НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.94:002.1.027.552

«ПОГОДЖЕНО»
Декан факультету
інформаційних технологій
Глазунова О.Г., д.пед.н., професор
_____
«__» _____ 20__р

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»
Завідувач кафедри комп'ютерних систем,мереж та кібербезпеки
Лахно В.А., д.т.н., професор
_____
«__» _____ 20__ р

МАГІСТЕРСЬКА РОБОТА

на тему: «Дослідження методів захисту персональних даних у сервісі SMART-вагів»

Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

Освітньо-професійна програма Комп'ютерні системи і мережі
виробнича, дослідницька

1504 – МР 1578 «С» 20.10.23.001.ПЗ

Керівник магістерської роботи
доц., к.т.н., _____ /Смолій В.В.//
(вчене звання і ступінь) (підпис) (ПІБ)

Виконав _____ /Хвост В.В.//
(підпис) (ПІБ студента)

КИЇВ – 2021

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри

Комп'ютерних систем, мереж та кібербезпеки
(назва кафедри)

доктор.т.н., професор _____ Лахно В.А.
(вчене звання і ступінь) (підпис) (ініціали і прізвище)

«_____» _____ 20 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТА

Хвоста Вадима Володимировича
прізвище,ім'я,по-батькові)

Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

Спеціалізація _____
(виробнича, дослідницька)

Магістерська программа Комп'ютерні системи і мережи

1.Тема магістерської роботи: Дослідження методів захисту персональних даних в сервісі SMART- вагів

затверджена наказом ректора НУБіП від «23» жовтня 2020 р. № 1578 "С"

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

3. Вихідні дані до магістерської роботи:
1. Завдання кафедри на виконання магістерської роботи
2. Наукова література з тематики магістерської роботи

4. Перелік питань, що підлягають дослідженню:
1. Аналіз предметної області
2. Моделювання системи

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської роботи _____ Смолій В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Хвост В.В.
(підпис) (прізвище та ініціали студента)

РЕФЕРАТ

Пояснювальна записка містить 3 розділи , 16 підрозділів, 12 рисунків та 1 таблиць. Її загальний кількісний обсяг – 72 сторінок; кількість позицій списку використаних джерел – 15.

Мета роботи. Мета роботи полягає у дослідженні методів захисту персональних даних в сервісі SMART-вагів.

Об'єкт дослідження. Об'єктом дослідження виступає безпека даних при використанні сервісу SMART- вагів.

Предмет дослідження –Підвищення рівня захисту персональних даних у сервісі SMART-вагів.

У першому розділі було детально розглянуто тему хмарних обчислень і проблем, які виникають при використанні даної технології. На базі опитувань, які проводила Міжнародна корпорація даних (IDC) у вересні 2009 року дана компанія за результатами ранжування даних, дійшла до висновку, що для всіх користувачів хмарних технологій найбільш важлива безпека конфіденційних даних.

У другому розділі розглянуто деталі інформаційно-орієнтованого підходу забезпечення безпеки інформації.Висвітлено класифікацію існуючих рішень і досліджено політики контролю доступу при забезпеченні безпеки і конфіденційності даних.Також було сформовано концептуальну основу підходу сфокусованого на безпеку інформації.

Третій розділ присвячений висвітленню конкретних засад реалізації методу забезпечення безпеки і конфіденційності даних описаного вище. Перший підрозділ розглядає китайську теорему про залишки, яка створює математичну і криптографічну базу для реалізації підходу. У другому підрозділі обговорюються засади реалізації управління доступом і розділення ключа для доступа до зашифрованих даних.Потім йде підрозділ присвячений реалізації пошуку по зашифрованих даних. Четвертий і наступні підрозділи присвячені питанням контролю доступу і питанням конфіденційності.

ABSTRACT

The explanatory note contains 3 sections, 16 subsections, 12 figures and 1 tables. Its total volume is 72 pages; the number of items in the list of sources used - 15.

The purpose of the work. The purpose of the work is to study the methods of personal data protection in the service of SMART-scales.

Object of study. The object of research is data security when using the SMART-scales service.

Subject of research - Improving the level of personal data protection in the service of SMART-scales.

The first section discussed in detail the topic of cloud computing and the problems that arise when using this technology. Based on a survey conducted by the International Data Corporation (IDC) in September 2009, the company concluded that the security of confidential data is the most important for all users of cloud technology.

The second section examines the details of the information-oriented approach to information security. The classification of existing solutions and policies to control access control in ensuring security and confidentiality of data.

The third section is devoted to highlighting the specific principles of the method of ensuring the security and confidentiality of data described above. The first section examines the Chinese residual theorem, which creates a mathematical and cryptographic basis for implementing the approach. The second section discusses the principles of implementation of access control and key sharing for access to encrypted data. Then there is a section on the implementation of the search for encrypted data. The fourth and subsequent sections focus on access control and privacy issues.

ЗМІСТ

ВСТУП

З кожним роком смарт-технології все глибше проникають в суспільство. Компанії виготовляють різні «розумні» пристрої, які повинні поліпшити наше життя і зробити його більш комфортним.

Смарт-технології передбачають використання комп'ютерних систем і мікропроцесорів для виконання щоденних завдань і обміну інформацією. На перший погляд, все складно, але в дійсності - ні. Смарт-технології нас оточують вже давно, але ми не завжди звертаємо на них увагу.

Технічний прогрес змінював різні покоління людей. Більшу частину населення планети зараз можна назвати смарт-суспільством. Ми живемо в світі взаємодії інтернету і різних технічних засобів, створених людьми. В результаті цього отримуємо нові технології, які повинні поліпшити соціальні та економічні сфери життя.

Смарт-суспільство постійно прагне розвиватися. Тому з часом в різних куточках світу почали з'являтися проекти «розумних» міст. В кінці смарт-технології покликані звільнити людей від докучливих пробок, сформувати енергонезалежність і допомогти покращувати життя міста. Звучить утопічно, але це не можна назвати неможливим.

У Нью-Йорку ще з 90-х років розробляються комп'ютерні системи для управління містом. Сьогодні це одне з найбезпечніших міст США, що стало результатом роботи електронної програми, спрямованої на профілактику вуличній злочинності. У Бостоні встановлюють на автомобілі датчики, які інформують комунальні служби про вибоїни на дорогах.

У Сан-Франциско впроваджують електромобілі, в країнах Прибалтики тестують машинних листонош, в Амстердамі роботи прибирають сміття, а в Південній Кореї побудували перше в світі «розумне» місто. Все це - смарт-технології. Вони - всюди, просто десь вони більше помітні, а десь - менше.

З даною темою були проведені виступи на таких конференціях:

- Постерна конференція магістрів 2-го року навчання 2021

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі обговорюються фундаментальні засади технологій хмарних обчислень та проблеми забезпечення конфіденційності даних, які виникають під час роботи SMART-вагів. Розуміння принципів роботи хмарних обчислень і яким чином ця концепція впроваджується у різні моделі надання послуг та розгортання допомагає визначити проблеми безпеки, які постають перед цією новою технологією.

1.1 Що із себе представляють хмарні розрахунки

Хмарні технології з'явилися зовсім недавно: у 2006 році один з найбільших американських інтернет-магазинів Amazon надав свої обчислювальні ресурси, що не використовуються (а на той час їх обсяг став величезним) абсолютно новим чином. Традиційно для оренди ресурсів у дата-центрах необхідно було скласти договір та внести плату за певний термін. Лінійка типорозмірів серверів (обсяг оперативної пам'яті, кількість ядер, розмір дискового простору та ін) досить велика і вибирається заздалегідь, до підписання договору. Можна орендувати багато серверів, зв'язати їх високопродуктивною мережею, підключити балансувальник навантаження та отримати систему, що обробляє велике навантаження. У подібній моделі використання ресурсів є суттєві незручності. При створенні програм часто невідомо, яке буде навантаження на систему, на який термін орендувати сервери програми. Або такий приклад: створюється стартап, орендуються сервери і до закінчення терміну оренди цей стартап "вмирає". Що робити з непотрібними орендованими серверами? Ще складніша справа з купівлею фізичних серверів. Адже їх, крім адміністрування операційної системи та встановлених додатків, необхідно обслуговувати фізично. Сюди входить підбір приміщення, електроживлення, системи охолодження, вентиляції... Усі

ці проблеми можна вирішити за допомогою еластичних обчислювальних ресурсів, що надаються хмарними провайдерами. (Платформа Amazon Web Services називає ці ресурси EC2 - Elastic Cloud Computers [1]) Ключові переваги хмарної моделі такі:

-ресурси надаються на вимогу та таким самим чином звільняються;

-плата нараховується за фактичний час використання ресурсів;

-Надання та звільнення ресурсів здійснюється самим споживачем ресурсів через веб-портал, без будь-якої паперової тяганини з договорами.

Крім віртуальних машин, у хмарних середовищах надаються різні послуги, що дозволяють будувати різні архітектури: послуги віртуальних мереж, підмереж, балансувальники навантаження, списки контролю доступу (Access Control Lists, ACL), виділенні IP-адреси та ін. Ці послуги складають основу інфраструктури як сервісу (Infrastructure as a Service, IaaS). Звісно,пряма вартість річної оренди фізичного сервера може бути меншою, ніж вартість оренди хмарного сервера з погодинною оплатою з такими ж характеристиками, але багато хмарних провайдерів (наприклад, AWS) надають можливість довгострокової оренди віртуальних машин за цінами суттєво меншими, ніж за погодинної оплати. Якщо сервери потрібні на невеликий час і заздалегідь не відомо, якого розміру має бути віртуальна машина, то еластичні віртуальні машини можуть стати єдиним прийнятним вибором. У разі ж прямої купівлі фізичних серверів завдання вибору, придбання, налаштування та обслуговування, а також їх продаж після застосування стає досить непростим. Щоб забезпечити можливість виділення користувачам ресурсів, хмарні провайдери мають великі, географічно рознесені дата-центри, веб-портали для отримання доступу до їх ресурсів, а також API для програмного доступу. Це дозволяє зробити те, що не можна виконати за допомогою будь-якої іншої традиційної технології: код програми

може сам собі виділяти стільки ресурсів, скільки йому потрібно. Або ж програми можуть створювати інфраструктуру, на якій вони виконуватимуться.

Крім «голої» інфраструктури, хмарні провайдери надають найбільш типові програми у вигляді веб-сервісів. Як приклад можна навести хмарне сховище даних (cloud storage), сервіс надання облікових записів (identity provider), сервіс хостингу веб-додатків, базу даних як сервіс, брокер повідомлень, концентратор повідомлень та ін. Всі ці послуги, що здаються на перший погляд розрізненим набором, надаються як загальна платформа. Доступ до них уніфікується як одноманітних API, SDK, можливі їх «з'єднання» між собою, загальний моніторинг логів і подій тощо. Це інший рівень використання ресурсів хмари: платформа як сервіс (Platformas a Service). PaaS дозволяє користувачам створювати не просто програмні продукти в рамках однієї операційної системи, веб-платформи та ін., але цілі інформаційні системи, компонентами яких будуть хмарні екземпляри сервісів. Подібно до IaaS, сервіси PaaS зазвичай допускають масштабування (як ручне, шляхом вибору відповідного розміру, так і автоматичне, за допомогою різних метрик і подій). Як правило, сервіси PaaS надають набагато менші права для доступу до обчислювальних ресурсів інфраструктури, що лежить у їх основі. Наприклад, сервіси хостингу веб-додатків не дозволяють встановити специфічні програми, COM-компоненти, поміняти бібліотеку DLL у GAC, змінити запис у реєстрі та ін, оскільки відсутній root-доступ. Але натомість вони надають зручні портали адміністрування, інтеграцію. з іншими сервісами, вбудовані засоби логування та моніторингу, доступність 99,99% часу та ін.

В даний час найбільшими хмарними провайдерами є Amazon Web Services (AWS), Microsoft Azure, Google Cloud, IBM Bluemix, Oracle.

1.2 Історія розвитку хмарних технологій

Початком розвитку хмарних технологій в IT-індустрії прийнято вважати 50-ті роки минулого століття. Тоді, внаслідок дорожнечі комп'ютерів, вчені вигадали використовувати одну обчислювальну машину, доступ до якої мали відразу кілька співробітників тієї, чи іншої компанії. Ідея про підключення низки користувачів до загального процесора з'явилася в 1954 році, реалізація почалася в 1959, а перше комерційно успішне рішення випустили в 1964. Тоді ж сформувалося ставлення до обчислювальних потужностей як ресурсу, відкрилися спеціальні комп'ютерні бюро: в них клієнти могли купувати необхідний обсяг потужності до виконання розрахунків.

Аж до 80-х років подібна модель надання потужностей була життєздатною, але потім поступово пішла в минуле, оскільки на ринку з'явилися відносно недорогі персональні комп'ютери.

У 1966 році виник також проект ARPANET(Рисунок 1.1), за створення якого відповідав вчений Джозеф Ліклайдер.

Рисунок 1.1 Карта мережі ARPANET

Його ідея полягала в тому, щоб усі люди з різних кінців Землі були взаємопов'язані та могли отримувати доступ до програм та даних із будь-якої точки світу.Він заклав основу для грід-обчислень, ранього попередника хмари, в яких географічно розподілені комп'ютери були об'єднані для створення слабозв'язаної мережі. Саме ядро цього проекту на початку 1990-х еволюціонувало до сучасного інтернету. Іншим важливим фактором, що вплинув появу «хмар», став розвиток систем віртуалізації. Йдеться про цифрові системи, які не залежать від конкретного обладнання та дозволяють починати та закінчувати роботу в будь-який момент. Комерційний варіант подібної технології випустила компанія IBM у 1972 році.

1.2.1 Як з'явилися SaaS та PaaS

Так чи інакше, активніший розвиток концепції хмарних технологій почався вже в 90-ті роки, зі значним збільшенням пропускної спроможності мережі Інтернет. Одним із піонерів майбутньої хмарної революції стала компанія Salesforce.com. її фахівці розробили концепцію доставки корпоративних програм через простий веб-сайт. Ця компанія спочатку була створена з розрахунком на надання CRM-систем клієнтам як послуги з передплати (SaaS).

Як переваги нової моделі представники Salesforce.com називали: можливість аутсорсингу інформаційних технологій, захист від збоїв та оперативну технічну підтримку (оскільки всі апаратні ресурси розташовуються в зоні фізичної доступності постачальника), зниження сукупної вартості володіння інформаційними технологіями. Тобто багато в чому компанія виступила провісником відомих сьогодні хмарних моделей надання послуг. Salesforce.com досі є одним із великих гравців на ринку SaaS поряд з Microsoft, Oracle та SAP.Поступово SaaS став популярною послугою –

за допомогою цієї абревіатури онлайн-сервіси розрізняли серед десктопних програм, що вимагають встановлення на комп'ютер.

Іншим важливим етапом у розвитку SaaS можна назвати 2009 рік, коли Google та інші великі розробники стали пропонувати програми на базі браузера (йдеться про Google Apps). Приблизно в цей час Microsoft закріпила себе на ринку бізнес-додатків, вклавши сили в розвиток хмарної версії Office 365.

Стрімкий розвиток інтернету призвів до регулярного зростання кількості розробників програмного забезпечення. Процес розміщення нових програм потрібно було спростити. Так на ринку, крім SaaS, з'явилася послуга PaaS (Platform as a Service — «платформа як послуга»). Першим подібним сервісом у 2006 році став Zimki. У 2008 році Google представила App Engine, який пізніше став хмарною платформою Google.

1.2.2 Як з'явився IaaS

Ще на початку 2000-х років корпорації мали у своєму розпорядженні значні обсяги обчислювальних ресурсів, частина яких практично не була задіяна і виступала резервом на випадки пікових навантажень (наприклад, «чорна п'ятниця» у випадку з онлайн-продавцями). Бізнес почав передавати свої потужності третім особам – так з'явилася модель IaaS (Infrastructure as a Service – «інфраструктура як послуга»).

Першим IaaS-сервісом (або хмарою у сучасному розумінні цього терміну) став Amazon Web Services, запущений у 2002 році. Компанія надала пакет хмарних інфраструктурних послуг, включаючи зберігання, обчислення та навіть можливості людського інтелекту через Amazon Mechanical Turk. У 2006 році Amazon запустив Elastic Compute Cloud (EC2) - комерційний веб-сервіс, який дозволяв невеликим компаніям та окремим особам орендувати частину IT-інфраструктури, на якій можна запускати будь-які програми.

Сьогодні Amazon залишається лідером у сфері хмарних послуг за моделлю IaaS. Згідно з аналітикою Gartner за 2017 рік, багато підприємств зараз витрачають понад п'ять мільйонів доларів на рік зі свого IT-бюджету на хмарні сервіси Amazon.

Поступово до Amazon на ринку IaaS приєдналися Microsoft (сервіс Azure, 2010) і Google (Google Compute Engine, 2012). Таким чином, на довгі роки сформувалася трійка лідерів у сфері надання сервісів за моделлю «інфраструктура як послуга».

Інші постачальники послуг, спостерігаючи діяльність гігантів ринку, намагалися відповідним чином змінити свої продукти та скоригувати бізнес-стратегії. Слідувати «великій трійці» намагалися, наприклад, HPE, Dell та VMware. Частина подібних компаній згодом відмовилася від спроб надання публічної хмари, не витримавши конкурентного тиску. У той же час VMware і Rackspace обрали дещо інший маршрут, позиціонуючи себе як організації, які можуть допомогти підприємствам керувати програмами та робочими навантаженнями у публічних хмарах своїх конкурентів. Так, діяльність Rackspace змістилася у сферу допомоги підприємствам у керуванні своїми хмарними розгортаннями на платформах Amazon, Microsoft, Google та інших. VMware, залишаючись одним з лідерів на ринку програмного забезпечення для віртуалізації, продала свій публічний хмарний бізнес французькому IaaS-конкуренту, OVH, у квітні 2017 року. В даний час VMware позиціонує себе як постачальник гібридних хмарних обчислень.

Що стосується загального стану справ на ринку, розвиток апаратного забезпечення (а саме створення багатоядерних процесорів та збільшення ємності накопичувачів інформації) та технологій віртуалізації (зокрема програмного забезпечення для створення віртуальної інфраструктури, наприклад, Xen-віртуалізація) сприяло не тільки розвитку, а й більшій доступності хмарних технологій. Сьогодні хмарні обчислення це те, чим майже кожен користується щодня. За даними Citrix та IDC більше 90% компаній у всьому світі орієнтовані на використання хмарних технологій.

1.3 Способи створення ресурсів у хмарі

Перш ніж розпочати описувати способи створення ресурсів, пояснимо, що це таке. Як зазначалося вище, хмарні провайдери мають в основі своїх сервісів величезні дата-центри, чиї обчислювальні ресурси з допомогою системи віртуалізації поділяються на невеликі частини: голі віртуальні машини різних розмірів із встановленою операційною системою (IaaS) та групи віртуальних машин із встановленим софтом, що надає доступ лише до своїх можливостей (PaaS). Так от, створити хмарний ресурс — значить надіслати запит контролеру ресурсів, розміщеному в хмарному ЦОДі, на виділення необхідних обчислювальних ресурсів з доступних пулу. Тобто, по суті, ресурс не створюється з нічого, а лише виділяється на вимогу. І тут можлива ситуація (рідко, але буває), коли користувач запитав ресурси у контролера, а вони з'явилися. Це трапляється через те, що фізичні ресурси, де розміщуються віртуальні, вже задіяні іншими користувачами. Завдання оптимального розподілу доступних ресурсів між користувачами повністю вирішує контролер. І якщо користувач під час створення ресурсів зіткнувся з проблемою, він повинен повторити спробу, вдавшись до різних варіацій (повторити через деякий час, змінити регіон і повторити, змінити обліковий запис і повторити тощо).

З точки зору користувача, виділення ресурсів виглядає як створення ресурсів: він виконав низку дій на веб-порталі, і в останньому з'явилися ресурси. Насправді, звичайно, вони були виділені, і про це не варто забувати. Однак для простоти та наочності застосовуватимемо термін «створення».

Існує чотири способи керування хмарною інфраструктурою (Рисунок 1.1).Перший, найпростіший і очевидніший — задіяти веб-портал. При цьому користувач повинен мати відповідні права на створення ресурсів. Ручний спосіб дуже простий: у всіх хмарних провайдерів є зручні портали, велика документація, відеоінструкції та ін. Не потрібні додаткові сервіси, SDK та ін.

Рисунок 1.2 Способи керування хмарною інфраструктурою

Однак цей спосіб має недоліки:

- тривалий час створення інфраструктури;
- недостатня надійність (у разі проблем з ресурсами їх доведеться перетворювати вручну, з усіма ручними налаштуваннями, конфігуруванням тощо);
- складність перенесення інфраструктури в новий регіон або обліковий запис - її знадобиться вручну клонувати або копіювати (даний недолік частково згладжується тим, що хмарні провайдери дозволяють копіювати або клонувати ресурси, але ця процедура все одно потребує ручного ініціювання);
- процес створення ресурсів у цьому випадку неможливо автоматизувати.

Другий спосіб – застосувати програмні бібліотеки (Software Development Kit, SDK), що забезпечують доступ до ресурсів хмари з коду

користувача програм. Як правило, SDK є набором класів і методів, що полегшують програмні операції з ресурсами хмари. Щоб забезпечити доступ до таких ресурсів, програма з хмарним SDK повинна містити ключі облікової запису, який матиме доступ до хмари. У хмарі ці ключі зареєстровані у вигляді користувача в активному каталозі хмарного облікового запису, що володіє правами виконувати програмне маніпулювання ресурсами хмари (такий користувач називається принципалом - service principal). І управління цими обліковими записами відбувається так само, як і обліковими записами користувачів хмарного веб-порталу. Серед переваг такого підходу — можливість створення програм, які самі собі створюють хмарні ресурси, а також автоматизованого керування хмарним акаунтом.

До третього способу створення хмарних ресурсів відносять спеціалізовані розширення для мов командного рядка - shell, CMD та ін. (наприклад Azure PowerShell, AWS CLI та ін.), що працюють у ній безпосередньо. Для підключення цих розширень до хмарних ресурсів необхідно імпортувати ключі або виконати вхід до облікового запису через форму введення логіна/паролю. Як і у випадку SDK для сценарних мов програмування, SDK для командної оболонки дозволяє описувати хмарну інфраструктуру у вигляді набору команд, кожна з яких створює чи конфігурує відповідний хмарний сервіс.

І SDK, і команди оболонки оперують зрештою з API хмарного провайдера (зазвичай REST API), доступ до яких також дозволить маніпулювати ресурсами хмари.

Четвертий спосіб створення хмарних ресурсів – застосувати шаблони. В цьому у випадку всі необхідні ресурси та зв'язки між ними описуються за допомогою текстового файлу у форматі YAML чи JSON. Такий шаблон може бути завантажений у відповідний хмарний сервіс безпосередньо через веб-портал чи через CLI-команди.

Опис інфраструктури через шаблон - дуже потужний механізм, широко застосовується конфігурування різних інфраструктур (наприклад, у системах

Ansible, Chef, Puppet та інших.). Як зазначалося, шаблони представляють собою текстові файли, які можуть зберігатися в репозиторії шаблонів або найчастіше у репозиторії GitHub. Для хмари AWS сервіс створення ресурсів за допомогою шаблонів називається CloudFormation (підтримує YAML та JSON), у Azure це ARM Template (нині підтримує тільки JSON). На веб-порталі AWS є спеціальний редактор, що спрощує створення та конфігурування шаблону. Останній може бути завантажений у файлове сховище S3, репозиторій CodeCommit або будь-яке інше місце, доступне для сервісу CloudFormation. Цей сервіс створює стек - набір ресурсів, керованих спільно (створення, видалення та оновлення).

Сервіс CloudFormation дуже зручний у застосуванні зі сторонніми сервісами конфігурування – наприклад, Ansible. Це широко використовуваний додаток, YAML для створення конфігураційних шаблонів, які служать для адміністрування групи серверів (переважно Linux, але є розширення і для Windows), не вимагаючи інсталяції на цих серверах «агентів». Для роботи Ansible потрібні тільки ключі доступу до ресурсів (SSH-ключі для Linux-хостів, сертифікат для PowerShell-доступу до Windows-хостів або ключі доступ до AWS). Шаблон CloudFormation для Ansible представлений у вигляді JINJA, що допускає передачу параметрів через змінні Ansible.

1.4 Безпека хмарних ресурсів

Поряд із незаперечними перевагами, зберігання та обробка даних у хмарних середовищах потенційно може доставити низку проблем, яких немає (або,вірніше, вони проявляються не так чітко) у разі розміщення та обробки даних у власних дата-центрах. Це зумовлено низкою причин. По перше, хмарні середовища самі собою публічно доступні і всі сервіси, якщо явно не налаштовано інше, доступні для всіх в Інтернеті. По-друге, відповідальність за захист даних та інфраструктури від ненавмисних дій користувачів лежить поза компетенцією хмарного провайдера. Крім того, хмарні інфраструктури, що

працюють з великими даними, часто містять у своєму складі великі кластери віртуальних машин, що вимагає застосування спеціальних заходів для забезпечення надійної роботи всієї системи. Крім цього, інформація фізично передаватиметься по незахищених каналах і є загроза її перехоплення. Розглянемо докладніше всі перелічені та деякі інші аспекти безпеки хмарного середовища.

Найбільш поширений спосіб захисту кінцевих точок хмарних сервісів – обмеження доступу до них за допомогою механізмів автентифікації та створення списків дозволених IP-адрес, з яких можна отримати доступ до точок. Розглянемо насамперед різні способи забезпечення доступу із заданого адресного простору.

Сервіси, що стосуються IaaS, а також у ряді випадків до PaaS, вимагають для свого створення конфігурованої хмарної віртуальної приватної мережі (VNet, VPC), розбитої на підмережі. Доступ до кінцевих точок сервісів, розташованих у цих підмережах, можна регулювати за допомогою конфігурування мережевих груп безпеки (Network Security Group, NSG) (рис. 1.2), які представляють собою списки контролю доступу, ACL.

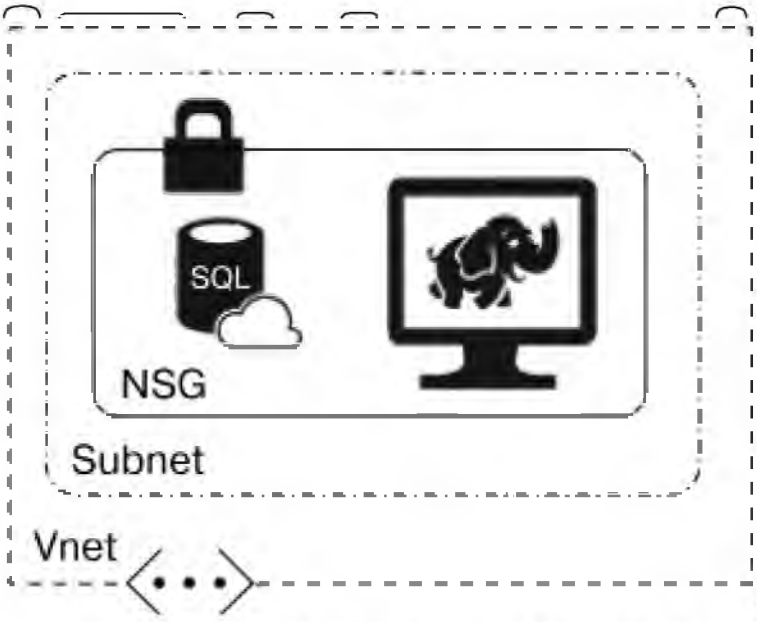

Рисунок 1.3 Обмеження доступу до кінцевих точок хмарних сервісів за допомогою мережевих груп безпеки

Отже, віртуальна частина мережі – один із базових сервісів IaaS. Він представляє являє собою хмарний аналог локальної мережі та служить для надання діапазону IP-адрес для розміщення в них ресурсів. Віртуальну приватну мережу можна розділити на підмережі (subnet), а між ними встановити правила маршрутизації IP-пакети. Крім того, на підмережі можна встановити списки контролю доступу, які називаються мережними групами безпеки. Це дозволяє логічно розділяти архітектури інформаційних систем на різні рівні (наприклад, рівень даних, бізнес-логіки, фронтенд) шляхом розміщення кожного рівня у своїй підмережі та встановлення правил маршрутизації.

NSG є список доступу, що містить набір записів. Кожна запис складається з таких елементів, як:

- назва;
- цифра, що визначає пріоритет перегляду списку записів;
- діапазон IP-адрес (для одиієї конкретної адреси це /32);
- номер порту;
- дія - ALLOW або DENY («Дозволити» або «Відхилити») по відношенню до запиту, що надійшов із цієї адреси.

Крім того, вказується протокол, до якого застосовується дія ALLOW або DENY (TCP, UDP, ICMP та ін.). Безпека кінцевих точок у цьому випадку забезпечується обмеженням до них доступу ззовні. Крім NSG, ряд хмарних сервісів, які не вимагають віртуальної приватної мережі (наприклад, Azure SQL), мають фаєрволи – списки «дозволених» та «заборонених» діапазонів. Хорошою практикою є повсюдне використання NSG та фаєрволів. При цьому необхідно, щоб усі порти, що стосуються віддаленого доступу/керування (наприклад, 22 для SSH, 3388 для RDP) або безпосередньо до сервісу (скажімо, 1433 для MS SQL), були недоступні з Інтернету поза діапазоном адрес віртуальної приватної мережі. Для отримання ж доступу до сервісів із «дозволеної» локальної мережі або з дозволеного комп'ютера слід встановити VPN-шлюз із локальної мережі або з комп'ютера до віртуальної приватної

мережі або безпосередньо до екземпляра сервісу. Крім шлюзу, при з'єднанні локальної мережі з віртуальною приватною мережею необхідно застосувати проміжний хост, проксі-хост (Рисунок 1.3), який транслюватиме запити та дозволятиме приватні адреси хмарних ресурсів з локальної мережі. Проксі-хост у різних реалізаціях можна розмістити як у хмарній мережі, так і в локальній. В останній може розташовуватися контролер домену, сервер БД з даними, які не можуть бути розміщені в хмарі, а також інші сервери, які можуть бути розміщені лише у локальній мережі.

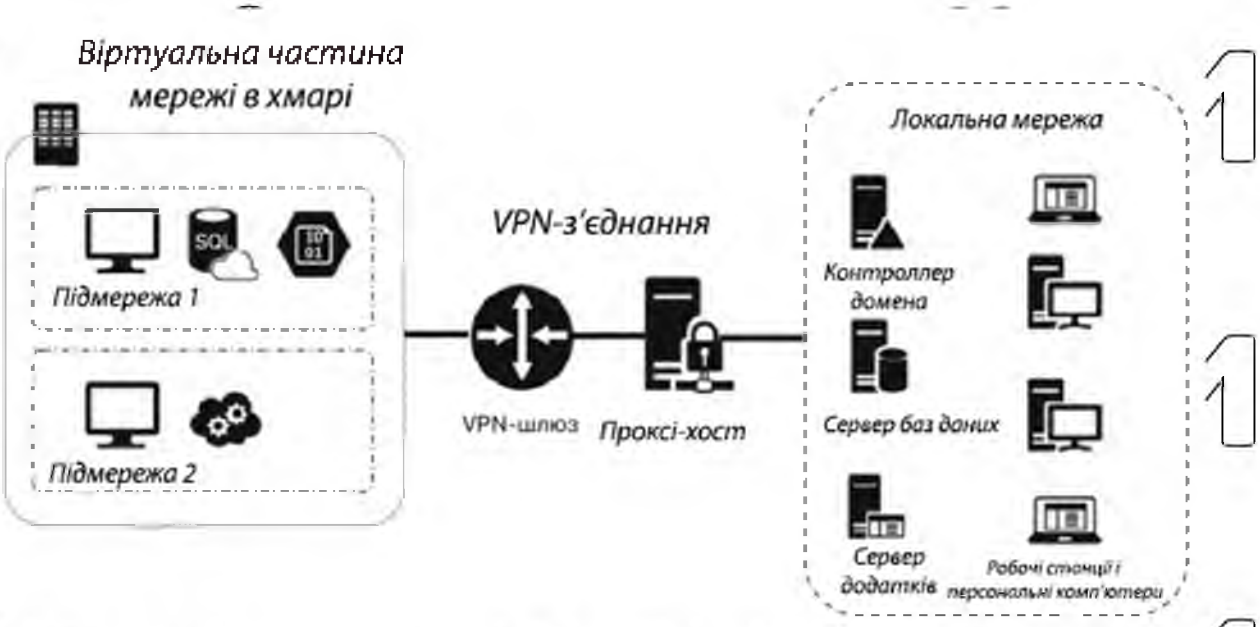

Рисунок 1.4 Забезпечення захищеного доступу до хмарних ресурсів

Деякі кінцеві точки (скажімо, API управління самого хмарного облікового запису) не обладнані ні фаєрволом, ні NSG. Для їх захисту використовується як шифрування трафіку (HTTPS), і спеціальні ключі доступу до ресурсів. Ключі можуть генеруватися безпосередньо сервісом, що захищається, і застосовуватися у заголовках REST-запитів. Наприклад, сервіс сховища Azure Storage використовує аутентифікацію на основі HMAC — Hash-based Message Authentication Code, який повинен містити підписаний алгоритмом SHA-256 токен як заголовок аутентифікації. Наступний

стандартний механізм автентифікації запитів — застосування протоколу OAuth 2.0, при якому облікові дані користувача зберігаються в сервісі зберігання облікових даних (у разі Azure це Azure Active Directory, а AWS - Cognito). Загалом хмарний сервіс зберігання облікових даних включає в себе користувачів, ролі та API-ресурси, що захищаються цим протоколом. Будь-який запит до REST API ресурсів, зареєстрованих у цьому сервісі (а це, по суті, все REST API кінцевих точок управління хмарними ресурсами), повинен у своєму заголовку містити токен, який можна отримати, якщо виконати процедуру введення облікових даних до каталогу.

Ще один «ешелон» захисту даних у хмарних ресурсах – це їхнє шифрування. Поширений підхід у разі — про прозоре шифрування даних (transparent data encryption, TDE). Суть його полягає в тому, що все шифрування та дешифрування даних відбувається «за лаштунками», без участі користувача, за допомогою ключів шифрування, які генеруються та зберігаються самим хмарним акаунтом. У разі Azure ці ключі зберігаються в Azure KeyVault, а у випадку AWS - в AWS KMS.

Наступна ланка захисту - послуги, що відповідають за моніторинг запитів, що надходять до ресурсів та здійснюють аудит користувальницької активності. Наприклад, Azure Security Center, який може виконувати моніторинг дуже багатьох ресурсів, видає рекомендації щодо їх захисту та стежить за вхідним трафіком. На практиці у реальній робочій системі такий сервіс дозволяє виявляти і успішно відбивати великі хакерські атаки на систему з використанням розподіленої мережі заражених серверів (ботнет).

І остання лінія захисту даних, вбудована у хмарні ресурси, включає у собі сервіси аналізу активності користувача, наприклад Audit & Threat Detection для Azure SQL. Цей сервіс забезпечує внутрішній аудит всіх запитів у базі даних Azure SQL та аналіз їх на предмет підозрілих дій. Цей вид захисту має один недолік: в результаті використання в реальній системі при включенні максимального рівня аудиту та захисту суттєво знижується чутливість та загальна продуктивність.

1.5 Хмарні обчислення: труднощі та проблеми

Під час використання сервісів хмарних обчислень всіх користувачів тривожить питання безпечності пересилання конфіденційних даних з персональних пристроїв в «хмару», яку використовує дуже багато інших людей. Міжнародна корпорація даних (IDC) провела опитування у вересні 2009 року про проблеми хмарних обчислень, які тривожать користувачів(Рисунок 1.5) [2]. Згідно з результатами ранжування найбільш важливою є безпека конфіденційних даних.

Рисунок 1.5 Ранжування проблем хмарних обчислень

Де:

- 1 – Мала гнучкість налаштування під себе;
- 2 – Складність інтеграції;
- 3 – Складність відмови від послуг «хмари»;
- 4 – Відсутність єдиних стандартів сумісності;
- 5 – Більш дорогі тарифні плани;

- 6 – Потужність хмарних обчислень;
- 7 – Висока доступність;
- 8 – Безпека даних.

Проблеми безпеки в хмарних обчисленнях формуються на основі недоліків або помилок допущених в реалізації основних технологічних компонентів, на які покладаються хмарні обчислення. Цими компонентами є:

– Веб-додатки та служби - найбільш часто використовувані технології 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 недоліки вищеописаних трьох основних технологічних компонентів можна розглядати як поля для атаки на хмарні обчислювальні системи. 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, всі очікувані проблеми, пов’язані з Інтернетом, також пов’язані з хмарними обчисленнями. Вразливості в ОС і інших програмах, реалізованих програмно і встановлених в хмарну інфраструктуру, також можна розглядати як такі, що пов’язані з вразливостями хмарних обчислень.

1.6 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.

1.6.1 Захист конфіденційності і цілісності даних від провайдерів хмарних обчислень

Конфіденційність і цілісність даних є важливими вимогами для різних мережевих додатків. Клієнти хмарних обчислень не тільки турбуються про компрометацію конфіденційності і цілісності своїх даних від можливих зловмисників, а й від потенційно цікавих до цього провайдерів хмарних обчислень. На жаль, порушення безпеки, підраховані в 2011 році і перераховані в [3], 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. 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.

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.

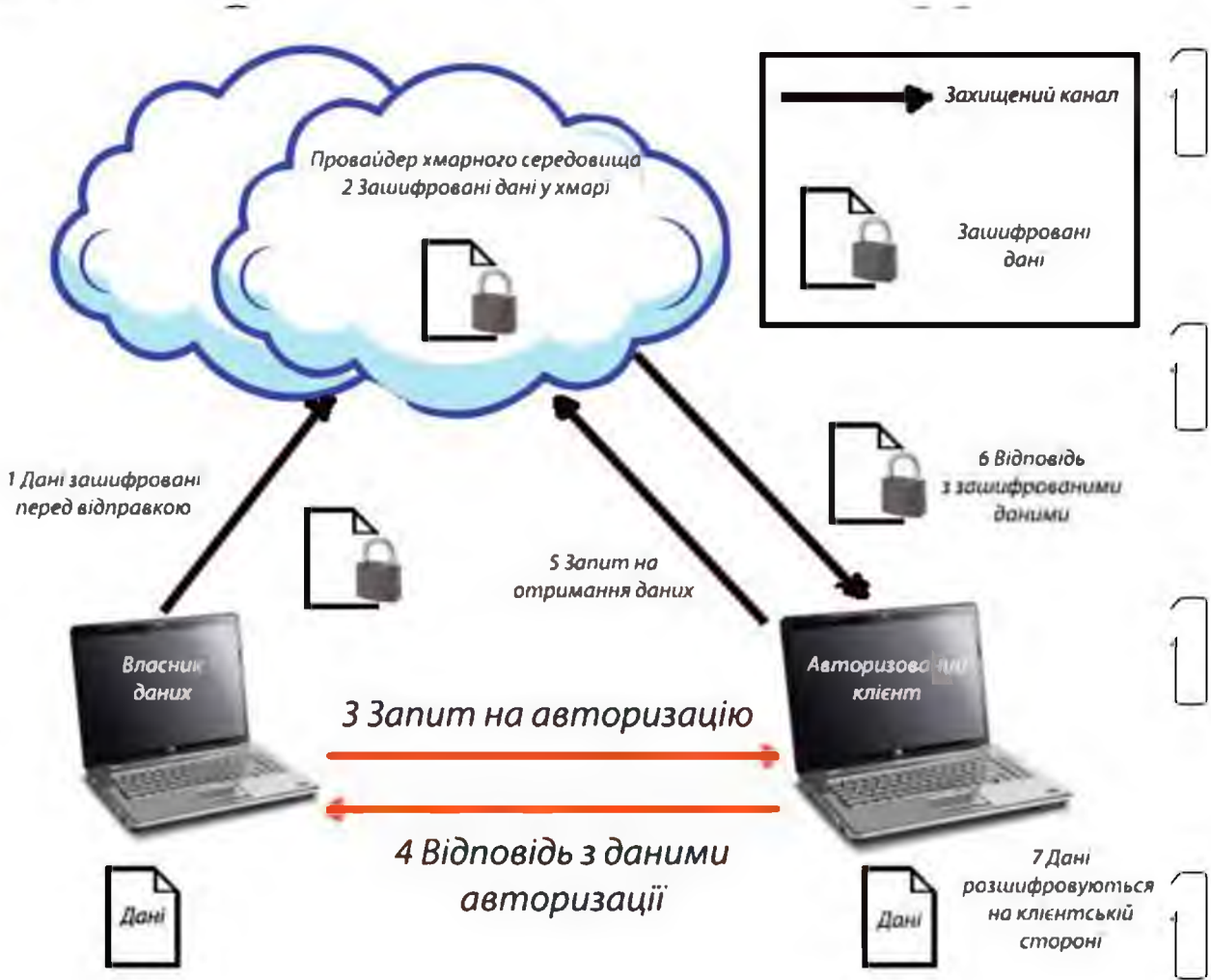

Рисунок 1.6 – Базова архітектура для збереження конфіденційності даних в хмарі

На рисунку 1.8 показано базову архітектуру шифрування даних захисту конфіденційності, при відправці в хмару. Потім дані залишаються зашифрованими у хмарі, і лише користувачі, авторизовані власником даних,

можуть отримати облікові дані для доступу до зашифрованих даних. Зашифровані дані можуть бути розшифровані лише після завантаження на авторизований комп'ютер користувача. У такому разі конфіденційність даних не залежить від неявного припущення про довіру сервера або індивідуальних угод про рівень послуг (SLA). Натомість захист конфіденційності залежить від методів шифрування, які використовуються для захисту даних. Інші проблеми полягають у тому, як дозволити власнику даних та авторизованим користувачам ділитися та шукати зашифровані дані та використовувати їх для деяких обчислень відповідно до їхніх прав доступу. Всі ці функції повинні виконуватися безпечно, без розкриття приватної інформації неавторизованим суб'єктам, включаючи хмарні провайдери. Нові криптографічні методи, схеми довірених обчислень та підходи, орієнтовані на безпеку інформації, можуть стати перспективним рішенням для подолання кількох проблем безпеки хмарних обчислень.

1.6.2 Криптографічні технології пристосовані до хмарної моделі

Нові технології криптографії необхідні для відповідності потребам хмарної моделі та підвищення безпеки 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. Тільки авторизовані користувачі можуть запитувати та отримувати зашифровані дані, не відкриваючи жодної приватної інформації ні про дані, ні про запит, який може містити інформацію про дані. Іншим прикладом є методи шифрування, що дозволяють проводити обчислення зашифрованих даних без їх дешифрування. У таких методах також захищаються результати обчислення, які зазвичай містять інформацію про захищені дані. Прикладом цих методів є гомоморфне шифрування. Є й інші методи криптографії, такі як шифрування на основі ідентифікаційної інформації (IBE) та шифрування на основі атрибутів (ABE), які можуть

покращити керування ключами та контроль доступу до хмарних систем. Ці нові криптографічні рішення можуть бути найбільш підходящими інструментами для вирішення кількох проблем безпеки та надання хмарним клієнтам більш ефективного контролю над своїми даними у хмарі.

1.6.3 Довірені обчислення

IT-спільнота, зокрема Trusted Computing Group (TCG), намагається створити набір технологій, таких як автентифікація, шифрування даних, керування ідентифікацією та доступом, керування паролями, керування доступом до мережі та аварійним відновленням, щоб гарантувати, що комп'ютерні системи виконуватимуть бажаний тип операцій. TCG застосовує схему довірених обчислень до Trusted Multi Tenant Infrastructure (TMI), щоб встановити довіреність ненадійного середовища, наприклад, послугу публічних хмарних обчислень. Нова концепція спрямована на те, щоб дозволити клієнтам оцінювати довіреність хмарних провайдерів, застосовуючи набір апаратних та програмних технологій. Атестація віддаленого сервера – одна з цих технологій, що 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 та RSA. Ці криптографічні функції необхідні для встановлення надійності в обчислювальному устаткуванні. Іншими словами, TCG спрямована на надання стандартних апаратних та програмних технологій для надійних обчислень, включаючи хмарні обчислення. Тому деякі рішення безпеки, зокрема ті, що пропонуються для забезпечення віртуальної ізоляції між віртуальними машинами та віртуальною машиною від провайдера сервера, ґрунтуються на технологіях TCG. Ці технології, як і раніше, стикаються з кількома проблемами і не можуть використовуватися виключно для забезпечення універсального вирішення всіх проблем із хмарною безпекою. Наприклад, якщо TPM, який є основним

компонентом TCG, скомпрометовано, всі рішення, щодо безпеки даних, будуть порушені. Крім того, концепція TCG не надає користувачам хмарних обчислень достатнього контролю над своїми даними з погляду конфіденційності та політики безпеки.

1.6.4 Підхід орієнтований на безпеку інформації

У традиційних методах захисту даних безпека забезпечується сервером, який зберігає інформацію. Методи, що використовуються для захисту даних, а також керування захищеними даними, контролюються адміністраторами сервера. Такий підхід можна класифікувати як системно-орієнтований підхід, який не підходить для захисту даних клієнтів у меш надійному хмарному середовищі. Очікується, що підхід, орієнтований на інформацію, буде більш ефективним та адаптивним для хмарних послуг. Термін орієнтований на безпеку інформації загалом свідчить про те, що захист сфокусована навколо даних. Цю термінологію можна використовувати по-різному. Для хмарних обчислень підхід орієнтований на безпеку інформації полягає у захисті даних зсередини, таким чином дані, відповідно до їх значення та 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.

1.7 Висновок

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

використовують ресурси та послуги в міру потреби та платять лише за те, що вони споживають. Хмарні послуги можуть постачатися в різних моделях, що базуються на типі послуги, що надається. Основними трьома хмарними послугами:

- Програмне забезпечення як послуга (SaaS);
- Платформа як послуга (PaaS);
- Інфраструктура як послуга (IaaS).

Загалом, існує п'ять відомих моделей розгортання послуг хмарних обчислень, а саме: приватна хмара, громадська хмара, публічна хмара, гібридна хмара та віртуальна приватна хмара. В даний час модель публічних хмар є популярною комерційною хмарною моделлю. З одного боку, ця розробка дає величезні переваги, такі як економічна ефективність, економія часу та масштабованість. З іншого боку, ця технологія стикається з низкою труднощів і проблем, коли проблеми конфіденційності та безпеки можуть вважатися найбільш складними. Таким чином, дослідні тенденції полягають у захисті конфіденційності даних та цілісності у хмарі навіть від самих хмарних провайдерів та у наданні клієнтам хмарних функцій можливості більш суворо контролювати свою політику безпеки даних у хмарі. Пропоновані рішення для підвищення безпеки даних у хмарі мають різні напрями: деякі зосереджені на використовуваних інструментах, в першу чергу криптографічних алгоритмах, для підвищення безпеки даних у хмарі, інші зосереджені на більш комплексних рішеннях, що об'єднують різні методи безпеки, засновані головним чином на двох підходах: Надійні обчислення (НО) та Підхід орієнтований на безпеку інформації (ОБІ).

2 ПІДХІД СФОКУСОВАНИЙ НА БЕЗПЕЦІ ІНФОРМАЦІЇ ПРИ ХМАРНИХ ОБЧИСЛЕННЯХ

У цьому розділі детальніше обговорюється підхід, сфокусований на безпеку інформації. Цей підхід є основним, який розглядається в цій роботі для підвищення безпеки та конфіденційності хмарних обчислень у сервісі SMART- вагів.

2.1 Розгляд підходу сфокусованого на безпеку інформації

У хмарних обчисленнях дані користувачів переважно зберігаються у віртуальних сховищах провайдерів хмарної інфраструктури. У публічних SaaS і DaaS моделях користувачі мають лише дані, що зберігаються. Все обладнання та програмне забезпечення, залучене до зберігання та оброблення інформації, знаходиться у власності сервісу провайдерів. В інших моделях, таких як публічні IaaS та PaaS моделі, користувач має доступ до обробки даних та програмного забезпечення, при цьому доступу до апаратного забезпечення немає. Відповідно, з перспективи користувача хмарного сервісу найціннішим активом у хмарному середовищі є його дані. Як і будь-які інші послуги в мережі Інтернет, хмарні сервіси також атакуються на системи безпеки. Компрометація хмарних послуг зазвичай призводить до короткострокових ефектів і пошкодження можуть бути відновлені. Компрометація конфіденційності та конфіденційності даних споживачів хмарних послуг може призвести до довгострокових ефектів і будь-які втрати можуть бути досить важкими для відновлення. Наприклад, коли кілька паролів з адміністративних облікових записів UK's National Healthcare System (NHS) були зламані 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.

Отже, власники даних стурбовані безпекою та конфіденційністю їх даних та бажають зберегти свої дані у безпеці, навіть від провайдерів послуг зберігання даних у хмарі. Крім того, вони вважають за краще власноруч керувати політикою безпеки своїх даних у безпечному режимі, начебто ці дані зберігалися б на їхньому ПК. Традиційна концепція безпеки зазвичай зосереджена навколо технологій та пристроїв, що використовуються для зберігання та обробки даних. Ця концепція може бути важко адаптована для забезпечення необхідного рівня безпеки. Наукове співтовариство нещодавно звернуло увагу на питання безпеки та конфіденційності даних у хмарах, запропоновані рішення в основному спрямовані на забезпечення безпеки Операційних Систем (ОС), що лежать в основі та хмарні сервіси віртуальних машин (VM). Таким чином, більшість рішень, як і раніше, засновані на традиційній концепції безпеки і в основному фокусуються на будь-якій ОС-орієнтованій або VM-орієнтованій безпеці. Деякі з цих рішень грунтуються на концепті надійних обчислень, запропонованому та розробленим групою Trusted Computing Group (TCG). TCG прагне розробити набір стандартів та технологій, таких як Trusted Platform Module (TPM), які можуть зберігати клієнтські дані та програми, що обробляються в межах хмарної інфраструктури, який безпечний 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 фізичного розташування даних у хмарі. Тим не менш, хмари клієнти повинні спочатку довіряти TCG технології з точки зору оцінки надійності провайдерів. Навіть якщо TPM та інші інструменти TCG є гарантією безпеки, фокус цих інструментів не полягає в тому, щоб надати користувачам бажаний контроль за безпекою та конфіденційністю даних. Натомість, з погляду клієнта, реалізація концепції MC дозволяє здійснювати клієнтам моніторинг чи аудит операцій, у тому числі над політикою контролю доступу для сервера через довірені

інструменти, які можуть забезпечити докази відповідності концепції МС для користувачів. Нова концепція була запропонована декількома дослідниками для вирішення конкретних питань безпеки хмарних обчислень, перемішуючи фокус 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 системноцентровані. Якщо рішення сфокусовано на окремому визначеному рівні, його класифіковано відповідно до цього специфічного рівня. 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 апаратний 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.

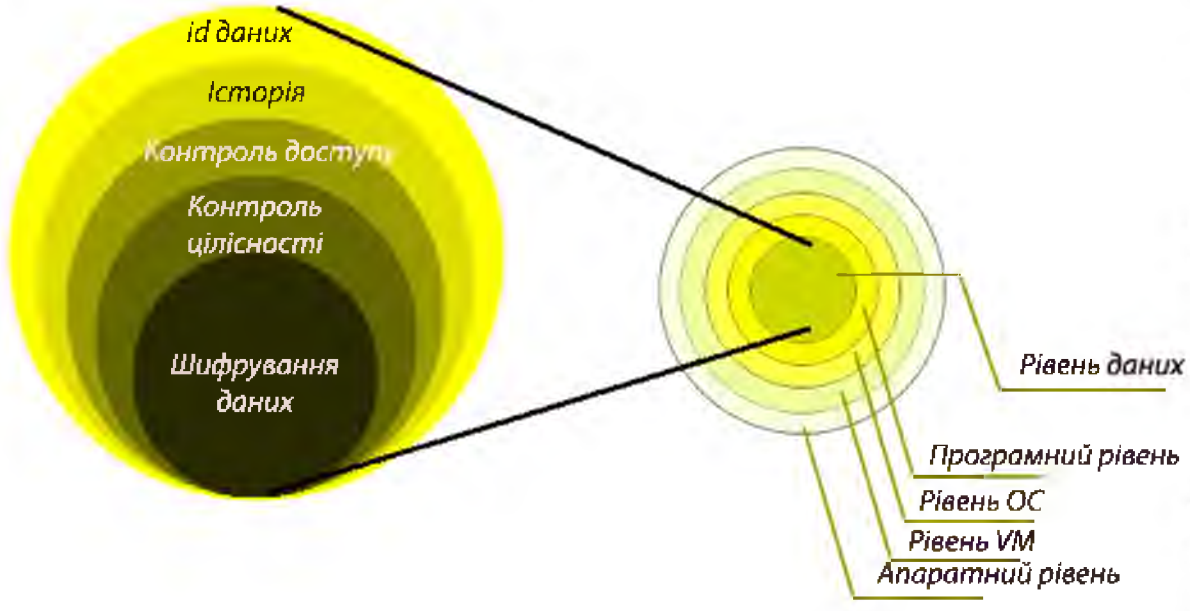

Рисунок 2.1 – Інформаційно-орієнтована та системно-орієнтована моделі

Рисунок 2.2 – Можливі рівні безпеки

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 дані, 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 орієнтації 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

Нині концепція ОБІ перебуває в етапі зародження, проте стрімко розвивається. Систематичний перегляд літератури у напрямі пошуку реалізованих систем інформаційно-орієнтованої безпеки не приніс плідних результатів. Однак можна відзначити роботу колег з Індії, які практично досліджували інформаційно орієнтовану систему безпеки на прикладі розподіленої системи охорони здоров'я. Значимість результатів дослідження підкріплюється авторитетом міжнародної конференції "Розподілених обчислень та мереж" [4]. Дослідники пропонують рішення, здатні забезпечити конфіденційність, контроль доступу та цілісність. Дані спочатку сегментуються і кожен сегмент шифрується за допомогою статичних симетричних наборів ключів, а потім динамічного симетричного ключа в момент передачі даних. Статичний ключ створюється за допомогою технології AES. Наступний симетричний ключ розроблений методом ECDH. Це процес,

що динамічно генерує ключ, тому не потрібно постійно зберігати ключ. Щоразу цей ключ виводиться на обох кінцях. Далі використовується асиметрична техніка з розділення ключів для генерації симетричного ключа, використовуючи KDF.

2.1.3 Класифікація даних

Власник даних зазвичай є найкращим суб'єктом для оцінки вимог безпеки своїх даних. Як правило, дані повинні оброблятися відповідно до їх значення, оскільки наступний принцип безпеки зазначає: «цінність даних, що захищаються, впливають на заходи, що вживаються для їх захисту». Отже, дані класифікуються виходячи з оцінки власника даних та вимог до безпеки. Наприклад, дані можуть бути класифіковані як повністю таємні, секретні чи 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, підключена до даних, або представлена на вимогу необхідних заходів безпеки, що застосовуються до даних.

2.1.4 Політики контролю доступу 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 та обмеження, що контролюють доступ, використання та потік даних. Основа концепції полягає у контролі того, хто чи що може отримати доступ до даних та з якими наданими правами, наприклад, читати, писати, що відповідають традиційній концепції політики контролю доступу. Питання полягає не тільки у визначенні цих політик, але й у забезпеченні дотримання цих політик безпечним чином, що зберігає

конфіденційність користувачів при їхньому доступі або спільному використанні даних. У підході ОБІ політики контролю доступу та механізму 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 поширені та легко модифіковані для адаптації до динамічних змін вимог до використання та спільного використання даних у хмарному обчислювальному середовищі. Оскільки у кожного набору даних є свої політики доступу, виникають інші складні проблеми взаємодії різних політик контролю доступу та права власності на новий набір даних, який створюється у хмарі від обробки кількох наборів даних із різними політиками та власниками даних.

2.1.5 Контроль 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 навіть сам хмарний провайдер може бути випадково чи навмисно протягом життєвого циклу набору. Тому власники даних та авторизовані користувачі вимагають, щоб ця функція гарантувала, що дані не будуть змінені некоректно. У підході ОБІ інформація для перевірки цілісності даних вбудована у дані та захищена. Перевірка не залежить від інформації, наданої поза самими даними. Отже, пряме підтвердження цілісності даних у хмарі без необхідності завантаження – це великий виклик, особливо якщо дані динамічно змінюються у хмарі.

2.1.6 Приватність та конфіденційність

Конфіденційність означає, що лише уповноважені сторони мають можливість доступу до захищених даних із відповідними правами та

привілеями, визначеними власником даних. Ризик компрометації конфіденційності даних зростає у хмарній моделі через збільшення числа сторін, включаючи інших користувачів в одній хмарі. Крім того, керування 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 та доступні іншим сторонам. Захист конфіденційності йде далі, ніж захист фактичних даних від несанкціонованого доступу. Він також запобігає витоку будь-якої інформації, що розглядається як приватна інформація при обробці даних. Будь-яке рішення безпеки має враховувати конфіденційність даних та конфіденційність хмарних клієнтів відповідно до різних правил та вимог.

2.1.7 Доступність

У хмарних обчисленнях доступність відноситься до послуг хмарних обчислень, наявних та доступних для авторизованих користувачів, коли вони їм потрібні. Крім того, ці служби можуть бути пов'язані з обробкою критично важливих даних та запуском важливих програм, які повинні бути доступні весь час і здатні обслуговувати безліч користувачів. Доступність послуг хмарних обчислень залежить від постійної безперервної роботи інфраструктур та мережевих ресурсів. Тому хмарний провайдер повинен мати надійні та зайві стратегії для забезпечення доступності системи. Хмарний провайдер також несе відповідальність за обслуговування своїх сервісів, навіть якщо є внутрішня чи зовнішня загроза, орієнтована 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 у будь-який час. Як уже згадувалося, на етапах життєвого циклу даних постачальник хмари повинен мати ефективну стратегію резервного копіювання та архівування для захисту

доступності даних. З погляду власника даних, дані є найважливішим активом, яким він володіє у хмарі. Розглянутий інформаційно-орієнтований підхід у цій роботі дає власнику даних більш істотну відповідальність і контроль за захистом своїх даних, які мають зберігатися серед хмарних обчислень.

2.1.8 Концептуальна основа ОБІ

У цьому пункті створено цілісну концептуальну основу ОБІ. Ця основа грунтується на огляді різних попередніх досліджень щодо концепцій ОБІ, що застосовуються до хмарної моделі, та на основі характеристик ОБІ, визначених раніше у пункті 1.6.4. В рамках основи концепції дані захищені перш ніж залишити довірений домен власника даних, які параметри безпеки прив'язані до даних як частину їх метаданих. В ідеальній ситуації лише облікові дані, які використовуються для скасування або доступу через захист, зберігаються за межами даних, а також власником даних та авторизованими користувачами відповідно до їхніх прав доступу. Будь-які інші параметри безпеки даних мають бути пов'язані з даними. Наприклад, у випадках, коли дані динамічно оновлюються, користувач, який звертається до даних, повинен перевірити, чи є дані найсучаснішими. Ця вимога повинна бути надана з фактичних даних або метаданих, прикріплених до даних, наприклад, функція історії є рівнем рівня даних, як показано на Рисунку 2.1. Як інший приклад, доказ достовірності джерела даних також може бути доданий разом з доказом цілісності, що зазвичай включає цифровий підпис власника даних. Будь-яка додаткова функція може бути додана як новий підрівень під рівнем даних або включена як один із зразкових підрівнів, проілюстрованих у лівій частині Рисунка 2.1. Крім того, всі ці метадані безпеки надаються на рівні даних та створюються власником даних. Вони також залишаються захищеними протягом усього циклу даних. Окремий набір даних має свої метадані безпеки, незалежні від інших наборів даних. У наступному списку наведено критерії,

які повинні задовольняти будь-яке надане рішення безпеки на основі підходу ОБМ.

– Кожен набір даних є самоописним, самозахищеним. Отже, вимоги та функції безпеки кожного набору даних надаються зсередини та 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.

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

– Усі операції, пов'язані з доступом до захищених даних, встановлені авторизованими користувачами, та примусове застосування політик безпеки даних виконуються без шкоди для конфіденційності користувачів або конфіденційності даних. Приклад концептуальної основи ОБІ для моделі хмарних обчислень показано на малюнку 2.2. Створення даних здійснюється у 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 та можливості відстеження. На етапі створення створюється безпечний набір даних і він інкапсулює вміст захищених даних та відповідні метадані. Контролер керування даними та відстеження дозволяє власнику даних керувати та відстежувати набір даних, наприклад, розташування, використання та історію доступу, з моменту створення, доки він не буде видалений з хмарного середовища. Крім того, передача між надійним доменом, в якому знаходиться користувач, та хмарним доменом, зазвичай здійснюється через Інтернет та захищена основним механізмом захисту, що забезпечується підходом ОБІ або протоколом інтернет-безпеки. У захищеного набору даних протягом його життєвого циклу в хмарі є можливість, виконавши додані коди, інформувати власника даних, якщо є які-

небудь зміни його стану. У той же час захищений набір даних включає виконуваний код, готовий до прийому і відповіді на запити власника даних для відстеження інформації та команд управління. Навіть якщо зв'язок між хмарним доменом та 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 малюнку 2.3, примусове застосування політик доступу буде виконуватися ресурсами хмарного сервера в домені хмари, але на основі параметрів безпеки, прикріплених до набору захищених даних. Правило провайдера хмарних обчислень у примусовому порядку дозволяє виконувати ці політики доступу, не розкриваючи деталі політики контролю доступу або вмісту набору даних. Власник даних може перевірити цілісність захищених даних, включаючи історію записів. Коли захищений набір даних має бути отриманий авторизованим користувачем, сервер хмари в хмарному домені перевірятиме права доступу користувачів до даних за допомогою пов'язаних параметрів безпеки. Авторизованим користувачам можна завантажувати захищений набір даних. Потім авторизовані користувачі можуть перевірити цілісність та оригінальність завантажених даних, використовуючи додані докази цілісності та справжності. Ця концептуальна основа ОБІ дозволяє, з одного боку, оптимізувати конфігурацію безпеки параметрів безпеки кожного набору даних, з другого – 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 та ТТР) відповідно до трьох рівнів відповідальності безпеки, проілюстрованих на Рисунку 2.1. Наприклад, хмарні провайдери несуть відповідальність за захист своїх систем, ТТР, відповідальних за активи, що надаються провайдерами заходи безпеки та власники даних несуть відповідальність за захист своїх даних у хмарі[5]. Інші

очікувані переваги застосування ОБІ до хмарної моделі обговорюються у наступному пункті.

Рисунок 2.3 – Основа ОБІ

2.2 Висновки

У цьому розділі наведено відомості про підхід ОБІ і його застосування до хмарної моделі. У цій дипломній роботі підхід ОБІ будується на трьох базових засадах:

– 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 даними.

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

Протягом усього життєвого циклу наборів даних власник даних може керувати своїми специфікаціями безпеки та відстежувати їх, включаючи історію використання. Найголовніше, що на першому етапі життєвого циклу даних, етап створення набору даних власник даних відповідає за побудову кожного набору даних з усіма вимогами безпеки перед відправкою його в хмару. На етапах, що залишилися, до етапу знищення набору даних підтримується відповідність політикам безпеки власників даних. Очікується, що застосування такого підходу до хмарної моделі покращить конфіденційність даних та безпеку у хмарі, щоб відповідати її масштабованості та еластичності. Однак повне застосування ідеальної концептуальної основи ОБІ, як і раніше, ускладнене кількома проблемами. Сфера застосування підходу ОБІ у цьому дослідженні визначається обмеженням таких проблем. Хоча область охоплення обмежена неструктурованими даними, що зберігаються і поділяються у публічному хмарному сховищі, пропоноване рішення охоплює найпоширеніші ситуації у хмарних сервісах. Більше того, рішення може бути змінено для застосування в інших формах даних та інших хмарних моделей. Основні вимоги безпеки для запропонованого рішення наведені в пункті 2.6. Вони в основному пов'язані із захистом конфіденційності та цілісності при доступі та пошуку зашифрованих даних, що зберігаються на хмарних серверах. На основі концепцій ОБІ розглядаються методи безпеки, які можуть відповідати цим вимогам. Методи,

що відповідають вимогам безпеки, будуть адаптовані та використані у запропонованому рішенні ефективним та практичним способом. У наступному розділі запропоноване рішення докладно роз'ясніються.

3 ЗАБЕЗПЕЧЕННЯ ПРИВАТНОСТІ ТА БЕЗПЕКИ ПРИ ВИКОРИСТАННІ SMART-ВАГІВ

Цей розділ визначає запропоноване рішення з основних досліджень цієї роботи. Це рішення базується на безпеці інформації, що описано у розділі 2. Рішення призначене для задоволення наступного переліку бажаних вимог до додатків у середовищі хмарних обчислень:

– Дані зашифровані та доступні лише для авторизованих користувачів.

– Дані доступні для пошуку без загрози для їхньої конфіденційності.

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

– Провайдер сервера не знає кількість чи особу користувачів, авторизованих та які мають право доступу до даних.

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

– Дані містять усю необхідну інформацію для перевірки їхньої цілісності 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, як мінімум, однією з зазначених вище вимог. Усі параметри, необхідні для досягнення функцій безпеки, прикріплюються до файлу даних і в результаті є файл під назвою OBI-файл. Функції безпеки включають захист конфіденційності, перевірку цілісності та автентифікацію.

3.1 Застосуваннія теореми про залишки

Перетворення китайської теореми про залишки (КТС) [6] є фундаментальною основою рішення широкого класу задач теорії чисел, а також прикладних завдань інженерії та інформатики. Незважаючи на свою простоту та давню історію, КТС продовжує представляти себе в новому світлі та відкривати нові перспективи застосування, особливо в математиці, інформатиці (машинна арифметика) [7], криптографії [8] тощо. Побудова непозиційної системи числення в обчислювальних системах (системи залишкових класів) для виконання операцій з великими числами [9], дискретне перетворення Фур'є, генерування таємних ключів в асиметричних криптосистемах, зв'язок з класичною поліноміальною інтерполяційною теорією, багатовимірні обчислення, можливість зведення надлишків за модулем m (де m – довільне ціле число) до вивчення кільця надлишків за модулем s (p – просте число), дослідження алгебраїчних кілець, можливість арифметичної самокорекції кодів та розпаралелювання обчислень, визначення послідовності великої кількості зразків ДНК – ось далеко не повний перелік сучасного застосування КТЗ.

Теорема 1. Китайська теорема про залишки [10]

Для будь-яких заданих цілих чисел a1, a2, …, ak, наступна система одночасної конгруенції має унікальне рішення X, яке лежить в межах $0 \le X < n = n1n2\ldots nk$, за умови, що невід’ємні цілі числа n1, n2, … nk взаємно прості.

$$\begin{aligned} X &\equiv a1 \bmod n_1 \\ X &\equiv a2 \bmod n_2 \\ X &\equiv ak \bmod n_k \end{aligned} \tag{3.1}$$

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 одиночасної конгруентності можна розрахувати за наступним рівнянням:

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

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

M_i^{-1} є зворотним $M_i\ mod\ n_i$, тобто $M_i M_i^{-1} \equiv 1 \pmod{n_1}$

Через те що M_i відносно первинне до n_i, є унікальний мультиплікативний зворот $\bmod a_i$. Отже, розрахунок мультиплікативного зворотного в модулі є істотною частиною визначення рішення КТЗ. Розширений алгоритм Евкліда [11] може бути використаний для обчислення мультиплікативного звороту та є істотним компонентом алгоритму Гарнера [12], який зазвичай використовується для визначення рішення КТЗ. Нижченаведений алгоритм 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. $C_i = u \cdot C_i \bmod n_i$.

2. $u \leftarrow a_i$, $X \leftarrow u$.

3. Для i = 2 до k справедливе:

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

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 $a_{k+1} \bmod n_{k+1}$ додається до конгруентності, нехай рішення для нової конгруенції буде X' . Нове рішення може бути знайдене шляхом обчислення рішення наступних двох конгруенцій:

$$X' \equiv X \bmod n_1, n_2 \ldots n_k \text{ та}$$

$$X' \equiv a_{k+1} \bmod n_{k+1}.$$

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 $X \equiv a_k \bmod n_k$ у Рівнянні (3.1) буде виключено. Нове рішення може бути вирахуване дією з одним модулем

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

Ці функції можуть використовуватися для мінімізації додаткових обчислень розрахунку рішення КТЗ у таких випадках. Наприклад, ця функція використовується у запропонованому рішенні у цьому пункті під час видачі прав доступу новому користувачеві до певних ресурсів. Це питання обговорюватиметься у пункті 3.2.3. КТЄ використовує кілька додатків у криптографії та зв'язку захищеними мережами через свою арифметичну природу, яка не споживає високих обчислювальних ресурсів. Таким чином, він підходить для обчислювальних та комунікаційних додатків з обмеженими ресурсами. Поділ секрету є однією з основних функцій безпеки з використанням КТЗ у криптографії. Це дозволяє виконувати поділ, виводячи

з нього кілька загальних, які можуть бути відновлені тільки з заздалегідь підготовленим набором значень. Багато програм засновані на цій схемі поділу секрету, такі як гранична криптографія та електронне голосування.

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

3.2 Управління доступом та розділення ключа використовуючи теорему про залишки

У цій роботі запропоноване рішення використовує КТС та криптосистему з публічним ключем для безпечного обміну даними захищених користувачів, що зберігаються в середовищі хмарних обчислень серед авторизованих користувачів [13]. Дані, що іноді згадуються як ресурси або файли, захищені симетричним методом шифрування, де в якості секретного ключа використовується Ks. У роботі ресурсом може бути набір даних чи файл, що містить дані будь-якого типу, зокрема текст, аудіо, зображення чи відео. Для поширення секретного ключа авторизованих користувачів він зашифрований з використанням методу публічного шифрування відкритого

ключа користувача. Шифрування також включає інше значення, Cr , яке буде використовуватися як відповідь на запит сервера, коли користувач запитує доступ до ресурсу. Лише авторизовані користувачі, які мають відповідний приватний ключ, можуть розшифрувати код Cr та Kr для отримання Cr, щоб отримати доступ до ресурсу r. Параметри Cr та Ks об'єднуються для формування Cr||Ks та розглядаються як єдине значення. Це значення шифрується за допомогою публічного ключа $E_{K_{pub\,i}}$ кожного авторизованого користувача u_i, в результаті код $E_{K_{pub\,i}}(C_r||K_s)$ для цього користувача u_i, де i=1, 2, 3, … , k, та k кількість авторизованих користувачів, що мають доступ до ресурсу r.

Щоб застосувати КТЗ до запропонованого рішення, для користувачів $u_1, u_2 \dots u_k$, кожен авторизований користувач пов'язаний з унікальним відносним простим числом $n_i = n_1, n_2, \dots n_k$, 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 $E_{K_{pub\,i}}(C_r||K_s)$, використовується для заміни a_i у Рівнянні (3.1) для виведення наступної спільної конгруентності:

$$\begin{aligned} X_r &\equiv (E_{K_{pub\,1}}(C_r||K_s)) \bmod n_1 \\ X_r &\equiv (E_{K_{pub\,2}}(C_r||K_s)) \bmod n_2 \\ X_r &\equiv (E_{K_{pub\,k}}(C_r||K_s)) \bmod n_k \end{aligned} \qquad (3.3)$$

Вирішення цієї конгруенції Xr, таке, що $0 \leq Xr < n = n_1, n_2, \dots n_k$, є загальним значенням для ресурсу r і він приєднаний до ресурсу, а ресурс і загальне значення зберігаються разом в хмарному сервері. Коли авторизований користувач u_i 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:

$$(\text{Cr}||\text{Ks} = D_{K_{prv\,i}}(X_r \bmod n_i)) \quad (3.4)$$

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 Cr 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 того, щоб розшифрувати файл та отримати його вміст.

3.2.1 Підвищення безпеки ключів шифрування та файлів

З точки зору підвищення безпеки у запропонованому рішенні, власник даних повинен використовувати унікальний симетричний ключ K_s для шифрування кожного файлу перед відправкою його на зберігання до хмари. Відповідно до підходу орієнтованого на безпеку даних, всі вимоги безпеки прикріплені до фактичних даних, отже, кожен симетричний ключ прикріплюється до цього файлу. З рівняння (3.3), симетричний ключ K_s захищений двома рівнями безпеки: спочатку шифрування його авторизованим користувачем, а потім за допомогою КТС, щоб знайти загальне значення Xr. Тільки авторизовані користувачі можуть обчислити K_s з Xr за допомогою рівняння (3.4), якому необхідний відповідний n_i приватний ключ авторизованого користувача. Для ефективного управління ключами власник даних додає себе як користувача при розрахунку Xr для кожного файлу за допомогою рівняння (3.3). Отже, ні власник даних, ні користувачі не зобов'язані зберігати K_s, але необхідний лише їх приватний ключ і призначене значення n_i. Таким чином, ключ K_s надійно та ефективно, спільно з

авторизованими користувачами, захищений від несанкціонованого доступу, включаючи постачальника послуг хостингу. Секретний ключ K_s, а також значення Cr є унікальними для кожного файлу. Якщо значення одного файлу було скомпрометовано, інші файли залишаються у безпеці. Секретне значення Cr використовується авторизованим користувачем, щоб показати серверу, що він має право отримати доступ до ресурсу r (тобто файлу). Власник даних надійно прикріплює секретне значення Cr до файлу, а також посилає його всередині Xr коду до сервера. Тільки авторизовані користувачі можуть розрахувати секретне значення Cr за допомогою розділеного значення Xr. Таким чином, лише авторизовані користувачі можуть знати та відкрити Cr сервер і довести, що вони мають право доступу до цього конкретного файлу. Секретне значення Cr є унікальним для кожного файлу, навіть якщо він використовується одним користувачем для різних файлів. Таким чином, якщо один Cr для певного файлу скомпрометовано, інші файли не будуть під загрозою. Крім того, ця функція корисна, якщо власник даних хоче змінити деталі авторизованих користувачів для файлу, наприклад, щоб додати нового авторизованого користувача, власнику даних потрібно лише змінити Xr для цього файлу. Оскільки параметр Cr може залишатися таким самим, немає необхідності повторно надсилати новий Cr на сервер. Докладніше дане питання освітлено в Пункті 3.2.3

3.2.2 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 Cr і розділеним значенням Xr, що розраховується власником даних. Провайдер може прочитати розділене значення Xr, але не може дізнатися K_s, який було використано для шифрування даних. Лише авторизовані користувачі можуть виявити K_s від Xr. Провайдер також може зчитувати секретне значення Cr 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 відносно 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 ($E_{K_{pub\,i}}(C_r||K_s)$) від спільного значення Xr. 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 Xr файлу і користувач отримав його/її n_i.

Повідомлення, якими обмінюються авторизовані користувачі і хостинг хмарним сервером загальних файлів показано на Рисунку 3.2. 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, nonce (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 $E_{K_{pub\,S}}(ID_r,\ ID_i,\ K_t,$ nonce). Сесійний ключ Kt створюється користувачем для безпечного обміну інформацією між користувачем і сервером. Сервер знаходить файл з IDr, читає його спільне значення Xr, збільшує nonce до (nonce +1) і шифрує їх за допомогою сесійного ключа Kt. Результат, можна представити як $E_{K_t}(X_r, \text{nonce} + 1)$, який повертається користувачу.

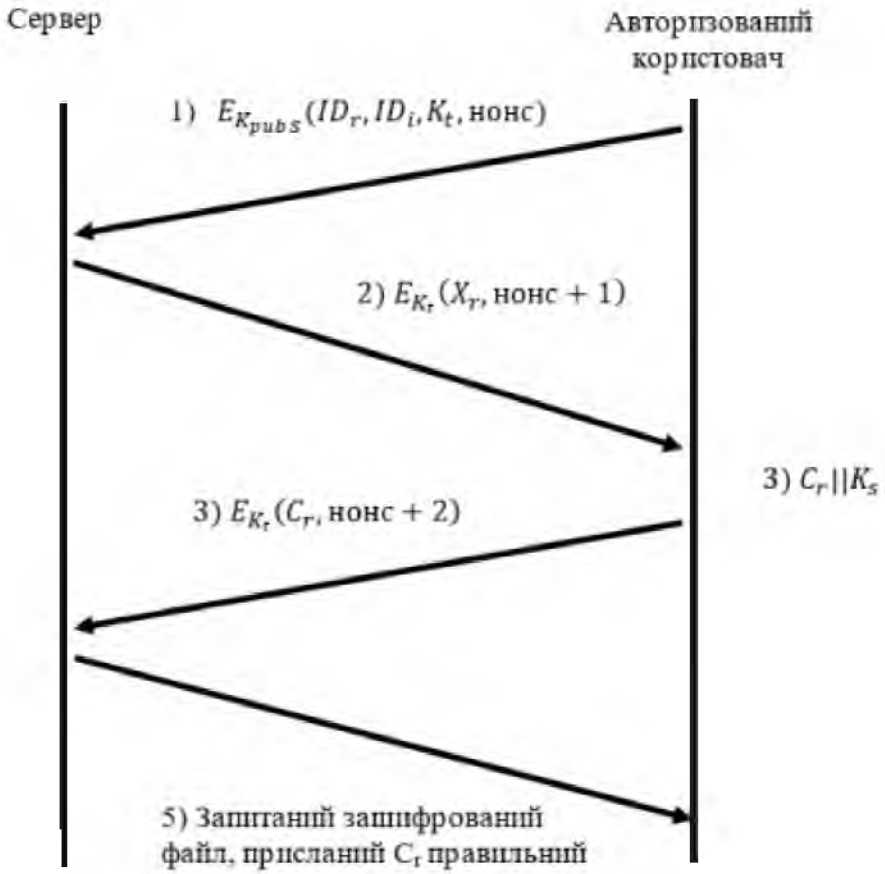

Рисунок 3.1 – Алгоритм контролю доступа

Після отримання інформації, представленої в повідомленні номер два на Рисунку 3.1, з сервера, користувач обчислює секретний параметр Cr, використовуючи Рівняння (3.4) і повертає Cr та (nonce + 2) на сервер і шифрується сесійний ключ K_t, тобто $E_{K_t}(C_r, \text{nonce} + 2)$. Сервер перевіряє Cr 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 3.1. Всі повідомлення між користувачем і сервером є зашифровані і інкрементуючий nonce використовується для запобігання традиційних атак, такі як Man-In-The

Middle (MITM). Як було згадано вище, для конфіденційного доступу особистість авторизованого користувача прихована від сервера, отже, процедури уникає використання публічного ключа користувача, щоб зберегти його особистість у таємниці. На противагу цьому користувач починає сесію з сервером з повідомленням зашифрованим за допомогою публічного ключа сервера для перевірки автентичності 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 nonce, а потім реагувати на зашифроване повідомлення, що містить nonce + 1. Процедура доступу показує виконання політики контролю доступу та забезпечення авторизованого користувача секретним ключем для дешифрування даних здійснюється в одному механізмі. Цей метод зменшує додаткове навантаження як на сервер хмар, так і на користувача з точки зору обчислення та управління ключами. До того ж використання КТЗ у цьому механізмі є ще одним фактором для зменшення додаткових витрат, оскільки операції КТЗ є простими модульними арифметичними операціями; зокрема, наприклад, немає модульних експонентних операцій. Ще одна важлива особливість, яка досягається за допомогою використання КТЗ, є те, що КТЗ дозволяє власнику даних приховувати число авторизованих користувачів.Через особливості роботи, які було описано в Пункті 3.1, код ($E_{K_{pub\,i}}(C_r||K_s)$) кожного авторизованого користувача u_i для i = 1, 2, ... k може бути представлений з використанням КТЗ з одним значенням Xr. Сервер не може дізнатися скільки користувачів представлені в цьому Xr, але без використання КТЗ власник даних повинен прикріпити всі коди, щоб розкрити серверу кількість користувачів.

3.2.3 Алгоритм надання та скасування доступу до даних

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 $X_r^{`}$.

$$X_r^{`} \equiv (E_{K_{pub\,1}}(C_r||K_s)) \bmod n_1$$
$$X_r^{`} \equiv (E_{K_{pub\,2}}(C_r||K_s)) \bmod n_2$$
$$X_r^{`} \equiv (E_{K_{pub\,k}}(C_r||K_s)) \bmod n_k \qquad (3.5)$$
$$X_r^{`} \equiv (E_{K_{pub\,k+1}}(C_r||K_s)) \bmod n_{k+1}$$

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

$$X_r^{`} \equiv X_r \bmod n_1, n_2 \ldots n_k$$
$$X_r^{`} \equiv (E_{K_{pub\,k+1}}(C_r||K_s)) \bmod n_{k+1} \qquad (3.6)$$

Власник даних надсилає нове загальне значення $XXrr'$ з ID файлу та ID власника даних до серверу для заміни старого Xr. Механізм додавання нового користувача ефективний з двох причин; необхідна лише одна асиметрична операція шифрування для фіксованої довжини інформації, тобто, $CCr||KKss$, та він знаходить рішення лише для двох конгруенцій, див. Рівняння (3.6).

Щоб скасувати право доступу користувача u_k до файлу r, власник даних повинен змінити секретне значення Cr на $C_r^{`}$ та перерахувати Xr на $X_r^{``}$ для користувачів, що зберігають право доступу до файлу, від 1 до k-1 наступним чином:

$$\begin{aligned} X_r^{``} &\equiv (E_{K_{pub\,1}}(C_r^{`} || K_s)) \bmod n_1 \\ X_r^{``} &\equiv (E_{K_{pub\,2}}(C_r^{`} || K_s)) \bmod n_2 \\ X_r^{``} &\equiv (E_{K_{pub\,k-1}}(C_r^{`} || K_s)) \bmod n_{k-1} \end{aligned} \quad (3.7)$$

Власник даних посилає на сервер нові значення ключів з ID файлу та ID власника файлу, щоб замінити відповідні старі значення для файлу. Якщо користувач отримав доступ до даних перед скасуванням старих значень, можливо, 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 анульований доступ до файлу і власник даних змінив тільки Cr для цього файлу, користувач може домовлятися з сервером, тому сервер дозволяє йому отримати доступ до файлу, і користувач надає серверу старий ключ Ks, що розшифровує файл. Незважаючи на те, що описано в Пункті 3.2.2, схема не розкриває особистість користувачів для сервера і може знизити ймовірність такого роду collusion attack, рішуче рекомендується змінити ключ Ks та повторно зашифрувати файл, особливо якщо відбулася зміна змісту. Потім замініть старий файл на новий, який був зашифрований 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.

3.3 Можливість пошуку по зашифрованим даним

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.

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

Від кожного ключового слова w_i, власник даних створює сі 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 підрозділ розглядає алгоритм того, як користувачі можуть безпечно здійснювати пошук їхніми зашифрованими даними за допомогою використання зашифрованих ключових слів.

3.4 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 (Kpub i). Користувач розшифровує повідомлення за допомогою свого приватного ключа та потім зашифровує T_w з публічним ключом сервера (Kpub S), перед відправкою його на сервер в якості пошукового запиту з ID власника, nonce і сесійний ключ Kt, як показано в Рівнянні (3.9).

$$E_{K_{pub\ s}}(T_w, ID \text{ власника даних}, K_t, \text{nonce}) \qquad (3.9)$$

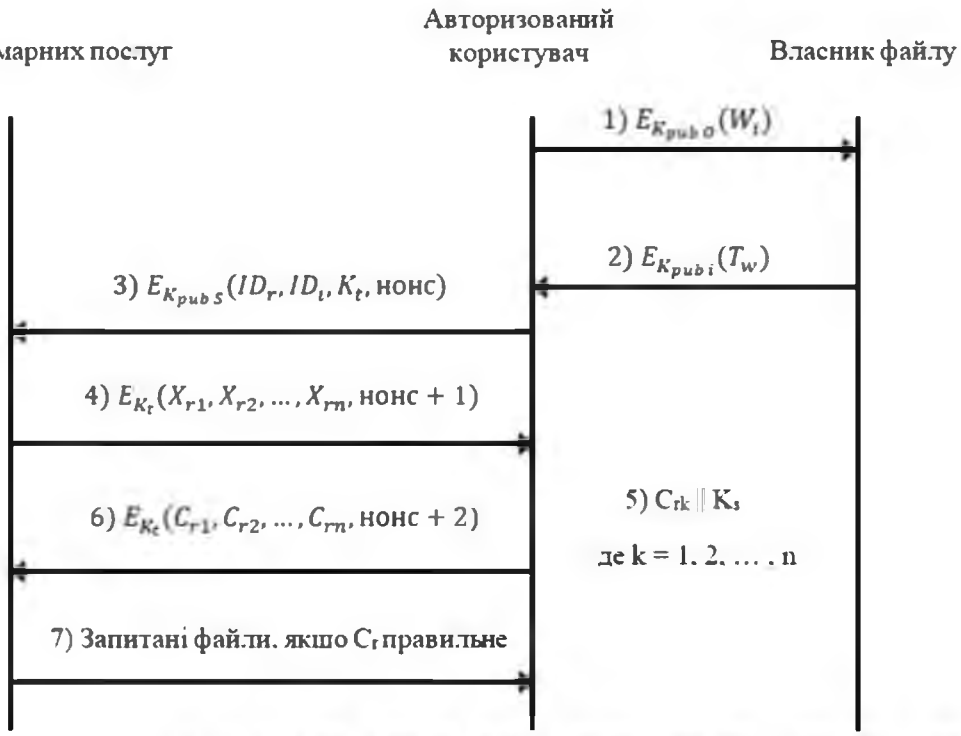

Рисунок 3.2 – Алгоритм безпечного пошуку по зашифрованим даним

Сервер повинен шукати тільки дані, пов'язані з ID власника, використовуючи Tw для обчислення $P = H_{T_w}(R)$, а потім для кожного ci сервер перевіряє результат $P \oplus$ ci, щоб побачити, якщо перший біт дорівнює флагу. Потім ID файлу, пов'язане з цим ci, додається до списку результатів пошуку

разом із власними двома параметрами Xr та Cr. Цілі та зміст елементів параметрів Xr та Cr описано в Пункті 3.3.

$$P = H_{Tw}(R), \text{ if } P \oplus ci = \text{флаг}||\text{зміщення} \therefore \text{є збіг} \quad (3.10)$$

Після пошуку всіх файлів, сервер надсилає лише те значення Xr ($Xr1$, $Xr2$, … Xr), де n - кількість файлів, які відповідають критеріям пошуку, та nonce+ 1 і всі зашифровані за допомогою сесійний ключ, Kt, назад клієнту.

$$E_{Kt}(Xr1, Xr2, \ldots Xrr, \text{nonce} + 1) \quad (3.11)$$

Користувач отримує секретне значення Cri для кожного загального значення Xr, де i = 1, 2, ..., n, за допомогою Рівняння (3.4), а потім повертає секретні значення і nonce + 2 на сервер зашифроване за допомогою сесійного ключа Kt.

$$E_{Kt}(\text{C}r1, \text{C}r2, \ldots \text{C}r\text{n}, \text{nonce} + 2) \quad (3.12)$$

Сервер перевіряє секретні значення ($Cr1$, $Cr2$, … , Cr) та надсилає клієнту лише ті данні, зашифровані дані , яких збігається з тими, які були надіслані користувачем. Тепер користувач може розшифрувати дані за допомогою секретного ключа Ks, витягнутого з Xri для кожного файлу. Рисунок 3.2 ілюструє безпечний алгоритм пошуку.

3.5 Створення захищеного файлу з перевірками цілісності даних та аутентичності вмісту

Підхід захисту інформації створює захищений контейнер, який включає в себе всі попередні модулі описані в підрозділах вище.Контейнер

інкапсулює вихідний файл даних перед його відправкою в хмарний сервер. Процес створення захищеного контейнеру передбачено проводити в повністю довіреній машині, яка належить власнику даних. Перед створенням є чотири основні операції:

1. шифрування контенту файлу за допомогою алгоритму симетричного шифрування (див. Рисунок 3.3 блок A)

2. створити, за допомогою китайської теореми залишків, загальне значення Xr (див. Рисунок 3.3 блок C)

3. створеннясекретних ключових слів для можливостей зашифрованого пошуку (див. Рисунок 3.3 блок D)

4. створення перевірок цілістиості даних та аутентичності вмісту для зашифрованого результуючого 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 ключ $K_{prv\ o}$ власника файлу. Потім на основі цих чотирьох операцій створюється захищений і конфіденційний контейнер.

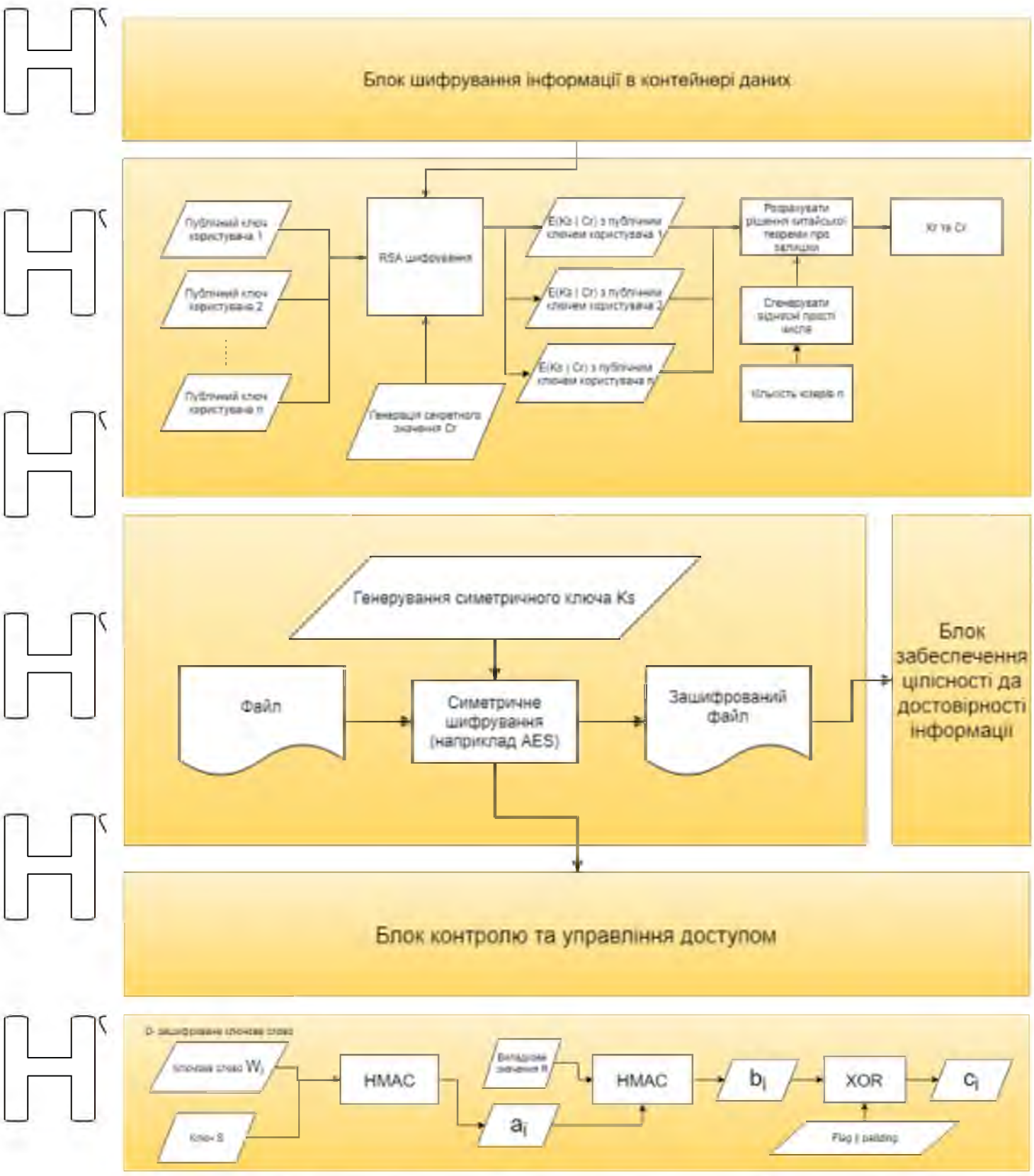

Рисунок 3.3 – Структура підходу орієнтованого на безпеку інформації

3.6 Збереження конфіденційності та цілісність запропонованого рішення

У цьому розділі представлено короткий виклад особливостей запропонованого рішення для забезпечення безпеки даних. Ми використовуємо чудовий симетричний ключ для шифрування та інше секретне значення для забезпечення дотримання параметрів контролю доступу для кожного файлу. Це покращення 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 Ks і/або секретне значення Cr 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.

– Досліджене рішення є більш витривалим до можливих атак користувачів, позбавлених права доступу, і хмарного серверу.

Оскільки кожен файл захищається унікальним симетричним ключем, сервер не може вступити в змову з користувачем, позбавленим права доступу, для того щоб переглянути вміст даних, якщо такий користувач не мав доступу перед позбавленням його таких прав. Особливо, якщо такий користувач отримав доступ до захищеного контейнеру і показав симетричний ключ Ks, то власник даних може видалити файл і використовувати новий ключ Ks для створення нового контейнеру. Таким чином, якщо сервер запобігає збереженню копії цього контейнеру, 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 Cr для сервера. Має бути змінено лише загальне значення Xr. Отже, знижується можливість компрометації значення Cr.

– Коли власник даних додає себе як авторизованого користувача для всіх файлів ОБІ, власник даних та авторизовані користувачі повинні залишати в безпеці лише свої пари публічних ключів.

– Запропоноване технічне рішення забезпечує цілісність та автентифікацію для кожного файлу та не потребує 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 захищеного контейнеру. Це дозволяє як хмарному сервіс-провайдеру, так і користувачеві працювати з кожним контейнером більш гнучко та ефективно. Узагальнюючи, слід зазначити, що обчислення КТЗ рішення більш ефективне порівняно з іншими методами криптографії, що використовуються в криптографічному методі контролю доступу, оскільки він використовує прості модульні операції без експоненційних операцій.

3.7 Висновки

У цьому розділі представлено рішення, яке використовує підхід орієнтований на безпеку інформації для хмарних обчислень. Рішення розроблене на основі модулів, кожен з яких забезпечує свої функції безпеки. Перший модуль призначений для шифрування даних перед їх передачею на хмарний сервер. 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 створення зашифрованих ключових слів. Четвертий модуль виділяє перевірки цілісності та аутентичності. Результати всіх попередніх модулів використовуються для створення захищеного контейнеру. 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 захищеному контейнері, залежать від криптографічних алгоритмів, які використовуються в даному рішенні і знаходяться під управлінням власника даних.

ВИСНОВКИ

Ця дипломна робота досліджує підвищення рівня захисту персональних даних у сервісі SMART-вагів, які працюють на основі хмарних обчислень.

У першому розділі було детально розглянуто тему хмарних обчислень і проблем, які виникають при використанні даної технології. На базі опитувань, які проводила Міжнародна корпорація даних (IDC) у вересні 2009 року дана компанія за результатами ранжування даних, дійшла до висновку, що для всіх користувачів хмарних технологій найбільш важлива безпека конфіденційних даних.

Проблеми з безпекою формуються на основі недоліків або помилок при реалізації основних технологічних компонентів, з яких будуються хмарні сервіси. Цими компонентами є:

-Веб-додатки та служби – найбільш часто використовувані технології доступу до хмарних обчислень;

-Віртуалізація є основною технологією надання хмарних обчислень;

-Криптографічні методи – в даний час є найбільш поширеними методами для досягнення задовільного рівня вимог безпеки для хмарних обчислень

У другому розділі розглянуто деталі інформаційно-орієнтованого підходу забезпечення безпеки інформації.Висвітлено класифікацію існуючих рішень і досліджено політики контролю доступу при забезпеченні безпеки і конфіденційності даних.Також було сформовано концептуальну основу підходу сфокусованого на безпеку інформації.

Третій розділ присвячений висвітленню конкретних засад реалізації методу забезпечення безпеки і конфіденційності даних описаного вищє. Перший підрозділ розглядає китайську теорему про залишки, яка створює математичну і криптографічну базу для реалізації підходу. У другому

підрозділі обговорюються засади реалізації управління доступом і розділення ключа для доступа до зашифрованих даних. Потім йде підрозділ присвячений реалізації пошуку по зашифрованих даних. Четвертий і наступні підрозділи присвячені питанням контролю доступу і питанням конфіденційності. Закінчується розділ розглядом структури захищеного контейнера з даними користувача.

Отже, використання такого захищеного контейнера здатне зберегти приватність, цілісність, аутентичність даних які було розміщено в хмарі, навіть від самого хмарного провайдеру з мінімальним впливом на функціональність зашифрованих даних у сервісі SMART- вагів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Робота з BigData у хмарах. Обробка та зберігання даних з прикладами з Microsoft Azure. - СПб.: Пітер, 2019. - 448 с.

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

3. 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.

4. 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.

5. Інформаційно орієнтована концепція забезпечення безпеки інформації [Електронний ресурс] // Режим доступу: https://iconfs.net/journals/number2/%D0%86nformatsijno-ori%D1%94ntovana-kontseptsiya-zabezpechennya-bezpeky-khmarnykh-obchyslen

6. Бухштаб А.А. Теория чисел. – М.: Просвещение, 1966. – 384 с.

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

8. Задірака В., Олексюк О. Комп’ютерна криптологія: Підручник.– К.:2002. – 504 с.

9. Задірака В.К., Олексюк О.С. Комп’ютерна арифметика багаторозрядних чисел: Наукове видання. – К.: 2003. – 264 с.

10. Китайська теорема про залишки [Електронний ресурс] / Режим доступу: http://shaman-nismo.narod.ru/Lahun/31.htm

11. Розширений алгоритм Евкліда [Електронний ресурс] / Режим доступу: http://e-maxx.ru/algo/export_extended_euclid_algorithm

12. Алгоритм Гарнера [Електронний ресурс] // Режим доступу:https://studfile.net/preview/5911354/page:6/

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

14. Псевдовипадкова функція [Електронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%B2%D0%B8%D0%BF%D0%B0%D0%B4%D0%BA%D0%BE%D0%B2%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F

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