичай здійснюється через веб-додаток або сервісні технології, тому інтерфейс управління хмарами, ймовірно, схильний до вразливості цих технологій.

Проблема відновлення даних: через природу віртуалізації і спільного використання хмарних послуг на апаратному рівні області пам’яті і зберігання, які були орендовані попередніми клієнтами, можуть бути перерозподілені для нових клієнтів. Можливо, що ці нові клієнти можуть відновлювати дані з цих областей пам’яті і зберігання, які можуть містити конфіденційну інформацію, що належить попереднім клієнтам.

Вразливість образу шаблону віртуальної машини (VM): нова віртуальна машина зазвичай створюється шляхом клонування образу шаблону попередньо сконфігурованої віртуальної машини, так як це економить час і зусилля. Таким чином, багато клієнтів будуть орендувати віртуальні машини з однаковими конфігураціями. Зловмисник може збирати інформацію про образи шаблонів хмарних систем, ставши клієнтом хмари з правами адміністратора. Як тільки зловмисник має доступ до образів шаблонів, він може шукати вразливості в цих образах, які також використовуються іншими клієнтами.

Можливість витоку даних: інша проблема вразливості, пов’язана з образами шаблонів віртуальних машин, полягає в тому, що хмарні провайдери можуть використовувати шаблони, створені іншими клієнтами для нових клієнтів. Ці шаблони можуть містити секретні бекдори, створені зловмисником, що прикидатися клієнтом, і дозволяють зловмисникові отримати доступ до віртуальних машин інших клієнтів.

Ін’єкційні вразливості: оскільки більшість хмарних сервісів використовують служби веб-додатків, можна вводити зловмисні коди в хмарну систему, використовуючи вразливості в таких службах веб-додатків, щоб зламати веб-сервери, які обслуговують ці служби. Існує багато прикладів способів взлому з використанням зловмисних кодів, таких як коди SQL, команда ОС або коди JavaScript. Як тільки веб-сервер зламаний, він може використовуватися в якості стартового майданчика для зловмисника для взламування інших цілей в системі, і ці цілі включають бази даних і операційні системи.

Метрики безпеки і труднощі моніторингу: клієнти повинні мати можливість вимірювати і контролювати ситуацію безпеки своїх хмарних послуг і ресурсів. Однак надання таких можливостей хмарним клієнтам як і раніше є проблемою, тому що доступні традиційні стандартні інструменти ще не підходять для хмарного середовища. Оскільки хмарне середовище має складні і динамічні ієрархічні сервіси, які можуть включати в себе різних провайдерів хмарних обчислень, хмарне середовище вимагає нових розподілених можливостей моніторингу, відповідних цим характеристикам.

Складність в управлінні цифровими ключами і випадковими числами: в хмарній системі існують різні типи ключів і випадкових чисел, необхідні для криптографічних операцій. Управління та зберігання різних ключів в хмарному середовищі - непрості завдання, тому що немає повної фізичної ізоляції між ресурсами зберігання, виділеними для різних клієнтів. Ефективність генерації випадкових чисел в основному залежить від апаратного годинника, використовуваного генератором випадкових чисел. Відсутність такої ефективності може бути випробувано в хмарному середовищі, де в різних сеансах кілька хмарних клієнтів використовують одні і ті ж ресурси генерації одночасно. Це може призвести до перевантаження ресурсів генерації випадкових чисел, або може привести до отримання слабких чисел. Таким чином, впровадження стандартних механізмів безпеки, таких як модуль апаратної безпеки, який спирається на ефективний ресурс генерації випадкових чисел, в хмарні системи є проблемою безпеки.

Проблема функціональної сумісності хмари: ця проблема пов’язана з тим, як різні хмарні провайдери дозволяють власникам даних безперешкодно переміщати свої дані від одного провайдера до іншого або від хмарного провайдера назад в свої локальні ресурси, коли їм це потрібно. Без функціональної сумісності між хмарними провайдерами, власник даних може заблокувати певного провайдера і не зможе легко перейти до інших провайдерів або оптимізувати послуги між різними провайдерами.

Спостереження за шаблонами активності: шаблони активності одного хмарного клієнта можуть спостерігатися або іншими клієнтами в одній хмарі, або хмарним провайдером. Це спостереження може бути кроком для атаки безпеки або може бути використано для виявлення ділових дій, які не можуть бути виявлені в звичайних обставинах. Наприклад, обмін інформацією між двома компаніями може свідчити про планування злиття.

Необхідність взаємних можливостей проведення аудиту, підзвітності і надійність: довіра повинна будуватися між провайдерами і клієнтами в хмарному середовищі. Прозорість через високий рівень можливостей проведення аудиту і підзвітності має важливе значення для створення довірливих відносин. Довірливі відносини будуть більш складними, якщо хмарні провайдери делегують деякі з

хмарних сервісів субконтрактам. Клієнти хмарних обчислень повинні знати, чи є які-небудь субконтракти, які також можуть відповідати за їх дані і додатки за хмарою. Наприклад, Linkup надав онлайн-службу зберігання через іншого субпідрядного провайдера під назвою Nirvanix, перш ніж він закрився в результаті втрати значної кількості клієнтських даних. Ймовірно, субпідрядник ніс відповідальність за втрату клієнтських даних.

1.3 Тенденції та напрямки рішень

Технологія хмарних обчислень стикається з різними проблемами, які не можна вирішувати безпосередньо традиційними рішеннями. Відповідне рішення має бути адаптоване до конкретних характеристик цієї нової обчислювальної парадигми. Дослідницькі напрямки вирішення проблем хмарних обчислень різні в залежності від того, на яких видах хмарних проблем зосереджуються дослідники. На рівні віртуалізації технології стверджується, що проблеми, пов’язані з ізолюванням віртуальних машин на одній фізичній машині, вимагають більшої уваги з точки зору безпеки та продуктивності. Для більш високого рівня важливості, з приводу складнощів довіри у розрахунку на багато клієнтів і вимог до можливостей взаємного аудиту в хмарних обчисленнях, можуть стати новими важливими завданнями. Проте, можна стверджувати, що конфіденційність і цілісність даних є основною вимогою безпеки, особливо в ненадійних хмарах. Також в надійних або частково надійних хмарах сильним механізмом конфіденційності може бути ключ до встановлення надійності. Ненадійний сервер хмарних обчислень може надавати персональні дані і шаблони активності клієнтів і повертати невірні дані від обчислювальних процесів клієнтам. Також можливо, що ненадійний провайдер може маніпулювати законним способом обробкою запитів користувачів в їх інтересах. Таким чином, захист даних, зокрема їх конфіденційності і цілісності, як від провайдерів хмарних послуг, так і від

зовнішніх зловмисників, як очікується, призведе до створення більш сильних архітектур безпеки хмар, які сприятимуть ширшому впровадженню хмарних сервісів.

1.3.1 Захист конфіденційності і цілісності даних від провайдерів хмарних обчислень

Конфіденційність і цілісність є важливими вимогами для різних додатків, таких як e-government і EHR (Electronic Health Record). Клієнти хмарних обчислень не тільки турбуються про компрометацію конфіденційності і цілісності своїх даних від можливих зловмисників, а й від потенційних цікавих до цього провайдерів хмарних обчислень. На жаль, порушення безпеки, підраховані в 2011 році і перераховані в [6], показують, що великі компанії, такі як Google, EMC/RSA, Sony, UK National Healthcare System (NHS) і Amazon EC2, всі стикалися з інцидентами з безпекою. В хмарних обчисленнях дані клієнтів передаються стороннім провайдерам, які можуть бути надійними або ненадійними. Термін ненадійний може використовуватися для вказівки того, що хмарному провайдеру не можна повністю довіряти. Наприклад, ненадійні постачальники хмарних послуг можуть не змінювати дані користувачів, але вони можуть пасивно порушувати конфіденційність даних або приховано змінювати протоколи для своєї фінансової вигоди. Іншими словами, сервер провайдера хмарних обчислень можна розглядати як чесний, але допитливий сервер. Отже, він заслуговує на довіру в наданні послуг з точки зору доступності даних, дотримання основних вимог контролю безпеки і обробки чесно дозволених запитів на збереження даних і повернення правильних результатів. Проте, можливі зловмисні дії всередині хмари можуть виконуватися зловмисним адміністратором або співробітником.

У зв’язку з більш широким впровадженням хмарних сервісів дослідники вивчають і розробляють нові методи, які зберігають конфіденційність і цілісність зовнішніх даних без повної залежності від провайдерів хмарних обчислень для забезпечення цих вимог безпеки. Рішення повинні надавати клієнтам більше контролю над захистом своїх даних і також захищати дані від провайдерів. Оскільки сервер хмарного провайдера, на якому розміщені аутсорсингові дані, може бути не

повністю надійним, кілька дослідників запропонували методи вирішення таких ситуацій. Загалом, запропоновані ними рішення базуються на шифруванні даних до того, як дані будуть відправлені на сервер провайдера хмарних обчислень.

Хоча зашифровані дані захищені від несанкціонованого доступу, вони не можуть бути повністю корисні, якщо вони не дешифровані. Наприклад, авторизовані користувачі не можуть шукати ключові слова в зашифрованих даних, використовувати зашифровані дані в якості вхідних даних для операцій обчислення або порівняння. Оскільки дешифрування даних в хмарі може розкривати їх контент серверам-провайдера, то принаймні більш безпечно буде розшифровувати дані тільки в надійних машинах, які контролюються користувачем, що був авторизований для доступу до цих даних.

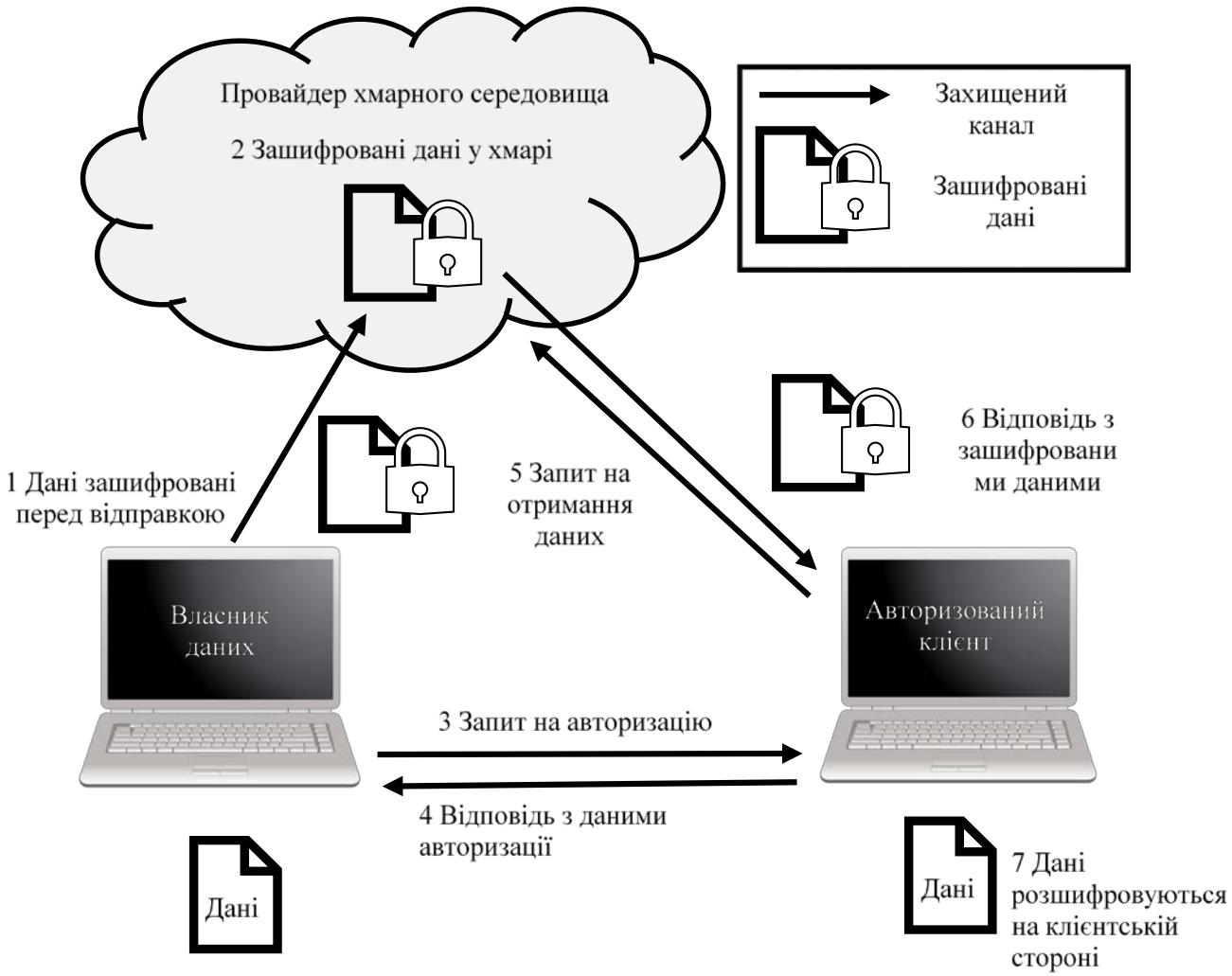

Рисунок 1.2 – Базова архітектура для збереження конфіденційності даних в хмарі

На рисунку 1.2. показана базова архітектура шифрування даних для захисту конфіденційності, перш ніж відправляти їх в хмару. Потім дані залишаються зашифрованими в хмарі, і тільки користувачі, авторизовані власником даних, можуть отримати облікові дані для доступу до зашифрованих даних. Зашифровані дані можуть бути розшифровані тільки після їх завантаження на авторизований комп'ютер користувача. У такому сценарії конфіденційність даних не залежить від неявного припущення про довіру сервера або індивідуальних угод про рівень послуг (SLA). Замість цього, захист конфіденційності залежить від методів шифрування, що використовуються для захисту даних. Решта проблем полягають в тому, як дозволити власнику даних і авторизованим користувачам ділитися і шукати зашифровані дані і використовувати їх для деяких обчислень відповідно до їх прав доступу. Всі ці функції повинні виконуватися безпечним чином, без розкривання приватної інформації неавторизованим суб'єктам, включаючи хмарних провайдерів. Нові криптографічні методи, схеми довіренних обчислень і підходи орієнтовані на безпеку інформації можуть стати перспективним рішенням для подолання декількох проблем безпеки хмарних обчислень.

1.3.2 Криптографічні технології пристосовані до хмарної моделі

Нові технології криптографії необхідні, щоб відповідати хмарній моделі і підвищити безпеку інформації з меншим впливом на зручність її використання. Наприклад, шифрування з можливістю пошуку - це одна з областей, де дослідники розробляють можливість пошуку в зашифрованих даних без їх дешифрування. Лише авторизовані користувачі можуть запитувати і отримувати зашифровані дані, не відкриваючи ніяку приватну інформацію ні про дані, ні про запит, який може містити інформацію про дані. Іншим прикладом є методи шифрування, які дозволяють виконувати обчислення зашифрованих даних без їх дешифрування. В таких методах також захищаються результати обчислення, які зазвичай містять інформацію про захищені дані. Прикладом цих методів є гомоморфне шифрування. Існують і інші

методи криптографії, такі як шифрування на основі ідентифікаційної інформації (IBE) і шифрування на основі атрибутів (ABE), які можуть поліпшити управління ключами і контроль доступу до хмарних систем. Чоу, Голле (Chow, Golle) та співавтори вважають, що ці нові криптографічні рішення можуть бути найбільш підходящими інструментами для вирішення декількох проблем безпеки і надання хмарним клієнтам більш ефективного контролю над своїми даними в хмарі, розглядаючи, як ці методи можуть бути поліпшені, щоб адаптувати їх до практичного хмарного середовища.

1.3.3 Довіренні обчислення (TC)

IT-спільнота, зокрема Trusted Computing Group (TCG), намагається створити набір технологій, таких як автентифікація, шифрування даних, управління ідентифікацією та доступом, управління паролями, управління доступом до мережі та аварійним відновленням, щоб гарантувати, що комп'ютерні системи будуть виконувати бажаний тип операції. TCG застосовує схему довіренних обчислень до Trusted Multi Tenant Infrastructure (TMI), щоб встановити довіренність ненадійного середовища, наприклад, в послузі публічних хмарних обчислень. Нова концепція спрямована на те, щоб дозволити клієнтам оцінювати довіренність хмарних провайдерів, застосовуючи набір апаратних і програмних технологій. Атестація віддаленого сервера - одна з цих технологій, яка дозволяє клієнтам перевіряти хости.

TCG починає будувати свої рішення на основі створення стандартного апаратного модуля - Trusted Platform Module (TPM), який виконує основні функції криптографії, такі як хеш-функція і RSA. Ці криптографічні функції необхідні для встановлення надійності в обчислювальному обладнанні. Іншими словами, TCG спрямована на надання стандартних апаратних і програмних технологій для надійних обчислень, включаючи хмарні обчислення. Тому деякі з рішень безпеки, зокрема ті, які пропонуються для забезпечення віртуальної ізоляції між віртуальними машинами і віртуальною машиною від провайдера сервера, засновані на технологіях TC. Ці технології як і раніше стикаються з декількома проблемами і не можуть виключно використовуватися для забезпечення універсального рішення всіх проблем з хмарної

безпекою. Наприклад, якщо TPM, який є основним компонентом ТС, скомпрометовано, всі рішення буде порушено, як вказав Крістофер Тарновський (Christopher Tarnovsky) в 2010 році. Крім того, концепція ТС не надає користувачам хмарних обчислень достатній контроль над своїми даними з точки зору конфіденційності та політики безпеки.

1.3.4 Підхід орієнтований на безпеку інформації

У традиційних методах захисту даних безпека забезпечується сервером, який зберігає інформацію. Методи, які використовуються для захисту даних, а також управління захищеними даними контролюються адміністраторами сервера. Такий підхід можна класифікувати як системно-орієнтований підхід, який не підходить для захисту даних клієнтів в менш надійному хмарному середовищі. Очікується, що підхід, орієнтований на інформацію, буде більш ефективним і адаптивним для хмарних послуг. Термін орієнтований на безпеку інформації в цілому вказує на те, що захист сфокусовано навколо даних. Ця термінологія може використовуватися по-різному. Для хмарних обчислень підхід орієнтований на безпеку інформації полягає в захисті даних зсередини, таким чином дані, відповідно до їх значення і класифікації, мають свої вимоги безпеки, вбудовані в фактичні дані, щоб забезпечити оптимальний захист даних на будь-якому етапі існування даних, незалежно від середовища, в якому зберігаються дані.

1.4 Висновок

Концепція хмарних обчислень пропонує нову обчислювальну парадигму, в якій, з одного боку, фізичні ресурси можуть спільно використовуватися клієнтами в Інтернеті, і з іншого боку, клієнти мають власний обчислювальний простір, використовуючи методи віртуалізації. Загальна концепція хмарних обчислень полягає в тому, що служби ІСТ та ресурси, що надаються ПХС через широкосмугові

мережі, в основному в Інтернеті, і клієнти використовують ресурси і послуги в міру необхідності і платять тільки за те, що вони споживають. Хмарні послуги можуть постачатися в різних моделях, що базуються на типі послуги, що надається ІСТ. Основними трьома хмарними послугами є: Програмне забезпечення як послуга (SaaS), Платформа як послуга (PaaS) і Інфраструктура як послуга (IaaS). Загалом, існує п’ять відомих моделей розгортання послуг хмарних обчислень, а саме: приватна хмара, громадська хмара, публічна хмара, гібридна хмара і віртуальна приватна хмара. В даний час модель публічних хмар є найпопулярнішою комерційною хмарною моделлю. З одного боку, ця технологія дає великі переваги, такі як економічна ефективність, економія часу і масштабованість. З іншого боку, ця технологія стикається з цілою низкою труднощів і проблем, коли проблеми конфіденційності та безпеки можуть вважатися найбільш складними. Таким чином, дослідницькі тенденції полягають у захисті конфіденційності даних і цілісності в хмарі навіть від самих хмарних провайдерів і в наданні клієнтам хмарних функцій можливості більш строго контролювати свою політику безпеки даних в хмарі. Запропоновані рішення для підвищення безпеки даних в хмарі мають різні напрямки: деякі зосереджені на використовуваних інструментах, в першу чергу криптографічних алгоритмах, для підвищення безпеки даних в хмарі, інші зосереджені на більш комплексних рішеннях, що поєднують різні методи безпеки, засновані головним чином на двох підходах: Надійні обчислення (ТС) та Підхід орієнтований на безпеку інформації (ОБІ).

2 ПІДХІД ОРІЄНТОВАНИЙ НА БЕЗПЕКУ ІНФОРМАЦІЇ ДЛЯ ХМАРНИХ ОБЧИСЛЕНЬ

У цьому розділі більш детально обговорюється підхід орієнтований на безпеку інформації (ОБІ). Цей підхід є фундаментальним, що використовується в даній роботі для підвищення безпеки і конфіденційності хмарних обчислень. Розділ розпочинається з аналізу та класифікації можливих рішень безпеки на основі концепції ОБІ в моделі хмарних обчислень. Спираючись на них в пункті 2.1.3 сформовано концептуальну основу реалізації ОБІ, запропоновану в цьому дослідженні. Очікувані переваги застосування ОБІ-підходу до хмарних обчислювальних середовищ обговорюються в пункті 2.2. Деякі з постійних змін, що виникають під час застосування висвітлено у пункті 2.3. Обсяг застосування підходу ОБІ до моделі хмарних обчислень визначено для цієї роботи в пункті 2.4. У пункті 2.5 роз’яснюються основні вимоги безпеки при застосуванні підходу ОБІ до цієї сфери. Крім того, у цьому пункті розглядаються доступні технології, які можуть бути використані для досягнення цих вимог. На основі здійснених оглядів, в пункті 2.6, оцінюється придатність цих технологій для отримання нового рішення, яке є одним з основних вкладів цього дослідження. Нарешті, короткий виклад цього розділу коротко представлено в останньому пункті.

2.1 Інформаційно-орієнтований підхід забезпечення безпеки

У хмарних обчисленнях дані користувачів, в основному, зберігаються у віртуальних сховищах провайдерів хмарної інфраструктури. В публічних SaaS та DaaS моделях користувачі володіють лише даними, які знаходяться на зберіганні. Все обладнання та програмне забезпечення, залучене до зберігання та обробки інформації, знаходиться у власності сервіс провайдерів. В інших моделях, таких як

публічні IaaS и PaaS моделі, користувач має доступ до обробки даних та до програмного забезпечення, при цьому доступу до апаратного забезпечення немає. Відповідно, з перспективи користувача хмарного сервісу, найбільш цінним активом в хмарному середовищі є його дані, особливо ті дані, що містять інформацію делікатного характеру і вимагають особливого ставлення, а саме: дані урядового характеру, охорони здоров'я та фінансового спектру. Переваги, що надаються за використання хмарних обчислень, дають користувачам, що раніше розміщували делікатні дані на своїх ПК, більш привабливі перспективи до аутсорсингу своїх даних в хмару. Як і будь-які інші послуги в мережі Інтернет, хмарні сервіси також зазнають атак на системи безпеки. Компрометація доступності хмарних послуг, як правило, призводить до короткострокових ефектів і пошкодження можуть бути відновлені. Компрометація конфіденційності та приватності даних споживачів хмарних послуг, може привести в свою чергу до довгострокових ефектів і будь-які втрати можуть бути достатньо важким для відновлення. Наприклад, коли кілька паролів з адміністративних облікових записів UK's National Healthcare System (NHS) було взламано в червні 2011 року, система NHS була закрита органами охорони здоров'я для захисту записів пацієнтів. Це показує, що для таких випадків конфіденційність даних є більш важливою ніж нормальне функціонування системи як такої. Внутрішні ризики можуть бути від користувачів-зловмисників, які використовують той самий хмарний сервіс, і пом'якшення ризиків, в такому випадку, залежить повністю від хмарного провайдера і виходить з-під контролю власника даних. З точки зору власників даних, хмарне середовище невидиме і власник даних не впевнений в тому як його/її дані захищені від ризиків пов'язаних з безпекою. Наприклад, виходячи з природи файлів, їх може бути переміщено через сервіс провайдерів, невідомих для власників даних, або в різних країнах, з різною юрисдикцією щодо конфіденційності даних. Як наслідок з цих проблем, клієнти хмари відчувають обмежений контроль над своїми даними і їм бракує впевненості щодо безпеки даних. З іншої точки зору, провайдери хмарних послуг має надмірний контроль над даними клієнтів, особливо щодо їх безпеки та приватності. Отже, власники даних стурбовані безпекою та приватністю їх даних і мають бажання зберегти свої дані в безпеці, навіть від

провайдерів послег зберігання даних в хмарі. Крім того, вони надають перевагу власноруч управляти політикою безпеки своїх даних в безпечному режимі, як ніби ці дані зберігалися на їх ПК.

Традиційна концепція безпеки зазвичай зосереджується навколо технологій і пристроїв, що використовуються для зберігання та обробки даних. Ця концепція може бути важко адаптованою для забезпечення необхідного відповідного рівня безпеки, особливо для делікатних та конфіденційних даних. Наукова спільнота нещодавно звернула увагу на питання безпеки та конфеденійності даних в хмарах, запропоновані рішення в основному спрямовані на забезпечення безпеки Операційних Систем (ОС), що лежать в основі та віртуальних машин (VM), що хостять хмарні сервіси. Таким чином, більшість рішень як і раніше засновані на традиційній концепції безпеки і в основному фокусуються на будь-який ОС-орієнтованій або VM-орієнтованій безпеці. Деякі з цих рішень засновані на концепті надійних обчислень, який було запропоновано і розроблено групою Trusted Computing Group (TCG). TCG прагне розробити набір стандартів і технологій, таких як Trusted Platform Module (TPM), які можуть зберігати клієнтські дані і додатки, оброблювані в межах хмарної інфраструктури, який убезпечений навіть від системних адміністраторів хмари. TCG, на основі їх технологій, фокусується на наданні набору інструментів, які можуть бути використані для надання допомоги клієнтам, щоб оцінити надійність провайдерів, стежити за дотриманням політики, а також створювати можливість прозорості щодо фізичного розташування даних в хмарі. Проте, клієнти хмари повинні спочатку довіряти TCG технологіям з точки зору оцінки надійності провайдерів та якщо TPM, який є основним компонентом щоб побудувати цю довіру, буде скомпрометовано, постраждає все рішення. У 2010 році, Крістофер Тарновський стверджував, що йому вдалося скомпрометувати TPM. Отже, дослідження і запропоновані рішення на основі на TPM, можливо, варто піддати переоцінці. Навіть якщо TPM та інші інструменти TCG є гарантією безпеки, фокус цих інструментів не на тому щоб надати користувачам бажаний контроль за безпекою та приватністю їх даних. Замість цього, з точки зору клієнта, реалізація концепції ТС дозволяє здійснювати клієнтам моніторинг або аудит операцій, в тому числі над політикою контролю доступу, для

серверу через довіренні інструменти, які можуть забезпечити докази відповідності концепції ТС для користувачів. Нова концепція була запропонована декількома дослідниками для вирішення конкретних питань безпеки хмарних обчислень, переміщаючи фокус з забезпечення безпеки для даних клієнтів на дані, як такі, і вони називають це інформаційно орієнтована безпека. Ця концепція все ще розвивається, і існують різні думки щодо її застосування до моделі хмари.

2.1.1 Класифікація існуючих рішень

У даній роботі, так і з точки зору розуміння концепції ОБІ, існуючі рішення можуть бути класифіковані за двома критеріями: перша класифікація заснована на тому, на якому рівні забезпечується безпека, а друга класифікація - на тому, хто несе відповідальність за забезпечення безпеки.

На правій частині Рисунку 2.1, проілюстровано рівні які можуть бути передбачені функцію безпеки по відношенню до даних. В цілому, рішення сфокусоване на забезпечення безпеки поза рівнем даних класифікуются як системно-центровані. Якщо рішення сфокусовано на окремому визначеному рівні, його класифіковано відповідно до цього специфічного рівня. Наприклад, рішення спрямовані на поліпшення безпечної ізоляції між віртуальною машиною та гіпервізором можуть бути класифіковані як VM-орієнтовані рішення в області безпеки. З іншого боку, рішення, спрямовані на забезпечення безпеки даних усередині самих даних, як показано на лівій частині Рисунку 2.1, класифікуються як інформаційно-орієнтовані підходи. Рівні, показані на Рисунку 2.1, є типовими рівнями. Там може бути більше або менше рівнів в практичній системі, на основі фактичних потреб та реалізації.

Приклад другої класифікації показано на Рисунку 2.2. Існує три рівні відповідальності безпеки:

– Рівень сервіс провайдерів, де забезпечується безпека та здійснюється підтримка хмарних провайдерів.

– Рівень довіренних обчислень, де забезпечується безпека та організатором виступає третя сторона.

– Рівень даних, на якому забезпечується безпека та організаторами виступають власники даних.

Для комплексного рішення питання безпеки, всі ці три класифікації безпеки повинні бути інтегровані і адаптовані до моделі хмари. Проте, залежність одного рівня безпеки від іншого рівня, повинна бути зведена до мінімуму. Так, наприклад, функції безпеки, що забезпечуються на рівні інформаційно-орієнтованої безпеки не повинні покладатися на інші рівні безпеки, зокрема, рівень гіпервізора та апаратнний рівень, так як ці два рівні, в публічній хмарі, знаходяться під контролем провайдера хмари в буд-якій моделі. Крім того, з точки зору відповідальності, управління безпекою інформаційно-орієнтованого рівня має здійснюватися лише власником даних, так як дані в хмарі належать власнику даних незалежно від моделі хмари, що хостить їх.

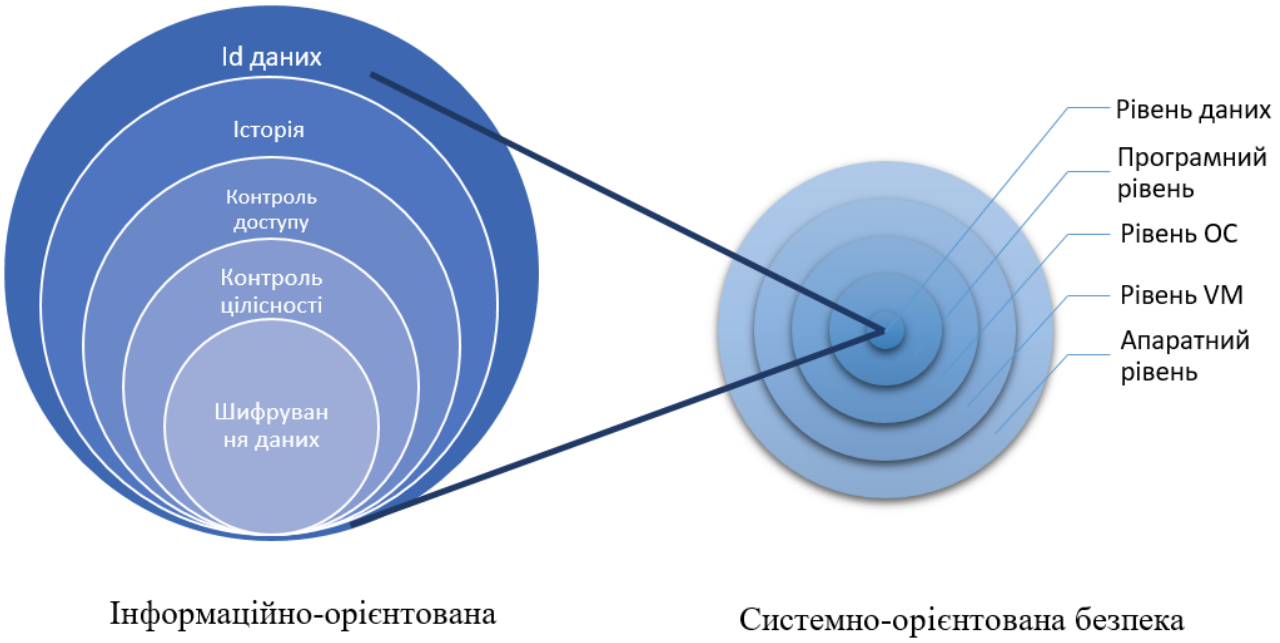

Рисунок 2.1 – Інформаційно-орієнтована та системно-орієнтована моделі

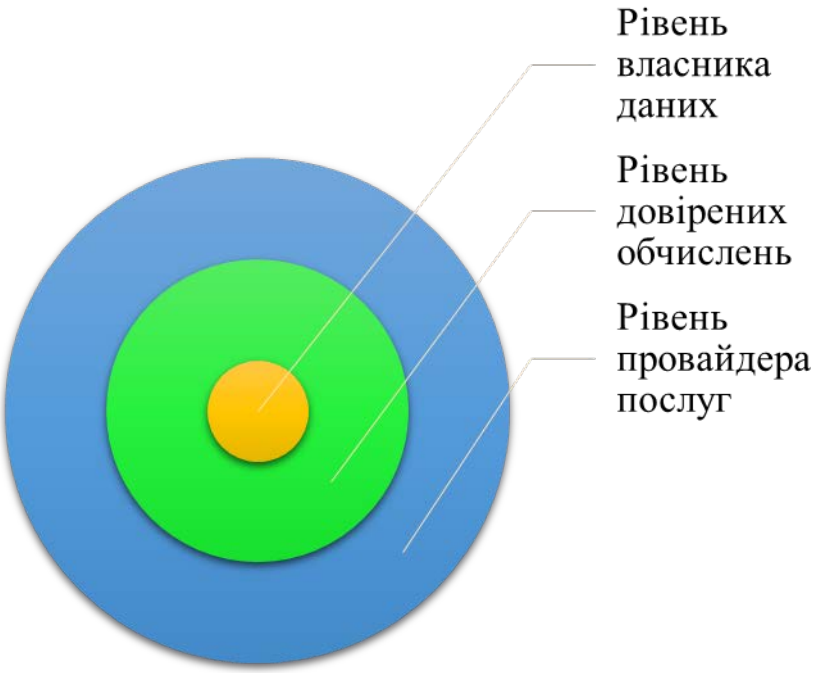

Рисунок 2.2 – Можливі рівні безпеки

З точки зору вивчення різноманітності поглядів в розумінні концепції ОБІ (або ОБІ), дослідники погоджуються зосередитися на інформації при розробці необхідного раціонального рівня безпеки. Однак у дослідників різні погляди на те, як забезпечити необхідний рівень. Чи надавати його за межами даних (додатки, обладнання і т. д., як показано в правій частині Рисунка 2.1. протягом його життєвого циклу) або всередині даних (як показано в лівій частині того ж рисунка)? Перші два посилання в списку орієнтовані на забезпечення безпеки за межами даних, оскільки концепція ОБІ застосовується до бізнес-процесів з використанням традиційних обчислювальних систем. Їх мета полягає в тому, що якщо система має кілька рівнів політик безпеки і механізмів, які розробляються незалежно, то система зможе застосувати належний рівень безпеки до інформації відповідно до її класифікації.

При застосуванні концепції ОБІ до хмарної обчислювальної системи мова йде про захист даних від самої хмарної системи. Отже, з одного боку, в деяких дослідженнях концепція ОБІ застосовується до хмари, все ще фокусуючись на забезпеченні належного рівня безпеки за межами даних, і вони намагаються захистити свої механізми безпеки і захищати дані від хмарної системи, яка розміщує

дані на основі концепції ТС. З іншого боку, в інших дослідженнях концепція ОБІ для хмарних обчислень базується на забезпеченні функцій безпеки через даних, тобто всередині даних, тому вона стає самоописовою, самозахищеною і самообороняємою. Однак в [7] ОБІ залежить від технологій ТС (НО) для оцінки надійності середовища, тому для забезпечення безпеки даних потрібен зовнішній захист даних. В орієнтація на безпеку інофрмації і збереження конфіденційності даних в хмарі представлена шляхом забезпечення конфіденційності даних і конфіденційності доступу до даних з використанням або підходу ТС (НО), або підходу ОБІ, де дані самозахищені. Що стосується відповідальності за безпеку, більшість уявлень про безпеку враховують, що власник даних відповідає за оцінку вимог безпеки даних і забезпечення безпеки даних до того, як захищені дані будуть відправлені в хмару. Однак після того, як дані перемістилися в хмару, дослідники не згодні з тим, хто несе відповідальність, тобто власник даних, провайдер хмарних обчислень або третя сторона, за підтримку і управління безпекою даних відповідно до концепції орієнтованої на безпеку інформації.

У цій роботі підхід ОБІ визначається як такий, що базується на забезпеченні вимог безпеки зсередини даних, тому захист даних і вся інформація, необхідна для захисту, прив’язані до даних. Більш того, вимоги до безпеки - відповідальність власників даних. Підхід ОБІ буде відповідати критеріям класифікації концепції ОБІ в двох вимірах: на якому рівні забезпечується безпека і хто несе за це відповідальність (див. Рисунок 2.1 і Рисунок 2.2). В наступному пункті описані основні функції безпеки підходу ОБІ, визначені в цій роботі на концептуальному рівні.

2.1.2 Огляд існуючих ОБІ рішень

На даний момент концепція ОБІ знаходиться на етапі зародження, проте стрімко розвивається. Систематичний перегляд літератури в напрямку пошуку реалізованих систем інформаційно-орієнтованої безпеки не дало плідних результатів. Проте можно відзначити роботу колег з Індії, які практично дослідили інформаційно-орієнтовану систему безпеки на прикладі розподіленої системи охорони

здоророзподіленої системи охорони здоровя. Значеність результатів дослідження підкріплюється авторитетом міжнародної конференції “Розподілених обчислень та мереж” [22].

Дослідники пропонують рішення, що здатне забезпечити конфіденційність, контроль доступу та цілісність. Дані спочатку сегментуються і кожен сегмент шифрується за допомогою статичних симетричних наборів ключів, а потім за допомогою динамічного симетричного ключа в момент передачі даних. Статичний ключ створюється за допомогою технології AES. Наступний симетричний ключ розроблений за допомогою методу ECDH. Це процес динамічно генерує ключ, тому не потрібно постійно зберігати ключ. Кожного разу цей ключ виводиться на обох кінцях.

Далі використовується асиметрична техніка позподілу ключів для генерації симетричного ключа використовуючи KDF.

2.1.3 Характеристики ОБІ

В цьому пункті підхід ОБІ обговорюється більш детально. Деякі з його характеристик виділені та описано для більш чіткого розуміння концептуальних вимог ОБІ. Всі ці характеристики пов’язані з проблемами конфіденційності та безпеки щодо захисту даних клієнтів у хмарних обчисленнях. Традиційно безпека в основному відноситься до вирішення трьох основних проблем: конфіденційності, цілісності та доступності. Як правило, модель CIA використовується в якості скорочення для цих трьох важливих вимог безпеки. З точки зору безпеки даних, вимоги CIA стосуються конфіденційності даних, цілісності даних і доступності даних. У хмарних обчисленнях або будь-яких інших обчислювальних системах, вимоги CIA також застосовуються до системи щодо конфіденційності системи, цілісності системи і доступності системи.

2.1.4 Класифікація даних

Власник даних зазвичай є найкращим суб’єктом для оцінки вимог безпеки своїх власних даних. Наприклад, якщо дані підключені до бізнес-моделі, вимоги до безпеки даних повинні бути результатом аналізу їх використання в бізнес-процесі. Як правило, дані повинні оброблятися відповідно до їх значенням, оскільки наступний принцип безпеки проголошує: «цінність того, що захищається, впливає на заходи, прийняті для його захисту». Отже, дані класифікуються на основі оцінки власника даних та вимог до безпеки. Наприклад, дані можуть бути просто класифіковані як цілком таємні, секретні або конфіденційні. Виходячи з цього, політики контролю доступу та властивості безпеки, необхідні для обробки даних, надаються відповідно до цих вимог безпеки. Класифікація даних на концептуальному рівні може бути представлена як інформація, приєднана до даних, або представлена на вимогу необхідних заходів безпеки, які застосовуються до даних.

2.1.5 Політики доступу до даних і їх застосування

Політика контролю доступу - це основна функція безпеки, в якій вимоги безпеки задаються для кожного набору даних відповідно до політик безпеки, які можуть включати будь-які потенційні юридичні зобов’язання. Ці політики включають правила і обмеження, які контролюють доступ, використання і потік даних. Основа концепції полягає в контролі того, хто або що може отримати доступ до даних і з якими наданими правами, наприклад, читати, писати, які відповідають традиційній концепції політики контролю доступу. Питання полягає не тільки у визначенні цих політик, але і в забезпеченні дотримання цих політик безпечним чином, який зберігає конфіденційність користувачів при їх доступі або спільному використанні даних. У підході ОБІ політики контролю доступу та механізм їх застосування визначається власником даних, а хмарна система несе відповідальність за дотримання механізму примусового виконання. Завдання полягає в тому, як політики контролю доступу та їх примусове застосування можуть бути поширеними

і легко модифікованими для адаптації до динамічних змін вимог до використання і спільного використання даних в хмарному обчислювальному середовищі. Оскільки у кожного набору даних є свої політики доступу, виникають інші складні проблеми щодо взаємодії різних політик контролю доступу та права власності на новий набір даних, який створюється в хмарі від обробки декількох наборів даних з різними політиками і власниками даних.

2.1.6 Самозахист

Конфіденційність вмісту набору даних захищена від будь-яких неавторизованих об’єктів, навіть хмарних провайдерів. Щоб самозахистити конфіденційність даних, вони шифруються за допомогою достатньої технології шифрування, і вони залишаються зашифрованими до тих пір, поки вони не будуть переведені в повністю надійний домен і не будуть розшифровані авторизованим користувачем. Тільки авторизовані користувачі можуть мати доступ до секретного ключа, який прикріплений до набору даних, а також захищений. Зашифровані дані упаковуються з їх параметрами безпеки, такими як захищений секретний ключ і політика використання даних. Тому кожен із самозахищених наборів даних має свої політики доступу, і ці політики консультуються, коли авторизований користувач запитує доступ до захищених даних. За допомогою самозахисту, де політики доступу впроваджені в дані, доступ до захищених даних може бути виконаний в будь-якому місці, якщо існує механізм для виконання політик вбудованого доступу. Механізм контролю доступу буде рудиментарним в його реалізації в локації. Тільки власник даних може керувати політиками контролю доступу та іншими параметрами і функціями, пов’язаними з самозахистом, характерним для кожного набору даних.

2.1.7 Перевірка цілісності даних

Дуже важливо, щоб цілісність набору даних могла бути перевірена на будь-якому етапі. При зберіганні в загальнодоступному домені набір даних піддається модифікації неавторизованими об’єктами, яким може бути навіть сам хмарний

провайдер, випадково або навмисно протягом життєвого циклу набору. Тому власники даних і авторизовані користувачі вимагають, щоб ця функція гарантувала, що дані не будуть некоректно змінені. У підході ОБІ інформація для перевірки цілісності даних вбудована в дані і також захищена. Перевірка не залежить від інформації, наданої поза самими даними. Отже, пряме підтвердження цілісності даних в хмарі без необхідності завантаження, це великий виклик, особливо якщо дані динамічно змінюються в хмарі.

2.1.8 Приватність та конфіденційність

Конфіденційність означає, що тільки уповноважені сторони мають можливість доступу до захищених даних з відповідними правами і привілеями, визначеними власником даних. Ризик компрометації конфіденційності даних зростає в хмарній моделі через збільшення числа сторін, включаючи інших користувачів в одній хмарі. Крім того, управління даними делегується провайдеру хмари, а хмарам не вистачає апаратного поділу між користувачами. Це призводить до збільшення ризику компрометації конфіденційності, оскільки дані стають неконтрольованими власниками даних і доступні для інших сторін.

Приватність має різні визначення в літературі. Приватність може використовуватися як синонім конфіденційності, в той час як деякі дослідники розглядають приватність як іншу концепцію. Концепція приватності розглядається по-різному навколо слова, і це відображено в різних законах про приватність. Приватність видається більш широкою концепцією, ніж конфіденційність, оскільки вона охоплює те, що на основі закону або культури вважається приватною інформацією, що стосується організацій і людей, навіть якщо ця інформація сама по собі не є конфіденційною. Ця концепція широти може бути знайдена в одному з визначень конфіденційності, наданих Американським інститутом сертифікованих громадських бухгалтерів і Канадським інститутом дипломованих бухгалтерів в стандарті загальноприйнятих принципів конфіденційності: «права і обов’язки окремих осіб і організацій щодо збирання, використання, утримання та розкриття

особистої інформації». Наприклад, ім’я або місцезнаходження звичайної особи зазвичай не вважається конфіденційним, але в деяких випадках воно може вважатися приватним. Тому захист приватності йде далі, ніж захист фактичних даних від несанкціонованого доступу. Він також запобігає витокам будь-якої інформації, яка розглядається як приватна інформація під час обробки даних. Наприклад, прикріплені політики контролю доступу вважаються приватною інформацією, і провайдер хмари не повинен розкривати їх. Застосування такого виду захисту до процесу управління доступом призводить до підходу до доступу до конфіденційності, який може використовуватися для запобігання витоку навіть часткової інформації про дані під час процесу доступу. Тому будь-яке рішення безпеки має враховувати конфіденційність даних і приватність хмарних клієнтів у відповідності з різними правилами та вимогами.

2.1.9 Доступність

У хмарних обчисленнях доступність відноситься до послуг хмарних обчислень, наявних і доступних для авторизованих користувачів, коли вони їм потрібні. Крім того, ці служби можуть бути пов’язані з обробкою критично важливих даних і запуском важливих додатків, які повинні бути доступні весь час і здатні обслуговувати велику кількість користувачів. Доступність послуг хмарних обчислень залежить від постійної безперервної здорової роботи інфраструктур і мережевих ресурсів. Тому хмарний провайдер повинен мати надійні і надлишкові стратегії для забезпечення доступності системи. Хмарний провайдер також несе відповідальність за обслуговування своїх сервісів, навіть якщо є внутрішня або зовнішня загроза, націлена на доступність системи. Іншою проблемою доступності є доступність даних клієнта для доступу власником даних або авторизованими користувачами. Авторизований користувач повинен мати можливість отримувати дані з хмари в будь-який час. Як згадувалося раніше, на етапах життєвого циклу даних, постачальник хмари повинен мати ефективну стратегію резервного копіювання та архівування для захисту доступності даних. З точки зору власника даних, дані є найважливішим

активом, яким він володіє в хмарі. Пропонований ОБІ підхід в цій роботі дає власнику даних більш істотну відповідальність і контроль над захистом своїх даних, які повинні зберігатися в середовищі хмарних обчислень.

2.1.10 Концептуальна основа ОБІ

В цьому пункті створено цілісну концептуальна основа ОБІ. Ця основа базується на огляді різних попередніх дослідницьких робіт щодо концепцій ОБІ, застосовуваних до хмарної моделі, і на основі характеристик ОБІ, визначених раніше у пункті 1.1.2. В рамках основи, що базується на концепції, дані захищені перш ніж покинути довірений домен власника даних, а їх параметри безпеки прив’язані до даних як частина їх метаданих. В ідеальній ситуації тільки облікові дані, які використовуються для відміни або доступу через захист, зберігаються за межами даних, а також власником даних і авторизованими користувачами відповідно до їх прав доступу. Будь-які інші параметри безпеки даних повинні бути прив’язані до даних. Наприклад, в тих випадках, коли дані динамічно оновлюються, користувач, який звертається до даних, повинен перевірити, чи дані є найсучаснішими. Ця вимога має бути надана з фактичних даних або метаданих, прикріплених до даних, наприклад, функція історії є під-рівнем від рівня даних, як показано на Рисунку 2.1. Як інший приклад, доказ достовірності джерела даних також може бути додано разом з доказом цілісності, що зазвичай включає цифровий підпис власника даних. Будь-яка додаткова функція може бути додана як новий під-рівень під рівнем даних або включена в якості одного зі зразкових підрівнів, проілюстрованих в лівій частині Рисунку 2.1. Крім того, всі ці метадані безпеки надаються на рівні даних і створюються власником даних. Вони також залишаються захищеними протягом усього життєвого циклу даних. У окремого набору даних є свої метадані безпеки, незалежні від інших наборів даних. У наступному списку наведено критерії, які повинні задовольняти будь-яке надане рішення безпеки на основі підходу ОБІ:

– Кожен набір даних є самоописовим, самозахищеним і самоохороняемим (тобто захищеним набором даних). Отже, вимоги та функції безпеки кожного набору

даних надаються зсередини і не залежать від установок поза набором даних, крім деяких базових процесів обробки даних.

– Захист даних не залежить від провайдера хмарних обчислень або довіреної третьої сторони (TTP).

– Тільки власник даних відповідає за створення і управління цими вимогами і функціями безпеки для кожного набору даних з моменту його створення до кінця життєвого циклу набору даних.

– Всі операції, пов’язані з доступом до захищених даних, встановлені авторизованими користувачами, і примусове застосування політик безпеки для даних виконуються без шкоди для конфіденційності користувачів або конфіденційності даних.

Приклад концептуальної основи ОБІ для моделі хмарних обчислень показаний на Рисунку 3.3. Створення даних виконується в надійному домені з боку власника даних. На етапі створення дані класифікуються на основі вимог безпеки і, відповідно, метадані безпеки прив’язані до даних, наприклад, їх політика контролю доступу, перевірка цілісності, запис їх історії і можливості відстеження. На етапі створення, створюється безпечний набір даних і він інкапсулює вміст захищених даних і відповідні метадані. Контролер управління даними і відстеження дозволяє власнику даних управляти і відстежувати набір даних, наприклад, розташування, використання та історію доступу, з моменту створення, поки він не буде видалений з хмарного середовища. Крім того, передача між надійним доменом, в якому знаходиться користувач, і хмарним доменом зазвичай здійснюється через Інтернет і захищена основним механізмом захисту, забезпечуваним підходом ОБІ або протоколом інтернет-безпеки. У захищеного набору даних протягом його життєвого циклу в хмарі є можливість, виконавши додані до нього коди, інформувати власника даних, якщо є будь-які зміни його стану. У той же час захищений набір даних включає в себе виконуваний код, готовий для прийому і відповіді на запити власника даних для відстеження інформації і команд управління. Навіть якщо зв’язок між хмарним доменом і надійним доменом не працює, оскільки набір даних містить всю необхідну

інформацію для виконання основних політик безпеки, пов’язаних з даними, захищений набір даних може бути постійно доступний авторизованим користувачам без участі власника даних.

Як показано на Рисунку 2.3, примусове застосування політик доступу буде виконуватися ресурсами хмарного сервера в хмарному домені, але на основі параметрів безпеки, прикріплених до набору захищених даних. Правило провайдера хмарних обчислень в примусовому порядку дозволяє виконання цих політик доступу, не розкриваючи деталі політики контролю доступу або вміст набору даних. Власник даних може перевірити цілісність вмісту захищених даних, включаючи його історію записів.

Коли захищений набір даних повинен бути отриманий авторизованим користувачем, хмарний сервер в хмарному домені перевірятиме права доступу користувачів до даних з використанням пов’язаних параметрів безпеки. Авторизованим користувачам дозволено завантажувати захищений набір даних. Потім авторизовані користувачі можуть перевірити цілісність і оригінальність завантажених даних, використовуючи додані докази цілісності та аутентичності. Ця концептуальна основа ОБІ дозволяє, з одного боку, оптимізувати конфігурацію безпеки параметрів безпеки для кожного набору даних, а з іншого - знижує складність загального управління безпекою. Це особливо важливо, коли відповідальність за хмарну безпеку також оптимально розподілена між потенційними сторонами хмарної системи (тобто провайдерами хмарних обчислень, користувачами і ТТР) відповідно до трьох рівнів відповідальності безпеки, проілюстрованих на Рисунку 3.2. Наприклад, хмарні провайдери несуть відповідальність за захист своїх систем, ТТР, відповідальних за активи, заходи безпеки, що надаються провайдерами, і власники даних несуть відповідальність за захист своїх даних в хмарі. Інші очікувані переваги застосування ОБІ до хмарної моделі обговорюються в наступному пункті.

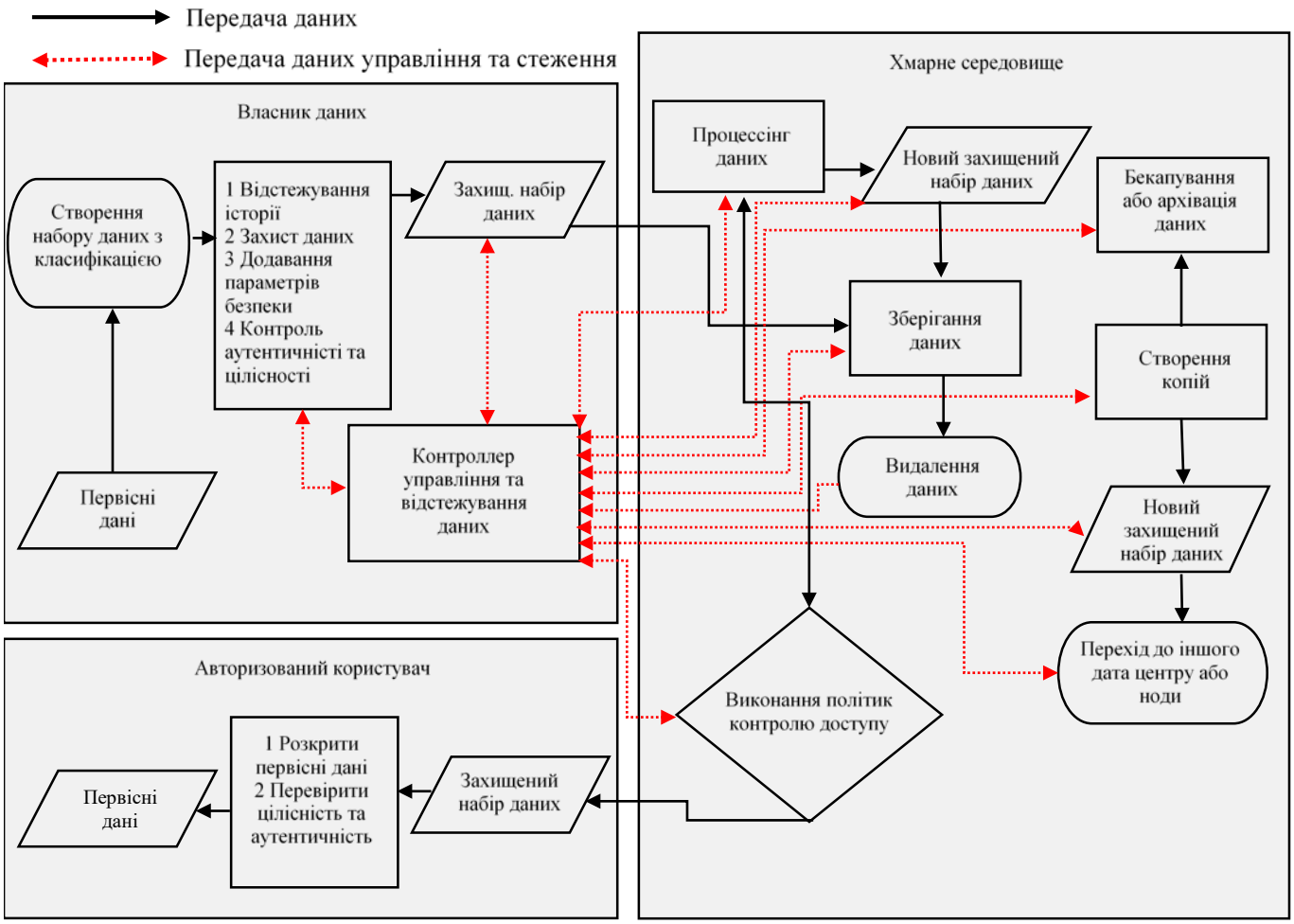

Рисунок 3.3 – Основа ОБІ

2.2 Висновки

У цьому розділі наведено відомості підхід ОБІ і його застосування до хмарної моделі. У цій дисертації підхід ОБІ грунтується на трьох фундаментальних концепціях:

– Вимоги безпеки забезпечуються всередині даних.

– Власник даних відповідає за ці вимоги безпеки протягом усього терміну служби даних.

– Вимоги безпеки не залежать від заходів безпеки, що надаються поза межами даних.

Ці концепції відрізняють підхід ОБІ від інших підходів для забезпечення безпеки і приватності даних в хмарі. Інші підходи забезпечують заходи безпеки на рівні обладнання, платформи або програми, спираючись на хмарного провайдера і/або третю сторону, включаючи технології ТС. На противагу цьому, підхід ОБІ забезпечує рішення для забезпечення безпеки і приватності даних для покриття всього життєвого циклу даних. Відповідно, кожен набір даних є самоописовим, самозахисним і з усіма доданими до нього специфікаціями безпеки, і виконання цих специфікацій здійснюється безпечним чином.

Протягом усього життєвого циклу наборів даних власник даних може керувати своїми специфікаціями безпеки і відстежувати їх, включаючи історію використання. Найголовніше, що на першому етапі життєвого циклу даних, етапі створення набору даних, власник даних відповідає за побудову кожного набору даних з усіма вимогами безпеки перед відправкою його в хмару. На тих етапах що залишилися до етапу знищення набору даних підтримується відповідність політикам безпеки власників даних. Очікується, що застосування такого підходу до хмарної моделі поліпшить приватність даних і безпеку в хмарі, щоб відповідати її масштабованості і еластичності. Однак повне застосування ідеальної концептуальної основи ОБІ, як і раніше, ускладнено декількома проблемами. Сфера застосування підходу ОБІ в цьому дослідженні визначається для обмеження таких проблем. Хоча область охоплення обмежена неструктурованими даними, що зберігаються і розділяються в публічному хмарному сховищі, пропоноване рішення охоплює найбільш поширені ситуації в хмарних сервісах. Більш того, рішення може бути змінено для застосування в інших формах даних і в інших хмарних моделях. Основні вимоги безпеки для пропонованого рішення перераховані в Пункті 2.6. Вони в основному пов’язані із захистом приватності і цілісності при доступі і пошуку зашифрованих даних, що зберігаються на хмарних серверах. На основі концепцій ОБІ розглядаються методи безпеки, які можуть задовольняти ці вимоги. Методи, що відповідають вимогам безпеки, будуть адаптовані і використані в запропонованому рішенні ефективним і практичним способом. У наступному розділі запропоноване рішення детально пояснюється.

3 ІНФОРМАЦІЙНО-ОРІЄНТОВАНЕ РІШЕННЯ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПРИВАТНОСТІ ТА БЕЗПЕКИ В ХМАРІ

Цей розділ описує запропоноване рішення щодо основних дослідницьких запитань даної роботи. Це рішення має за основу ОБІ підхід описаний в Розділі 2. Мета даного розділу полягає в тому, щоб розробити рішення, яке робить оригінальний дослідницький внесок в концепцію ОБІ, що вважається адекватним підходом до вирішення питань безпеки та конфіденційності в хмарних обчисленнях. Рішення призначене для задоволення наступного переліку бажаних вимог до додатків в середовищі хмарних обчислень:

– Дані зашифровані та доступні лише для авторизованих користувачів.

– Дані доступні для пошуку без загрози для їх конфіденційності.

– Дані відповідають вимогам самозахисту та необхідним параметрам безпеки.

– Параметри контролю доступу приховані від провайдерів хмарних серверів та інших користувачів.

– Провайдер серверу не знає кількість або особистість користувачів, що авторизовані та мають право доступу до даних.

– Несанкціоновані об’єкти, в тому числі постачальники послуг, не можуть отримати доступ до даних або отримати інформацію про дані від авторизованих процесів що проводяться на даних.

– Дані містять всю необхідну інформацію для перевірки їх цілосності для авторизованих користувачів, які мають доступ до даних.

– Взаємодія між власниками та авторизованими користувачами має бути мінімальною, особливо щодо ключових цілей управління.

Ці вимоги визначені набором модулів, кожен з яких точно описаний принаймні однією із зазначених вище вимог. Всі параметри, необхідні для досягнення функцій безпеки прикріплюються до файлу даних і в результаті є файл з назвою ОБІ-файл. Функції безпеки включають у себе захист конфеденційності, перевірку цілесності та

аутентифікацію. Цей розділ починається з опису умов китайської теореми про залишки (КТЗ), що є основним алгоритмом запропонованого рішення. Пункт 3.2 описує модуль який використовується для забезпечення і управління параметрами необхідними для контролю доступу та спільного використання ключа з використанням КТЗ. Потім пункт 3.3 пояснює яким чином включено в алгоритм можливість до пошуку і як вона інтегрована з процедурою контролю доступу. Пункт 3.5 описує архітектуру всіх модулів залучених в генерацію ОБІ-файлу на основі даних з їх цілісністю та перевірками аутентифікації. Нарешті, в останньому пункті розглядаються вигоди від запропонованого рішення і здійснено підсумок даного розділу.

3.1 Китайська теорема про залишки та її застосування

КТЗ було засновано на античних манускриптах авторства китайських математиків для аналізу арифметичних задач [10]. Вона відома як одна із найдавніших математичних теорем, яка використовувалася для створення календарів протягом першого століття нашої ери. КТЗ розвивали математики по всьому світу, поки вона не досягла сучасного свого вигляду формули.

Теорема 1. Китайська теорема про залишки [11]

Для будь-яких заданих цілих чисел a_1, a_2, …, a_k, наступна система одночасної конгруенції має унікальне рішення X, яке лежить в межах $0 \leq X < n = n_1 n_2 \ldots n_k$, за умови, що невід'ємні цілі числа $n_1, n_2, \ldots n_k$ взаємно прості.

$$\begin{aligned} X &\equiv a_1 \bmod n_1 \\ X &\equiv a_2 \bmod n_2 \\ X &\equiv a_k \bmod n_k \end{aligned} \tag{3.1}$$

Якщо X дано в наведеній вище конгруентності, його можна вирахувати за формулою:

$$a_i = X \bmod n_1 \tag{3.2}$$

для i = 1,2, …, k.

Унікальне рішення X для одночасної конгруентності можна розрахувати за наступним рівнянням:

$$X = \sum_{i=1}^{k} a_1 \; M_i \, M_i^{-1} \text{ (mod M)}$$

де $M = n_1 \, n_2 \ldots n_k$, $M_i = M/n_1$

M_i^{-1} є зворотним $M_i \bmod n_i$, тобто $M_i M_i^{-1} \equiv 1 \; (mod \; n_1)$

Через те що M_i відносно первинне до n_i, є унікальний мультиплікативний зворот $mod \; a_i$. Отже, розрахунок мультиплікативного зворотного в модулі є істотною частиною визначення рішення КТЗ. Розширений алгоритм Евкліда [12] може бути використаний для обчислення мультиплікативного звороту та є істотним компонентом алгоритму Гарнера [13], який зазвичай використовується для визначення рішення КТЗ.

Нижченаведений алгоритм 3.2 використовується для запропонованого рішення для визначення рішення КТЗ:

Вхід: додатні цілі числа $n = \prod_{i=1}^{k} n_i > 1$, де gcd ($n_i$, n_j) = 1 для всіх $i \neq j$, та модульним виразом $a_i = a_1, a_2, \ldots, a_k$, якими можуть бути будь-які цілі числа.

Вихід**:** ціле число X як основа системи вирахування b виразу.

Кроки алгоритму:

1. Для i = 2 до k справедливе:

   1.1.$C_1 \leftarrow 1$.

   1.2.Для j = 1 до (i - 1) справедливе:

1.2.1. $u \leftarrow n_j^{-1} \, mod \, n_i$.

1.2.2. $C_i = u \cdot C_i \, mod \, n_i$.

2. $u \leftarrow a_i, X \leftarrow u$.
3. Для i = 2 до k справедливе:

3.1. $u \leftarrow (a_i - X) \cdot C_i \, mod \, n_i, X \leftarrow X + u \cdot \prod_{j=1}^{i-1} n_j$.

4. Повернути (X).

де n_j^{-1} це мультиплікативний зворот n_j в модулі n_i.

Однією із корисних функцій КТЗ є те, що мінімальна зміна для поточної відповідності це додавання нового модулю до конгруентності. Наприклад, якщо рішення X для Рівняння (3.1) вже було знайдено та якщо новий модуль $a_{k=1} \, mod \, n_{k+1}$ додається до конгруентності, нехай рішення для нової конгруенції буде X'. Нове рішення може бути знайдене шляхом обчислення рішення наступних двох конгруенцій:

$$X' \equiv X \, mod \, n_1 n_2 \ldots n_k \text{ та}$$
$$X' \equiv a_{k+1} mod \, n_{k+1}.$$

Щоб прийти до нового рішення не має необхідності оцінювати інші конгруентності.

Простіше, якщо існуючу конгруентність видалити з конгруентностей, нове рішення може бути отримане простим розрахунком. Наприклад, нехай нове рішення отримає позначення X'' після останньої конгруентності $X \equiv a_k \, mod \, n_k$ у Рівнянні (3.1) буде виключено. Нове рішення може бути вирахуване дією з одним модулем

$$X'' \equiv X \, mod \, n_1 n_2 \ldots n_{k-1}$$

Ці функції можуть бути використані для мінімізації додаткових обчислень розрахунку рішення КТЗ в таких випадках. Наприклад, ця функція використовується

в запропонованому рішенні в цьому пункті при видачі прав доступу новому користувачу до певних ресурсів. Це питання буде обговорюватися в пункті 3.3.3.

КТЗ використовує кілька додатків в криптографії та зв’язок захищеними сітями через свою арифметичну природу, що не споживає високих обчислювальних ресурсів. Таким чином, він підходить для обчислювальних і комунікаційних додатків з обмеженим ресурсами. Розділення секрету є однією із основних функцій безпеки з використанням КТЗ в криптографії. Це дозволяє виконувати поділ виводячи з нього декілька спільних, які можуть бути відновлені тільки з певним заздалегідь підготовленим набором значень. Багато додатків засновані на цій схемі розділення секрету, такі як порогова криптографія та електронне голосування.

У бездротових мережах, де пристрої можуть мати обмежені ресурси, особливо обчислювальні ресурси і ресурси зберігання, запропоновані рішення на основі КТЗ можуть бути використані в декількох схемах. Зокрема, безпосередньо при аутентифікації та управлінні доступом до схем. Наприклад, бездротова сенсорна мережа застосована в аутентифікації на основі КТЗ. В іншому прикладі схема управління доступом запропонована і заснована на КТЗ для бездротового мультимедійного сенсорного датчика мережі. КТЗ також використовується для зменшення рівня споживаної енергії при передачі пакетів між вузлами бездротових сенсорних мереж, які призводять до збільшення тривалості життя та енергоефективності мережі. Спеціальні мобільні мережі та інші бездротові системи, де КТЗ запропоновано використовувати для забезпечення безпеки з меншим впливом на системні ресурси. Існує також непряме використання КТЗ в додатках для прискорення та зниження споживання ресурсів при генерації ключів.

3.3 Управління доступом та розділення ключа використовуючи КТЗ

У даній роботі, запропоноване рішення використовує КТЗ і криптосистема з публічним ключем для безпечного обміну даними захищених користувачів, що

зберігаються в середовищі хмарних обчислень серед авторизованих користувачів [16]. Дані, які іноді згадуються як ресурси або файли, захищені симетричним методом шифрування, де як секретний ключ використовується K_s. У даній роботі, ресурсом може бути набір даних або файл, який містить дані будь-якого типу, в тому числі текст, аудіо, зображення або відео. Для поширення секретного ключа для авторизованих користувачів, його зашифровано з використанням методу публічного шифрування відкритого ключа користувача. Шифрування також включає в себе інше значення, C_r, яка буде використовуватися в якості відповіді на запит сервера, коли користувач робить запит щодо доступу до ресурсу. Лише авторизовані користувачі, які мають відповідний приватний ключ, можуть розшифрувати шифротекст C_r та K_r для отримання C_r, щоб отримати доступ до ресурсу r. Параметри C_r та K_s зчеплені для формування $C_r||K_s$ та розглядаються як єдине значення. Це значення шифрується за допомогою публічного ключа $E_{K_{pub\,i}}$ кожного авторизованого користувача u_i, в результаті шифротекст $\left(E_{K_{pub\,i}}(C_r||K_s)\right)$ для цього користувача u_i, де $i=1, 2, 3, \ldots, k$, та k кількість авторизованих користувачів, що мають доступ до ресурсу r.

Щоб застосувати КТЗ до запропонованого рішення, для користувачів u_1, u_2, ..., u_k, кожен авторизований користувач пов’язаний з унікальним відносним простим числом $n_i = n_1, n_2, ..., n_k$, де k це кількість авторизованих користувачів. Всі n_i, $1 \leq i \leq k$, є відносні прості числа. Потім, шифротекст $C_r \,||\, K_s$ генерується для кожного користувача, тобто $\left(E_{K_{pub\,i}}(C_r||K_s)\right)$, використовується для заміни a_i у Рівнянні (3.1) для виведення наступної спільної конгруентності:

$$\begin{aligned} X_r &\equiv \left(E_{K_{pub\,1}}(C_r||K_s)\right) \; mod \; n_1 \\ X_r &\equiv \left(E_{K_{pub\,2}}(C_r||K_s)\right) \; mod \; n_2 \\ X_r &\equiv \left(E_{K_{pub\,k}}(C_r||K_s)\right) \; mod \; n_k \end{aligned} \tag{3.3}$$

Вирішення цієї конгруенції Xr, таке, що 0 ≤ Xr <n = n1 n2 ... nk, є загальним значенням для ресурсу r і він приєднаний до ресурсу, а ресурс і загальна значення зберігаються разом в хмарному сервері. Коли авторизований користувач ui надсилає запит щодо доступу до ресурсу r, хмарний сервер надсилає йому розділене значення Xr. Коли користувач отримує Xr, він використовує відповідний private key для розшифровки Xr, щоб отримати значення Cr||Ks, як показано в Рівнянні (3.4) нижче:

$$\left(\mathrm{Cr}||\mathrm{Ks} = \ D_{K_{prv\,i}}\,(X_r mod\ n_i)\right) \tag{3.4}$$

де $D_{K_{prv\,i}}$ це операція дешифрування з використанням приватного ключа користувача u_i, та n_i як відносного простого числа специфічного для даного користувача. Значення C_r потім вирушає назад до серверу користувачем, щоб довести володіння правом доступу до ресурсу. Потім сервер надсилає ресурс для користувача і ключ K_S, який використовується користувачем для щоб розшифрувати файл та отримати його вміст.

3.3.1 Підвищення безпеки ключів та файлів

З точки зору підвищення безпеки в запропонованому рішенні, власник даних повинен використовувати унікальний симетричний ключ, K_S, для шифрування кожного файлу перед відправкою його на зберігання в хмару. Відповідно до ОБІ підходу, всі вимоги до безпеки прикріплені до фактичних даних, отже, кожен симетричний ключ прикріплюється до його файлу. З Рівняння (3.3), симетричний ключ K_S захищений двома рівнями: спочатку шифруванням його авторизованим користувачем, а потім за допомогою КТЗ, щоб знайти загальне значення X_r. Лише авторизовані користувачі можуть обчислити K_S з X_r за допомогою Рівняння (3.4), якому потрібен відповідний n_i та приватний ключ авторизованого користувача. Для ефективного управління ключами, власник даних додає себе в якості користувача при розрахунку X_r для кожного файлу за допомогою Рівняння (3.3). Отже, ані власник даних, ані користувачі не зобов'язані зберігати K_S, але їм потрібен тільки їх приватний

ключ і призначене значення n_i. Таким чином, ключ K_S надійно та ефективно, спільно з авторизованими користувачами, захищено від несанкціонованого доступу, включаючи постачальника послуг хостингу файлу.

Секретний ключ K_S, а також значення C_r є унікальними для кожного файлу. Якщо значення для одного файлу було скомпрометовано, інші файли залишаються в безпеці. Секретне значення C_r використовується авторизованим користувачем, щоб показати серверу, що він або вона має право отримати доступ до ресурсу r (тобто файлу). Власник даних надійно прикріплює секретне значення C_r до файлу, а також надсилає його всередині шифротекст X_r до сервера. Лише авторизовані користувачі можуть розрахувати секретне значення C_r з використанням розділенного значення X_r. Таким чином, лише авторизовані користувачі можуть знати і відкрити C_r серверу і довести, що вони мають право доступу до цього конкретного файлу. Секретне значення C_r є унікальним для кожного файлу, навіть якщо він використовується одним користувач для різних файлів. Таким чином, якщо один C_r для конкретного файлу скомпрометований, ніякі інші файли не будуть під загрозою. Крім того, ця функція корисна, якщо власник даних хоче змінити деталі щодо авторизованих користувачів для файлу, наприклад, щоб додати нового авторизованого користувача, власнику даних лише необхідно змінити X_r для цього файлу. Оскільки параметр C_r може залишатися таким же самим, немає необхідності для повторного надсилання нового C_r до серверу. Це, в свою чергу, дозволить знизити ймовірність компрометації цього значення в процесі підтримки динамічного механізму оновлення списку авторизованих користувачів. Проте, при забороні користувачу доступу до файлу, до якого він/вона вже мають доступ, значення C_r має бути змінене з міркувань безпеки. Більш детальна інформація описана в Пункті 3.3.3.

3.3.2 Безпека процедури контроля доступу

Запропоноване рішення поєднує в собі контроль доступу та спільного використання ключа в одному механізмі з використанням КТЗ. Хмарний сервер зберігає зашифровані дані з відповідним секретним C_r і розділеним значенням X_r, що

розраховується власником даних. Провайдер може прочитати розділене значення X_r, але не може дізнатися K_S, який було використано для шифрування даних. Лише авторизовані користувачі можуть виявити K_S від X_r. Провайдер також може зчитувати секретне значення C_r для кожного файлу. Проте, число або ідентичність користувачів які можуть отримати доступ до файлу, навіть якщо вони вже отримали доступ до файлу, залишаються прихованими від провайдера.

Для полегшення необхідних розрахунків, можна відзначити, що, коли авторизований користувач u_i має доступ до файлу r, власник даних вже повинен був надіслати його відносне просте n_i призначене для нього, яке відправляється один раз для кожного користувача. Тоді власник даних використовує його з відкритим ключем користувача для всіх файлів, правом доступу до яких володіє користувач. Для більшої безпеки, n_i повинен бути надісланий користувачеві надійно, наприклад, за допомогою шифрування його з відкритим ключем користувача. Таким чином, зловмисникам буде важче виявити шифротекст $\left(E_{K_{pub\,k}}(C_r||K_s)\right)$ від спільного значення X_r. Проте, навіть якщо n_i будь-якого користувача розкривається несанкціонованим об'єктам, файли цього користувача залишаються в безпеці до тих пір, поки не буде скомпрометовано приватний ключ користувача. Тому, перед тим, як користувач має право отримати доступ до файлу, власник даних файлу має відкритий ключ користувача при розрахунку X_r файлу і користувач отримав його/її n_i.

Повідомлення, якими обмінюються авторизовані користувачі і хостинг хмарним сервером загальних файлів показано на Рисунку 4.1. У пропонованому рішенні, коли користувачу u_i потрібен доступ до файлу r, користувач надсилає запит на сервер, що містить ID файлу, ID_i власника файлу, нонс (будь-яке випадкове число), і сесійний ключ K_t. Все шифрується за допомогою відкритого ключа сервера $E_{K_{pub\,S}}$. Результат можна представити у вигляді $E_{K_{pub\,S}}(ID_r, ID_i, K_t, \text{нонс})$. Сесійний ключ K_t створюється користувачем для безпечного обміну інформацією між користувачем і сервером. Сервер знаходить файл з ID_r, читає його спільне значення X_r, збільшує нонс до (*нонс* +1) і шифрує їх за допомогою сесійного ключа K_t. Результат, можна представити як $E_{K_t}(X_r, \text{нонс} + 1)$, який повертається користувачу.

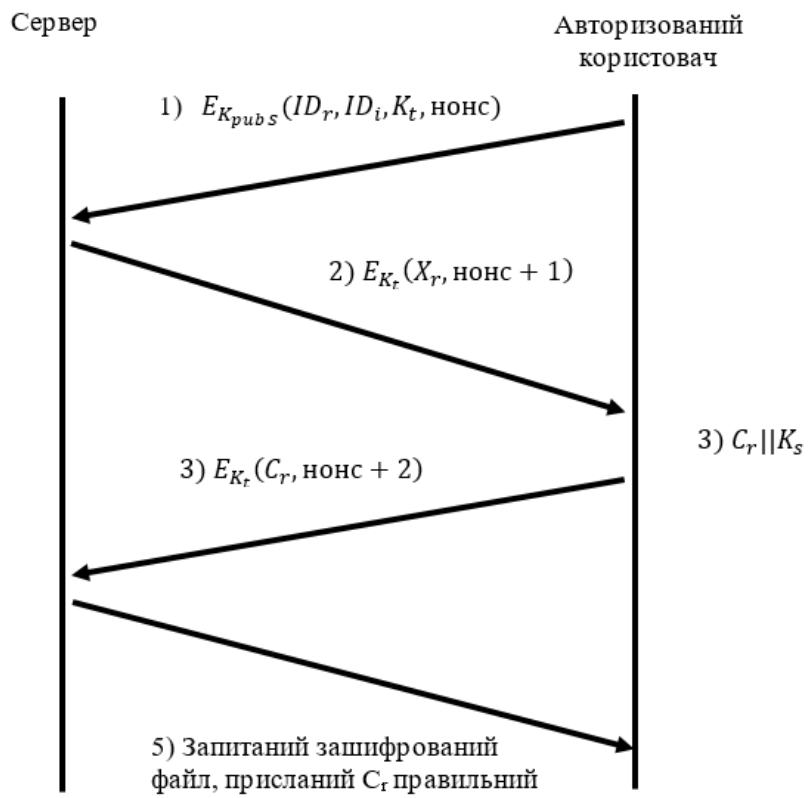

Рисунок 4.1 – Процедура контролю доступа

Після отримання інформації, представленої в повідомленні номер два на Рисунку 3.1, з сервера, користувач обчислює секретний параметр C_r, використовуючи Рівняння (3.4) і повертає C_r та (нонс + 2) на сервер і шифрується сесійний ключ K_t, тобто $E_{K_t}(C_r, \text{нонс} + 2)$. Сервер перевіряє C_r і, якщо він збігається з тим, що прикріплений до файлу на який було надіслано запит, сервер надсилає зашифрований файл користувачеві. Користувач може розшифрувати файл за допомогою ключа K_S, який було отримано раніше в результаті використання Рівняння (3.4) на третьому етапі на Рисунку 4.1. Всі повідомлення між користувачем і сервером є зашифровані і інкрментуючий нонс використовується для запобігання традиційних атак, такі як Man-In-The Middle (MIM). Як було згадано вище, для конфеденційного доступа, особистість авторизованого користувача приховано від сервера, отже, процедури уникає використання публічного ключа користувача, щоб зберегти його особистість в таємниці. На противагу цьому, користувач починає сессію з сервером з повідомленням зашифрованим за допомогою публічного ключа сервера для перевірки аутентичності сервера. Таким чином, якщо користувач здійснює зв’язок з

певним конкретним сервером, тільки цей сервер може розшифрувати повідомлення для того щоб розкрити сесійний ключ K_t і нонс, а потім реагувати зашифрованим повідомленням яке містить нонс + 1.

Процедура доступу показує, що виконання політики контролю доступу та забезпечення авторизованого користувача секретний ключ для дешифрування даних здійснюється в одному механізмі. Цей метод зменшує додаткове навантаження як на хмарному сервері, так і користувача, з точки зору обчислення та управління ключами. До того ж, використання КТЗ в цьому механізмі, є ще одним фактором для зменшення додаткових витрат, оскільки операції КТЗ є простими модульними арифметичними операціями; зокрема, наприклад, не має модульних експонентних операцій. Ще одна важлива особливість що досягається за допомогою використання КТЗ, є те, що КТЗ дозволяє власнику даних приховувати число авторизованих користувачів. Через природу, яку було описано в Пункті 3.1, шифротекст $\left(E_{K_{pub\,i}}(C_r||K_s)\right)$ кожного авторизованого користувача u_i для $i = 1, 2, \ldots k$ може бути представлений з використанням КТЗ з одним значенням X_r. Сервер не може дізнатися скільки користувачів представлені в цьому X_r, але без використання КТЗ власник даних повинен прикріпити всі шифротекст, щоб розкрити серверу кількість користувачів.

3.3.3 Процедура надання та скасування доступу

Щоб надати доступ до файлу r новому користувачу u_{k+1}, до КТЗ додається нова конгруентність, як показано в Рівнянні (3.5). У такому випадку, власнику необхідно перерахувати загальне значення X_r'.

$$\begin{aligned} X_r' &\equiv \left(E_{K_{pub\,1}}(C_r||K_s)\right) mod\ n_1 \\ X_r' &\equiv \left(E_{K_{pub\,2}}(C_r||K_s)\right) mod\ n_2 \\ X_r' &\equiv \left(E_{K_{pub\,k}}(C_r||K_s)\right) mod\ n_k \end{aligned} \tag{3.5}$$

$$X'_r \equiv \left(E_{K_{pub\,k+1}}(C_r||K_s)\right) mod\ n_{k+1}$$

$K_{pub\,k+1}$ є відкритим ключем нового користувача. Як було описано в Пункті 3.2, рішення X'_r вищевказаної одночасної конгруентності можна розрахувати з X_r, який вже додано до файлу, з меншою кількістю модульних розрахунків, якщо нова система конгруентності для X'_r має один і той же модуль попереднього X_r плюс новий модуль; іншими словами, якщо власник даних вирішує додати нового користувача для доступу до існуючого файлу і не збирається змінювати попередні значення, включаючи C_r та K_s, які були використані для обчислити X_r. Отже, щоб знайти X'_r більш ефективно, можна знайти рішення використовуючи наступний шлях:

$$X'_r \equiv X_r\ mod\ n_1 n_2 \ldots n_k$$
$$X'_r \equiv \left(E_{K_{pub\,k+1}}(C_r||K_s)\right) mod\ n_{k+1} \quad (3.6)$$

Власник даних надсилає нове загальне значення X'_r з ID файлу та ID власника даних до серверу для заміни старого X_r. Механізм додавання нового користувача ефективний з двох причин; необхідна лише одна асиметричний операція шифрування для фіксованої довжини інформації, тобто, $C_r||K_s$, та він знаходить рішення лише для двох конгруенцій, див. Рівняння (3.6). Хоча загальні значення X'_r базуються на основі секретних значень, лише авторизовані користувачі мають доступ до них.

Щоб скасувати право доступу користувача u_k до файлу r, власник даних повинен змінити секретне значення C_r на C'_r та перерахувати X_r на X''_r для користувачів, що зберігають право доступу до файлу, від 1 до k-1 наступним чином:

$$X''_r \equiv \left(E_{K_{pub\,1}}(C'_r||K_s)\right) mod\ n_1$$
$$X''_r \equiv \left(E_{K_{pub\,2}}(C'_r||K_s)\right) mod\ n_2 \quad (3.7)$$

$$X_r^{"} \equiv \left(E_{K_{pub\,k-1}}(C_r' || K_s)\right) mod\ n_{k-1}$$

Власник даних надсилає на сервер нові значення C_r' та $X_r^{"}$ з ID файлу та ID власника файлу щоб замінити відповідні старі значення для файлу. Якщо користувач отримав доступ до даних перед скасуванням старих значень, то можливо, що користувач і сервер можуть змовитися для доступу до даних після скасування. Коли авторизований користувач отримав доступ до файлу, ключ цього файлу стає відомим для нього. Пізніше, якщо для цього користувача буде анульовано доступ до файлу і власник даних змінив лише C_r для цього файлу, користувач може домовлятися з сервером, так що сервер дозволяє йому отримати доступ до файлу, і користувач надає серверу старий ключ K_s, що досі розшифровує файл. Незважаючи на те, що описано в Пункті 3.3.2, схема не розкриває особистість користувачів для сервера і може зменшити імовірність такого роду collusion attack, рішуче рекомендується змінити ключ Ks і повторно зашифрувати файл, особливо якщо відбулася зміна змісту. Потім замінити старий файл на новий, який було зашифровано з новим ключем і який має нові C_r' та $X_r^{"}$ прикріплені до нього. Таким чином, сервер і користувач, позбавлений права доступу, не можуть вступити в змову з метою отримання доступу до нового вмісту файлу.

3.4 Можливість зашифрованого пошуку

У даній роботі ми використовували методи для пошуку зашифрованих даних, які сумісні з ОБІ підходом, який описано в Розділі 3. Власник даних повинен вказати один секретний ключ K_w, який використовується при шифруванні кожного ключового слова. Власник файлу використовує Keyed Pseudorandom Function (PRF) для шифрування ключових слів [14]. Нехай $H_K(m)$ буде Hash-based Message Authentication Code (HMAC), який може бути використаний будь-якою

криптографічною хеш-функцію, такою як MD5, SHA1, SHA256 або SHA512, з ключем K та вхідне повідомлення m [15]. Припустимо, w_i це і-ий ключ в списку ключових слів, R це випадкове число, флаг – це постійний бітовий патерн з фіксованою довжиною 1 біт і зміщенням випадковим патерном бітів.

$$a_i = H_{K_w}(w_i), \quad b_i = H_{a_i}(R), \qquad c_i = b_i \oplus (\text{флаг}||\text{зміщення}) \tag{3.8}$$

Від кожного ключового слова w_i, власник даних створює c_i та надсилає його з випадковим числом R до сервера як додаток до відповідного файлу. c_i є зашифрованою формою ключового слова w_i; таким чином, сервер не може відкрити w_i від c_i. Наступний пункт описує, як користувачі можуть безпечно здійснювати пошук їхніми зашифрованими файлами за допомогою використання зашифрованих ключових слів.

3.5 Процедура доступу з підтримкою можливості безпечного пошуку

У запропонованому нами рішенні для пошуку зашифрованих даних, що зберігаються в хмарному середовищі, коли користувач u_i здійснює пошук ключового слова w_i, користувач має надіслати запит власнику даних для отримання можливості пошуку, що містить ключове слово w_i зашифроване за допомогою публічного ключа власника ($K_{pub\ O}$) і підписаного приватного ключа користувача ($K_{prv\ i}$). Власник даних потім відповідає з $T_w = H_{K_w}(w_i)$, зашифрованому за допомогою публічного ключа користувача ($K_{pub\ i}$). Користувач розшифровує повідомлення за допомогою свого приватного ключа та потім зашифровує T_w з публічним ключом сервера ($K_{pub\ S}$), перед відправкою його на сервер в якості пошукового запиту з ID власника, нонс і сесійний ключ K_t, як показано в Рівнянні (3.9).

$$E_{K_{pub\ s}}(T_w, ID\ \text{власника}, K_t, \text{нонс}) \tag{3.9}$$

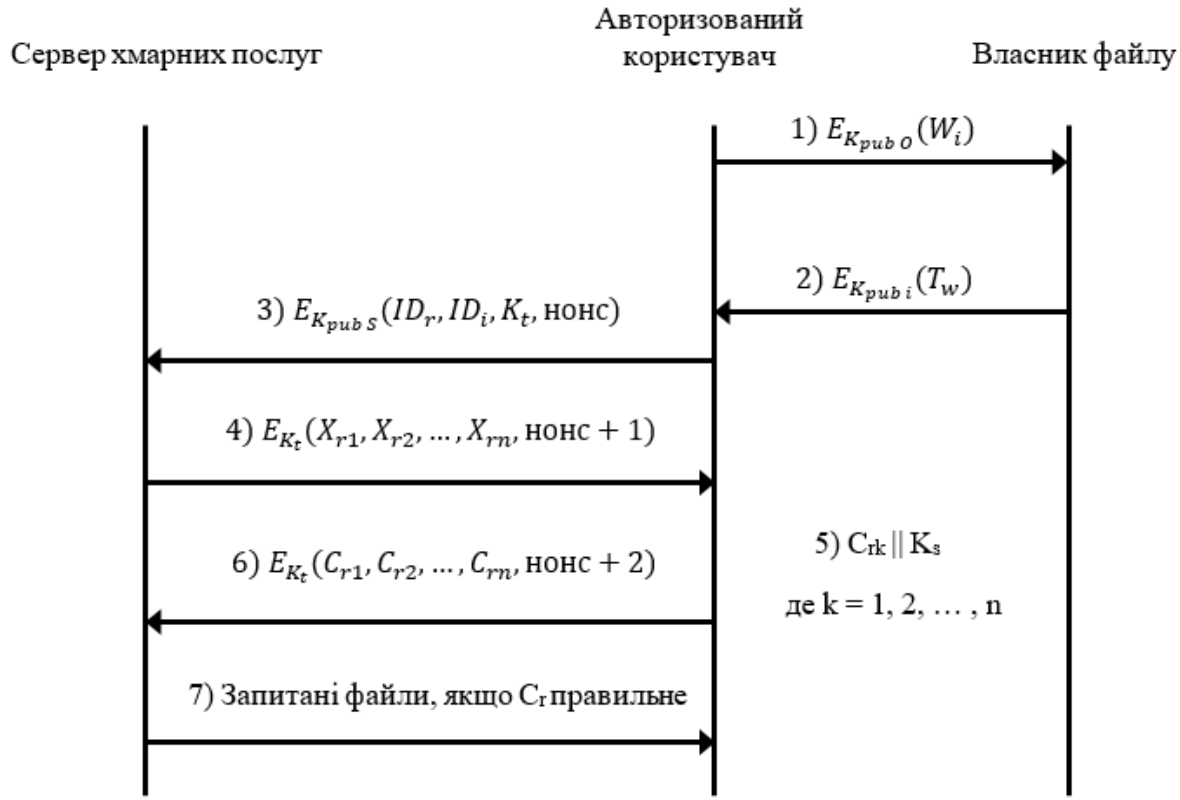

Рисунок 4.2 – Процедура безпечного пошуку

Сервер повинен шукати тільки дані, пов’язані з ID власника, використовуючи T_w для обчислення $P = H_{T_w}(R)$, а потім для кожного c_i сервер перевіряє результат $P \oplus c_i$, щоб побачити, якщо перший біт дорівнює флагу. Потім ID файлу, пов’язане з цим c_i, додається до списку результатів пошуку разом із власними двома параметрами X_r та C_r. Цілі та зміст елементів параметрів X_r та C_r описано в Пункті 3.3.

$$P = H_{T_w}(R), \qquad if\ P \oplus c_i = \text{флаг}||\text{зміщення} \therefore \ \in \text{збіг} \tag{3.10}$$

Після пошуку всіх файлів, сервер надсилає лише те значення X_r $(X_{r1}, X_{r2}, \dots X_{rn})$, де n - кількість файлів, які відповідають критеріям пошуку, та нонс + 1 і всі зашифровані за допомогою сесійний ключ, K_t, назад клієнту.

$$E_{K_t}(X_{r1}, X_{r2}, \dots X_{rn}, \text{нонс} + 1) \quad (3.11)$$

Користувач отримує секретне значення C_{ri} для кожного загального значення X_{ri}, де *i = 1, 2, ..., n*, за допомогою Рівняння (4.4), а потім повертає секретні значення і нонс + 2 на сервер зашифроване за допомогою сесійного ключа K_t.

$$E_{K_t}(C_{r1}, C_{r2}, \dots C_{rn}, \text{нонс} + 2) \quad (3.12)$$

Сервер перевіряє секретні значення $(C_{r1}, C_{r2}, \dots, C_{rn})$ та надсилає клієнту лише ті файли, секретні значення яких збігається з тими, які були надіслані користувачем. Тепер користувач може розшифрувати файли за допомогою секретного ключа K_s, витягнутого з X_{ri} для кожного файлу. Рисунок 3.2 ілюструє безпечну процедуру пошуку.

3.6 Побудова ОБІ-файлу з перевірками цілесності та аутентичності

Концепція ОБІ-файлу створює захищений контейнер, який включає в себе всі попередні модулі. ОБІ-файл інкапсулює вихідний файл даних перед його відправкою в хмарний сервер і таким чином зберігає файл даних захищеним від несанкціонованого доступу, в тому числі хмарного серверу. Лише авторизовані користувачі можуть здійснювати пошук і отримати доступ до файлового вмісту, відповідно до доданих параметрів політики безпеки, представлених в Cr та Xr, які встановлюються та управляються, головним чином, власником файлу. Процес створення ОБІ-файлу передбачено проводити в повністю довіренній машині, яка належить власнику даних. Перед створенням ОБІ-файлу є чотири основні операції:

1. шифрування вихідного вмісту файлу за допомогою алгоритму симетричного шифрування (див. Рисунок 3.3 блок А)
2. створити, за допомогою КТЗ, загальну значення X_r (див. Рисунок 3.3 блок С)

3. генерування секретних ключових слів для можливостей пошуку (див. Рисунок 3.3 блок D)

4. створення перевірок цілестності та аутентичності для зашифрованого вихідного файлу даних (див. Рисунок 3.3 блок B). Зашифрований файл захешований, а потім отриманий дайджест значення має цифровий підпис за допомогою його шифрування використовуючи приватний клбч $K_{prv\ O}$ власника файлу.

Потім ОБІ-файл створюється на основі цих чотирьох операцій. Рисунок 4.3 (блок F) показує вихідну структуру ОБІ-файлу. Лише авторизовані користувачі мають можливість отримати вихідний файл з ОБІ-файлу.

Всі зашифровані ключові слова, де c_i для $i=1,2, ..., z$ та z, як кількість ключових слів, прикріплюються до ОБІ-файлу разом з асоційованими рандомними числами R_i.

ОБІ-файл тепер може бути відправлений для зберігання в хмарне середовище. Приватність та цілістність вихідного файлу надійно зберігається і не залежать від хмарного серверу для забезпечення вимог безпеки, за винятком деяких операцій для їх виконання.

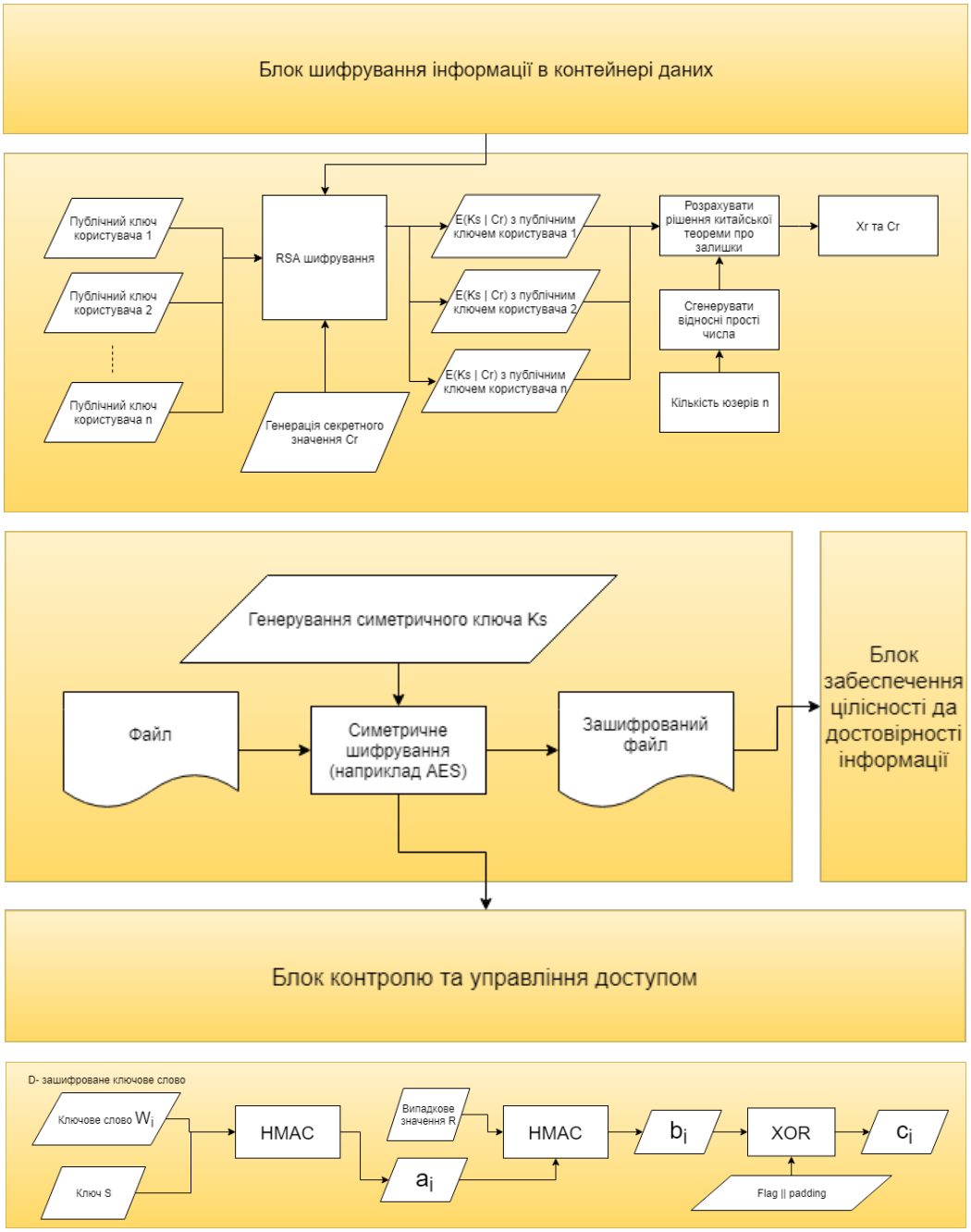

Рисунок 3.3 – Результати моделювання структури ОБІ підходу

3.7 Збереження конфіденційності та цілісність запропонованого рішення

У цьому розділі наведено короткий виклад особливостей запропонованого рішення для забезпечення конфіденційності і безпеки даних користувача даних, що

зберігаються в хмарі. У порівнянні з попередніми роботами, запропоноване рішення є більш безпечним, так як ми використовуємо відмінний симетричний ключ для шифрування кожного файлу та інше секретне значення для забезпечення дотримання параметрів контролю доступу для кожного файлу. Це поліпшення знаходить своє відображення в кількох аспектах безпеки та приватності, які наведено нижче:

– Користувач не може отримати доступ до зашифрованих даних, перш ніж він/вона забезпечує надійність секретного значення для серверу.

– Особистість користувача завжди прихована.

– Якщо симетричний ключ K_s і/або секретне значення C_r одного файлу скомпрометовано, інші файли залишаються в безпеці.

– Запропоноване рішення поєднує в собі контроль доступу та спільного використання секретного ключа в одному механізмі, з використанням КТЗ, що призводить до мінімізації обчислювальних та управлінських накладних витрат.

– Запропоноване рішення є більш стійким до можливих атак користувачів, позбавлених права доступу, і хмарного серверу. Оскільки кожен файл шифрується з іншим симетричним ключем, сервер не може вступити в змову з користувачем, позбавленим права доступу, для того щоб розкрити вміст даних, якщо такий користувач не мав доступу перед позбавленням його таких прав. Особливо, якщо такий користувач отримав доступ до ОБІ-файлу і показав симетричний ключ K_s, то власник даних може видалити файл і використовувати новий ключ K_s для створення нового ОБІ-файлу. Таким чином, якщо сервер запобігає збереженню копії цього ОБІ-файлу, то він не може вступити в змову з користувачами, які були позбавлені права доступу.

– При додаванні нового користувача до списку авторизованих користувачів, які можуть отримати доступ до файлу, власнику даних не потрібно змінювати секретне значення C_r для серверу. Має бути змінене лише загальне значення X_r. Отже, знижується можливість компрометації значення C_r. Крім того, для позбавлення користувача прав доступу, власнику даних лише необхідно змінити C_r цього файлу, тому що ніякі інші файли не використовують одне і те саме старе значення.

– Коли власник даних додає себе в якості авторизованого користувача для всіх ОБІ-файлів, власник даних і авторизовані користувачі мають залишати у безпеці лише свої пари публічних ключів.

– Запропоноване технічне рішення забезпечує цілісність та перевірку аутентичності для кожного файлу і не вимагає зміни параметрів при позбавленні існуючого користувача прав доступу та надання доступу для нового користувача.

– Використовуючи запропоноване рішення, всі параметри безпеки незалежно один від одного додаються до кожного ОБІ-файлу. Це дозволяє як хмарному провайдеру, так і користувачеві працювати з кожним ОБІ-файлом більш гнучко та ефективно. Наприклад, користувач або сервер може перемістити ОБІ-файл, як правило, не піклуючись про свою безпеку або політику контролю доступу, так як ці функції вже є частиною цього процесу і не залежать від зовнішніх об’єктів. Наприклад, безпека ОБІ-файлу не залежить від управління базами даних або доступом до системи, таким чином, не потрібно, звертати увагу на ці параметри при переміщенні даних в хмарі або між хмарами.

Узагальнюючи, варто зазначити, що обчислення КТЗ рішення більш ефективне у порівнянні з іншими методами криптографії, що використовують в криптографічному методі контролю доступу так як він використовує прості модульні операції без необхідності експоненціальних операцій.

3.8 Висновки

У цьому розділі представлено рішення, яке використовує ОБІ підхід для хмарних обчислень. Рішення розроблене на основі модулів, кожен з яких забезпечує набір сервісів безпеки, які в основному знаходяться в управлінні власника даних. Перший модуль призначений для шифрування даних перед їх аутсорсингом. Другий модуль створює можливості для більш ефективного та дієвого способу управління політикою контролю доступу, які виражені секретним та спільним значенням, які

призначені для спільного використання і доступу до контролю даних. Третій модуль використовується для забезпечення можливості безпечного пошуку для зашифрованих даних через генерацію зашифрованих ключових слів. Четвертий модуль виділяє перевіки цілісності та аутентичності. Результати всіх попередніх модулів використовуються для створення ОБІ-файлу. Використання ОБІ-файлу є результатом пошуку рішення що підвищує безпеку і конфіденційність та відповідає середовищу хмарних обчислень. Використання ОБІ-файлу, як частини запропонованого рішення, здатне зберегти приватність, цілісність, аутентичність даних які було розміщено в хмарі, навіть від самого хмарного провайдеру з мінімальним впливом на функціональність зашифрованих даних, які відповідають можливостям пошуку та поширення для авторизованих користувачів. На останок, важливо згадати, що функції безпеки, запропоновані в ОБІ-файлі, залежать від криптографічних алгоритмів, які використовуються в даному рішенні і знаходяться під управлінням власника даних.

4 РЕАЛІЗАЦІЯ МОДЕЛІ ТА ЇЇ ДОСЛІДЖЕННЯ

В цьому розділі розглядаються питання реалізації запропонованого в 4 розділі рішення. Мета цієї розділу – оцінити вплив використання цього рішення на стороні клієнта і сервера в основному з точки зору ресурсів обчислень і зберігання. Засоби реалізації і експериментальне середовище описані в пункті 4.1. Реалізація операцій, необхідних для створення файлу ОБІ на стороні власника даних і операцій, необхідних для авторизованої користувальницької сторони, описані в пункті «Забезпечення сторонніх клієнтів». У пункті 4.3 обговорюється реалізація і експерименти з пошуку ключових слів у файлах ОБІ, що зберігаються на стороні сервера. Питання реалізації на стороні клієнта і результати експериментів обговорюються далі у пункті 4.4. Резюме і висновки цього розділу наведені в останньому пункті.

4.1 Інструменти реалізації і середовище експерименту

Засоби реалізації і середовище експерименту були підготовлені для сервера і клієнтської сторони. Для реалізації було викритано мову C#, що базувалася на Eclipse IDE. Віртуальна серверна платформа, що представляє хмарний сервер, була встановлена на персональному комп'ютері з наступними характеристиками::

- Процесор: i5-7400 3.0GHz/8GT/s/6MB
- Оперативна пам’ять: 16 ГБ
- Загальна сховище на жорсткому диску: 256 ГБ (SSD).

VMware VSphere 6 була платформою віртуалізації для створення віртуального сервера. VMware є одним зі світових лідерів в області віртуалізації і хмарної інфраструктури. Віртуальний сервер був налаштований з такими специфікаціями:

- ОС: Windows Server 2012 R2 Enterprise 64-розрядної версії

- Віртуальна пам’ять: 8 ГБ
- Віртуальний простір для зберігання: 100 ГБ.
- Процесор: 2 ядра Intel i5-7400 3.0 ГГц

Інші віртуальні машини були встановлені на тому ж виділеному апаратному сервері для створення віртуального середовища, що імітує середу віртуалізації, яка використовується в публічній хмарі. Як обговорювалося в Розділі 2, в публічній хмарі кілька клієнтів використовують один і той же апаратний сервер, використовуючи свої віртуальні машини, що працюють на одному і тому ж апаратному сервері. Ці віртуальні машини ізольовані віртуально з використанням платформи віртуалізації.

В експерименті використовувався інший фізичний комп’ютер. Комп’ютер використовувався для моделювання як сторони власника даних, так і авторизованої користувальницького боку. Це був ноутбук з наступними характеристиками:

• Процесор: Intel CORE i3 CPU 2.2 ГГц

• Пам’ять: 16 ГБ

• ОС: Windows 10 64 біт

З боку власника даних було реалізовано і протестовано операції зі створення ОБІ-файлу перед його аутсорсингом на хмарний сервер. Для авторизованого користувача реалізація та тестування проводилися для операцій шляхом обчислення значення $C_r||K_s$ із загального значення X_r, використовуючи Рівняння (3.4), і шляхом дешифрування даних з файлу ОБІ, що містить зашифровані дані. На стороні сервера було виконано пошук ключових слів у ОБІ-файлі. На Рисунку 4.1 показано налаштування середовища експерименту, повідомлень, переданих між об’єктами, і основних операцій з кожного боку. На етапі реалізації одним із завдань був вимір службових даних сховища і накладних витрат на обчислення для виконання операцій ОБІ.

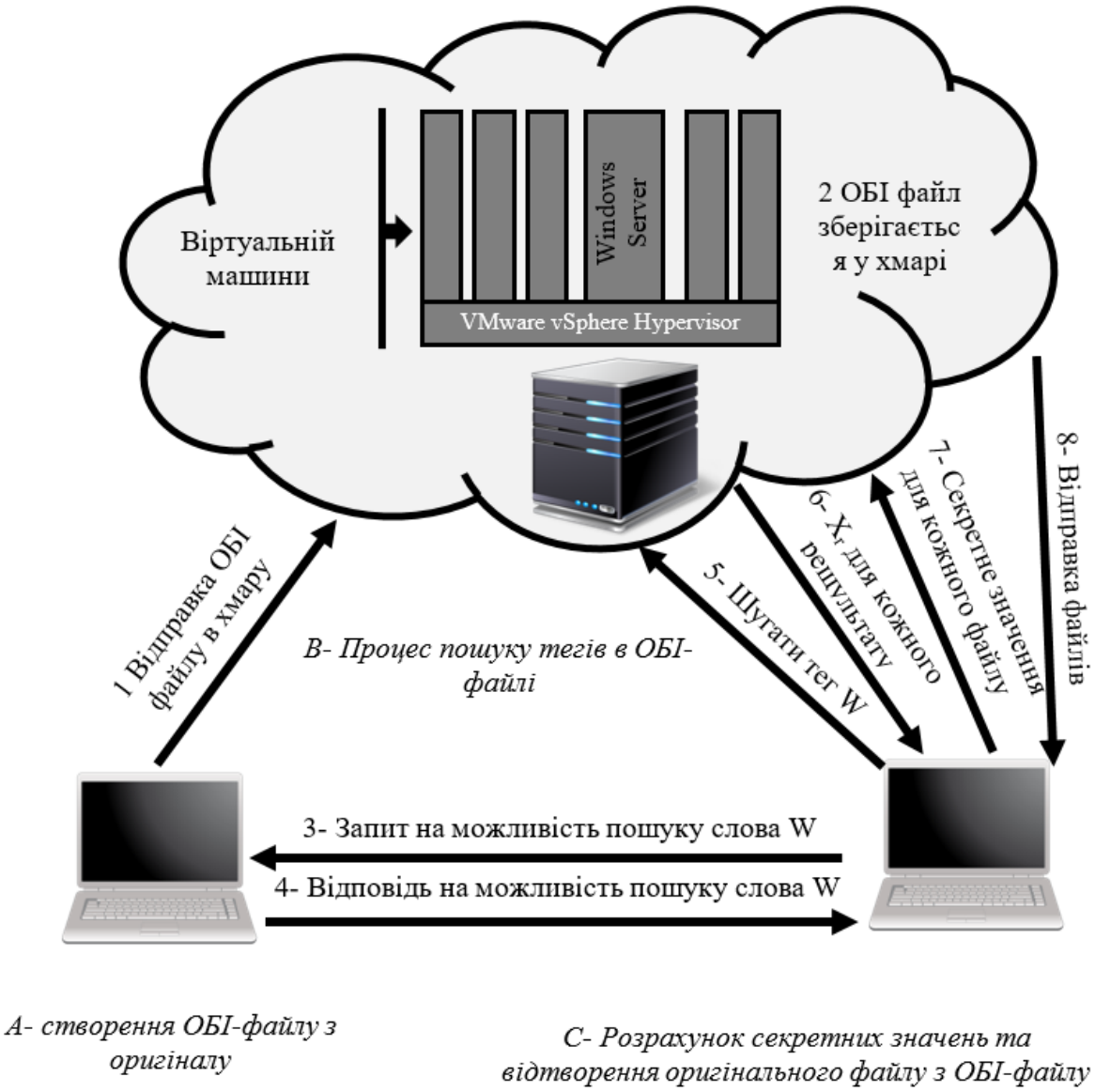

Рисунок 4.1 – Експериментальне середовище для запропонованого рішення

4.2 Реалізація на стороні клієнта

Клієнтська сторона може бути власником даних або авторизованим користувачем. Багато функцій, такі як асиметричний алгоритм шифрування, алгоритм симетричного шифрування та обчислення хеша файлу, використовуються як власником даних, так і авторизованим користувачем. Пошук рішення КТЗ і

шифрування ключових слів виконується тільки власником даних. У наступних пунктах описується реалізація необхідних функцій для клієнтської сторони:

4.2.1 Симетричне шифрування вихідного файлу

Шифрування даних за допомогою симетричного алгоритму шифрування є найбільш важливою операцією. Для надійного захисту дані повинні бути зашифровані, навіть якщо вони залишаться у власності власника даних. Тому перша дія після класифікації даних як конфіденційних даних – це шифрування даних з використанням адекватного алгоритму шифрування і сили ключа. У загальному випадку алгоритм симетричного шифрування використовується замість асиметричного шифрування для шифрування великої кількості даних, оскільки він більш ефективний. У цій реалізації файли зашифровуються за допомогою Advanced Encryption Standard (AES), що забезпечується програмою AES Crypt з відкритим вихідним кодом для платформ Windows з використанням 256-бітної довжини ключа, сильнішої довжини ключа для цього алгоритму. У наступному пункті описується реалізація алгоритму КТЗ, використовуваного для спільного використання K_s з секретним значенням C_r.

4.2.2 Реалізація розрахункового рішення КТЗ

КТЗ використовується для розподілу двох спільних секретних значень для кожного файлу серед авторизованих користувачів; симетричний ключ K_s, який використовується для шифрування певного файлу і секретного значення C_r, використовуваного для забезпечення дотримання політик контролю доступу для цього певного файлу. У реалізації довжина значення C_r повинна бути зафіксована для всіх файлів, тому легко відрізнити її від симетричного ключа K_s, коли два значення об’єднані як одне значення $C_r||K_s$. Ці два значення зашифровуються одним з асиметричних алгоритмів шифрування $E_{K_{pub\,i}}$ з публічним ключем кожного авторизованого користувача u_i, в результаті шифротекст $a_i = \left(E_{K_{pub\,i}}(C_r||K_s)\right)$ для

i=1, 2, … , k, де k кількість авторизованих користувачів для певного файлу. Отриманий в результаті шифротекст для кожного користувача, якого було авторизовано для доступу до файлу, буде вихідним для операції КТЗ для обчислення загального значення X_r для цього файлу, використовуючи Рівняння (3.3) з Пункту 3.2. Потім для реалізації потрібно знайти рішення одночасної конгруентності КТЗ, представленої в Рівнянні (3.3). Отже, наприклад, якщо ми використовуємо алгоритм RSA для шифрування значення $C_r||K_s$, мінімальна довжина результату шифротексту буде становити не менше 1024 біт. Для реалізації платформа .NET надає бібліотеку класів System.Numerics.BigInteger для обробки довгих цілих значень довжиною понад 64 біти. Крім того, для КТЗ потрібна функція для генерації відносних простих чисел $n_i = n_1, n_2, ... n_k$, де k - кількість авторизованих користувачів. Генерація відносних простих чисел може бути виконана один раз для всіх користувачів, а потім визначена унікально для кожного користувача. Програма, яка використовується для вирішення одночасної конгруентності КТЗ, базується на алгоритмі Гарнера та Розширеному алгоритмі Евкліда (РАЕ) і використовується для обчислення модульного мультиплікативного звороту. Вихідні коди для цих двох алгоритмів доступні у вигляді програми з відкритим вихідним кодом в . Однак оригінальні вихідні коди реалізовані для підтримки цілих чисел з довжиною в 64 біти. Для реалізації запропонованого рішення вихідні коди модифікуються на основі бібліотеки класів System.Numerics.BigInteger для підтримки цілих чисел більше 64 бітів, класифікованих на C# як Biglntegers. Отже, результуюча програма для вирішення КТЗ може обробляти цілі числа довжиною понад 64 біти. CalculateKTZ.cs і EuclidHelper.cs – це два файли класів C#, які містять оновлені програмні коди для пошуку рішення КТЗ і для обчислення модульного мультиплікативного звороту. Вихідні коди цих двох файлів класів C# описані в Додатку Б.

Функція CalculateKTZ.Execute (Biglnteger [] input_a, Biglnteger [] input_n) використовується для знаходження рішення КТЗ, де a і n – вхідні масиви Biglnteger значень a_i і n_i. На рисунку 4.2 показано вихідний код для вирішення КТЗ, що підтримує Biglntegers. У функції Execute, модульний мультиплікативний зворот обчислюється з використанням функції EuclidHelper.ExecuteInverse (Biglnteger

input_p, Biglnteger input_g), як показано на рисунку 4.3. Поверненим значенням зворотної функції є g^{-1} (mod p). Найбільший спільний дільник (GCD) двох цілих чисел можна обчислити, використовуючи функцію Biglnteger p.gcd (Biglnteger g), включену в бібліотеку класу Biglnteger.

```
using System;

public class CalculateKTZ
{
    // Розрахунок рішення для китайської теореми про залишки
    public static System.Numerics.BigInteger Execute(System.Numerics.BigInteger[] input_a, System.Numerics.BigInteger[]
    {
        if (input_a == null || input_n == null) {
            Console.WriteLine("Жоден з аргументів не може бути пустим");
            return null;
        }
        if (input_a.Length < 2 || input_n.Length < 2) {
            Console.WriteLine("Довжина кожного елемента повинна бути більше двох");
            return null;
        }
        if (input_a.Length != input_n.Length) {
            Console.WriteLine("Довжина обох аргументів повинна дорівнювати");
            return null;
        }
    System.Numerics.BigInteger x = input_a[0];
    System.Numerics.BigInteger nInverse, y, z;
    System.Numerics.BigInteger ni = System.Numerics.BigInteger.One;

    int i = 0;
    while (i < input_a.Length - 1) {
        ni = ni * input_n[i];
        //перевіримо чи n відносно просте
        if (!System.Numerics.BigInteger.One.Equals(input_n[i + 1].gcd(ni))) {
            Console.WriteLine("The n's are not relatively prime->" + input_n[i + 1] + "," + ni);
            return null;
        }
        nInverse = cryptoBig.inverse(n[i + 1], ni);
        if (nInverse.compareTo(0) == -1) {
            nInverse = nInverse + input_n[i + 1];
        }
        z = input_a[i + 1] - x;
        z = z * nInverse;
        y = z.remainder(input_n[i + 1]);
        if (y.compareTo(0) == -1) {
            y = y + n[i + 1];
        }
        x = x + ni * y;
        i++;
    }
    return x; // рішення КТЗ
    }
}
```

Рисунок 4.1 – Код для віднаходження рішення КТЗ

```
public class EuclidHelper
{
    //повертає g в ступены (-1) (mod p)
    public static System.Numerics.BigInteger
        ExecuteInverse(System.Numerics.BigInteger input_p, System.Numerics.BigInteger input_g)
    {
        System.Numerics.BigInteger[] result = ExtendedEuclid(input_p, input_g);
        if (result[2].compareTo(0 as System.Numerics.BigInteger) != -1) {
            return input_p + result[2];
        }

        return result[2];
    }


    //Ця функція виконає розширений алгоритм Евкліда, щоб знайти GCD для значень input_a та
    public static System.Numerics.BigInteger[]
        ExecuteEEA(System.Numerics.BigInteger input_a, System.Numerics.BigInteger input_b)
    {
        System.Numerics.BigInteger[] result = new BigInteger[3];
        System.Numerics.BigInteger q;

        if (input_b.Equals(0 as System.Numerics.BigInteger)) {
            result[0] = input_a;
            result[1] = 1 as System.Numerics.BigInteger;
            result[2] = 0 as System.Numerics.BigInteger;
        }
        else
        {
            // В іншому випадку зробити рекурсивний виклик
            q = input_a / input_b;
            result = ExecuteEEA(input_b, a.remainder(input_a));

            System.Numerics.BigInteger s = result[2] * q;
            System.Numerics.BigInteger temp = result[1] - s; // - ans[2]*q;

            result[1] = result[2];
            result[2] = temp;
        }

        return result;
    }
}
```

Рисунок 4.3 – Код для обчислення модульного мультиплікативного звороту

4.2.3 Шифрування асиметричного ключа

В реалізації було використано RSA в якості асиметричного алгоритму шифрування ключа для шифрування значення $C_r||K_s$ для кожного авторизованого

користувача. Довжина K_s може бути 128, 192 або 256 біти і це для секретного значення C_r має бути обрано так, що загальна довжина $C_r||K_s$ буде менше 1024 бітів. Це гарантує, що зашифроване значення $C_r||K_s$ обмежене 1024 бітами. Більш того, обмеження довжини значення $C_r||K_s$ призведе до зменшення обчислювальних витрат операцій шифрування і дешифрування RSA. Платформа .NET надає пакет який містить кілька бібліотек класів, які включають в себе функції, які використовуються для генерації пар ключів RSA і RSA-операцій шифрування/дешифрування. З цих функцій було запрограмовано два файли C#-класів: клас GenerateRSAkey.cs і клас RSAfurBytes.cs. Клас GenerateRSAkey.cs призначений для створення пари ключів RSA і зберігання їх у файлі. Клас RSAforBytes.cs містить дві функції: rsaEncrypt (byte [] data, String PubKeyFile) для шифрування масиву байтів і rsaDecrypt (byte [] data, String PrivKeyFile) для дешифрування цього масиву байтів. Коди двох класів перераховані в Додатку Б.

На Рисунку 4.4 показано деякі результати генерації пар ключів RSA і шифрування значення $C_r||K_s$ з використанням згенерованих ключів. Значення $C_r||K_s$ було вибране рівне «104443322525597233998882325374 79». Це ціле значення було перетворене з Biglnteger в масив байтів, тому що функція rsaEncrypt запрограмована для шифрування даних в форматі масиву байтів. Перетворення результувало у 14 байтах, а шифрування цього значення для десяти користувачів зайняло 15 мілісекунд, як показано в знімку екрана на Рисунку 4.4. Отримане зашифроване значення для кожного користувача має довжину 128 байтів (1024 біти), оскільки використовуваний відкритий ключ генерується з довжиною 1024 біти. Кожен зашифрований вихід повинен бути перетворений назад в Biglnteger. Потім отримані зашифровані значення Biglnteger будуть використовуватися в якості вхідних даних для функції CRTforBiglnteger.findSolution (Biglnteger [] a, Biglnteger [] n), де масив a [] містить значення a_i і n [], містить значення n_i, як представлено в Рівнянні (3.1) де $a_i = \left(E_{K_{pub\,i}}(C_r||K_s)\right)$ та n_i – відносне просте число для користувача u_i для $i = 1, 2, \ldots k$, де k – кількість авторизованих користувачів для цього певного файлу. Потім отримане

рішення КТЗ для X_r з findSolution буде прикріплене до захищених даних як частина ОБІ файлу.

CreateSecureFile.cs | CalculateKTZ.cs | cryptobig.cs | generateRSAKey.cs | StringHelper.cs | DateTimeHelper.cs

```
26
27          //Кінець блоку шифрування тегів
28
29          // Блок шифрування RSA
30          Stopwatch swRSA = new Stopwatch();
31          swRSA.Start();
32          BigInteger C_K = System.Numerics.BigInteger.Parse("1044433225255972339988823253747 9");
33          long c = 1044433225; //
34          Console.WriteLine("Значення Cr " + c);
35
36          sbyte[] data = C_K.toByteArray();
37          /* RSA шифрування масиву байтів Cr||Ks з кожним публічнім ключем авторизованого користувача */
38          sbyte[] rsaEncoded1 = RSAforBytes.rsaEncrypt(data, "public.key");
39          sbyte[] rsaEncoded2 = RSAforBytes.rsaEncrypt(data, "public2.key");
40          sbyte[] rsaEncoded3 = RSAforBytes.rsaEncrypt(data, "public3.key");
41          sbyte[] rsaEncoded4 = RSAforBytes.rsaEncrypt(data, "public4.key");
42          sbyte[] rsaEncoded5 = RSAforBytes.rsaEncrypt(data, "public5.key");
43          sbyte[] rsaEncoded6 = RSAforBytes.rsaEncrypt(data, "public6.key");
44          sbyte[] rsaEncoded7 = RSAforBytes.rsaEncrypt(data, "public7.key");
45          sbyte[] rsaEncoded8 = RSAforBytes.rsaEncrypt(data, "public8.key");
46          sbyte[] rsaEncoded9 = RSAforBytes.rsaEncrypt(data, "public9.key");
47          sbyte[] rsaEncoded10 = RSAforBytes.rsaEncrypt(data, "public10.key");
48
49          swRSA.Stop();
50           //Кінець RSA шифрування
51          Console.WriteLine("RSA шифрування зайняло:" + swRSA.elapsed() + " мс");
52          Console.WriteLine("Довжина Cr||Ks:" + data.Length + " байт");
53          Console.WriteLine("Довжина зашифрованого Cr||Ks:" + rsaEncoded1.Length + " байт");
54          // Кінець шифрування
55          // Транформування зашифрованого масиву даних в BigInteger
56          BigInteger transformed1 = new BigInteger(rsaEncoded1);
57          BigInteger transformed2 = new BigInteger(rsaEncoded2);
58          BigInteger transformed3 = new BigInteger(rsaEncoded3);
59          BigInteger transformed4 = new BigInteger(rsaEncoded4);
60          BigInteger transformed5 = new BigInteger(rsaEncoded5);
```

Рисунок 4.4 – Знімок екрана з шифруванням RSA для десяти користувачів

Результати роботи RSA для 10 користувачів:

– Час виконання: 15 мс

– Довжина Cr||Ks: 14 байт

– Довжина зашифрованого: 128 байт

4.2.4 Шифрування ключових слів

В реалізації ключові слова для пошуку зашифровуються з використанням коду повідомлень на основі Hash (HMAC) з алгоритмом хешування SHA256. Клас C# HMACSHA256.C# містить функцію encode (String key, String data), яка обчислює

значення HMAC-SHA256 для строкових вхідних даних з ключем, який вводиться у вигляді рядка. Функція encode повертає зашифрований рядок у форматі байтового масиву довжиною 256 біти. На рисунку 4.5 показано код функції кодування. Ключ, використовуваний для цієї функції, однаковий для всіх ключових слів.

```
public class HMACSHA256
{
    public static sbyte[] ExecuteEncode(string key_input, string data_input)
    {
        Mac hash_algo = Mac.getInstance("HmacSHA256"); // Вибір хеш алгоритму
        SecretKeySpec secret_key = new SecretKeySpec(key_input.GetBytes(), "HmacSHA256");
        hash_algo.init(secret_key); //Генерація на основі переданого ключа

        return hash_algo.doFinal(data_input.GetBytes()); //Складання хеша
    }
}
```

Рисунок 4.5 – Код для вирахування HMAC для ключового слова

Рисунок 4.6 показує знімок екрану з запуском програмного коду для шифрування п’яти ключових слів з використанням функції кодування. Шифрування зайняло 187 мілісекунд.

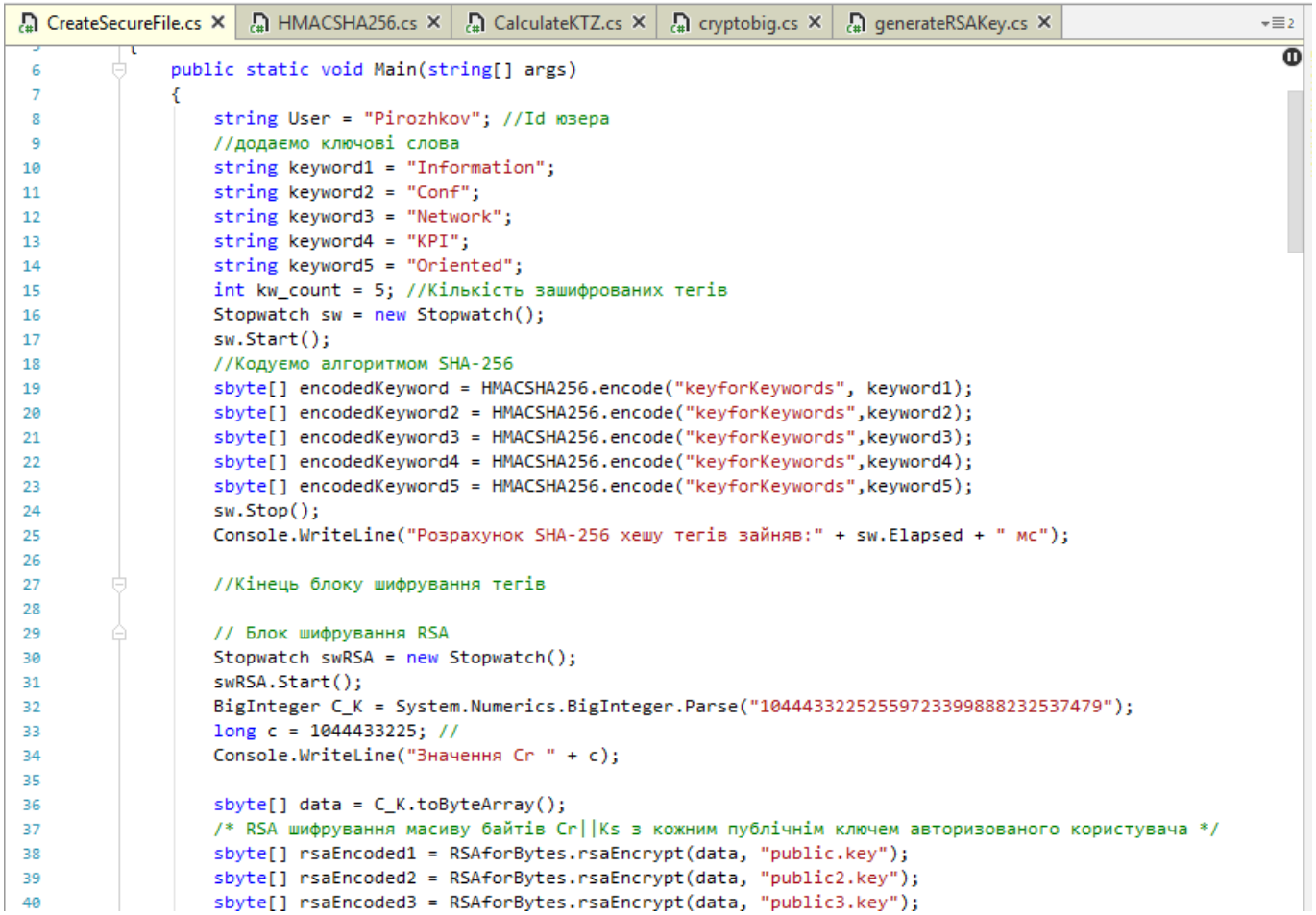

Рисунок 4.6 – Знімок екрану з вирахуванням HMAC-SHA256 для п’яти ключових слів

4.2.5 Обчислення доказів цілісності та достовірності

Перевірка цілісності та достовірності обчислюється шляхом хешування зашифрованого файлу і подальшого шифрування отриманого дайджесту особистим ключем власника даних. Функція createFileHash (String filename, String method) в класі Hash.cs використовується для обчислення дайджесту файлу з використанням одного з хеш-алгоритмів, таких як SHA-I, SHA-256 і MD5 (див. Код Hash.cs в Додатку Б). CreateFileHash приймає в якості вхідних даних ім’я файлу і ім’я хеш-алгоритму (наприклад, SHA-1, SHA-2 або MD5).

Після обчислення дайджесту даного файлу відповідно до зазначеного алгоритмом хеша функція повертає значення дайджесту в форматі масиву байтів. Довжина отриманого дайджесту залежить від використовуваного алгоритму.

Наприклад, при використанні SHA-256 підсумкова довжина дайджесту становить 256 біти. Потім отриманий дайджест зашифровано з використанням функції шифрування RSA rsaEncrypt (байт [] data, String PrivKeyFile), де ключ є закритим ключем власника даних (тобто цифровим підписом власника даних). Значення цифрового підпису буде 1024 біти при використанні 1024-бітного закритого ключа і 2048 біт при використанні 2048-бітного ключа. Рисунок 4.7 містить програмний код для обчислення значення, FSignature в програмному коді, який використовується для перевірки цілісності та достовірності.

```
// Розрахунок вхідного дайджесту файла і сигнатури власника даних
string private_key_file = "prkey_dataowner.key"; //приватний ключ власника даних
Stopwatch swDigest = new Stopwatch();
swDigest.Start();
//Підрахунок дайджесту файла
sbyte[] fileDigest = Hash.createFileHash(InputFile, "SHA-256");
// Підпис файлу
sbyte[] fileSignature = RSAforBytes.rsaEncrypt(fileDigest, private_key_file);
```

Рисунок 4.7 – Код для обчислення доказів цілісності та достовірності

4.2.6 Створення захищеного файлу (ОБІ файл) власником даних

Створення ОБІ-файлу в основному пов’язане з раніше описаними операціями, а саме: симетричним шифруванням, пошуком рішення КТЗ, HMAC-SHA256, шифруванням RSA і підписом зашифрованого дайджесту повідомлень вхідного файлу. Всі виходи цих операцій і пов’язаної з ними інформацією включаються до складу вмісту файлу ОБІ. Клас CreateSecureFile.cs створено і розроблено для використання у створенні файлу ОБІ з зашифрованого файлу. Перед запуском програми CreateSecureFile вихідний файл повинен вже бути зашифрований програмою AES Crypt і визначений наступними параметрами, з прикладами присвоєних значеннь параметру, вказаного для кожного з параметрів:

1. Визначене наперед ім’я власника або ID.

string User = "Pirozhkov"; //Id юзера

2. Визначені наперед рядки, число і ключ шифрування ключових слів пошуку.

//додаємо ключові слова

```
string keyword1 = "Information";
string keyword2 = "Conf";
int kw_count = 2; //Кількість зашифрованих тегів
//Кодуємо алгоритмом SHA-256
sbyte[] encodedKeyword = HMACSHA256.encode("keyforKeywords",
keyword1);
sbyte[] encodedKeyword2 =
HMACSHA256.encode("keyforKeywords",keyword2);
```

3. Визначені наперед окремо значення $C_r||K_s$ і C_r.

```
BigInteger C_K =
System.Numerics.BigInteger.Parse("10444332252559723399888232537479");
long c = 1044433225;
```

4. Визначені наперед всі відносні прості значення (тобто елементи масиву Biglnteger n []) авторизованими користувачами цього файлу.

```
BigInteger prime1 = System.Numerics.BigInteger.Parse("215143111331331”); // Користувач 1
BigInteger prime2 = System.Numerics.BigInteger.Parse("103143111331”); // Користувач 2
```

5. Визначені наперед файли, що містять публічний ключ для кожного з авторизованих користувачів.

```
sbyte[] rsaEncoded1 = RSAforBytes.rsaEncrypt(data, "public.key"); // Користувач 1
sbyte[] rsaEncoded2 = RSAforBytes.rsaEncrypt(data, "public2.key"); // Користувач 2
```

6. Визначений наперед файл, що містить приватний ключ власника даних.

```
string private_key_file = "prkey_dataowner.key"; //приватний ключ власника даних
```

7. Визначені наперед алгоритми хешування (наприклад, SHA-1 або SHA-256) для дайджесту зашифрованого файлу.

```
sbyte[] fileContainingDigest = Hash.createFileHash(InputFile, "SHA-256");
```

Після визначення вищевказаних параметрів програма готова до виконання. Виконання спочатку обчислює HMAC-SHA256 ключових слів, як показано на Рисунку 4.6 вище. Потім вона шифрує значення $C_r||K_s$ з відкритим ключем кожного користувача, приклад якого показано на Рисунку 3.7. Отримані зашифровані значення з відносними цілими числами користувачів використовуються для знаходження загального значення за допомогою функції CRTforBigInteger.findSolution. Наприклад, обчислення КТЗ-рішення для п’яти користувачів, де довжина їх відносних простих чисел становить 1025 біти, а значення $C_r||K_s$ становило 1024 біта, зайняло близько 5,1 мілісекунди, а отриманий X_r був 5122 біти, що отриманр кодом на Рисунку 4.8.

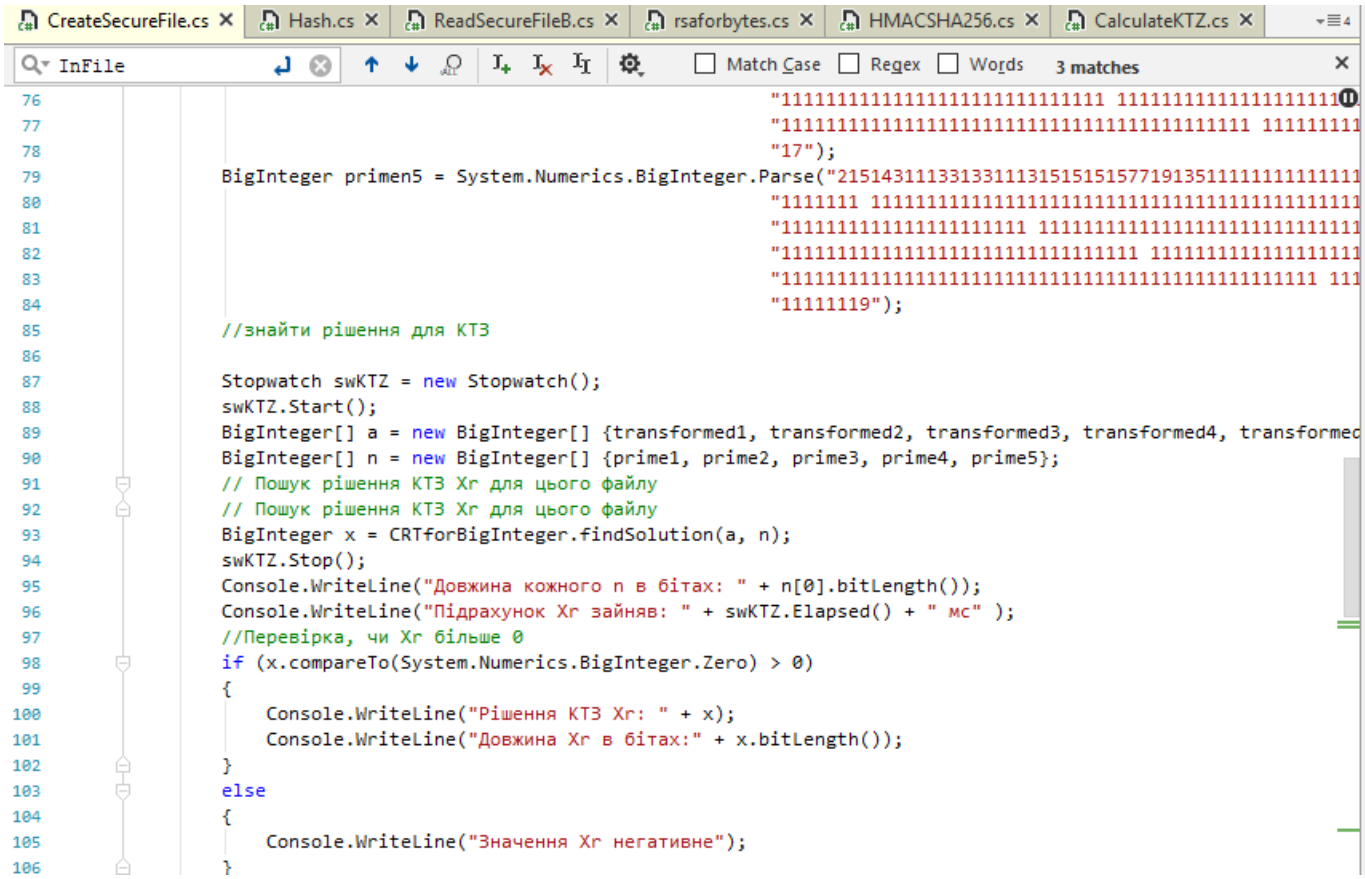

Рисунок 4.8 – Знімок екрану з вирахуванням КТЗ рішення для п’яти користувачів

Потім програма запропонує користувачеві ввести зашифрований файл, який буде перетворений в файл ОБІ з прикріпленими параметрами безпеки. Програма обчислює дайджест вхідного файлу і підписує дайджест особистим ключем власника даних. На Рисунку 4.9 показаний приклад використання розробленої програми для

обчислення дайджесту файлу розміром 40,7 МБ і створення цифрового підпису файлу. Це зайняло 780 мілісекунд, коли алгоритм хеша SHA-256 і 1024-бітний закритий ключ RSA були використані для цього конкретного файлу.

```
            Console.WriteLine("Довжина Xr в бітах:" + x.bitLength());
        }
        else
        {
            Console.WriteLine("Значення Xr негативне");
        }
//вкажіть зашифрований файл для створення ОБІ-файлу
    Console.Write("Введіть ім'я файла з розширенням: ");
    string InputFile;
    Console.ReadLine(InputFile);
    File InFile = new File(InputFile); //input file
    if (!InFile.exists())
    {
        Console.WriteLine("File does not exist.");
        Environment.Exit(0);
    }
    // Розрахунок вхідного дайджесту файла і сигнатури власника даних
    string private_key_file = "prkey_dataowner.key"; //приватний ключ власника даних
    Stopwatch swDigest = new Stopwatch();
    swDigest.Start();
    //Підрахунок дайджесту файла
    sbyte[] fileContainingDigest = Hash.createFileHash(InputFile, "SHA-256");
    // Підпис файлу
    sbyte[] fileSignature = RSAforBytes.rsaEncrypt(fileContainingDigest, private_key_file);
    Console.WriteLine("Довжіна підпису файла у байтах: " + FSignature.Length);
    //Зупинка таймеру обчислювання дайджеста та сігнатури
    swDigest.Stop();
    Console.WriteLine("Підрахунок дайджеста та сігнатури зайняв:" + swDigest.Elapsed() + " ms");
    // Кінець розрахунків дайджеста та сігнатури
    //геенрування ОБІ-файла
```

Рисунок 4.9 – Знімок екрану з виразуванням дайджесту файлу та підпису для файлу 40,7 МБ

Після обчислення всіх необхідних параметрів програма запропонує користувачеві ввести шлях, ім’я та розширення для вихідного файлу ОБІ з розширенням «dcs» за замовчуванням. Потім програма починає створювати ОБІ-файл в якості вихідного.. Повний програмний код CreateSecureFile.cs можна знайти в Додатку Б.

На мові програмування C# для вставки нового потоку біт на початку файлу потрібно створити новий файл з новою інформацією метаданих файлу і вихідним вмістом файла. Тому для створення ОБІ-файлу потрібно створити новий файл з параметрами, в якості метаданих і зашифрованого вмісту файлу. Це додасть

обчислювальних витрат, особливо для великих файлів, в копіюванні вмісту зашифрованого файлу в файл ОБІ.

4.2.7 Операції на стороні авторизованого користувача

Авторизований користувач вимагає, в основному, чотирьох операцій, тобто обчислення значення $C_r||K_s$ із загального значення X_r, відтворення вихідного зашифрованого файлу з файлу ОБІ, перевірки цілісності та перевірки аутентичності і, нарешті, дешифрування зашифрованого файлу для створення файлу у вигляді нешифрованого тексту. Всі ці операції реалізовані в одному файлі класу ReadSecureFileB.cs. Користувач повинен визначити наступні параметри. Приклади присвоєних значень параметру наведені для кожного з параметрів:

1. Унікальний відносно простий користувач
   BigInteger n = new BigInteger(“3812938192391”);
2. Файл, що містить закритий ключ користувача.
   string pr_key_file = “pr.key”;
3. Файл, що містить відкритий ключ власника даних.
   String publ_key_file = “publicUser.key”;

Прості і закриті ключі користувача використовуються для обчислення $C_r||K_s$ із значення X_r, а відкритий ключ власника даних використовується для перевірки цілісності та аутентичності файлу ОБІ.

Вирахування секретного значення $C_r||K_s$ із загального значення X_r

Після визначення значення $C_r||K_s$ авторизований користувач u_i тепер може використовувати C_r для завершення процедури контролю доступу та K_s для дешифрування вихідного зашифрованого файлу після його завантаження. Оскільки комунікаційна частина процедури контролю доступу не розглядається в цій реалізації, то код читає X_r безпосередньо з файлу ОБІ, поки сервер повинен це зробити і відправити його користувачеві під час процедури контролю доступу. Основна увага в цій частині реалізації була присвячена вимірюванню накладних витрат обчислень, викликаних обчисленням $C_r||K_s$ на стороні користувача. Як показано в Таблиці 4.2

вище програмний код виконує в основному дві функції. Перша функція пов’язана з алгоритмом КТЗ, який заснований на простій модульної функції (тобто X_r mod ni), і, отже, розрахунок може бути виконаний дуже швидко. Наприклад, використовуючи цю функцію, обчислюючи значення X_r з довжиною 5121 біти, для п’яти користувачів, та n з довжиною 1024 біти займає близько 96,565 мікросекунд. Друга функція - це операція розшифровки КТЗ для отриманого зашифрованого значення з першої функції. Ця операція дешифрування зайняла близько 516 мілісекунд.

Читання вбудованої інформації в ОБІ-файлі і повторне відтворення вихідного зашифрованого файлу

Файл ОБІ містить інформацію, таку як значення X_r, докази цілісності і аутентичності, на додаток до зашифрованих даних у файлі. Коли клас ReadSecureFileB.cs запущений, програма попросить користувача ввести вхідний файл ОБІ, а потім ім’я і розширення для вихідного файлу. Програмний код для зчитування інформації з вхідного файлу ОБІ і подальшого виведення зашифрованих даних у файлі.

Код, призначений для того, щоб дозволити авторизованому користувачу або провайдеру хмари читати з ОБІ-файлу, ім’я власника даних, X_r, C_r і підписаний дайджест файлу зашифрованих даних. Однак тільки авторизовані користувачі можуть отримати ключ K_s, який потім може використовуватися для дешифрування зашифрованого файлу. Крім того, авторизований користувач може також перевірити цілісність і достовірність файлу даних.

Перевірка цілісності та достовірності

Після отримання зашифрованого файлу даних з файлу ОБІ і читання прикріпленого підпису до файлу, авторизований користувач може перевірити цілісність і аутентичність файлу зашифрованих даних.

Процес перевірки призводить до обчислювальних накладних витрат, а кількість накладних витрат залежить від розміру файлу. Наприклад, перевірка цілісності та аутентичності файлу розміром 40,7 МБ займає близько 720 мілісекунд,

Розшифровка зашифрованого файлу

Після перевірки цілісності та аутентичності зашифрованого файлу даних зашифрований файл даних може бути дешифрований за допомогою програмного забезпечення AES Crypto з ключем K_s для цього файлу. Велика частина обчислювальних витрат пов’язана з розшифровкою зашифрованого файлу, особливо для великих файлів, як в результатах, показані в пункті 4.5.2.

4.3 Реалізація та експерименти на стороні сервера

На стороні сервера основна реалізована операція - це процес пошуку в файлі ОБІ прикріплених зашифрованих ключових слів. Клас search.cs включає код, необхідний для пошуку файлів ОБІ. Для прискорення процесу пошуку використовується алгоритм пошуку рядків Knuth-Morris-Pratt (KMP). Алгоритм KMP є одним з найбільш ефективних алгоритмів пошуку, і він може знайти точну відповідність шаблонів в ряді цифрових даних. Клас KMPMatch.cs містить реалізацію алгоритму KMP з відкритим вихідним кодом. KMPMatch.cs забезпечує функцію:

indexOf(bytes[] data_input, bytes[] pattern_input)

Функція використовує алгоритм KMP для пошуку першого входження певного образу масиву байтів (тобто byte[] pattern) в заданому діапазоні масиву байтів (тобто byte[] data). Ця функція використовується в класі search.cs для пошуку заданого зашифрованого ключового слова в файлах ОБІ на хмарному сервері. Реалізований алгоритм пошуку, заснований на алгоритмі KMP, призначений для пошуку тільки з боку файлу ОБІ, який містить масив байтів, що містять зашифровані ключові слова. Наприклад, для п’яти зашифрованих ключових слів діапазон буде 164 байта, оскільки перші 4 байта файлу ОБІ представляють кількість ключових слів і кожне зашифроване ключове слово зайняло 32 байти. Обмеження діапазону пошуку дуже важливе для скорочення часу, необхідного для пошуку, особливо для великих файлів. Якщо процес пошуку не обмежується діапазоном, що містить зашифровані ключові слова, пошук по всьому файлу ОБІ буде витрачати ресурси і час на обчислення,

оскільки в зашифрованому файлі даних немає зашифрованих ключових слів з можливістю пошуку.

Перед запуском класу search.cs необхідно вказати шлях до каталогу, який містить файли ОБІ для пошуку, як показано нижче:

String path = “C:\\some_fold”;

Код, запитує рядок ключового слова, а потім обчислює значення HMAC-SHA256 для ключового слова. Отримане значення поміщається в масив байтів. Цей код включений в клас search.cs для експериментів. У реальному житті цей код запускається на стороні власника даних. Хмарний сервер отримує тільки зашифровані ключові слова (тобто масив байтів), які використовуються для пошуку файлів ОБІ, що містять ключові слова.

Після введення ключового слова і обчислення його HMAC-SHA256, search.cs відкриває кожен з файлів ОБІ і зчитує кількість зашифрованих ключових слів, вбудованих в файл. З цілого числа ключових слів, прикріплених до ОБІ-файлу, код обчислює діапазон пошуку, а потім виконує пошук ключового слова з використанням алгоритму KMP. Якщо образ відповідає, код виводить на екран шлях і ім’я файлу ОБІ, що містить це ключове слово. Код повторює ту ж процедуру для всіх файлів ОБІ в зазначеній директорії. Для всіх файлів код показує час, необхідний для пошуку кожного з файлів. Повний код класу search.cs наведено в Додатку Б.

4.4 Результати та обговорення

У цьому пункті обговорюються результати і проблеми впровадження в відношенні накладних витрат з точки зору зберігання і обчислень. Накладні витрати визначаються як час, витрачений на обчислення. Результати засновані на припущенні, що всі значення параметрів вже згенеровані, такі як пари ключів RSA, відносні прості числа, значення C_r і K_s. В цьому пункті основна увага приділяється стороні власника

даних і авторизованій користувальницькій стороні, оскільки серверна сторона вже розглянута в попередньому пункті 4.3.

4.4.1 На стороні власника даних

У цьому пункті обговорюються проблеми з впровадженням і накладні витрати при створенні файлу ОБІ на машині власника даних. Спочатку файл даних буде зашифрований з використанням AES (використовується інша хмара з симетричними алгоритмами шифрування), і отриманий файл зашифрованих даних стане основною частиною файлу ОБІ. У Таблиці 4.1 показані накладні витрати на зберігання і час виконання шифрування файлів різних типів даних з використанням алгоритму AES з розміром ключа 256 біти. Накладні витрати на зберігання складають близько 300 байт для всіх файлів. Збільшується обчислювальне навантаження зі збільшенням розміру файлу, щоб досягти більш ніж однієї хвилини для файлу з розміром 571 МБ. Ця операція є основою для будь-якого методу, заснованого на шифруванні даних, перш ніж дані будуть відправлені в хмару. Крім того, важливо вибрати сильний алгоритм шифрування і довший ключ для досягнення більш високої безпеки. Однак власники даних мають гнучкість, відповідно до їх вимог безпеки, щоб зменшити обчислювальні накладні витрати, використовуючи більш короткий ключ або слабший алгоритм шифрування з меншими обчислювальними витратами. Загалом, процес шифрування несе в собі найбільші обчислювальні витрати на стороні власника даних в запропонованому в цій роботі методі.

Таблиця 4.1 – Зберігання та тимчасові накладні витрати для шифрування AES

Назва файлу	Розмір в байтах	Розмір після шифрування в байтах	Збільшення розміру в байтах	AES шифрування в секундах
Sample.docx	14,175	14,485	310	<1
visio.vsd	45,029	45,341	312	<l

Продовження таблиці 4.1

photo.jpg	64,819	65,130	318	<1
video1.wmv	25,327,026	25,327,338	312	1.6
video2.mov	41,670,503	41,670,812	309	2.3
video3.avi	136,318,116	136,318,429	313	14.8
video4.mov	598,787,322	598,787,634	311	86.7

Рисунок 4.11 – Залежність часу AES шифрування від розміру файлу

Обчислення значення Xr

Обчислення КТЗ-рішення для визначення значення X_r є суттєвою частиною запропонованого рішення. На основі цієї цінності засновані політики обміну ключами і контролю доступу. На розрахунок КТЗ не впливає розмір файлу, але на нього впливає кількість користувачів і алгоритм асиметричного шифрування, використовуваний для шифрування значення Cr||Ks. У Таблиці 4.1 показані витрати часу на зберігання і час при обчисленні КТЗ-рішення для різних користувачів з двох до шести. Ці обчислення виконувалися, коли значення Cr||Ks було зашифровано з 1024-бітовим RSA і кожне відносне просте число було 1022-бітним. Результати показують, що сама операція КТЗ вимагає незначного часу виконання. Наприклад, для шести користувачів обчислення КТЗ-рішення зайняло близько 6 мілісекунд. Час

виконання обчислення значення обумовлено головним чином процесом шифрування КТЗ і збільшується зі збільшенням кількості користувачів. Однак, оскільки процес шифрування КТЗ виконується для фіксованого значення довжини Cr||Ks, загальний час виконання пошуку все ще незначний і задовільний. Наприклад, загальний час становив 31 мілісекунду для шести користувачів, а загальний час збільшувалася приблизно на 2 мілісекунди для кожного додаткового користувача. Більшість витрат, викликаних операцією КТЗ - це накладні витрати на зберігання, які збільшуються зі збільшенням числа користувачів. На Рисунку 4.12 показано, що на підставі результатів Таблиці 4.2 розмір X_r лінійно збільшується приблизно на 1000 біт для кожного додаткового користувача, коли довжина ключа КТЗ дорівнює 1024 біти. Наприклад, для шести користувачів розмір дорівнював 6142 біт.

Таблиця 4.2 – Зберігання та тимчасові накладні витрати для розрахунку загального значення X_r

№ користувача	КТЗ (1024 біти) шифрування для Cr||Ks в мс	Підрахунок КТЗ-рішення X_r в мс	Загальний час підрахунку X_r в мс	Довжина X_r в бітах
1	21	3	24	2049
2	22	4	26	3073
3	23	4	27	4097
4	24	5	29	5118
5	25	6	31	6142

Рисунок 4.12 – Розмір X_r в бітах у зв'язку з кількістю користувачів

Додавання зашифрованих ключових слів з можливістю пошуку

Додавання зашифрованих ключових слів в файл ОБІ необхідне для можливості пошуку, як описано в пункті 4.3.

У Таблиці 4.3 показані накладні витрати на зберігання і час виконання, пов'язані з додаванням від одного до п'яти зашифрованих ключових слів. Результати показують помірні обчислювальні накладні витрати, близько 200 мілісекунд, а накладні витрати на зберігання збільшується на 256 біт для кожного додаткового ключового слова. Виконується тільки один раунд HMAC, в той час як пропоноване рішення в Розділі 4 вимагає двох раундів і операції XOR. Причиною цього є скорочення обчислювальних витрат, а також використання алгоритму KMP для пошуку на сервері. Коли операція XOR використовується при шифруванні ключових слів, алгоритм KMP не може застосовуватися, оскільки при використанні операції процес пошуку XOR буде включати операцію XOR перед пошуком того ж шаблону, який суперечить механізму пошуку KMP. Більш того, використання одного раунду HMAC з SHA-256 або SHA-512 має забезпечувати достатній рівень безпеки. Реалізація дозволяє користувачам використовувати HMAC-SHA512 для більшої безпеки, але накладні витрати на зберігання збільшаться в два рази. Наприклад, при

використанні HMAC-SHA256 для п’яти ключових слів загальна необхідна пам’ять становить 1280 біт. При використанні HMAC-SHA512 для п’яти ключових слів необхідне сховище буде 2560 біт. Власник даних повинен віднайти компроміс між рівнем безпеки, необхідним для ключових слів і накладними витратами на сховище і час виконання. Також необхідно ретельно визначити кількість необхідних ключових слів, щоб зменшити накладні витрати, не втрачаючи зручність пошуку.

Таблиця 4.3 – Накладні витрати на сховище і час виконання для додавання зашифрованих ключових слів

№ ключового слова	Тривалість HMAC-SHA256 в мс	Загальний розмір виходу в байтах
1	196	256
2	196	512
3	201	768
4	197	1024
5	199	1280

Додавання перевірки цілісності та аутентичності та створення ОБІ файлу

Додавання доказів цілісності та аутентичності в файл ОБІ має важливе значення, але це призводить до додаткових витрат обчислення і зберігання. У Таблиці 4.3 показані накладні витрати на зберігання і час виконання при розрахунку цілісності та аутентичності файлів різних типів і розмірів. Хеш-функція, яка використовується для дайджесту зашифрованих файлів була SHA-256, а ключ RSA для шифрування дайджесту був 1024-біт. Через збільшення розміру файлу збільшується обчислювальне навантаження, а накладні витрати на зберігання фіксуються на 1024 біт, що є результатом шифрування 256-бітного дайджесту файлу RSA з довжиною в 1024 біт.

Таблиця 4.4 – Зберігання та тимчасові накладні витрати для підтвердження цілісності та достовірності

Вхідне ім’я файлу	Розмір в байтах	Час підрахунку дайджесту та підпису в мс	Накладні витрати на зберігання в байтах
Sample.docx.aes	14,485	51	128
visio.vsd.aes	45,341	51	128
photo.jpg.aes	65,130	133	128
video1.wmv.aes	25,327,338	1910	128
video2.mov.aes	41,670,812	775	128
video3.avi.aes	136,318,429	8714	128
video4.mov.aes	598,787,634	50977	128

Створення ОБІ файлу

Для створення ОБІ-файлу необхідно, щоб всі параметри були вже обчислені. Загальні накладні витрати на зберігання залежать від кількості користувачів, яких авторизовано для доступу до файлу, кількості прикріплених ключових слів і специфікацій використовуваних алгоритмів. У Таблиці 4.5 показано, що загальні накладні витрати при створенні ОБІ-файлу складають 952 байта для файлів різного розміру. Файли ОБІ були створені з п’ятьма прикріпленими зашифрованими ключовими словами і повинні використовуватися п’ятьма авторизованими користувачами. Алгоритми використовувалися зі специфікаціями, визначеними в реалізаціях, тобто RSA-1024, AES-256, SHA-256. У разі великих файлів цими накладними витратами на зберігання може бути знехтувано.

Таблиця 4.5 – Загальні накладні витрати на зберігання для створення файлу ОБІ для п’яти користувачів і п’яти ключових слів

Вхідне ім’я файлу	Розмір в байтах	Розмір ОБІ-файлу в байтах	Накладні витрати на зберігання в байтах
Sample.docx.aes	14,485	15,437	952

Продовження таблиці 4.5

visio.vsd.aes	45,341	46,293	952
photo.jpg.aes	65,130	66,082	952
video1.wmv.aes	25,327,338	25,328,290	952
video2.mov.aes	41,670,812	41,671,764	952
video3.avi.aes	136,318,429	136,319,381	952
video4.mov.aes	598,787,634	598,788,586	952

Процес створення файлу ОБІ вимагає копіювання вмісту зашифрованого файлу в новий файл ОБІ. Ця операція вимагає додаткового час, як показано в Таблиці 4.6. Цього часу можна уникнути, якщо одночасно шифрування AES і обчислення зашифрованого файлового дайджесту виконуються одночасно при створенні ОБІ-файлу. Іншими словами, вихідний файл даних зашифрований в блоках, а потім кожен блок систематизується і використовується для створення файлу ОБІ за один раунд. Тому на цей раз це пов’язано з реалізацією і не розглядається як накладні витрати щодо запонованого рішення.

Таблиця 4.6 – Тимчасові накладні витрати для копіювання вмісту зашифрованого файлу в файл ОБІ

Вхідне ім’я файлу	Розмір в байтах	Копіювання зашифрованого вмісту файлу в ОБІ-файл в мс
Sample.docx.aes	14,485	33
visio.vsd.aes	45,341	42
photo.jpg.aes	65,130	90
video1.wmv.aes	25,327,338	252
video2.mov.aes	41,670,812	391
video3.avi.aes	136,318,429	1284
video4.mov.aes	598,787,634	79560

Основні операції, що виконуються власником даних для створення файлу ОБІ – це шифрування AES, обчислення значення X_r, обчислення HMAC для ключових слів, обчислення дайджесту зашифрованого файлу і шифрування дайджесту. Таблиця

4.7. показує загальний час виконання цих операцій для різних розмірів файлів. Значення X_r було розраховано для п’яти користувачів і було додано п’ять зашифрованих ключових слів.

Таблиця 4.7 – Загальний час, необхідний для створення ОБІ-файлу

Розмір вхідного файлу в байтах	AES в секундах	Підрахунок X_r для 5 користувачів в секундах	Підрахунок дайджесту файлу та підпису в секундах	HMAC-SHA256 для 5 ключових слів в секундах	Загальний час в секундах
14,074	<1	0.029	0.003	0.199	-1
44,032	<l	0.029	0.004	0.199	-1
65,812	<1	0.029	0.005	0.199	-1
26,246,026	1.6	0.029	0.474	0.199	2.302
42,750,493	2.3	0.029	0.720	0.199	3.248
137,237,016	14.8	0.029	5.050	0.199	20.078
598,787,322	86.7	0.029	17.897	0.199	104.825

Рисунок 4.14 – Залежність сумарного часу від розміру файла

Загальний час виконання становid менше 4 секунд для файлів розміром менше 50 МБ і близько 20 секунд для файлу розміром 130 МБ. Для файлу в 571 МБ загальний час виконання становив близько 1,7 хвилини. З результатів, показаних в Таблиці 4.7, ясно, що обчислення параметрів, необхідних для контролю доступу, спільного використання ключів і пошуку, має менший вплив на обчислення. На противагу цьому, шифрування AES, за яким слід обчислювати цілісність та достовірність, має найбільший вплив на обчислення. Тому в запропонованому рішенні операція симетричного шифрування бере на себе більшість обчислювальних накладних витрат, особливо для великих файлів. Однак будь-яке рішення, засноване на шифруванні даних перед відправкою даних в хмару, вимагає принаймні одного раунду шифрування даних.

4.4.2 На стороні авторизованого користувача

Авторизований користувач повинен провести три операції; обчислення значення Cr||Ks із значення X_r, перевірку цілісності та аутентичності зашифрованого файлу даних і дешифрування файлу зашифрованих даних. Таблиця 4.7 показує час виконання операцій для різних розмірів файлів ОБІ. Для розшифровки AES потрібно найбільший час виконання, за яким слід перевірити цілісність та автентичність зашифрованого файлу даних. Обчислення значення Cr||Ks займає невеликий час виконання. Відтворення вихідного файлу зашифрованих даних з файлу ОБІ можна виконати на стороні сервера, а потім сервер відправляє вихідний файл зашифрованих даних з його підписаним дайджестом замість файлу ОБІ. Отже, накладні витрати на копіювання зашифрованого вмісту з файлу ОБІ в новий відтворений файл виключаються на стороні користувача.

Таблиця 4.8 – Виконання операцій на стороні користувача для різних файлів ОБІ

Вхідна назва ОБІ-файлу	Розмір в байтах	Створення оригінального зашифрованого файлу в мс	Перевірка цілісності та аутентичності в мс	Віднаходженн яCr|| Ks із X_r в мс	AES дешифрування в секундах
Sample.docx.dcs	15,337	214	3	29	<l
visio.vsd.dcs	45,290	220	4	29	<1
photo.jpg.dcs	67,082	350	*5*	29	<l
video1.wmv.dcs	26,247,290	1448	474	29	3.5
video2.mov.dcs	42,751,754	1457	720	29	2
video3.avi.dcs	137,238,282	11520	5050	29	32
video4.mov.dcs	598,788,586	13198	17897	29	32.7

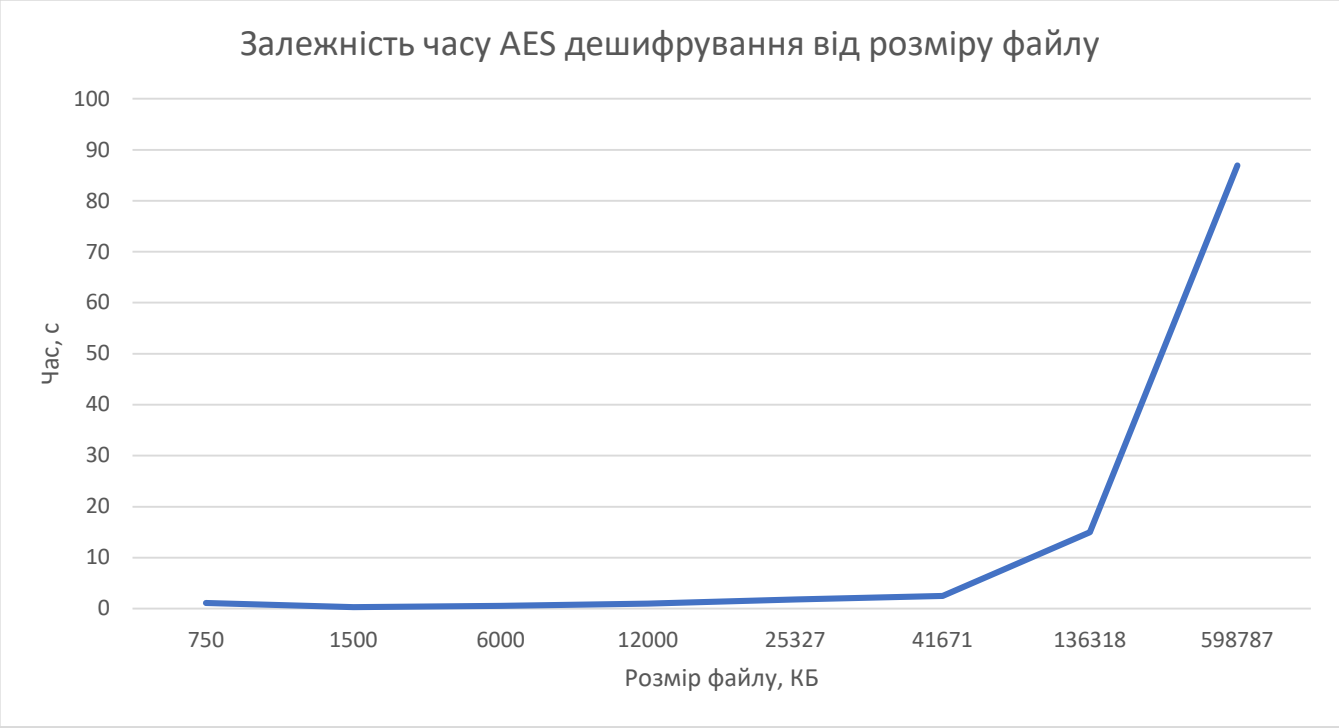

Рисунок 4.15 – Графік залежності від часу

4.5 Порівняльний аналіз отриманих результатів

В таблиці 4.9 приведені результати роботи системи інформаційно-орієнтованої ситстеми охорони здоров'я WSN-хмари представленої на конференції ICDCN, що розглядалась в пункті 2.1.2.

Таблиця 4.9 – Результати роботи інформаційно-орієнтована системи охорони здоров'я

Розмір файлу (КБ)	Час шифрування (мс)
750	344
1500	421

Рисунок 4.16 – Швидкість роботи системи охорони здоров'я

Далі для дослідження цієї дисертації розмір двох файлів доводяться до приблизно такого ж розміру, а потім апроксимуються до точок 750 КБ, 1500 КБ. Результати порівняння предствлені на рисунку 4.17

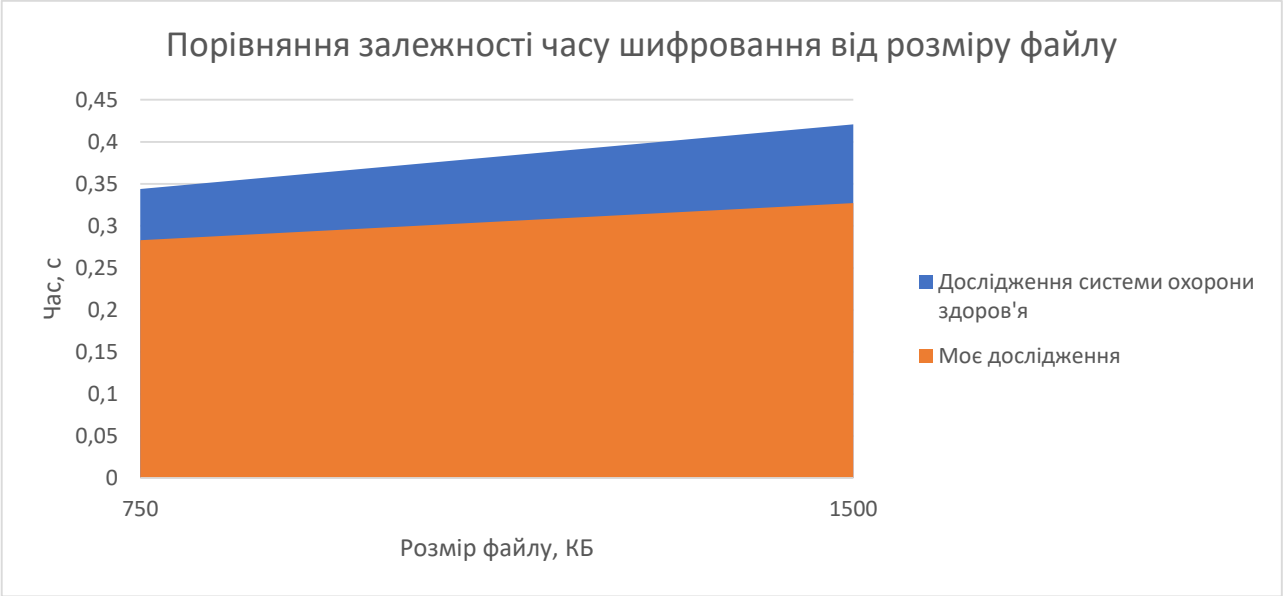

Рисунок 4.17 – Порівняння швидкості рботи для малих файлів

4.6 Резюме і висновок

У цьому пункті обговорюються питання реалізації запропонованого рішення в Розділі 3. Основна увага приділялася навантаженню на зберігання і обчислення, викликаним використанням запропонованого рішення з боку клієнта і сервера. Для обох сторін була встановлена експериментальна платформа, і для реалізації основних операцій звпропонованого рішення була використана платформа Rider від JetBrains для мови програмування C#. Для операцій власника даних, основний код для створення файлу ОБІ з зашифрованого файлу був записаний у файлі класу C# з ім’ям CreatSecureFile.cs. Цей клас обчислює загальний занчення X_r для будь-яких користувачів, викликавши функцію findSolution з CRTfurBiginteger.cs, який було реалізовано для обчислення КТЗ-рішення при використанні цілих чисел розміром більше 64 біти. Крім того, було створено CreatSecureFile для обчислення інших параметрів безпеки, а саме зашифрованих ключових слів для можливості пошуку, перевірки цілісності та достовірності. Потім він прив’язав всі ці параметри до файлу ОБІ таким чином, який можна легко прочитати пізніше.

ReadSecureFile.cs був реалізований для читання параметрів безпеки в ОБІ-файлі і відтворення вихідної зашифрованої файлової форми. Також він містить операції, необхідні на стороні авторизованого користувача, а саме, виявлення секретних значень C_r і K_s з X_r; і перевірки цілісності та аутентичності зашифрованого файлу. Відкрите вихідне програмне забезпечення AES Crypto використовувалося для шифрування вихідного файлу на стороні власника даних, а потім для його дешифрування на стороні користувача.

Файл класу Search.cs був реалізований на сервері для пошуку заданого ключового слова в файлах ОБІ. Для підвищення продуктивності при створенні ОБІ-файлу була додана інформація про те, скільки ключових слів і яка їх довжина. Отже, функція пошуку може обчислювати діапазон всередині ОБІ-файлу, який містить зашифровані ключові слова, тому процес пошуку буде обмежений цим діапазоном. Таким чином, час пошуку не буде залежати від довжини файлу. Крім того, використовувався алгоритм пошуку KMP для підвищення швидкості пошуку.

Результати проведених операцій показують, що запропоноване рішення є простим і не вимагає складних операцій. Крім того, накладні витрати на зберігання при створенні файлу ОБІ є низькими в порівнянні з наданими функціями. Цими функціями є: створення ОБІ-файлу з можливостями пошуку, політикою прихованого контролю доступу та цілісністю і достовірністю, незалежно від того, де він зберігається в хмарі. Запропоноване рішення може бути практично реалізовано з мінімальними витратами на обчислення і зберігання. Доведено, що запропоноване рішення дуже ефективне, так як воно не вимагає складних методів деривації ключів, і файл даних не потрібно шифрувати більше одного разу.

5 СТАРТАП ПРОЕКТ

Завдання розділу полягає в маркетинговому аналізі перспектив реалізації запропонованих науково-технічних рішень та пропозицій, оцінювання можливостей їх ринкового впровадження.

5.1 Опис ідеї проекту

Опис ідеї проекту наведений в таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка клієнтського програмного забезпечення, що виконує шифрування згідно принципів інформаційно-орієнтованої безпеки на довіреній машині користувача	В хмарних обчисленнях	Забезпечення безпеки та конфіденційності даних клієнта, що буде виконуватися в скомпрометованому хмарному середовищі, оскільки відповідальність за шифрування даних несе власник даних та його локальне програмне середовище
	В іншій інформаційній інфраструктурі	Наскрізне шифрування даних; немає необхідності в синхронізації даних з віддаленим центром обробки даних для забезпечення безпеки

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (табл. 5.2).

Інформаційно-орієнтований підхід суттєво відрізняється від традиційних інструментів безпеки на ринку, оскільки забезпечує високий рівень надійності безпеки даних навіть якщо провайдера послуг або інфраструктури хмари скомпрометували.

На ринку потенційні та конкретні конкуренти відсутні, присутні тільки товари замінники – Trusted Computing Module (TCM), що представлена компанією Trusted Computing Group та Next Generation Secure Computing Base (NGSCB).

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Trusted Computing Module (TCM)	Next Generation Secure Computing Base (NGSCB)			
1.	Гнучкі ціни	гнучкі, середні	гнучкі, середня	середня		Ціни залежать від об’єму переданих даних	Об’єм даних низький, оскільки обчислення виконуються на клієнти
2.	Розмір капіталовкладень в інтеграцію з постачальником хмар	60%	70%	100%	Незначні затрти на інтеграцію з хмарними провайдерами потрібні	Немає потреби у значних капіталовкладень	
3.	Рівень безпеки даних	найвищий	середній	найменш надійний	Накладні витрати на підтримку рівня безпеки		Наскрізне шифрування забезпечує кращу безпеку
4.	Затрати обчислювальних ресурсів на роботу системи	10%	100%	25%	На слабких клієнтах можуть бути проблеми з ефективністю роботи		Основні обчислення проходять на стороні клієнта

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

5.2 Технологічний аудит ідеї проекту

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 5.3):

– за якою технологією буде виготовлено товар згідно ідеї проекту?
– чи існують такі технології, чи їх потрібно розробити/доробити?
– чи доступні такі технології авторам проекту?

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розробка клієнта, що виконує шифрування згідно принципів інформаційно-орієнтованої безпеки на довіреній машині користувача	Розробка клієнтського програмного забезпечення використовуючи мову програмування відповідну пристрою, наприклад Java, C++, C#, Objective-C, Swift	+	+
2	Розробка довіреного модуля безпеки	Розробка сервісу використовуючи сучасну серверну	+	+

Продовження таблиці 5.3

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	по принципу ТС, що забезпечить роботу по моделі інформаційно-орієнтованої безпеки і буде інтегруватися з сервісами хмарних послуг	мову програмування, таку як Java або C#		
Обрана технологія реалізації ідеї проекту: Розробка клієнта та довіреного модуля, що виконує шифрування згідно принців інформаційно-орієнтованої безпеки				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Аналіз попиту наведений у таблиці 5.4.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	1700 за рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Сертифікація
6	Середня норма рентабельності в галузі (або по ринку), %	20%

Середня норма рентабельності проекту більша за середній банківський відсоток на вкладення, тому ринок є привабливим для входження за попереднім оцінюванням.

Визначені потенційні групи клієнтів, їх характеристики, та сформований орієнтовний перелік вимог до товару для кожної групи наведений у таблиці 5.5.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Забезпечення високого рівня конфіденційності та безпеки даних	Потенційна група клієнтів є підприємці, представники	- модель оплати - ціна послуг	- модель оплати pay-per-use - можливість забезпечення

Продовження таблиці 5.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
		менеджмент борду, що відповідальні за безпеку інформації в компанії	- необхідний рівень безпеки даних	безпеки в скомпрометованому хмарному середовищі - не значна переплата за додатковий рівень безпеки

Проведений аналіз ринкового середовища, складено таблицю факторів, що перешкоджають ринковому впровадженню проекту (табл. 5.6). Фактори в таблиці подано в порядку зменшення значущості.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Збільшення ціни на розроблення	Збільшення ціни заробітної праці розробників, понаднормові надлишкові витрати	Зміна цінової політики, питання перерозподілу грошових ресурсів
2	З’являються нові конкуренти	Зменшення долі ринку, на ринку	Цінова конкуренція, аналіз недоліків та переваг

Продовження таблиці 5.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
		формуються конкуренти з більш якісними продуктами	конкурентів, зміна стратегії та цілепокладання

Складений перелік факторів, що сприяють ринковому впровадженню проекту, наведено в таблиці 5.7.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Компрометація великих гравців ринку хмарних послуг і як наслідок збільшення попиту	Ріст потреби в конфіденційності даних. Користувачі усвідомлено потребуватимуть більш надійних методів забезпечення безпеки	Маркетингова компанія, залучення інвестицій, глибша автоматизація процесів, розширення робочого штату
2	Зменшення ціни на розроблення	Зменшення вартості заробітної плати розробників	Зміна цінової політики, інвестиції в просування на ринку
3	Технічний прогрес	Придбання нових технічних засобів призведе до необхідності утилізації старих, але	Впровадження більш простих процесів інтеграції в наявну інфраструктуру

Продовження таблиці 5.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Компрометація великих гравців ринку хмарних послуг і як наслідок збільшення попиту	Ріст потреби в конфіденційності даних. Користувачі усвідомлено потребуватимуть більш надійних методів забезпечення безпеки	Маркетингова компанія, залучення інвестицій, глибша автоматизація процесів, розширення робочого штату
		їх можна використати для збільшення обчислювальних можливостей інфраструктури	

Проведений аналіз пропозиції: визначені загальні риси конкуренції на ринку описані в таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	Олігополія, домінує мала кількість фірм	Пришвидшення розробки унікального продукту; впровадження

Продовження таблиці 5.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
		реклами та ефективної PR-стратегії
2. За рівнем конкурентної боротьби - локальний/національний/…	Національний	Вихід на всесвітній ринок
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	Міжгалузева	Розроблення універсального додатку, розроблення додаткових доповнень для конкретної галузі
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	Товарно-видова, різновиди однієї категорії товару, які здатні задовольнити конкретне бажання покупця	Відсутня
5. За характером конкурентних переваг - цінова / нецінова	Не цінова, ціна не є основним фактором конкуренції	Поштовх до зменшення ціни продукту
6. За інтенсивністю - марочна/не марочна	Не прив’язаний до певної марки, може використовувати будь-де	Співпраця з різними марками

М. Портер вирізняє п’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції. Це:

- конкурент, що вже є у галузі,
- потенційні конкуренти,
- наявність товарів-замінників,
- постачальники, що конкурують за ринкову владу,
- споживачі.

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючи їм власні умови співпраці. Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик.

На основі аналізу складових моделі 5 сил М. Портера розроблено перелік факторів конкурентоспроможності, що наведений у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Немає	Немає бар’єрів	Немає необхідності в постачальниках	Надійність інфраструктури, точність аналізу, підтримка системи та ПЗ	На даний час відсутні
Висновки:	Прямі конкуренти	Можливості входу на	Постачальники відсутні,	Клієнти диктують	Необхідним є

Продовження таблиці 5.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Немає	Немає бар'єрів	Немає необхідності в постачальниках	Надійність інфраструктури, точність аналізу, підтримка системи та ПЗ	На даний час відсутні
	на ринку відсутні.	ринок є, проте, це буде дещо складно (через наявних потенційних конкурентів). На ринок можна вийти до 1 року, за рахунок гарної команди спеціалістів	на ринок не впливають; потрібні розробники продукту	умови. Якісний, дешевий, швидкодіючий продукт.	дослідження та моніторинг ринку, можливий аналіз та розробка нових можливостей ОБІ-підходу та вдосконалення реалізованих

На основі аналізу конкуренції, наведеного в таблиці 5.9, а також із урахуванням характеристик ідеї проекту (табл. 5.2), вимог споживачів до товару (табл. 5.9) та

факторів маркетингового середовища (таблиці 5.7 та 5.6), визначено та обґрунтовано перелік факторів конкурентоспроможності, що наведений у таблиці 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Швидкість шифрування	Алгоритм за короткий час дає потрібний результат, на відміну від класичних методів
2	Підтримка клієнтів різних ОС	Користувачі могли б використовувати ОБІ-клієнт з різних пристроїв та різних операційних систем
3	Простота розробки	Швидка розробка проекту з високим якісним результатом

За визначеними факторами конкурентоспроможності (табл. 5.3) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 5.11).

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з потенційними конкурентами						
			–3	–2	–1	0	+1	+2	+3
1	Швидкість шифрування	18							+
2	Підтримка клієнтів різних ОС	20		+					
3	Простота розробки	18					+		

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak)

сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 5.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 5.11).

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: зменшення часу на шифрування даних, зменшення обчислювальних потужностей, надійність	Слабкі сторони: складність алгоритму розробки, особливості ціноутворення
Можливості: високий попит на товар, зменшення коштів/ресурсів на розроблення	Загрози: поява конкурентів, уповільнена швидкість росту ринку, збільшення ціни на розроблення

На основі SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. табл. 5.7).

Визначені альтернативи проаналізовано з точки зору строків та ймовірності отримання ресурсів (табл. 5.13).

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Здешевлення розробки ПЗ	27%	-
2	Розроблення стійкого алгоритму	94%	До 6 міс.
3	Запуск маркетингової компанії	62%	До 1 міс.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 5.14).

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Малий бізнес	так	32%	немає	середня
2	Середній бізнес	так	44%	немає	середня
3	Великий бізнес	так	65%	немає	висока

Продовження таблиці 5.14

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Які цільові групи обрано: малий/середній/великий бізнес					

За результатами аналізу потенційних груп споживачів (сегментів) обирають цільові групи, для яких пропонуватиметься товар, та визначається стратегію охоплення ринку:

- якщо компанія зосереджується на одному сегменті – обирається стратегія концентрованого маркетингу;
- якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – використовується стратегія диференційованого маркетингу;
- якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – використовується масовий маркетинг.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (табл. 5.15).

Таблиця 5.1 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	Стратегія виклику лідера	Стратегія диференційованого маркетингу	Індивідуальний підхід до клієнта; краща якість, ніж у конкурентів	Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмін-ним від товарів можливих конкурентів

Вибір стратегії конкурентної поведінки наведено у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Так	Шукати нових та переманювати клієнтів у конкурентів	Ні. За основу буде взятий підхід орієнтований на безпеку інформації, що не є традиційним рішенням серед конкурентів на ринку хмарних послуг	Стратегія лідера

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 5.5), а також в залежності від обраної базової стратегії розвитку (табл. 5.15) та стратегії конкурентної поведінки (табл. 5.16) розроблено стратегію позиціонування (табл. 5.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.2 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Підтримка ПЗ	Стратегія диференціації	Підтримка клієнтів різних ОС та пристроїв	Функціональність, зручність, підтримка
2	Розроблення ПЗ			
3	Розгортання та масштабування серверної інфраструктури	Стратегія лідерства по витратах	Гнучкі ціни на розгортання системи, швидкість	Низька ціна, надійність доступність
4	Інтеграція з новими провайдерами хмарних послуг			

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.18 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Високий рівень безпеки	Власник даних гарантує шифрування даних. Хмарне середовище не знає про зміст ОБІ-файлу	Конкуренти використовують більш класичні системно-орієнтовані підході до забезпечення безпеки даних
2	Швидкість	Витрати часу на обчислення мінімальні	Конкуренти частково програють у швидкості роботи
3	Можливість пошуку	Можливість пошуку в зашифрованих даних	Конкуренти не реалізують таку можливість або частково дешифрують дані піддаючи їх ризику
4	Забезпечення контролю доступу	Власник даних має можливість контролювати хто має доступ до ОБІ-файлу	Конкуренти мають подібні технології

Надалі розробляється трирівнева маркетингова модель товару: уточняється ідея продукту, його фізичні складові, особливості процесу його надання (табл. 5.19).

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Можна виділити наступні вигоди від використання продукту: високий рівень забезпечення безпеки та конфіденційності даних; можливість пошуку по захищеним даним; контроль доступу</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Складність алгоритмів шифрування
Функція пошуку по зашифрованим файлам
Швидкість кодування та декодування
Універсальність при реалізації інтеграції з постачальниками хмарних послуг
Відсутній термін дії</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: відповідає нормам розробки програмного забезпечення</td></tr>
<tr><td colspan="3">Пакування: Продукт представлений на сайті у вигляді дистрибутивів для Windows, MacOS, Ubuntu. Представлені посилання на клієнти мобільний пристроїв. На сайті міститься: загальна назва продукту, власна назва; опис продукту; функції, які представлені; інструкція для користування; контакти для зв’язку з розробником; підтримка для клієнтів.
Товар буде захищено від копіювання. Товар буде запатентований, та захищений за допомогою спеціальних утиліт.</td></tr>
<tr><td colspan="3">Бренд: «DataLab»</td></tr>
<tr><td></td><td colspan="3">До продажу: тріальна версія</td></tr>
</table>

Продовження таблиці 5.19

Рівні товару	Сутність та складові
III. Товар із підкріпленням	Після продажу: підтримка за ліцензією
Потенційний товар захищений від копіювання через ліцензії та ключі активації	

Наступним кроком визначено цінові межи, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари-субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 5.20).

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	Відсутні	54 грн за 1ТВ даних	800 000 тис. грн	240 грн за 1 ТВ даних

Визначено оптимальну системи збуту, в межах якої приймається рішення (табл. 5.21):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Відходження на сайті продукту, оплата продукту, відсилання на електрону пошту ключа активації	Зберігання, доставка на електрону пошту ключа активації	Виробник - споживач	За допомогою веб-сайту

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 5.22).

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Дуже ретельно обирають товар, багато порівнюють	Інтернет, веб-сайти, e-mail, телефон	Підтримка, індивідуальний підхід, постійне оновлення	Донесення переваг до потенційних клієнтів, інформування про переваги,	Повна безпека в хмарі навіть після компрометації

Продовження таблиці 5.22

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
				розширення ЦА	

5.6 Висновки за розділом

На основі проведеного аналізу можна стверджувати, що є можливість ринкової комерціалізації проекту.

Визначені сильні, слабкі та нейтральні характеристики ідеї проекту, проаналізована технологічна здійсненність ідеї проекту і обрана технологія реалізації ідеї проекту: розробка клієнта та довіреного модуля, що виконує шифрування згідно принців інформаційно-орієнтованої безпеки.

За результатами попередньої характеристики потенційного ринку стартап-проекту, потенційних клієнтів проекту, ступеневого аналізу конкуренції на ринку можна сказати, що проект конкурентоспроможний, попит наявний, пряма конкуренція відсутня, а динаміка ринку відображає необхідність швидкого розвитку та впровадження продукту. Відмічена потреба в підтримці багатьох платформ.

Середня норма рентабельності по ринку визначена на рівні 20%, але визначені фактори конкурентоспроможності сприятимуть рентабельності стартап-проекту вище середнього в обраному сегменті надання товарів і послуг. Можлива ринкова комерціалізація проекту. Серед обраних потенційних груп клієнтів обрано малий/середній/великий бізнес, оскільки проект пропонує інноваційне рішення традиційної задачі, що в перспективі дозволить йому вийти на міжнародний рівень.

Обрана альтернатива розвитку проекту – стратегія виклику лідера; стратегія охоплення ринку – стратегія диференційованого маркетингу; ключові конкурентоспроможні позиції відповідно до обраної альтернативи – низька кількість індивідуальний підхід до клієнта; краща якість, ніж у конкурентів; базова стратегія розвитку – стратегія диференціації.

Отже, подальша імплементація проекту відповідно до проведеного аналізу є доцільною.

6 ВИСНОВКИ

Ця дисертація досліджує концепцію орієнтованої на безпеку інформації (ОБІ) як важливий підхід до підвищення безпеки та приватності даних, що зберігаються клієнтами в хмарних обчислювальних системах, особливо в середовищі публічних хмарних обчислень, де дані можуть часто переміщатися з одного хмарного сервера на інший і можуть бути доступні іншим кінцевим об’єктам. У цьому дослідженні основна увага приділяється захисту фактичних даних від можливих ризиків безпеки в хмарному середовищі без повної впевненості в тому, що хмарний провайдер або третя сторона є надійними. На відміну від підходів, спрямованих на забезпечення безпеки обладнання і додатків, що обробляють дані в хмарі, які вимагають щоб власник даних довіряв провайдеру хмари або третій стороні для забезпечення необхідних вимог безпеки та приватності. Як згадувалося раніше, через природу публічної хмари, дані клієнтів можуть переміщатися всередині хмарного середовища між різними провайдерами або локаціями, які мають різні заходи безпеки і правила конфіденційності. Як обговорювалося в Пункті 3.2, підхід ОБІ ще не є стандартизованою концептуальною основою застосування його до парадигми хмарних обчислень, і це дослідження пропонує вклад на цьому концептуальному рівні.

У цьому дослідженні запропоновано концептуальну основа для застосування підходу ОБІ, метою якого є підвищення безпеки і приватності даних в хмарі. Запропонована структура заснована на визначенні вимог безпеки безпосередньо у фактичних даних, тому дані є самоописовими, самозахищеними та самоохороняємими протягом усього їх життєвого циклу. Власник даних зберігає повну відповідальність за визначення і управління безпекою та приватністю своїх даних, переданих в хмару. Крім того, приватність даних і приватність користувачів, які звертаються до даних, залишаються захищеними навіть від хмарних провайдерів. Захист даних не залежить від надійності провайдера хмарних обчислень і не вимагає надійності третьої сторони. Очікується, що пропонована концептуальна структура

додасть надійний рівень безпеки для даних у хмарі, і, отже, потенційні користувачі можуть мати більше впевненості в переміщенні своїх даних в хмарні сховища. В результаті підвищення рівня безпеки та приватності, що досягається за допомогою запропонованої структури, переваги використання хмарних сервісів можуть бути досягнуті без втрати контролю над безпекою та приватністю даних власником даних або оприлюднення даних за можливих порушень безпеки і приватності, напротивагу зберіганню даних у власному приватному обчислювальному середовищі користувача. Іншими словами, ОБІ підхід намагається надати користувачам хмар можливість захищати свої дані, як якщо б дані все ще перебували під їх контролем в плані безпеки і приватності. У Пункті 3.3 обговорювалися очікувані переваги застосування підходу ОБІ до хмарної моделі. Як обговорювалося в Пункті 3.2, хоча деякі рішення дослідників у вирішенні проблем безпеки в хмарах були мотивовані концепцією ОБІ, між ними немає ніякої згоди щодо будь-яких чітко визначених критеріїв і вимог цієї концепції. Це дослідження, викладене в Пункті 3.2, забезпечує більш чітку і ширшу концептуальну структуру ОБІ для застосування підходу ОБІ до хмарної моделі з трьома основними критеріями. Нижче перераховані ці три основних критерії:

– Всі заходи безпеки і вимоги надаються з самого набору даних.

– Власник даних відповідає за конфігурацію, управління і моніторинг даних та їх характеристик безпеки протягом усього терміну служби даних.

– Безпека і приватність даних і користувачів, які мають доступ до даних, не повинні покладатися на провайдерів хмарних обчислень або довірених третіх осіб.

Другим внеском цієї роботи є пропозиція рішення, заснованого на підході ОБІ для підвищення безпеки і приватності будь-якого типу файлів даних, таких як документи, зображення і відео, що зберігаються і спільно використовуваних в публічному хмарному сховище (див. Розділ 4). Запропоноване рішення засноване на створенні файлу, який називається ОБІ-файл, власником даних для кожного файлу даних, який повинен бути переданий в хмару. Кожен файл ОБІ містить зашифрований вихідний файл даних і набір параметрів безпеки, які використовуються для забезпечення дотримання політик контролю доступу, пошуку ключових слів і

перевірки цілісності та аутентичності файлу даних. Виключно, власник даних встановлює і управляє цими параметрами безпеки для кожного файлу ОБІ протягом усього його життєвого циклу. Файл ОБІ зберігає захищений файл даних в хмарному середовищі, і тільки авторизовані користувачі можуть отримати доступ до зашифрованого файлу даних в своїх довірених доменах. Отже, дані залишаються захищеними від можливих загроз безпеки всередині і поза хмарного середовища, навіть від можливих шкідливих дій самого провайдера хмарних обчислень.

Кожен файл ОБІ має свої власні політики контролю доступу, прикріплені до нього, а політики приховані навіть від провайдера хмарних обчислень, який несе відповідальність за дотримання політик доступу. Крім того, кількість і особистості авторизованих користувачів, які можуть мати доступ до даних прихований навіть від провайдера хмарних обчислень. Китайська теорема про залишки (КТЗ) ефективно використовується для обчислення загального значення X_r, що містить два секретних значення. У запропонованому рішенні КТЗ використовується для розподілу двох секретних значень; одним значенням є секретний ключ K_s для дешифрування файлу даних та інше секретне значення C_r для аутентифікації авторизованих користувачів в процедурі доступу без виявлення особистостей користувачів. В результаті запропоноване рішення поєднує в собі загальний секретний ключ і примусове застосування політик контролю доступу в одному механізмі, що зводить до мінімуму накладні витрати обчислень і управління ключами. Кожен ОБІ-файл має своє загальне значення X_r, що додається, та яке включає певні політики доступу і симетричний ключ K_s цього файлу ОБІ. Ні власник даних, ні користувачі не повинні зберігати або замінювати цей ключ, так як кожен ключ буде надійно прикріплений до відповідного файлу. Тільки авторизовані користувачі можуть отримувати значення K_s і C_r із загального значення X_r. Отже, тільки авторизовані користувачі можуть отримати доступ і дешифрувати файл даних в ОБІ-файлі. Навіть хмарний сервер, який зберігає файли ОБІ, не може виявити секретне значення K_s із загального значення X_r. Більш того, якщо один файл ОБІ стає скомпрометованим, інші файли ОБІ не будуть порушені, тому що кожен файл має свої унікальні секретні значення. Як описано в Розділі 4, тільки власник даних може змінювати політики контролю доступу для

кожного ОБІ файлу шляхом безпечної і ефективної зміни його значення X_r. Реалізація та результати експерименту, представлені в Розділі 5, показують, що обчислення X_r має низькі обчислювальні витрати.

В файлі ОБІ також можна безпечно здійснювати пошук, прикріплюючи до нього зашифровані ключові слова. Тому пошук можна здійснювати навіть у нетекстових файлах по секретним ключовим словам. Ключові слова зашифровані і прикріплені до ОБІ-файлу власником даних, і тільки авторизовані користувачі можуть шукати ключові слова в файлах ОБІ. Прикріплені ключові слова зашифровуються власником даних, щоб приховати їх від неавторизованих осіб, включаючи провайдера хмарних обчислень. Ця можливість пошуку не залежить від бази даних, що зберігається у провайдера хмарних обчислень, і вимагає лише звичайного процесу шифрування і простого методу пошуку. На етапі імплантації процедура пошуку призначена для прискорення процесу пошуку, використовуючи ефективний алгоритм пошуку рядків, тобто алгоритм Кнута-Морріса-Пратта (Knuth-Morris-Pratt, - KMP), як описано в Розділі 5. Результати виконання реалізації показує задовільну ефективність пошуку, в той час як ключові слова залишаються надійно захищеними.

Пропоноване рішення, засноване на підході ОБІ, забезпечує платформу, незалежну від обчислювального середовища, і підходить для складної і динамічної природи хмарної середовища, особливо у випадку публічної хмари. Файл ОБІ має свій зашифрований файл даних, а вимоги безпеки до нього прив’язані як параметри безпеки, які також відсилають до метаданих безпеки. Кожен ОБІ-файл має свої незалежні параметри безпеки, включаючи докази цілісності та аутентичності, прикріплені до нього, і може управлятися індивідуально власником даних. Тому власники даних можуть бути впевнені в безпеці своїх файлів ОБІ, навіть якщо файли передаються між різними хмарними провайдерами та різними географічними локаціями. Незалежно від хмарного сервера, в якому знаходиться ОБІ-файл в хмарному середовищі, авторизовані користувачі можуть здійснювати пошук, доступ та перевірку цілісності, оскільки вони вбудовані в файл і не залежать від обчислювального середовища. Більш того, коли провайдер хмари створює копії для

резервного копіювання або поліпшення доступу для декількох користувачів або з яких-небудь інших причин, параметри безпеки вихідного файлу ОБІ підтримуються скопійованими версіями.

Результати, отримані при тестуванні реалізованих операцій запропонованого рішення, показують, що реалізація, а отже і відповідне рішення, проста і не вимагає складних операцій, що вимагають великих обчислювальних ресурсів, а також високого обслуговування. Крім того, запропоноване рішення може бути практично використане для будь-якого типу файлу даних з помірними накладними витратами з точки зору зберігання і обчислювальних ресурсів. Крім того, запропоноване рішення дозволяє власнику даних гнучко вибирати алгоритми шифрування і хешування на основі необхідних сильних сторін безпеки і додатків.

ПЕРЕЛІК ПОСИЛАНЬ

1. A break in the clouds: towards a cloud definition / M.Lindner, J. Caceres, R. Luis, V. Luis. // ACM SIGCOMM Computer Communication Review. – 2009. – С. 50–55.

2. Mell P. The NIST Definition of Cloud Computing [Електронний ресурс] / P. Mell, T. Grance // National Institute of Standards and Technology. – 2011. – Режим доступу до ресурсу: https://csrc.nist.gov/publications/detail/sp/800-145/final (дата звернення 05.03.2018)

3. Yeun C. Cloud computing security management / C. Yeun, S. Almulla. // Engineering Systems Management and Its Applications (ICESMA). – 2010. – 2nd International Conference on Engineering System Management & Applications

4. Mutavdžić R. Cloud computing architectures for national, regional and local government / Ratko Mutavdžić. // MIPRO. – 2010.

5. Patidar S. A Survey Paper on Cloud Computing / S. Patidar, D. Rane, P. Jain. // Advanced Computing & Communication Technologies (ACCT). – 2012.

6. Lin D. Data protection models for service provisioning in the cloud / D. Lin, A. Squicciarini. // The ACM Symposium on Access Control Models and Technologies (SACMAT). – 2010. – С. 183–192.

7. Controlling data in the cloud: outsourcing computation without outsourcing control / [J. Staddon, R. Masuoka, J. Molina та ін.]. // ACM Conference on Computer and Communications Securit. – 2009. – №9. – С. 85–90.

8. Privacy preserving cloud data access with multi-authorities / J.Taeho, L. Xiang-Yang, W. Zhiguo, W. Meng. // INFOCOM. – 2013 – 2013 Proceedings IEEE.

9. Chen L. Adaptive Data Replicas Management Based on Active Data-centric Framework in Cloud Environment / L. Chen, D. Hoang. // High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC). – 2013 – №10.

10. Kangsheng S. Historical development of the Chinese remainder theorem / Shen Kangsheng. // Archive for History of Exact Sciences. – 1988. – №38. – С. 285–305.

11. Bogomolny A. Chinese Remainder Theorem [Електронний ресурс] / Alexander Bogomolny // Interactive Mathematics Miscellany and Puzzles – Режим доступу до ресурсу: https://www.cut-the-knot.org/blue/chinese.shtml (дата звернення 04.03.2018)

12. Sorenson J. Genetic algorithms for the extended GCD problem / V. Piehl, J. Sorenson, and N. Tiedeman // Journal of Symbolic Computation. – 1997.

13. Fast Parallel Garner Algorithm for Chinese Remainder Theorem / [L. Yongnan, X. Limin, L. Aihua та ін.]. // International Conference on Network and Parallel Computing (NPC). – 2017. – №9. – С. 164–171.

14. Wassenberg J. Fast keyed hash/pseudo-random function using SIMD multiply and permute / J. Wassenberg, J. Alakuijala. // Google Research. – 2016. – №2016.

15. Khali H. A system-level architecture for hash message authentication code / H. Khali, R. Mehdi, A. Araar. // ICECS. – 2005. – №12.

16. Ferguson N. Practical Cryptography / N. Ferguson, B. Schneier., 2003.

17. big-O notation [Електронний ресурс] // U.S. National Institute of Standards and Technology. – 2004. – Режим доступу до ресурсу: https://xlinux.nist.gov/dads/HTML/bigOnotation.html (дата звернення 01.03.2018)

18. Distributed Searchable Symmetric Encryption / [C. Bosch, P. Andreas, B. Leenders та ін.]. // Privacy, Security and Trust (PST). – 2014. – №12.

19. Zonghua Y. Research on asymmetric searchable encryption / Y. Zonghua, W. Yudong. // AIP Conference Proceedings. – 2017.

20. Безпека даних в хмарних середовищах: матеріали V міжнар. наук.-практ. конф. з інформаційних систем та технологій, 1-2 грудня 2017 р., Київ / відп. ред. А. В. Писаренко. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 28 с.

21. Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень [Текст] / Пирожков О., Савчук О. // Інфокомунікаційні системи та технології. – 2018. – № 2(2). – С. 32-36

22. A cloud security framework for a data centric WSN application / S.Sayantani, D. Rounak, D. Suman, N. Sarmistha. // International Conference on Distributed Computing and Networking. – 2016. – №17.

23. Smoot S.R. Private Cloud Computing / Stephen R. Smoot, Nam-KeeTan. – Morgan Kaufmann Publishers B, 2011. – 424 с.

24. Rountree D. The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in Theory and Practice / Derrick Rountree, IleanaCastrillo. – Newnes, 2013. – 172 с.

25. Qing Li Applications integration in a hybrid cloud computing environment: modelling and platform / Qing Li, Zeyuan W., Weihua Li, Jun Li, Cheng Wang, Ruiyang Du // Enterprise Information Systems. – 2013. – 7 (3). – С. 237–271

26. NIST Special Publication 500-293, US Government Cloud Computing Technology Roadmap, Release 1.0 (Draft), Volume II Useful Information for Cloud Adopters, 2011. – 85 с.

27. Mell P. Effectively and Securely Using the Cloud Computing Paradigm / P. Mell, T. Grance. – NIST. Information Technology Laboratory. – 2009. – Том 10 – С. 7.

Додаток А. Результати створення структури ОБІ підходу

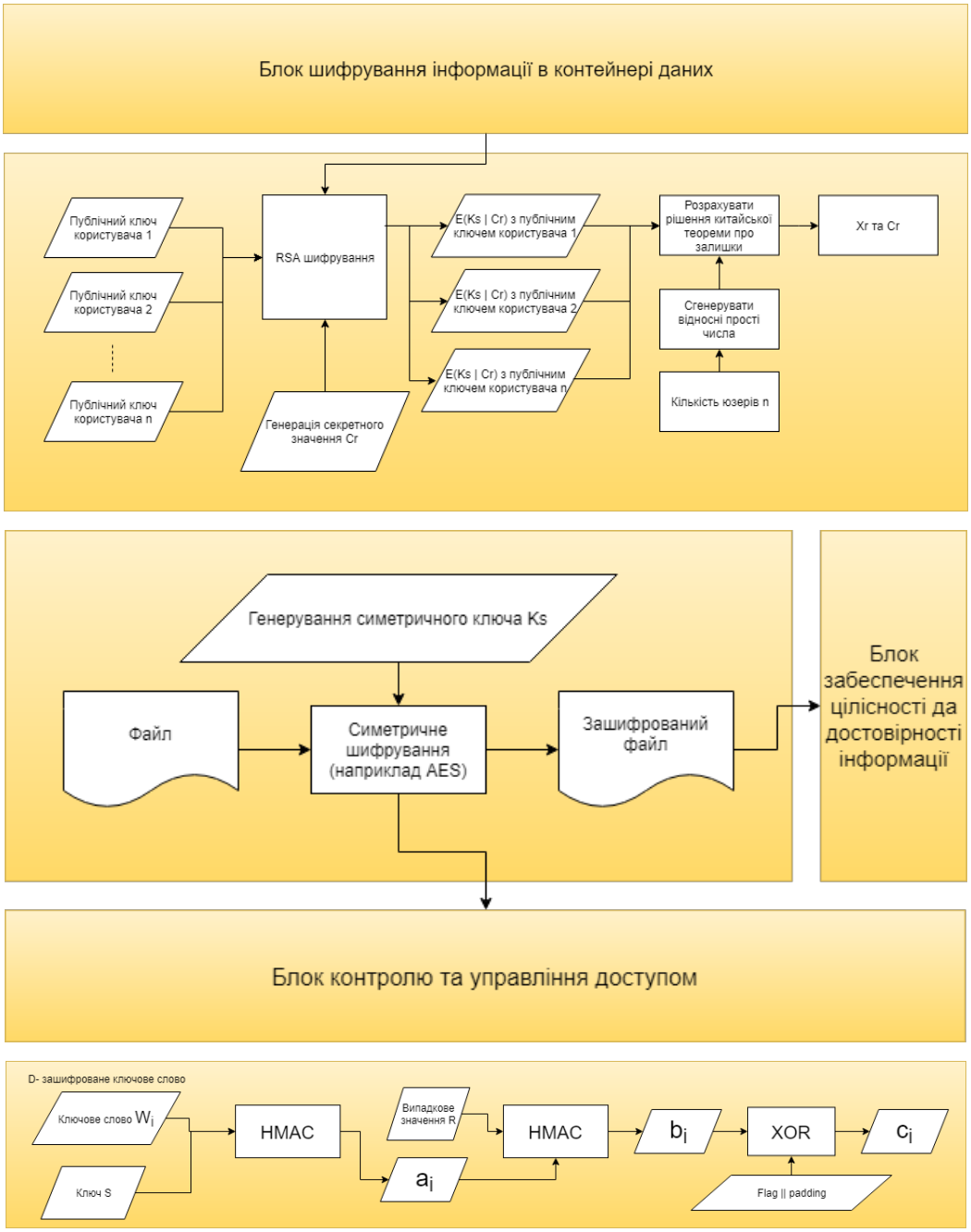

Рисунок А.1 – Результати створення структури ОБІ підходу

Додаток Б. Лістинг програми

```
using System;
using System.Diagnostics;

public class CreateSecureFile
{
    public static void Main(string[] args)
    {
        string User = "Pirozhkov"; //Id юзера
        //додаємо ключові слова
        string keyword1 = "Information";
        string keyword2 = "Conf";
        int kw_count = 2; //Кількість зашифрованих тегів
        Stopwatch sw = new Stopwatch();
        sw.Start();
        //Кодуємо алгоритмом SHA-256
        sbyte[] encodedKeyword = HMACSHA256.encode("keyforKeywords", keyword1);
        sbyte[] encodedKeyword2 = HMACSHA256.encode("keyforKeywords",keyword2);
        sw.Stop();
        Console.WriteLine("Розрахунок SHA-256 хешу тегів зайняв:" + sw.Elapsed + " мс");

        //Кінець блоку шифрування тегів

        // Блок шифрування RSA
        Stopwatch swRSA = new Stopwatch();
```

```
swRSA.Start();
BigInteger C_K = System.Numerics.BigInteger.Parse("10444332252559723399888232537479");
long c = 1044433225;
Console.WriteLine("Значення Cr " + c);

sbyte[] data = C_K.toByteArray();
/* RSA шифрування масиву байтів Cr||Ks з кожним публічнім ключем авторизованого користувача */
    sbyte[] rsaEncoded1 = RSAforBytes.rsaEncrypt(data, "public.key");
    sbyte[] rsaEncoded2 = RSAforBytes.rsaEncrypt(data, "public2.key");
sbyte[] rsaEncoded3 = RSAforBytes.rsaEncrypt(data, "public3.key");
sbyte[] rsaEncoded4 = RSAforBytes.rsaEncrypt(data, "public4.key");
sbyte[] rsaEncoded5 = RSAforBytes .rsaEncrypt(data, "public5.key");
sbyte[] rsaEncoded6 = RSAforBytes.rsaEncrypt(data, "public6.key");
sbyte[] rsaEncoded7 = RSAforBytes.rsaEncrypt(data, "public7.key");
sbyte[] rsaEncoded8 = RSAforBytes.rsaEncrypt(data, "public8.key");
sbyte[] rsaEncoded9 = RSAforBytes.rsaEncrypt(data, "public9.key");
sbyte[] rsaEncoded10 = RSAforBytes.rsaEncrypt(data, "public10.key");

swRSA.Stop();
 //Кінець RSA шифрування
Console.WriteLine("RSA шифрування зайняло:" + swRSA.elapsed() + " мс");
Console.WriteLine("Довжина Cr||Ks:" + data.Length + " байт");
Console.WriteLine("Довжина зашифрованого Cr||Ks:" + rsaEncoded1.Length + " байт");
// Кінець шифрування
// Транформування зашифрованого масиву даних в BigInteger
BigInteger transformed1 = new BigInteger(rsaEncoded1);
```

```
            BigInteger transformed2 = new BigInteger(rsaEncoded2);
            BigInteger transformed3 = new BigInteger(rsaEncoded3);
            BigInteger transformed4 = new BigInteger(rsaEncoded4);
            BigInteger transformed5 = new BigInteger(rsaEncoded5);

            // Відносні прості числа
            BigInteger prime1 =
System.Numerics.BigInteger.Parse("21514311133133111315151515771913511111111111
1111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
11
11111111111111111111111111111111111111111111111111111111111111111111
11
11111111111111111111111111111111111111111111111111111111111111111111
11 11111111111111111111");
            BigInteger prime2 =
System.Numerics.BigInteger.Parse("21514311133133111315151515771913511111111111
1111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
11
11111111111111111111111111111111111111111111111111111111111111111111
11
11111111111111111111111111111111111111111111111111111111111111111111
11 11111111111111111113");
            BigInteger prime3 =
System.Numerics.BigInteger.Parse("21514311133133111315151515771913511111111111
1111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
11
11111111111111111111111111111111111111111111111111111111111111111111
```

```
11
1111111111111111111111111111111111111111111111111111111111111111111111
11 11111111111111111111115");
            BigInteger prime4 =
System.Numerics.BigInteger.Parse("21514311133133111315151515771913511111111111
1111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111
11
1111111111111111111111111111111111111111111111111111111111111111111111
11
1111111111111111111111111111111111111111111111111111111111111111111111
11 11111111111111111111117");
            BigInteger primen5 =
System.Numerics.BigInteger.Parse("21514311133133111315151515771913511111111111
1111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111
11
1111111111111111111111111111111111111111111111111111111111111111111111
1
1111111111111111111111111111111111111111111111111111111111111111111111
11 11111111111111111111119");
            //знайти рішення для КТЗ

            Stopwatch swKTZ = new Stopwatch();
            swKTZ.Start();
            BigInteger[] a = new BigInteger[] {transformed1, transformed2,
transformed3, transformed4, transformed5};
            BigInteger[] n = new BigInteger[] {prime1, prime2, prime3, prime4, prime5};
            // Пошук рішення КТЗ Xr для цього файлу
            // Пошук рішення КТЗ Xr для цього файлу
```

```
            BigInteger x = CRTforBigInteger.findSolution(a, n);
            swKTZ.Stop();
            Console.WriteLine("Довжина кожного n в бітах: " + n[0].bitLength());
            Console.WriteLine("Підрахунок Xr зайняв: " + swKTZ.Elapsed() + " мс" );
            //Перевірка, чи Xr більше 0
            if (x.compareTo(System.Numerics.BigInteger.Zero) > 0)
            {
                Console.WriteLine("Рішення КТЗ Xr: " + x);
                Console.WriteLine("Довжина Xr в бітах:" + x.bitLength());
            }
            else
            {
                Console.WriteLine("Значення Xr негативне");
            }
        //вкажіть зашифрований файл для створення ОБІ-файлу
            Console.Write("Введіть ім'я файла з розширенням: ");
            string InputFile;
            Console.ReadLine(InputFile);
            File InFile = new File(InputFile); //input file
            if (!InFile.exists())
            {
                Console.WriteLine("File does not exist.");
                Environment.Exit(0);
            }
            // Розрахунок вхідного дайджесту файла і сигнатури власника даних
            string private_key_file = "prkey_dataowner.key"; //приватний ключ власника
даних
            Stopwatch swDigest = new Stopwatch();
            swDigest.Start();
            //Підрахунок дайджесту файла
```

```
        sbyte[] fileContainingDigest = Hash.createFileHash(InputFile, "SHA-256");
        // Підпис файлу
        sbyte[] fileSignature = RSAforBytes.rsaEncrypt(fileContainingDigest, private_key_file);
        Console.WriteLine("Довжіна підпису файла у байтах: " + FSignature.Length);
        //Зупинка таймеру обчислювання дайджеста та сігнатури
        swDigest.Stop();
        Console.WriteLine("Підрахунок дайджеста та сігнатури зайняв:" + swDigest.Elapsed() + " ms");
        // Кінець розрахунків дайджеста та сігнатури
        //геенрування ОБІ-файла
        Console.Write("Введіть назву та росширення вихідного ОБІ файла: ");
        BufferedReader outc = new System.IO.StreamReader(System.in);
        string outfileExtension = outc.ReadLine();
        File file = new File(outfileExtension); //output file
        Stopwatch swOutput = new Stopwatch();
            swOutput.Start();

        RandomAccessFile outputFile = new RandomAccessFile(file, "rw");

        outputFile.writeInt(kw_count); //кількість ключових слів

        outputFile.write(encodedKeyword);
        outputFile.write(encodedKeyword2);
        outputFile.write(encodedKeyword3);
        outputFile.write(encodedKeyword4);
        outputFile.write(encodedKeyword5);
        outputFile.writeUTF(User); // запис Id користувача
        sbyte[] xr = x.toByteArray(); //Конвертація BigInteger Xr в масив байтів
        // calculating the length of the xr so we can read it later
```

```
int xlength = xr.Length;
outputFile.writeLong(Cr); // Cr
outputFile.writeInt(xlength); // довжина Xr
outputFile.write(xr); //спільне Xr
outputFile.write(FSignature); //додавання цифрового підпису
//зчитування
RandomAccessFile in = new RandomAccessFile(InFile, "r");
// Перезапис байтів з вхідного у вихідний
sbyte[] buf = new sbyte[1024];
int len;
while ((len = in.read(buf)) > 0)
{
f.write(buf, 0, len);
}
Console.WriteLine("Нова довжина ОБІ-файлу:" + f.length());
f.seek(0);
f.close();
in.close();
Console.WriteLine("ОБІ-файл зроблено");
      swOutput.Stop();
Console.WriteLine("ОБІ-файл сгенеровано за:" +swOutput.Elapsed() + " мс");
Console.WriteLine(DateTime); // show end date and time
}
private static string DateTime
{
      get
      {
      DateFormat df = new SimpleDateFormat("yyyy-MMdd_ hh:mm:ss");
      df.TimeZone = TimeZone.getTimeZone("PST");
      return df.format(DateTime.Now);
```

```
            }
        }
    }

    public class HMACSHA256
    {
        public static sbyte[] ExecuteEncode(string key_input, string data_input)
        {
            Mac hash_algo = Mac.getInstance("HmacSHA256"); // Вибір хеш алгоритму
            SecretKeySpec secret_key = new SecretKeySpec(key_input.GetBytes(), "HmacSHA256");
            hash_algo.init(secret_key); //Генерація на основі переданого ключа

            return hash_algo.doFinal(data_input.GetBytes()); //Складання хеша
        }
    }

    using System;

    public class CalculateKTZ
    {
        // Розрахунок рішення для китайської теореми про залишки
        public static System.Numerics.BigInteger Execute(System.Numerics.BigInteger[] input_a, System.Numerics.BigInteger[] input_n)
        {
            if (input_a == null || input_n == null) {
                Console.WriteLine("Жоден з аргументів не може бути пустим");
                return null;
            }
            if (input_a.Length < 2 || input_n.Length < 2) {
```

```
            Console.WriteLine("Довжина кожного елемента повинна бути більше двох");
            return null;
        }
        if (input_a.Length != input_n.Length) {
            Console.WriteLine("Довжина обох аргументів повинна дорівнювати");
            return null;
        }
        System.Numerics.BigInteger x = input_a[0];
        System.Numerics.BigInteger nInverse, y, z;
        System.Numerics.BigInteger ni = System.Numerics.BigInteger.One;
        int i = 0;
        while (i < input_a.Length - 1) {
            ni = ni * input_n[i];
            //перевіримо чи n відносно просте
            if (!System.Numerics.BigInteger.One.Equals(input_n[i + 1].gcd(ni))) {
                Console.WriteLine("The n's are not relatively prime->" + input_n[i + 1] + "," + ni);
                return null;
            }
            nInverse = cryptoBig.inverse(n[i + 1], ni);
            if (nInverse.compareTo(0) == -1) {
                nInverse = nInverse + input_n[i + 1];
            }
            z = input_a[i + 1] - x;
            z = z * nInverse;
            y = z.remainder(input_n[i + 1]);
            if (y.compareTo(0) == -1) {
                y = y + n[i + 1];
            }
```

```
            x = x + ni * y;
            i++;
        }
        return x; // рішення КТЗ
        }
    }

    public class EuclidHelper
    {
        //повертає g в ступены (-1) (mod p)
        public static System.Numerics.BigInteger
            ExecuteInverse(System.Numerics.BigInteger input_p,
System.Numerics.BigInteger input_g)
        {
            System.Numerics.BigInteger[] result = ExtendedEuclid(input_p, input_g);
            if (result[2].compareTo(0 as System.Numerics.BigInteger) != -1) {
                return input_p + result[2];
            }

            return result[2];
        }


        //Ця функція виконає розширений алгоритм Евкліда, щоб знайти GCD для
значень input_a та input_b
        public static System.Numerics.BigInteger[]
            ExecuteEEA(System.Numerics.BigInteger input_a,
System.Numerics.BigInteger input_b)
        {
            System.Numerics.BigInteger[] result = new BigInteger[3];
```

```
        System.Numerics.BigInteger q;

        if (input_b.Equals(0 as System.Numerics.BigInteger)) {
            result[0] = input_a;
            result[1] = 1 as System.Numerics.BigInteger;
            result[2] = 0 as System.Numerics.BigInteger;
        }
        else
        {
            // В іншому випадку зробити рекурсивний виклик
            q = input_a / input_b;
            result = ExecuteEEA(input_b, a.remainder(input_a));

            System.Numerics.BigInteger s = result[2] * q;
            System.Numerics.BigInteger temp = result[1] - s; // - ans[2]*q;

            result[1] = result[2];
            result[2] = temp;
        }

        return result;
    }
}
```

Додаток В. Тези доповіді «Безпека даних в хмарних середовищах» на V Міжнародній науково-практичній конференції «Winter InfoCom Advanced Solutions 2017»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНСТИТУТ МОДЕРНІЗАЦІЇ ЗМІСТУ ОСВІТИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

WINTER INFOCOM ADVANCED SOLUTIONS 2017

МАТЕРІАЛИ

V МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

CONFERENCE PROCEEDINGS

5th SCIENTIFIC AND PRACTICAL CONFERENCE

КИЇВ, УКРАЇНА

1-2 ГРУДНЯ 2017 РОКУ

ЗМІСТ/CONTENTS

Безпека даних в хмарних середовищах

Пирожков Олексій Юрійович
КПІ ім. Ігоря Сікорського
Київ, Україна
a.y.pirozhkov@gmail.com

Савчук Олена Володимирівна
КПІ ім. Ігоря Сікорського
Київ, Україна
savchuk_l1@ukr.net

Анотація. **Робота присвячена концепції забезпечення безпеки та конфіденційності даних в хмарних обчисленнях, використовуючи підхід орієнтації на безпеку даних. Приводиться спосіб контролю доступу до ресурсів хмарного середовища.**

Ключові слова: безпека хмарних обчислень, безпека даних, орієнтація на безпеку даних.

Вступ

У хмарних обчисленнях дані користувачів, в основному, зберігаються у віртуальних сховищах хмарних інфраструктур. В таких публічних моделях як "програма як послуга" (SaaS) та "робоче місце як сервіс" (DaaS) користувачі володіють лише даними, які знаходяться на зберіганні [1, 2]. Все обладнання та програмне забезпечення, залучене до зберігання та обробки інформації, знаходиться у власності сервіс провайдерів. В інших моделях, таких як публічні IaaS и PaaS моделі, користувач має доступ до обробки даних та до програмного забезпечення, при цьому доступу до апаратного забезпечення немає [3].

Відповідно, з точки зору користувача хмарного сервісу, найбільш цінним активом в хмарному середовищі є його дані, особливо ті дані, що містять інформацію делікатного характеру і вимагають особливого ставлення, а саме: дані урядового характеру, охорони здоров'я та фінансового спектру. Переваги, що надаються за використання хмарних обчислень, дають користувачам, що раніше розміщували делікатні дані на своїх ПК, більш привабливі перспективи до аутсорсингу своїх даних в хмару. Як і будь-які інші послуги в мережі Інтернет, хмарні сервіси також зазнають атак на системи безпеки. Компрометація доступності хмарних послуг, як правило, призводить до короткострокових ефектів і пошкодження можуть бути відновлені. Компрометація конфіденційності та приватності даних споживачів хмарних послуг, може привести в свою чергу до довгострокових ефектів і будь-які втрати можуть бути достатньо важким для відновлення.

Внутрішні ризики можуть бути від користувачів-зловмисників, які використовують той самий хмарний сервіс, і пом'якшення ризиків, в такому випадку, залежить повністю від хмарного провайдера і виходить з-під контролю власника даних. З точки зору власників даних, хмарне середовище невидиме і власник даних не впевнений в тому, як його/її дані захищені від ризиків, пов'язаних з безпекою [4]. Отже, власники даних стурбовані безпекою та приватністю їх даних і мають бажання зберегти свої дані в безпеці, навіть від сервіс провайдера хмарних послуг. Крім того, вони надають перевагу власноруч управляти політикою безпеки своїх даних в безпечному режимі, як ніби ці дані зберігалися на їх ПК.

Концепція орієнтованого на безпеку даних підходу

В даному пункті визначена концепція орієнтованого на безпеку даних відходу (care data). Ця концепція бере за основу огляд попередніх дослідницьких робіт по концепціям орієнтації на безпеку даних (ОБД) стосовно моделі хмари [5, 6]. В даній концепції безпека даних забезпечена перш, ніж покинути довірений домен власника даних. А параметри безпеки приєднані до даних як частина їх метаданих. В ідеальній ситуації, лише облікові дані, які використовуються для відміни або доступу за допомогою захисту, залишаються поза даними і власником даних та авторизованих користувачів відповідно до їх прав доступу. Будь-які інші необхідні параметри безпеки мають бути додані до даних.

Наступний список виділяє критерії, яким має відповідати забезпечене рішення безпеки, на основі ОБД підходу:

1. Кожен набір даних – це автономний контейнер, що здатен самостійно описувати себе та захищати. Таким чином, вимоги безпеки для кожного набору даних та властивостей забезпечуються всередині нього та не залежать від об'єктів за межами набору даних, за винятком деяких основних процесів обробки даних.

2. Захист даних не залежить від сервіс-провайдеру або довіреної третьої особи.

3. Лише власник даних несе відповідальність за створення і управління цими вимогами безпеки та характеристики для кожного набору даних з моменту його створення до кінець життєвого циклу набору даних.

4. Всі операції, пов'язані з доступом до захищених даних, встановлюються уповноваженими користувачами, забезпечуються дотриманням політики безпеки даних та виконуються без компрометації приватності або порушення конфіденційності користувача.

Data-орієнтоване рішення

Data-орієнтоване рішення призначене для досягнення наступного переліку бажаних вимог до застосунків в середовищі хмарних обчислень [7]:

– дані зашифровані та доступні лише для авторизованих користувачів;

– дані доступні для пошуку без загрози для їх конфіденційності;

– дані відповідають вимогам самозахисту та необхідним параметрам безпеки;

– параметри контролю доступу приховані від провайдерів хмарних серверів та інших користувачів;

– провайдер серверу не знає кількості або особистості користувачів, що авторизовані та мають право доступу до даних;

– несанкціоновані суб'єкти, в тому числі постачальники послуг, не можуть отримати доступ до даних або інформацію про дії, що проводяться на даних авторизованими користувачами;

– дані містять всю необхідну інформацію для перевірки їх цілісності да достовірності для авторизованих користувачів;

– взаємодія між власниками даних та авторизованими користувачами має бути мінімальною, особливо щодо ключових цілей управління.

Ці вимоги визначені набором модулів, кожен з яких точно реалізує принаймні одну із зазначених вище вимог. Всі параметри, необхідні для досягнення функцій безпеки, прикріплюються до файлу даних і в результаті формують орієнтований на безпеку даних файл з відповідною назвою.

Контроль доступу

У даній роботі запропоноване рішення використовує китайську теорему о залишках [8] і реалізує криптосистему з публічним ключем для безпечного обміну даними, що зберігаються в середовищі хмарних обчислень серед авторизованих користувачів. Ці дані, які іноді згадуються як ресурси, або файли, захищені симетричним методом шифрування із секретним ключом K_s. У даній роботі, ресурсом може бути набір даних або файл, який містить дані будь-якого типу, в тому числі текст, аудіо, зображення або відео. Для поширення секретного ключу для авторизованих користувачів, його зашифровано з використанням методу публічного шифрування відкритого ключа користувача. Шифрування також включає в себе інше значення C_r, яка буде використовуватися в якості відповіді на запит сервера, коли користувач робить запит щодо доступу до ресурсу. Лише авторизовані користувачі, які мають відповідний приватний ключ, можуть розшифрувати шифротекст C_r та K_r для отримання C_r, щоб отримати доступ до ресурсу r. Параметри C_r та K_s зчеплені для формування $C_r||K_s$ та розглядаються як єдине значення [9]. Це значення шифрується за допомогою публічного ключа E_{Kpubi} для кожного авторизованого користувача u_i. В результаті отримуємо шифротекст $(E_{Kpubi}(Cr||Ks))$ для цього користувача u_i, де $i=1, 2, 3, ..., k$, та k - кількість авторизованих користувачів, що мають доступ до ресурсу r.

Секретний ключ K_S, а також значення C_r є унікальними для кожного файлу, навіть якщо він використовується одним користувачем для різних файлів. Якщо значення для одного файлу було скомпрометовано, інші файли залишаються в безпеці. Секретне значення C_r використовується авторизованим користувачем, щоб показати серверу, що він або вона має право отримати доступ до ресурсу r (тобто файлу). Власник даних надійно прикріплює секретне значення C_r до файлу, а також надсилає всередині нього шифротекст X_r до сервера. Лише авторизовані користувачі можуть розрахувати секретне значення C_r з використанням спільного значення X_r. Таким чином, лише вони можуть знати і відкрити C_r серверу і довести своє право доступу до цього конкретного файлу. Таким чином, якщо один C_r для конкретного файлу скомпрометований, ніякі інші файли не будуть під загрозою. Крім того, ця функція корисна, якщо власник даних хоче змінити деталі щодо авторизованих користувачів для файлу. Наприклад, щоб додати нового авторизованого користувача, власнику даних лише необхідно змінити X_r для цього файлу. Оскільки параметр C_r може залишатися таким же самим, немає необхідності для повторного надсилання нового C_r до серверу. Це, в свою чергу, дозволить знизити ймовірність компрометації цього значення в процесі підтримки динамічного механізму оновлення списку авторизованих користувачів. Проте, при забороні користувачу доступу до файлу, до якого він чи вона вже мають доступ, значення C_r має бути змінене з міркувань безпеки.

Запропоноване рішення поєднує в собі контроль доступу та спільного використання ключа в одному механізмі з використанням китайської теореми о залишках. Хмарний сервер зберігає зашифровані дані з відповідним секретним C_r і секретним значенням X_r, що розраховується власником даних. Провайдер може прочитати секретне значення X_r, але не може дізнатися K_S, який було використано для шифрування даних. Лише авторизовані користувачі можуть виявити K_S від X_r. Провайдер також може зчитувати секретне значення C_r для кожного файлу. Проте, число або ідентичність користувачів, які можуть отримати доступ до файлу, навіть якщо вони вже отримали доступ до файлу, залишаються прихованими від провайдера.

Висновки

У даній роботі виконаний орієнтований на безпеку даних підхід для хмарних обчислень.

Запропоноване рішення контролю доступу дозволило вирішити поставлені задачі, а саме:

– шифрування даних, надання доступу лише для авторизованих користувачів;

– відповідність даних вимогам безпеки;

– прихованість параметрів контролю доступу та безпосередньо даних від провайдерів хмарних серверів та інших користувачів;

– забезпечення необхідної інформації для перевірки цілісності да достовірності даних для авторизованих користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Armbrust, M., Fox, A. & Griffith, R. Above the Clouds: A Berkeley View of Cloud Computing.// Electrical Engineering and Computer Sciences University of California at Berkeley, 2009, pp. 1-8.

2. Chou, D. C., & Chou, A. Y. . Software as a Service (SaaS) as an Outsourcing Mode // An Economic Analysis, 2007, pp. 386-391.

3. Buyya, R., Yeo, C., Venugopal, S., Broberg, J. & Brandic, I. Cloud Computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25, pp. 599- 616.

4. Mukhin, V.&Volokyta, A. Integrated safety mechanisms based on security risks minimization for the

distributed computer systems // I.J. Computer Network and Information Security, 2013, vol. 2, pp. 21-28.

5. I. Iankoulova, M. Daneva: Cloud Computing Security Requirements: a Systematic Review // Sixth International Conference on Research Challenges in Information Science, 2012, 7, pp., 2012.

6. B. Grobauer, T. Walloschek, and E. Stocker, Understanding Cloud Computing Vulnerabilities// Security & Privacy, IEEE, 2011, vol. 9, pp. 50-57.

7. T. Dillon, W. Chen, and E. Chang. Cloud Computing: Issues and Challenges, in Advanced Information Networking and Applications (AINA)// 24th IEEE International Conference, 2010, pp. 27-33.

8. Kaya, Kamer & Aydin Selçuk, Ali. Secret Sharing Extensions based on the Chinese Remainder Theorem // IACR Cryptology ePrint Archive. 2010. 96.

9. C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," Journal of Computer Security, 2011, vol. 19, pp. 367-397.

Додаток Г. Стаття «Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень» у журналі Інфокомунікаційні системи та технології № 2(2)

Infocommunication Systems
and
Technologies

Інфокомунікаційні системи
та
технології

Инфокоммуникационные системы
и
технологии

Науково-практичне видання
Виходить 4 рази на рік
Засноване у січні 2017 року

№ 2(2) / 2018

ЗМІСТ/CONTENTS

Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень

Пирожков Олексій Юрійович
КПІ ім. Ігоря Сікорського
Україна, Київ
a.y.pirozhkov@gmail.com

Савчук Олена Володимирівна
КПІ ім. Ігоря Сікорського
Україна, Київ
Savchuk_l1@ukr.net

Анотація. **В даній статті розглядається підхід до забезпечення безпеки даних в хмарі, що на відміну від традиційних підходів гарантує конфіденційність та безпеку інформації навіть при компрометації провайдера хмарних послуг або його інфраструктури. Представлені алгоритми роботи основних складових елементів даної моделі.**

Ключові слова: інформаційно-орієнтована безпека, безпека даних, безпека даних в хмарному середовищі.

Вступ

Технологія хмарних обчислень стала популярною альтернативою традиційним обчислювальним технологіям. Ця технологія забезпечує нову концепцію плати за використання корисних обчислювальних ресурсів, що базується переважно на технологіях віртуалізації. Основними перевагами хмарних обчислювальних послуг є: самообслуговування, широкий доступ до мережі, об'єднання ресурсів, швидке масштабування. Незважаючи на ці переваги, широке використання цієї нової технології зіштовхуються з низкою перешкод, включаючи безпеку та конфіденційність. На додаток до традиційних ризиків безпеки ризик як у будь-якої обчислювальної системи, підключеної до Інтернету, хмарні системи мають особливі проблеми безпеки і конфіденційності через віртуалізацію і свою багаторівневу природу [1]. Через те, що ресурси хмар поділяються між різними клієнтами, потенційно шкідливі клієнти можуть використовувати вразливі місця віртуалізації для здійснення атак на дані та застосунки інших клієнтів. Крім того, користувачі хмар володіють обмеженим контролем над своїми даними в хмарі, а постачальники хмарних послуг отримують надмірні права по контролю даних клієнта, включаючи отримання права доступу до контенту клієнта, а також контроль над редагуванням прав доступу. Також, у хмарних обчисленнях дані користувача можуть переміщатись між різними постачальниками хмарних послуг. Крім того, постачальник хмарних послуг потенційно може бути зацікавлений в даних клієнта. Через ці проблеми клієнти не охоче передають свої дані в хмари зважаючи на проблеми конфіденційності інформації [10]. Відповідно, потреба в більш надійній методах забезпечення безпеки інформації в хмарному середовищі приводить до дослідження методів, які можуть захистити дані клієнтів у хмарі навіть від самих постачальників хмар.

Інформаційно-орієнтований підхід

У хмарних обчисленнях дані користувачів, в основному, зберігаються у віртуальних сховищах провайдерів хмарної інфраструктури. В публічних SaaS та DaaS моделях користувачі володіють лише даними, які знаходяться на зберіганні. Все обладнання та програмне забезпечення, залучене до зберігання та обробки інформації, знаходиться у власності сервіс провайдерів. В інших моделях, таких як публічні IaaS и PaaS моделі, користувач має доступ до обробки даних та до програмного забезпечення, при цьому доступу до апаратного забезпечення немає. Відповідно, з перспективи користувача хмарного сервісу, найбільш цінним активом в хмарному середовищі є його дані, особливо ті дані, що містять інформацію делікатного характеру і вимагають особливого ставлення, а саме: дані урядового характеру, охорони здоров'я та фінансового спектру. Переваги, що надаються за використання хмарних обчислень, дають користувачам, що раніше розміщували делікатні дані на своїх ПК, більш привабливі перспективи до аутсорсингу своїх даних в хмару [2]. Як і будь-які інші послуги в мережі Інтернет, хмарні сервіси також зазнають атак на системи безпеки. Компрометація доступності хмарних послуг, як правило, призводить до короткострокових ефектів і пошкодження можуть бути відновлені. Компрометація конфіденційності та приватності даних споживачів хмарних послуг, може привести в свою чергу до довгострокових ефектів і будь-які втрати можуть бути достатньо важким для відновлення. Наприклад, коли кілька паролів з адміністративних облікових записів UK's National Healthcare System (NHS) було зламано в червні 2011 року, система NHS була закрита органами охорони здоров'я для захисту записів пацієнтів [5]. Це показує, що для таких випадків конфіденційність даних є більш важливою ніж нормальне функціонування системи як такої. Внутрішні ризики можуть бути від користувачів-зловмисників, які використовують той самий хмарний сервіс, і пом'якшення ризиків, в такому випадку, залежить повністю від хмарного провайдера і виходить з-під контролю власника даних. З точки зору власників даних, хмарне середовище невидиме і власник даних не впевнений в тому як його/її дані захищені від ризиків пов'язаних з безпекою. Наприклад, виходячи з природи файлів, їх може бути переміщено через провайдерів послуг, невідомих для власників даних, в різних країнах, з різною

юрисдикцією щодо конфіденційності даних. Як наслідок з цією проблемою, клієнти хмари відчувають обмежений контроль над своїми даними і їм бракує впевненості щодо безпеки даних. З іншої точки зору, провайдер хмарних послуг має надмірний контроль над даними клієнтів, особливо щодо їх безпеки та приватності. Отже, власники даних стурбовані безпекою та приватністю їх даних і мають бажання зберегти свої дані в безпеці, навіть від провайдерів інфраструктури хмар. Крім того, клієнти надають перевагу власноруч управляти політикою безпеки своїх даних в безпечному режимі, як ніби ці дані зберігалися на їх ПК.

Традиційна концепція безпеки зазвичай зосереджується навколо технологій і пристроїв, що використовуються для зберігання та обробки даних [6]. Ця концепція може бути важко адаптованою для забезпечення необхідного відповідного рівня безпеки, особливо для делікатних та конфіденційних даних. Наукова спільнота нещодавно звернула увагу на питання безпеки та конфіденційності, запропоновані рішення в основному спрямовані на забезпечення безпеки в віртуальних машинах та операційних системах, що ними оперують. Таким чином, більшість рішень як і раніше засновані на традиційній концепції безпеки і в основному фокусуються на системно-орієнтованому або VM (віртуальна машина)-орієнтованому підході [9]. Деякі з цих рішень засновані на концепті надійних обчислень (НО), який було запропоновано і розроблено групою TCG (Trusted Computing Group). TCG прагне розробити набір стандартів і технологій, таких як Trusted Platform Module, які можуть зберігати клієнтські дані і додатки, оброблювані в заявленому довіреному контейнері, який убезпечений навіть від системних адміністраторів хмари [3]. TCG, на основі їх технологій, фокусується на наданні набору інструментів, які можуть бути використані для надання допомоги користувачам, щоб оцінити надійність провайдерів, стежити за дотриманням політики, а також створювати можливість прозорості щодо фізичного розташування даних в хмарі. Проте, користувачі повинні спочатку довіряти TCG технологіям з точки зору оцінки надійності провайдера та якщо TPM, який є основним компонентом щоб побудувати цю систему, буде скомпрометовано, постраждає все рішення. У 2010 році Кріс Тарновський стверджував, що йому вдалося скомпрометувати TPM. Отже, дослідження і запропоновані рішення на основі на TPM, можливо, варто піддати переоцінці. Навіть якщо TPM та інші інструменти TCG є гарантією безпеки, фокус цих інструментів не на тому щоб надати користувачам бажаний контроль за безпекою та приватністю їх даних. Замість цього, з точки зору клієнта, реалізація концепції НО дозволяє здійснювати клієнтам моніторинг або аудит операцій, в тому числі над політикою контролю доступу, для хмари через довірені інструменти, які можуть забезпечити докази відповідності концепції НО для користувачі. Нова концепція була запропонована декількома дослідниками для вирішення конкретних питань безпеки хмарних обчислень, переміщаючи фокус з забезпечення безпеки інформації для клієнтів саме на інформації, і вони називають це інформаційно-орієнтованим підходом.

Принципи роботи інформаційно-орієнтованої моделі

Рішення призначене для досягнення наступного переліку бажаних вимог до застосунків в середовищі хмарних обчислень:

– дані зашифровані та доступні лише для авторизованих користувачів;

– дані доступні для пошуку без загрози для їх конфіденційності;

– дані відповідають вимогам самозахисту та необхідним параметрам безпеки;

– параметри контролю доступу приховані від провайдерів хмарних серверів та інших користувачів;

– провайдер серверу не знає кількість або особистість користувачів, що авторизовані та мають право доступу до даних;

– несанкціоновані об'єкти, в тому числі постачальники послуг, не можуть отримати доступ до даних або отримати інформацію про дані від авторизованих процесів що проводяться на даних;

– дані містять всю необхідну інформацію для перевірки їх цілісності да достовірності для авторизованих користувачів, які мають доступ до даних;

– взаємодія між власниками даних та авторизованими користувачами має бути мінімальною, особливо щодо ключових цілей управління. [4]

Блок контролю та управління доступом

Рис. 1. Шифрування інформації в контенері з даними

У даній роботі, запропоноване рішення використовує китайську теорему о залишках і криптосистему з публічним ключем для безпечного обміну даними захищених користувачів, що зберігаються в середовищі хмарних обчислень серед авторизованих користувачів [7]. Дані, які іноді згадуються як ресурси або файли, захищені симетричним методом шифрування, де як секретний ключ використовується K_s. У даній роботі, ресурсом може бути набір даних або файл, який містить дані будь-якого типу, в тому числі текст, аудіо, зображення або відео. Для поширення секретного ключу для авторизованих користувачів, його зашифровано з використанням методу публічного шифрування відкритого ключа користувача. Шифрування також включає в себе інше значення, C_r, яка буде

використовуватися в якості відповіді на запит сервера, коли користувач робить запит щодо доступу до ресурсу. Лише авторизовані користувачі, які мають відповідний приватний ключ, можуть розшифрувати шифротекст C_r та K_r для отримання C_r, щоб отримати доступ до ресурсу r. Параметри C_r та K_s зчеплені для формування $C_r||K_s$ та розглядаються як єдине значення. Це значення шифрується за допомогою публічного ключа $E_{K_{pub\,i}}$ кожного авторизованого користувача u_i, в результаті шифротекст $\left(E_{K_{pub\,i}}(C_r||K_s)\right)$ для цього користувача u_i, де *i=1, 2, 3, ... , k,* та k кількість авторизованих користувачів, що мають доступ до ресурсу r.

Щоб застосувати китайську теорему о залишках до запропонованого рішення, для користувачів u_1, u_2, ..., u_k, кожен авторизований користувач пов'язаний з унікальним відносним простим числом $n_i = n_1, n_2, ..., n_k$, де k це кількість авторизованих користувачів. Всі n_i, $1 \le i \le k$, є відносні прості числа. Потім, шифротекст $C_r \parallel K_s$ генерується для кожного користувача, тобто $\left(E_{K_{pub\,i}}(C_r||K_s)\right)$, використовується для заміни a_i у рівнянні китайської теореми про залишки для виведення наступної спільної конгруентності:

$$X_r \equiv \left(E_{K_{pub\,1}}(C_r||K_s)\right) \; mod \; n_1,$$

$$X_r \equiv \left(E_{K_{pub\,2}}(C_r||K_s)\right) \; mod \; n_2,$$

$$X_r \equiv \left(E_{K_{pub\,k}}(C_r||K_s)\right) \; mod \; n_k.$$

Вирішення цієї конгруенції X_r, таке, що $0 \le X_r < n = n_1 n_2 ... n_k$, є загальним значенням для ресурсу r і він приєднаний до ресурсу, а ресурс і загальна значення зберігаються разом в хмарному сервері. Коли авторизований користувач u_i надсилає запит щодо доступу до ресурсу r, хмарний сервер надсилає йому спільне значення X_r. Коли користувач отримує спільного значення X_r, він використовує відповідний приватний ключ для розшифровки X_r, щоб отримати значення $C_r||K_s$, як показано нижче:

$$\left(\mathrm{Cr||Ks} = \; D_{K_{prv\,i}}(X_r mod \; n_i)\right),$$

де $D_{K_{prv\,i}}$ це операція дешифрування з використанням приватного ключа користувача u_i, та n_i як відносного простого числа специфічного для даного користувача. Значення C_r потім вирушає назад до серверу користувачем, щоб довести володіння правом доступу до ресурсу. Потім сервер надсилає ресурс для користувача і ключ K_S, який використовується користувачем для щоб розшифрувати файл та отримати його вміст.

Блок шифрування інформації в контейнері даних

З точки зору підвищення безпеки в запропонованому рішенні, власник даних повинен використовувати унікальний симетричний ключ, K_S, для шифрування кожного файлу перед відправкою

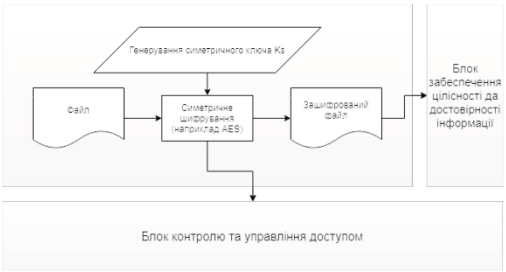

Рис. 2. Шифрування інформації в контенері з даними

його на зберігання в хмару. Відповідно до орієнтованого на безпеку інформації підходу, всі вимоги до безпеки прикріплені до фактичних даних, отже, кожен симетричний ключ прикріплюється до його файлу безпечним методом. Симетричний ключ K_S захищений двома рівнями: спочатку шифруванням його авторизованим користувачем, а потім за допомогою китайської теореми о залишках, щоб знайти загальне значення Xr. Лише авторизовані користувачі можуть обчислити K_S з Xr, яко-му потрібен відповідний n_i та приватний ключ авторизованого користувача. Для ефективного управління ключами, власник даних додає себе в якості користувача при розрахунку Xr для кожного файлу. Отже, ані власник даних, ані користувачі не зобов'язані зберігати K_S, але їм потрібен тільки їх приватний ключ і призначене значення n_i. Таким чином, ключ K_S надійно та ефективно, спільно з авторизованими користувачами, захищено від несанкціонованого доступу, включаючи постачальника послуг хостингу файлу.

Секретний ключ K_S, а також значення C_r є унікальними для кожного файлу. Якщо значення для одного файлу було скомпрометовано, інші файли залишаються в безпеці. Секретне значення C_r використовується авторизованим користувачем, щоб показати серверу, що він або вона має право отримати доступ до ресурсу r (тобто файлу). Власник даних надійно прикріплює секретне значення C_r до файлу, а також надсилає його всередині шифротекст X_r до сервера. Лише авторизовані користувачі можуть розрахувати секретне значення C_r з використанням спільного значення X_r. Таким чином, лише авторизовані користувачі можуть знати і відкрити C_r серверу і довести, що вони мають право доступу до цього конкретного файлу. Секретне значення C_r є унікальним для кожного файлу, навіть якщо він використовується одним користувач для різних файлів. Таким чином, якщо один C_r для конкретного файлу скомпрометований, ніякі інші файли не будуть під загрозою. Крім того, ця функція корисна, якщо власник даних хоче змінити деталі щодо авторизованих користувачів для файлу, наприклад, щоб додати нового авторизованого користувача, власнику даних лише необхідно змінити X_r для цього файлу. Оскільки параметр C_r може залишатися таким же самим, немає необхідності для повторного надсилання нового C_r до серверу. Це, в свою чергу,

дозволить знизити ймовірність компрометації цього значення в процесі підтримки динамічного механізму оновлення списку авторизованих користувачів. Проте, при забороні користувачу доступу до файлу, до якого він/вона вже мають доступ, значення Cr має бути змінене з міркувань безпеки.

БЛОК ПОШУКУ ПО КЛЮЧОВИМ СЛОВАМ В ЗАШИФРОВАНИХ ДАНИХ

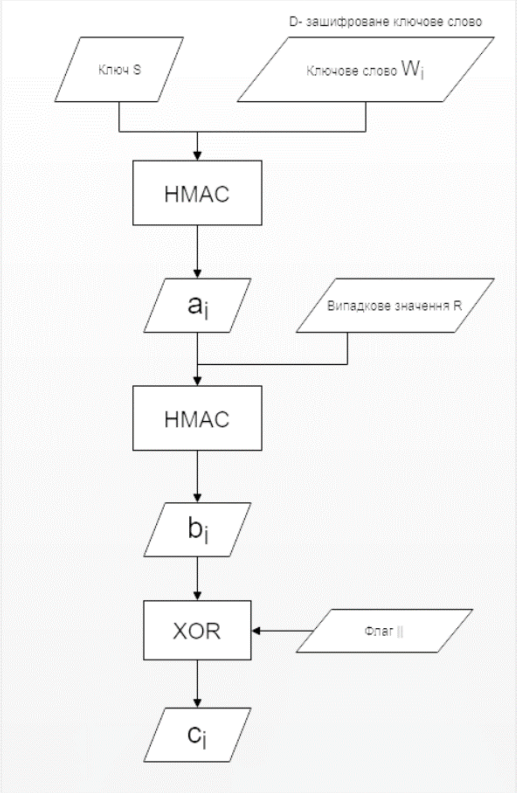

Рис. 3. Блок забезпечення можливості пошуку

Далі буде представлена модель для пошуку зашифрованих даних, яка сумісна з інформаційно-орієнтованим підходом. Власник даних повинен вказати один секретний ключ K_w, який використовується при шифруванні кожного ключового слова. Власник файлу використовує псевдовипадкову функцію для шифрування ключових слів. Нехай $H_K(m)$ буде хеш-код аутентифікації повідомлень (HMAC), який може бути використаний будь-якою криптографічною хеш-функцією, такою як MD5, SHA1, SHA256 або SHA512, з ключем K та вхідним повідомленням m [8]. Припустимо, w_i це і-е ключове слово в списку ключових слів, R це випадкове число, флаг це постійний біт-патерн з фіксованою довжиною l біт біти заповнювачі деякий патерн бітів.

$$a_i = H_{K_w}(w_i),$$

$$b_i = H_{a_i}(R),$$

$$c_i = b_i \oplus (\text{флаг}||\text{заповнювач}).$$

Від кожного ключового слова w_i, власник даних створює c_i та надсилає його з випадковим числом R до сервера як додаток до відповідного файлу. c_i є зашифрованою формою ключового слова w_i; таким чином, сервер не може відкрити w_i від c_i. Наступний пункт описує, як користувачі можуть безпечно здійснювати пошук їхніми зашифрованими файлами за допомогою використання зашифрованих ключових слів.

Для пошуку зашифрованих даних, що зберігаються в хмарному середовищі, коли користувач u_i здійснює пошук ключового слова w_i, користувач має надіслати запит власнику даних для отримання можливості пошуку, що містить ключове слово w_i зашифроване за допомогою публічний ключ власника ($K_{pub\ O}$) і підписане приватний ключ користувача ($K_{prv\ i}$). Власник даних потім відповідає з $T_w = H_{K_w}(w_i)$, зашифрованому за допомогою публічний ключ користувача ($K_{pub\ i}$). Користувач розшифровує повідомлення за допомогою свого приватний ключ та потім зашифровує T_w з публічний ключ сервера ($K_{pub\ S}$), перед відправкою його на сервер в якості пошукового запиту з ID власника, нонс і сеансовий ключ K_t, як показано в (1):

$$E_{K_{pub\,s}}(T_w, Id \text{ власника}, K_t, \text{нонс}). \quad (1)$$

Сервер повинен шукати тільки дані, пов'язані з ID власника, використовуючи T_w для обчислення $P = H_{T_w}(R)$, а потім для кожного c_i сервер перевіряє результат $P \oplus c_i$, щоб побачити, якщо перший біт дорівню значенню флага. Потім ID файлу, пов'язане з цим c_i, додається до списку результатів пошуку разом із власними двома параметрами X_r та C_r.

Після пошуку всіх файлів, сервер надсилає лише те значення X_r ($X_{r1}, X_{r2}, \ldots X_{rn}$), де n - кількість файлів, які відповідають критеріям пошуку, та нонс+1 і всі зашифровані за допомогою сеансовий ключ, K_t, назад клієнту, як показано в (2):

$$E_{K_t}(X_{r1}, X_{r2}, \ldots X_{rn}, \text{нонс} + 1). \quad (2)$$

Користувач отримує секретне значення C_{ri} для кожного загального значення X_{ri}, де $i = 1, 2, \ldots, n$, а потім повертає секретні значення і нонс + 2 на сервер зашифроване за допомогою сеансового ключа K_t, як показано в (3):

$$E_{K_t}(C_{r1}, C_{r2}, \ldots C_{rn}, \text{нонс} + 2). \quad (3)$$

Сервер перевіряє секретні значення $(C_{r1}, C_{r2}, \ldots, C_{rn})$ та надсилає клієнту лише ті файли, секретні значення яких збігається з тими, які були надіслані користувачем. Тепер користувач може розшифрувати файли за допомогою секретного ключа K_s, витягнутого з X_{ri} для кожного файлу. Рис. 3 ілюструє безпечну процедуру пошуку.

СТВОРЕННЯ РЕЗУЛЬТУЮЧОГО ЗАХИЩЕННОГО КОНТЕЙНЕРУ З ДАНИМИ

Концепція передбачає створення орійєнтованого на безпеку інформації файлу (далі ОБІ-файлу) створює захищений контейнер, який включає в себе всі попередні модулі. ОБІ-файл інкапсулює вихідний

файл даних перед його відправкою в хмарний сервер і таким чином зберігає файл даних захищеним від несанкціонованого доступу, в тому числі хмарного серверу. Лише авторизовані користувачі можуть здійснювати пошук і отримати доступ до файлового вмісту, відповідно до доданих параметрів політики безпеки, представлених в Cr та Xr, які встановлюються та управляються, головним чином, власником файлу. Процес створення ОБІ-файлу передбачено проводити в довіреному середовищі, яка належить власнику даних. Перед створенням ОБІ-файлу є чотири основні операції:

1) Шифрування вихідного вмісту файлу за допомогою алгоритму симетричного шифрування. (рис. 2).

2) Створення, за допомогою китайскої теореми про залишки, загального значення Xr (рис. 1).

3) Генерування секретних ключових слів для можливостей пошуку (рис. 3).

4) Створення захисту цілісності та достовірності для зашифрованого вихідного файлу даних. Мається на увазі цифровий підпис.

Лише авторизовані користувачі мають можливість отримати вихідний файл з ОБІ-файлу.

Всі зашифровані ключові слова, де c_i для i=1,2, ..., z, де z - кількість ключових слів, прикріплений до ОБІ-файлу разом з асоційованими випадковими числами Ri.

ОБІ-файл тепер може бути відправлений для зберігання в хмарне середовище. Приватній, цілісність та достовірність вихідного файлу надійно зберігається і не залежать від хмарного серверу для забезпечення вимог безпеки, за винятком деяких операцій.

Висновки

Рішення розроблене на основі модулів, кожен з яких забезпечує набір сервісів, які в основному знаходяться в управлінні власника даних. Перший модуль призначений для шифрування даних перед їх аутсорсингом в хмарну інфраструктуру. Другий модуль створює можливості для більш ефективного та дієвого способу управління політикою контролю доступу, які виражені секретний та спільний ключ, які призначені для спільного використання і доступу до контролю даних. Третій модуль використовується для забезпечення можливості безпечного пошуку для зашифрованих даних через генерацію зашифрованих ключових слів. Результати всіх попередніх модулів використовуються для створення результуючого контейнера. Використання контейнера є результатом пошуку рішення що підвищує безпеку і конфіденційність та відповідає середовищу хмарних обчислень. Використання контейнера, як частини запропонованого рішення, здатне зберегти приватність, цілісність та достовірність даних які було розміщено в хмарі, навіть від самого хмарного провайдеру з мінімальним впливом на функціональність зашифрованих даних, які відповідають можливостям пошуку та поширення для авторизованих користувачів. На останок, важливо згадати, що функції безпеки, запропоновані в контейнері, залежать від криптографічних алгоритмів, які використовуються в даному рішенні і знаходяться під управлінням власника даних.

Література

1. Jose M. Alcaraz Calero, Nigel Edwards, Johannes Kirschnick, Lawrence Wilcock & Mike WrayClouds: Toward a multi-tenancy authorization system for cloud services// IEEE Security and Privacy, 2010, pp. 16-122.

2. Chou, D. C., & Chou, A. Y. . Software as a Service (SaaS) as an Outsourcing Mode // An Economic Analysis, 2007, pp. 386-391.

3. Jun Feng, Yu Chen. Bridging the Missing link of Cloud data storage security in AWS // IEEE conf on CCNC, 2010.

4. I. Iankoulova, M. Daneva: Cloud Computing Security Requirements: a Systematic Review // Sixth International Conference on Research Challenges in Information Science, 2012, 7, pp., 2012.

5. B. Grobauer, T. Walloschek, and E. Stocker, Understanding Cloud Computing Vulnerabilities// Security & Privacy, IEEE, 2011, vol. 9, pp. 50-57.

6. T. Dillon, W. Chen, and E. Chang. Cloud Computing: Issues and Challenges, in Advanced Information Networking and Applications (AINA)// 24th IEEE International Conference, 2010, pp. 27-33.

7. Kaya, Kamer & Aydin Selçuk, Ali. Secret Sharing Extensions based on the Chinese Remainder Theorem // IACR Cryptology ePrint Archive. 2010. 96.

8. Jun-Ho Lee, Min-Woo Park: Multi level Intrusion Detection System and Log management in Cloud Computing // ICACT. 2011. pp. 316-397.

9. Rohan Raj Gupta, Gaurav Mishra & Subham Katara: Data storage security in cloud computing using container clustering // IEEE. 2016

10. Rohit Bhadauria, Sugata Sanyal: Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques // International Journal of Computer Applications. 2016