МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

УДК 004.946

До захисту допускається
Т.в.о завідувача кафедри
комп'ютерних наук та інженерії
к.т.н, доц. Сафонова С. О.
_____
«____»_____ 2020 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

«Інформаційно-орієнтований підхід забезпечення безпеки даних у хмарному середовищі»

Освітньо-кваліфікаційний рівень «Магістр»
Спеціальність 123 «Комп'ютерна інженерія»

Науковий керівник роботи: _____ Нестеров М. В.
(підпис) (ініціали, прізвище)

Консультант з охорони праці: _____ Критська Я. О.
(підпис) (ініціали, прізвище)

Студент: _____ Циганок Ю. С.
(підпис) (ініціали, прізвище)

Група: КІ-18 зм

Сєвєродонецьк – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Спеціальність 123 «Комп'ютерна інженерія»

«**ЗАТВЕРДЖУЮ**»
Т.в.о завідувача кафедри
комп'ютерних наук та інженерії
к.т.н, доц. Сафонова С. О.

"____" _____ 2020 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Циганок Юлії Сргіївні

(прізвище, ім'я, по-батькові)

1. **Тема проекту (роботи):** «Інформаційно-орієнтований підхід забезпечення безпеки даних у хмарному середовищі» затверджена наказом по університету № 136/15.15 від «11» жовтня 2019 р.

2. Строк здачі студентом закінченого проекту (роботи): 10.01.2020 р.

3. Вихідні дані проекту (роботи): матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

1. Дослідження традиційних методів забезпечення безпеки даних в хмарному середовищі;
2. Дослідження існуючих концепцій в області інформаційно-орієнтованої безпеки;
3 Розроблення концептуальної основи підходу до забезпечення безпеки даних;
4. Розроблення моделі інформаційно- орієнтованої безпеки даних;
5. Розроблення прикладних модулів на основі моделі
6. Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точною назвою обов'язкових креслень):

електронні плакати

6. Консультанти роботи, з вказівкою розділів, що до них відносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основна частина	Нестеров М. В.		
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я. О.		

7. Дата видачі завдання: 11.10.2019 р.

Керівник _____ Нестеров М. В.

(підпис)

Завдання до виконання прийняв _____ Циганок Ю. С.

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітки
1.	Отримання завдання, збір матеріалів	11.10.19- 24.10.19	
2.	Дослідження традиційних методів забезпечення безпеки даних в хмарному середовищі	25.10.19 –28.10.19	
3.	Дослідження існуючих концепцій і результатів в області інформаційно-орієнтованої безпеки	29.10.19 – 28.11.19	
4.	Розроблення концептуальної основи підходу до забезпечення безпеки даних	28.11.19 –31.12.19	
5.	Розроблення моделі інформаційно-орієнтованої безпеки даних та прикладних модулів	03.01.20 – 04.01.20	
6.	Оформлення пояснювальної записки	05.01.20 – 08.01.20	
7.	Підготовка та подання магістерської роботи до захисту	09.01.20 – 10.01.20	

Студент _____

(підпис)

Науковий керівник _____

(підпис)

АНОТАЦІЯ

Циганок Ю.С. Інформаційно-орієнтований підхід забезпечення безпеки даних у хмарному середовищі.

В роботі розглянуто інформаційно-орієнтований підхід забезпечення безпеки даних у хмарному середовищі. Виконано дослідження традиційних методів забезпечення безпеки у хмарному середовищі, існуючих концепцій, характеристик та критеріїв підходу для досягнення максимальної ефективності. Розроблено концептуальні основи інформаційно-орієнтованого підходу забезпечення безпеки даних у хмарному середовищі. Досліджено та обрано алгоритмів шифрування та підтримки цілісності даних, алгоритми забезпечення контролю доступу та перевірки аутентифікації. Розроблено складову частину інформаційно-орієнтовного підходу, програму тестування клієнт-серверної моделі, що імітує хмарне середовище.

Ключові слова: хмарне середовище, інформаційно-орієнтована безпека даних, контроль доступу, перевірка цілісності, автентичність.

АННОТАЦИЯ

Цыганок Ю.С. Информационно-ориентированный подход обеспечения безопасности данных в облачной среде.

В работе рассмотрен информационно-ориентированный подход обеспечения безопасности данных в облачной среде. Выполнены исследования традиционных методов обеспечения безопасности в облачной среде, существующих концепций, характеристик и критериев подхода для достижения максимальной эффективности. Определен алгоритмы шифрования и поддержки целостности данных, обеспечения контроля доступа и проверки подлинности. Разработана составная часть информационно-ориентированного подхода для шифрования данных, программа тестирования клиент-серверной модели, имитирующей облачную среду.

Ключевые слова: облачная среда, информационно-ориентированная безопасность данных, контроль доступа, проверка целостности, подлинность.

THE ABSTRACT

Tsyganok Y. S. An information-oriented approach to ensuring data security in a cloud environment.

The paper considers an information-oriented approach to data security in a cloud environment. Conducted research on traditional methods of security in a cloud environment, research on existing concepts focused on information security, defining the characteristics and criteria of the approach for maximum efficiency. Conceptual bases of information-oriented approach of data security in the cloud environment are developed. The algorithms of data encryption and data integrity support, algorithms for access control and authentication are investigated and selected.

An integral part of the information-oriented approach has been developed that provides secure search in encrypted data, a program for testing a client-server model that simulates a cloud environment.

Keywords: cloud environment, information-oriented data security, access control, integrity checking, authentication.

ЗМІСТ

ВСТУП

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

Software as a Service (SaaS) – модель, в якій споживачеві надається можливість використання прикладного програмного забезпечення провайдера, який працює в хмарної інфраструктурі і доступного з різних клієнтських пристроїв або за допомогою тонкого клієнта, наприклад, з браузера (наприклад, веб-пошта) або за допомогою інтерфейсу програми. Контроль і управління фізичною і віртуальною інфраструктурою хмари, в тому числі мережі, серверів, операційних систем, зберігання, або навіть індивідуальних можливостей додатка (за винятком обмеженого набору призначених для користувача налаштувань конфігурації програми) здійснюється хмарним провайдером.

Хмара є гарячою темою для малого бізнесу аж до глобальних підприємств, але залишається широкою концепцією, яка охоплює велику кількість Інтернет-території. Коли розглядається питання про перехід бізнесу на хмару, будь то для додатків чи розгортання інфраструктури, важливіше, ніж будь-коли, зрозуміти відмінності та переваги різних хмарних служб.

Програмне забезпечення як сервіс (SaaS) швидко стало привабливою альтернативою локальному розгортанню, і в багатьох категоріях воно вже перевершило місцеві місця за новими продажами. Це дослідження кількісно визначає поточні інвестиційні тенденції для SaaS та визначає вигоди, що сприяють компаніям розширювати свої впровадження SaaS. Він також визначає, які з найрізноманітніших SaaS-прикладів сприяють прийняттю, та оцінює тенденції щодо прийняття та інвестицій за розмірами та географією організації. Нарешті, ми вивчаємо позитивний досвід ROI та TCO у приймаючих та закінчуємо практичними порадами для тих, хто розглядає можливість впровадження рішень SaaS.

Модель програмного забезпечення як сервісу (SaaS) продовжує набирати «оберти» у всіх куточках ділового світу, і це є зрозумілою причиною. Також відоме як програмне забезпечення на замовлення, розміщене програмне забезпечення або веб-програмне забезпечення, SaaS відмовляється від традиційних підходів до встановлення, обслуговування та управління на користь доставки хмарних додатків через Інтернет. Завдяки SaaS, партнери

постачальників послуг беруть на себе відповідальність за безпеку, доступність та продуктивність системи.

Метою роботи є підвищення ефективності безпеки даних у хмарному середовищі.

Об’єкт дослідження – хмарні середовища.

Предмет дослідження – методи захисту даних у хмарних середовищах.

Методи дослідження. Аналіз існуючих традиційних підходів методів захисту веб-сервісів та концептуальних складових інформаційної безпеки. При формалізації задачі дослідження використано модель Software as a Service (SaaS).

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

− дослідження традиційних методів забезпечення безпеки у хмарному середовищі;

− дослідження існуючих концепцій орієнтованих на безпеку інформації, визначення характеристик та критеріїв підходу для досягнення максимальної ефективності;

− розроблення концептуальної основи інформаційно-орієнтованого підходу;

− дослідження та вибір алгоритмів шифрування даних та підтримки цілісності;

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

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

− тестування клієнт-серверної моделі, що імітує хмарне середовище.

Методи дослідження – емпіричний аналіз існуючих традиційних підходів безпеки хмарних обчислень, емпіричний аналіз концептуальних складових інформаційно-орієнтованої безпеки (ІОБ), синтез концептуальної основи ІОБ, формалізація концептуальної основи за результатами дослідження.

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

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

Практична значимість отриманих результатів полягає у тому, що рішення можна використовувати у якості модулів при комерційній реалізації системи, що використовує ОБІ

підхід. Отримане рішення слугуватиме ключовими криптографічними елементами системи. Прикладами таких систем можуть бути клієнт-серверні рішення.

Апробація результатів роботи. Основні результати роботи представлені у наступних публікаціях:

1. Циганок Ю. С. Тенденції та напрямки рішень технології хмарних обчислень / Матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Майбутній науковець-2019» (12 грудня 2019 р.). – Сєвєродонецьк. – С. 158-160.

2. Циганок Ю.С. Забезпечення безпеки даних у хмарних середовищах / Збірник науково-практичних праць V молодіжного форуму «ІТ-Ідея 2019» (6 грудня 2019 р.). – Сєвєродонецьк. – С. 33-36.

Наукова новизна магістерської роботи полягає в подальшому дослідженні створених методів захисту даних у хмарних середовищах. На основі проведених досліджень вироблені рекомендації щодо використання запропонованих методів захисту даних.

Практичне використання. Результати дослідження, запропоновані рішення дозволять підвищити безпеку зберігання даних у хмарному середовищі від несанкціонованого доступу до них, а також можуть бути використані у навчальному процесі кафедри комп’ютерних наук та інженерії при вивченні дисципліни «Захист інформації в комп’ютерних системах».

Структура і обсяг роботи.

Магістерська робота складається зі вступу, 5 розділів, висновків, переліку посилань до розділів з 48 найменувань, додатку на 6 сторінках. Загальний обсяг роботи складає 131 сторінку. Магістерська робота містить 31 рисунок та 13 таблиць.

РОЗДІЛ 1
АНАЛІЗ ЗАХИСТУ ДАНИХ У ХМАРНИХ ТЕХНОЛОГІЯХ

1.1 Теоретичні відомості про моделі хмарних обчислень

Програмне забезпечення як послуга (SaaS) - модель розповсюдження програмного забезпечення, в якій сторонній постачальник розміщує програми та робить їх доступними для клієнтів через Інтернет [1].

SaaS - одна з трьох основних категорій хмарних обчислень, поряд з інфраструктурою як сервіс (IaaS) і платформою як послугою (PaaS).

На рисунку 1.1 наведені варіанти розміщення програмних продуктів у хмарних ресурсах.

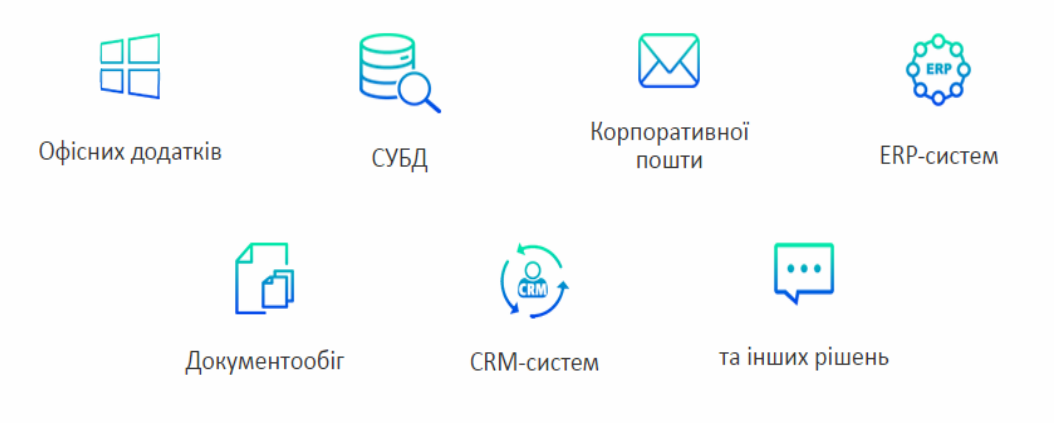


Рисунок 1.1 – Розміщення програмних продуктів у хмарних ресурсах

Хмара є гарячою темою для малого бізнесу аж до глобальних підприємств, але залишається широкою концепцією, яка охоплює велику кількість Інтернет-території. Коли ви починаєте розглянути питання про перехід вашого бізнесу на хмару, будь то для додатків чи розгортання інфраструктури, важливіше, ніж будь-коли, зрозуміти відмінності та переваги різних хмарних служб.

Зазвичай існує три моделі хмарного сервісу для порівняння: Програмне забезпечення як послуга (SaaS), Платформа як послуга (PaaS) та Інфраструктура як послуга (IaaS). Кожен із них має свої переваги, а також недоліки, і необхідно зрозуміти відмінності SaaS, PaaS та IaaS, щоб знати, як найкраще вибрати для своєї організації.

1.2 Основні відмінності розміщення веб-сервісів у хмарних технологіях

Програмне забезпечення як послуга, також відоме як хмарні служби додатків, являє собою найбільш часто використовуваний варіант для підприємств на хмарному ринку. SaaS використовує Інтернет, щоб доставляти додатки, якими керується сторонній постачальник, своїм користувачам. Більшість програм SaaS запускаються безпосередньо через ваш веб-браузер, а це означає, що вони не потребують завантаження або встановлення на стороні клієнта [2].

SaaS надає численні переваги працівникам та компаніям, значно скорочуючи час і гроші, витрачені на стомлюючі завдання, такі як встановлення, управління та оновлення програмного забезпечення. Це звільняє достатньо часу, щоб технічний персонал витрачав на більш нагальні питання та проблеми в організації.

На рисунку 1.2 наведені основні відмінності розміщення інформації в хмарних технологіях.

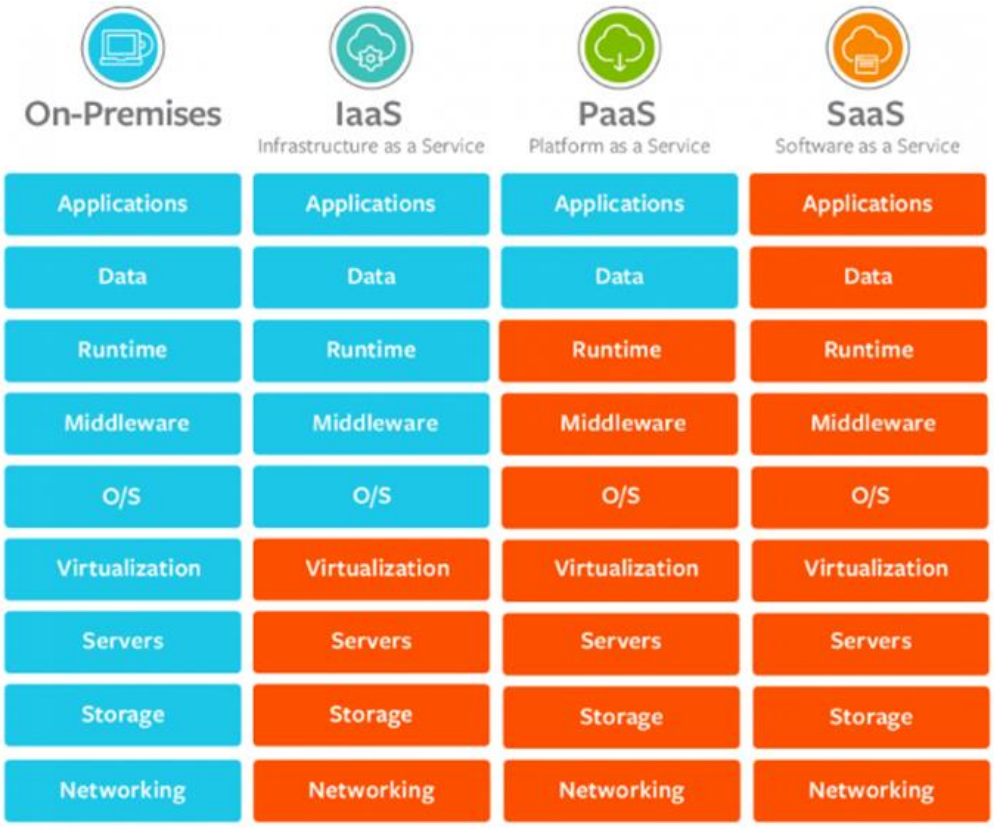


Рисунок 1.2 – Основні відмінності технологій розміщення інформації у хмарних технологіях.

1.3. Модель «Програмне забезпечення як послуга (SaaS)»

Існує кілька способів визначення використання технології SaaS [3]:

– керується з центрального місця;

– розміщений на віддаленому сервері;

– доступний через Інтернет;

– користувачі, які не несуть відповідальності за оновлення обладнання або програмного забезпечення.

На рисунку 1.3 наведена структура послуг моделі SaaS.

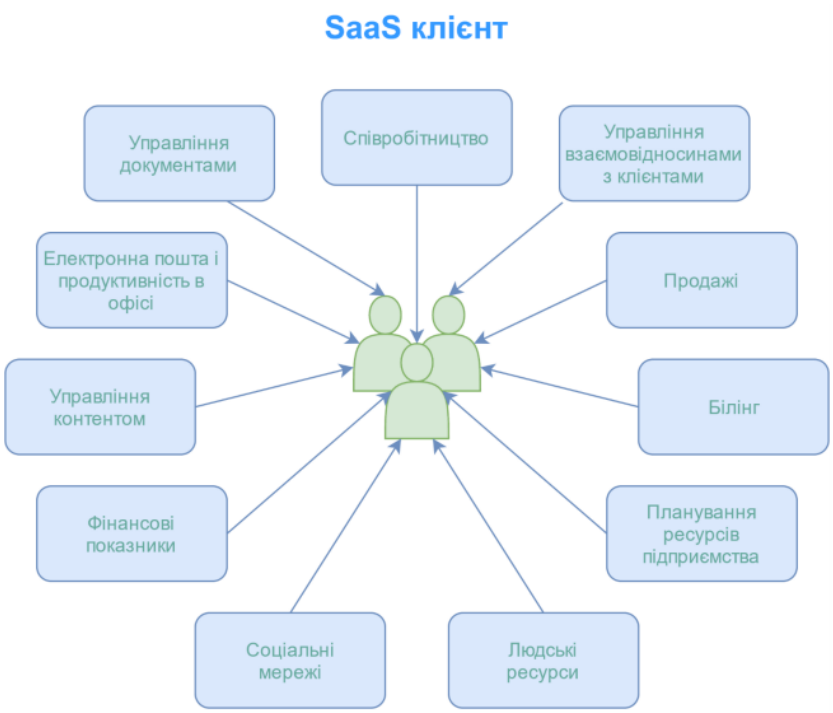


Рисунок 1.3 – Структура моделі SaaS

Послуги, що надаються дата-центрами щодо організації SaaS:

– оренда обчислювальних ресурсів (ядра, пам'ять, диски) у хмарі в дата-центрі з підтвердженим високим рівнем надійності TIER III;

– оренда обчислювальних ресурсів (ядра, пам'ять, диски) в хмарі в європейських дата-центрах

– оренда програмного забезпечення Майкрософт (SPLA/CSP) та хмарні рішення партнерівОренда програмного забезпечення в рамках програм Service Provider License Agreement (SPLA) та Cloud Solution Provider (CSP), дає унікальну можливість користуватися

завжди актуальною версію продуктів Microsoft. Використовуючи ліцензії за принципом «плати, якщо користуєшся» (pay as you use) витрати на ліцензії, можуть бути до 25% менше в порівнянні з іншими ліцензіями, наприклад з OLP.

Разом з позитивними моментами, слід зауважити на наступні обмеження використання SaaS:

1. Взаємодія. Інтеграція з існуючими програмами та послугами може стати головною проблемою, якщо програма SaaS не розроблена таким чином, щоб відповідати відкритим стандартам інтеграції. У цьому випадку організаціям, можливо, доведеться розробити власні інтеграційні системи або зменшити залежності від сервісів SaaS, що може бути не завжди можливим.

2. Блокування провайдера. Постачальники можуть полегшити приєднання до послуги та важко вийти з неї. Наприклад, дані можуть не переноситись технічно чи економічно за допомогою програм SaaS від інших постачальників, не несучи значних витрат або інженерно-технічної роботи. Не кожен постачальник дотримується стандартних API, протоколів та інструментів, але функції можуть бути необхідними для певних бізнес-завдань.

3. Відсутність інтеграційної підтримки. Багато організацій потребують глибокої інтеграції з локальними додатками, даними та послугами. Продавець SaaS може запропонувати обмежену підтримку в цьому плані, змушуючи організації вкладати внутрішні ресурси в розробку та управління інтеграцією. Складність інтеграції може додатково обмежувати використання програми SaaS або інших залежних сервісів.

4. Безпека даних. Великі обсяги даних, можливо, доведеться обмінятись в резервних центрах обробки даних програм SaaS, щоб виконати необхідну функціональність програмного забезпечення. Передача конфіденційної ділової інформації службі SaaS, яка базується на хмарному просторі, може призвести до погіршення безпеки та дотримання, крім значних витрат на міграцію великих навантажень даних.

5. Настроювання. Програми SaaS пропонують мінімальні можливості налаштування. Оскільки рішення, яке відповідає всім розмірам, не існує, користувачі можуть бути обмежені певними функціональними можливостями, продуктивністю та інтеграцією, як пропонує постачальник. На відміну від них, локальні рішення, що постачаються з декількома наборами для розробки програмного забезпечення (SDK), пропонують високу ступінь можливостей налаштування.

6. Відсутність контролю. Рішення SaaS включає передачу контролю сторонньому постачальнику послуг. Ці елементи керування не обмежуються програмним забезпеченням - щодо версії, оновлень або зовнішнього вигляду - а також даними та керуванням. Таким

чином, клієнтам може знадобитися переосмислити свої моделі захисту даних та управління, щоб відповідати особливостям та функціональності послуги SaaS.

7. Обмеження функції. Оскільки програми SaaS часто надходять у стандартизованому вигляді, вибір функцій може бути компрометуючим компромісом щодо безпеки, вартості, продуктивності чи іншої організаційної політики. Крім того, проблеми з блокуванням, витратами чи безпекою постачальника можуть означати, що в майбутньому не можливо переключити постачальників чи послуг, щоб вони обслуговували нові вимоги до функцій.

7. Продуктивність та простої. Оскільки постачальник контролює та керує послугою SaaS, ваші клієнти тепер залежать від постачальників, щоб підтримувати безпеку та продуктивність послуги. Планові та непланові послуги з обслуговування, кібератаки чи проблеми з мережею можуть вплинути на ефективність програми SaaS, незважаючи на захист відповідного рівня угод про рівень обслуговування (SLA).

Слід зауважити, що модель SaaS використовують всесвітньо відомі компанії такі як, Google GSuite (Apps), Dropbox, Salesforce, Cisco WebEx, SAP Concur та GoToMeeting.

1.4. Модель «Платформа як послуга (PaaS)»

Послуги хмарної платформи, також відомі як Платформа як послуга (PaaS), надають хмарні компоненти певному програмному забезпеченню, використовуючи в основному для додатків. PaaS пропонує рамку для розробників, яку вони можуть будувати та використовувати для створення спеціальних додатків. Усіма серверами, сховищами та мережами може керуватися підприємство або сторонній постачальник, тоді як розробники можуть підтримувати управління додатками [4].

Модель доставки PaaS схожа на SaaS, за винятком того, що замість доставки програмного забезпечення через Інтернет, PaaS пропонує платформу для створення програмного забезпечення. Ця платформа надається через Інтернет, що дає розробникам свободу сконцентруватися на створенні програмного забезпечення, не турбуючись про операційні системи, оновлення програмного забезпечення, сховище чи інфраструктуру.

PaaS дозволяє бізнесу розробляти та створювати додатки, вбудовані в PaaS за допомогою спеціальних програмних компонентів. Ці програми, які іноді називають середнім програмним забезпеченням, є масштабованими та широко доступними, оскільки вони набувають певних характеристик хмари.

На рисунку 1.4 наведена структура моделі PaaS.

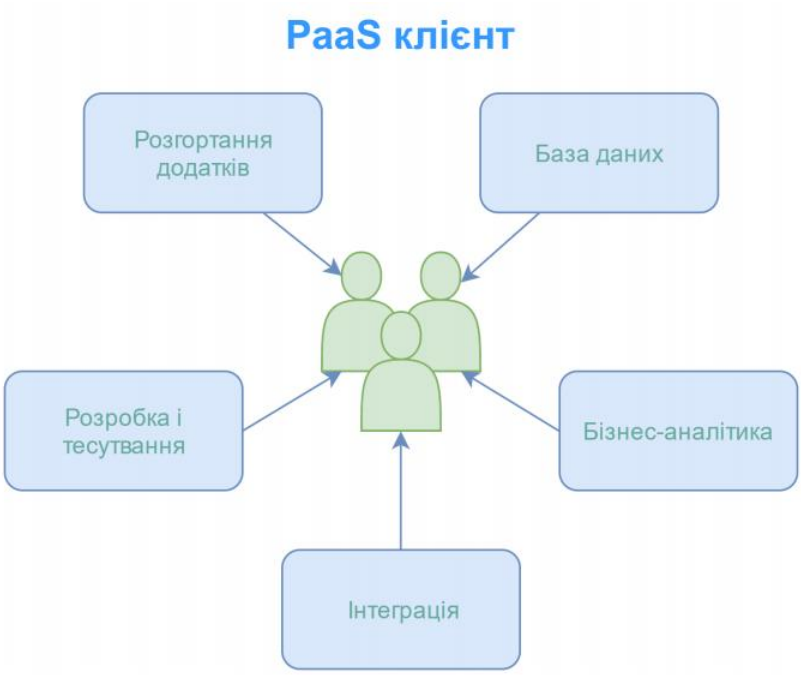


Рисунок 1.4 – Структура моделі PaaS

Незалежно від розміру компанії, використання PaaS пропонує численні переваги, зокрема:

– проста, економічно ефективна розробка та розгортання додатків;

– масштабованість та доступність;

– розробники можуть налаштовувати додатки без головного болю щодо підтримки програмного забезпечення;

– значне зменшення необхідної кількості кодування;

– автоматизація бізнес-політики;

– легка міграція до гібридної моделі.

PaaS має багато характеристик, які визначають його як хмарну послугу, включаючи:

– базується на технології віртуалізації, тому ресурси можуть бути легко зменшені вгору або вниз по мірі зміни вашого бізнесу

– надає різноманітні послуги для розробки, тестування та розгортання додатків

– доступний для численних користувачів через один і той же додаток для розробки

– інтегрує веб-сервіси та бази даних

Використання PaaS вигідне, іноді навіть необхідне, в декількох ситуаціях. Наприклад, PaaS може впорядкувати робочі процеси, коли декілька розробників працюють над одним проектом розробки. Якщо повинні бути включені інші постачальники, PaaS може забезпечити велику швидкість і гнучкість у всьому процесі. PaaS особливо корисний, якщо

вам потрібно створити спеціалізовані програми. Цей хмарний сервіс також може значно знизити витрати і може спростити деякі проблеми, які виникають, якщо ви швидко розвиваєте або розгортаєте додаток.

Разом з позитивними моментами слід вказати на деякі обмеження використання моделі PaaS:

1. Безпека даних. Організації можуть запускати власні програми та сервіси, використовуючи рішення PaaS, однак дані, що знаходяться на сторонніх, керованих постачальниками хмарних серверах, становлять ризики та проблеми безпеки. Ваші параметри безпеки можуть бути обмеженими, оскільки клієнти можуть не мати можливості розгорнути сервіси з певною політикою хостингу.

2. Інтеграції. Збільшується складність підключення даних, що зберігаються в локальному центрі обробки даних, або в хмарі поза приміщенням, що може вплинути на те, які додатки та послуги можна прийняти за допомогою PaaS. Особливо, коли не кожен компонент застарілої IT-системи побудований для хмари, інтеграція з існуючими сервісами та інфраструктурою може стати проблемою.

3. Блокування провайдера. Бізнес та технічні вимоги, які визначають рішення щодо конкретного рішення PaaS, можуть не застосовуватися в майбутньому. Якщо постачальник не запропонував зручну міграційну політику, перехід на альтернативні варіанти PaaS може виявитися неможливим без впливу на бізнес.

4. Настроювання застарілих систем. PaaS може бути не підключенням до існуючих застарілих додатків і служб. Натомість для застарілих систем для роботи зі службою PaaS може знадобитися кілька налаштувань та змін конфігурації. Отримане в результаті налаштування може спричинити складну IT-систему, яка може взагалі обмежити вартість інвестицій PaaS.

5. Питання виконання. Окрім обмежень, пов’язаних із конкретними програмами та послугами, рішення PaaS можуть не бути оптимізовані під мову та рамки на ваш вибір. Конкретні версії фреймворку можуть бути недоступними або працювати оптимально за допомогою послуги PaaS. Клієнти можуть не мати можливості розробляти власні залежності з платформою.

6. Оперативні обмеження. Індивідуальні хмарні операції з робочими процесами автоматизації управління можуть не застосовуватися до рішень PaaS, оскільки платформа має тенденцію обмежувати експлуатаційні можливості для кінцевих користувачів. Хоча це покликане зменшити експлуатаційне навантаження на кінцевих користувачів, втрата оперативного контролю може вплинути на керування, оснащення та експлуатацію рішень PaaS.

Наведемо приклади використання PaaS такими популярними компаніями як AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine і OpenShift.

1.5 Модель «Інфраструктура як послуга (IaaS)»

Послуги хмарної інфраструктури, відомі як інфраструктура як послуга (IaaS), складаються з високомасштабованих та автоматизованих обчислювальних ресурсів. IaaS - це повністю самообслуговування доступу та моніторингу комп'ютерів, мереж, зберігання та інших послуг. IaaS дозволяє підприємствам купувати ресурси за потребою та за потребою, а не купувати обладнання безпосередньо [5].

IaaS забезпечує інфраструктуру хмарних обчислень, включаючи сервери, мережу, операційні системи та сховище, за допомогою технології віртуалізації. Ці хмарні сервери зазвичай надаються організації через інформаційну панель або API, що дає клієнтам IaaS повний контроль над усією інфраструктурою. IaaS надає ті самі технології та можливості, що і традиційний центр обробки даних, не потребуючи фізичного обслуговування або управління ним. Клієнти IaaS все ще можуть отримувати прямий доступ до своїх серверів та сховища, але все це передається аутсорсингом через «віртуальний центр даних» у хмарі.

На рисунку 1.5 наведена структура моделі IaaS.

Рисунок 1.5 – Структура моделі IaaS

На відміну від SaaS або PaaS, клієнти IaaS відповідають за управління такими аспектами, як програми, час виконання, ОС, проміжне програмне забезпечення та дані.

Однак провайдери IaaS управляють серверами, жорсткими дисками, мережами, віртуалізацією та зберіганням даних. Деякі провайдери навіть пропонують більше послуг поза рівнем віртуалізації, таких як бази даних або черга повідомлень.

Модель IaaS пропонує безліч переваг, зокрема:

– найбільш гнучка та високомасштабована модель хмарних обчислень;

– легке автоматизоване розгортання накопичувачів, мереж, серверів та процесорної потужності;

– купівля обладнання може бути заснована на споживанні;

– клієнти зберігають повний контроль над своєю інфраструктурою;

– ресурси можна придбати за потребою;

Характеристики, що визначають IaaS, включають:

– ресурси доступні як послуга;

– вартість варіюється в залежності від споживання;

– послуги високо масштабовані;

– кілька користувачів на одному обладнання;

– організація зберігає повний контроль над інфраструктурою;

– динамічний і гнучкий сервіс, що змінюєтьс.

Як і у SaaS та PaaS, існують конкретні ситуації, коли IaaS є найбільш вигідним.

Стартапи та невеликі компанії можуть віддавати перевагу IaaS, щоб не витрачати час і гроші на придбання та створення апаратного та програмного забезпечення. Більші компанії можуть вважати за краще зберігати повний контроль над своїми додатками та інфраструктурою, але вони хочуть придбати лише те, що вони насправді споживають або потребують. Компанії, що відчувають стрімкий ріст, як масштабність IaaS, і вони можуть легко змінювати конкретне обладнання та програмне забезпечення у міру розвитку своїх потреб. У будь-який час ви не впевнені в вимогах нової програми, IaaS пропонує велику гнучкість та масштабованість.

Багато моделей, пов’язаних із моделями SaaS та PaaS - такі як безпека даних, перевищення витрат, блокування постачальників та проблеми налаштування - також застосовуються до моделі IaaS.

Конкретні обмеження щодо IaaS включають:

1. Безпека. Поки замовник контролює додатки, дані, проміжне програмне забезпечення та платформу ОС, загрози безпеці все ще можна отримувати від хоста або інших віртуальних машин (VM). Внутрішня загроза або вразливості системи можуть піддавати передачу даних між хост-інфраструктурою та VM стороннім особам.

2. Спадкові системи, що працюють у хмарі. Незважаючи на те, що клієнти можуть запускати застарілі програми у хмарі, інфраструктура може не бути розроблена для надання конкретних елементів управління для захисту застарілих програм. Перед міграцією їх у хмару може знадобитися незначне вдосконалення застарілих додатків, що може призвести до нових проблем безпеки, якщо вони не будуть адекватно перевірені на безпеку та ефективність в системах IaaS.

3. Внутрішні ресурси та навчання. Для робочої сили можуть знадобитися додаткові ресурси та навчання, щоб навчитися ефективно керувати інфраструктурою. Клієнти будуть нести відповідальність за безпеку даних, резервне копіювання та безперервність бізнесу. Через недостатній контроль над інфраструктурою, моніторинг та управління ресурсами можуть бути утрудненими без відповідної підготовки та ресурсів, що є в приміщенні.

4. Захист для багатьох орендарів. Оскільки апаратні ресурси динамічно розподіляються між користувачами, коли вони стають доступними, постачальник зобов'язаний переконатися, що інші клієнти не можуть отримати доступ до даних, переданих у сховища попередніх клієнтів. Аналогічно, клієнти повинні розраховувати на постачальника, щоб забезпечити адекватну ізоляцію VM в межах багатосторонньої хмарної архітектури.

Популярні приклади використання моделі IaaS включають такі компанії як DigitalOcean, Linode, Rackspace, веб-сервіси Amazon (AWS), Cisco Metacloud, Microsoft Azure та Google Compute Engine (GCE).

1.6. Приклади використання моделей SaaS, PaaS та IaaS

Кожна хмарна модель пропонує конкретні особливості та функціональні можливості, і важливо, щоб організація при виборі моделі зрозуміла відмінності. Незалежно від того, чи потрібне хмарне програмне забезпечення для варіантів зберігання, гладка платформа, яка дозволяє створювати спеціалізовані програми або повний контроль над усією інфраструктурою без фізичного обслуговування, для є хмарний сервіс. Незалежно від того, який варіант обираємо, перехід до хмари - це майбутнє бізнесу та технологій.

На рисунку 1.6 наведена узагальнена таблиця використання відомими світовими лідерами IT-індустрії моделей SaaS, PaaS та IaaS.

Platform Type	Common Examples
SaaS	Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting
PaaS	AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift
IaaS	DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

Рисунок 1.6 – Використання моделей відомим фірмами

1.7 Приклади популярного програмного забезпечення як сервісу (SaaS)

Програмне забезпечення як послуга (SaaS) швидко стало привабливою альтернативою локальному розгортанню, і в багатьох категоріях воно вже перевершило місцеві за розміром нових продажів. Вперше відтоді, як ми розпочали щорічне опитування тенденцій технологій, більшість (60%) компаній, що реагують, зараз повідомляють про прийняття хоча б деяких SaaS, і інвестиції залишаються високими [6].

«При правильному розгортанні SaaS обіцяє зменшення інфраструктури, швидкість впровадження та порівняний досвід клієнтів. Це також може заощадити на перших витратах», – сказав Девід Вагнер, віце-президент з досліджень з питань комп'ютерної економіки, Ірвін, Каліфорнія – «Не дивно, що компанії здійснюють перехід».

Повне дослідження кількісно визначає поточні інвестиційні тенденції для Software-as-a-Service та визначає вигоди, що сприяють компаніям розширювати свої впровадження SaaS. Він також визначає, які з найрізноманітніших SaaS-прикладів сприяють прийняттю, а також оцінює тенденції прийняття та інвестиції за розмірами та географією організації. Нарешті, ми вивчаємо позитивний досвід ROI та TCO у приймаючих та закінчуємо практичними порадами для тих, хто розглядає можливість впровадження рішень SaaS.

На рисунку 1.7 «Тенденції прийняття SaaS та досвід клієнтів» за даними журналу Computer Economics (2016 р.) показано відсоток організацій на кожному з п’яти етапів прийняття. Тридцять шість відсотків повідомляють, що у них є деякі SaaS і мають намір збільшити інвестиції. Ще 24% звіту мають SaaS, але не планують більше інвестувати. Коли 44% організацій здійснюють або збільшують свої інвестиції в SaaS, рівень прийняття продовжить зростати в осяжному майбутньому [7].

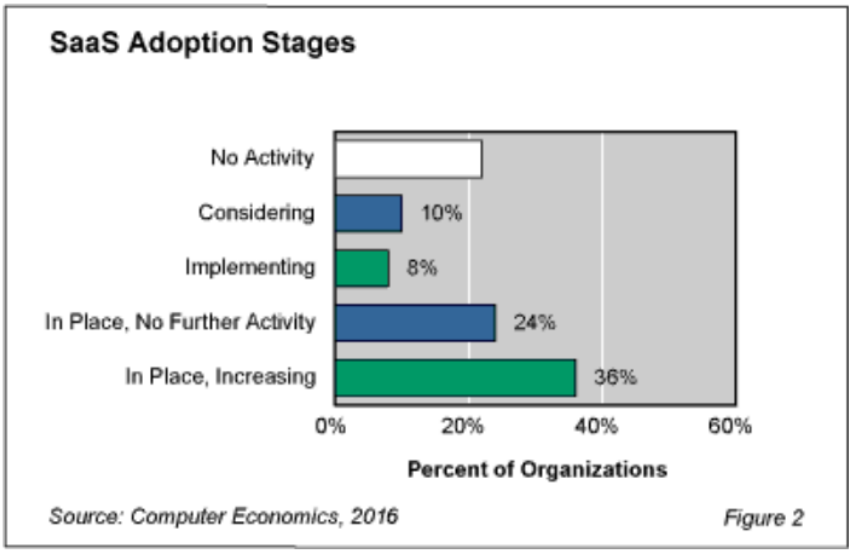


Рисунок 1.7 – Тенденції прийняття SaaS

Повне дослідження кількісно визначає поточні інвестиційні тенденції для Software-as-a-Service та визначає вигоди, що сприяють компаніям розширювати свої впровадження SaaS. Він також визначає, які з найрізноманітніших SaaS-прикладів сприяють прийняттю, а також оцінює тенденції прийняття та інвестиції за розмірами та географією організації. Нарешті, ми вивчаємо позитивний досвід ROI та TCO у приймаючих та закінчуємо практичними порадами для тих, хто розглядає можливість впровадження рішень SaaS.

Організації будь-яких форм і розмірів сприймають філософію SaaS як альтернативу локальному апаратному та програмному застосуванню. Провайдер метричних систем управління інформацією Computer Economics повідомляє, що 60 % усіх компаній зараз інтегрували принаймні деякі рішення SaaS у свій бізнес, 36 % мають намір збільшити свої інвестиції в наступні місяці.

1.8 Плюси і мінуси SaaS як сервісу

SaaS - це природна придатність для підприємств, які мають намір скоротити обов'язки та витрати на IT. В середньому фірми, які переходять на програмне забезпечення як послугу з передплатників із встановлення великої капітальної інфраструктури, технічного обслуговування та модернізації, користуються зниженням витрат на IT більш ніж на 15 %, згідно з даними, зібраними компанією Computer World.

SaaS особливо добре підходить для малого бізнесу. Замість того, щоб вкладати кошти в додаткові потужності власного сервера та ліцензії на програмне забезпечення, компанії просто можуть щомісяця коригувати своє Програмне забезпечення як підписку на послугу, збільшуючи вимоги споживання вгору та вниз, виходячи з потреб проекту та інших змінних. Також збільшується пропускна здатність людини: IT-співробітники компанії звільняються від завдань, пов'язаних з локальним обладнанням та програмним забезпеченням, що дозволяє їм вирішувати проекти, які є більш важливими для майбутнього зростання компанії. Оскільки IT-інфраструктура знаходиться в центрі даних постачальника послуг, ваша організація може негайно відновитись і запуститись у разі відключення послуги або більш різкого збою.

Звичайно, ніщо не є ідеальним, і SaaS не є винятком. Компанії, які приймають кілька службових програм як програмного забезпечення або планують підключити розміщене програмне забезпечення до існуючих локальних програм, можуть зіткнутися з проблемами інтеграції програм.

1.9 Хмарні обчислення: труднощі та проблеми

Безпека - це ще одна спільна проблема для підприємств, що замислюються над варіантами SaaS: Щоразу, коли конфіденційні дані компанії та бізнес-процеси доручаються стороннім постачальникам послуг, такі питання, як управління особистістю та доступом, повинні вирішуватися. Підприємства також повинні враховувати урядові норми дотримання правил, які притаманні зберіганню даних клієнтів у віддаленому центрі обробки даних.

У послугах хмарних обчислень клієнти стурбовані переміщенням своїх конфіденційних даних і додатків зі своїх приватних обчислювальних середовищ в хмарну середовище, яка спільно використовується різними клієнтами і яке зазвичай доступне через загальнодоступну мережу. Опитування, проведене у вересні 2016 року Міжнародною корпорацією даних (IDC), показує кілька проблем, які стосуються клієнтів хмарних обчислень.

Як показано на рисунку 1.9, безпека визначається як найвища проблема [8].

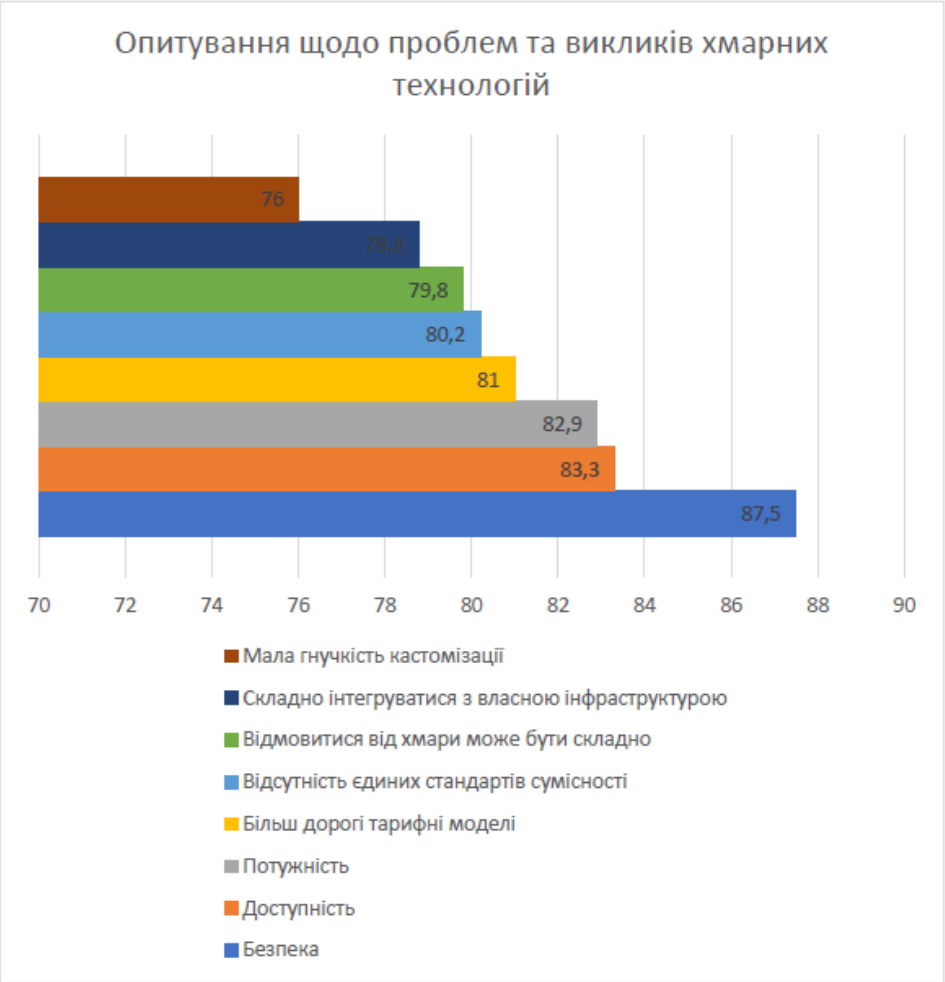


Рисунок 1.8 – Діаграма проблем та викликів хмарних технологій

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

− веб-додатки та служби є найбільш часто використовуваними технологіями для доступу до хмарних обчислень.

− віртуалізація є основною технологією надання хмарних обчислень. Обидві SaaS і PaaS засновані на віртуалізації інфраструктури, що надається на рівні IaaS.

− криптографічні методи в даний час є найбільш поширеними методами для досягнення задовільного рівня вимог безпеки для хмарних обчислень.

Отже, будь-які відомі вразливості вищезазначених трьох основних технологічних компонентів можна розглядати як вразливості хмарних обчислювальних систем. Наприклад, протокол HTTP, який використовується в веб-технологіях, піддається перехопленню сеансів та атакам сеансового рівня. Тому хмарні обчислювальні системи вразливі для такого роду

атак і повинні подолати цю слабкість. Віртуалізація - ще одна вразлива проблема. Зловмисник може прорватися через віртуальний поділ і отримати доступ до даних і ресурсів або пасивно, спостерігаючи за даними, або активно, змінюючи дані і конфігурації. Крім того, існують різні інші можливі вразливості, які можуть бути присутніми в хмарних обчислювальних системах, і вони пов'язані з його інфраструктурою та середовищем. Оскільки хмарні послуги зазвичай надаються через Інтернет, все очікувані проблеми, пов'язані з Інтернетом, також пов'язані з хмарними обчисленнями. Вразливості в операційних системах і інших програмах, реалізованих програмно і встановлених в хмарну інфраструктуру, також можна розглядати як такі, що пов'язані з уразливостями хмарних обчислень.

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

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

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

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

Можливість витоку даних: інша проблема вразливості, пов'язана з образами шаблонів віртуальних машин, полягає в тому, що хмарні провайдери можуть використовувати шаблони, створені іншими клієнтами для нових клієнтів. Ці шаблони можуть містити

секретні бекдори, створені зловмисником, що прикидатися клієнтом, і дозволяють зловмисникові отримати доступ до віртуальних машин інших клієнтів.

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

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

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

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

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

Необхідність взаємних можливостей проведення аудиту, підзвітності і надійність: довіра повинна будуватися між провайдерами і клієнтами в хмарному середовищі. Прозорість через високий рівень можливостей проведення аудиту і підзвітності має важливе значення для створення довірливих відносин. Довірливі відносини будуть більш складними, якщо хмарні провайдери делегують деякі з хмарних сервісів субконтрактам. Клієнти хмарних обчислень повинні знати, чи є які-небудь субконтракти, які також можуть відповідати за їх дані і додатки за хмарою. Наприклад, Linkup надав онлайн-службу зберігання через іншого субпідрядного провайдера під назвою Nirvanix, перш ніж він закрився в результаті втрати значної кількості клієнтських даних. Ймовірно, субпідрядник ніс відповідальність за втрату клієнтських даних.

1.10 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.

1.11 Захист конфіденційності і цілісності даних від провайдерів хмарних обчислень

Конфіденційність і цілісність є важливими вимогами для різних додатків, таких як e-government і EHR (Electronic Health Record). Клієнти хмарних обчислень не тільки турбуються про компрометацію конфіденційності і цілісності своїх даних від можливих зловмисників, а й від потенційних цікавих до цього провайдерів хмарних обчислень. На жаль, порушення безпеки, підраховані в 2011 році і перераховані в [6], показують, що великі компанії, такі як Google, EMC/RSA, Sony, UK National Healthcare System (NHS) і Amazon EC2, всі стикалися з інцидентами з безпекою. В хмарних обчисленнях дані клієнтів передаються стороннім провайдерам, які можуть бути надійними або ненадійними. Термін ненадійний може використовуватися для вказівки того, що хмарному провайдеру не можна повністю довіряти. Наприклад, ненадійні постачальники хмарних послуг можуть не змінювати дані користувачів, але вони можуть пасивно порушувати конфіденційність даних або приховано змінювати протоколи для своєї фінансової вигоди. Іншими словами, сервер провайдера хмарних обчислень можна розглядати як чесний, але допитливий сервер. Отже, він заслуговує на довіру в наданні послуг з точки зору доступності даних, дотримання основних вимог контролю безпеки і обробки чесно дозволених запитів на збереження даних і повернення правильних результатів. Проте, можливі зловмисні дії всередині хмари можуть виконуватися зловмисним адміністратором або співробітником [9] .

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.

На рисунку 1.9 показана базова архітектура шифрування даних для захисту конфіденційності, перш ніж відправляти їх в хмару.

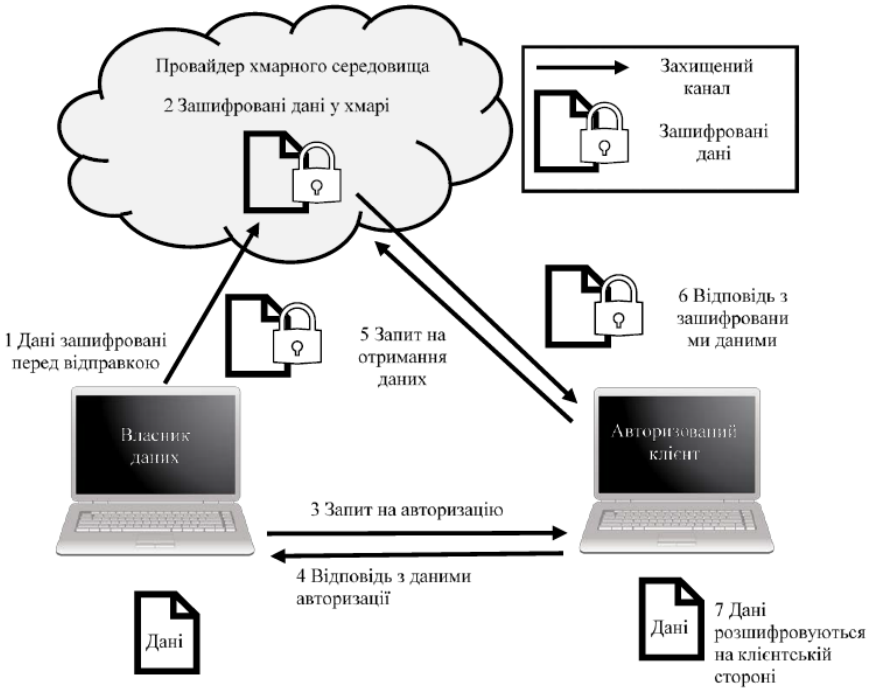


Рисунок 1.9 – Базова архітектура для збереження конфіденційності даних у хмарі

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

і підходи орієнтовані на безпеку інформації можуть стати перспективним рішенням для подолання декількох проблем безпеки хмарних обчислень.

1.12 Криптографічні технології пристосовані до хмарної моделі

Нові технології криптографії необхідні, щоб відповідати хмарній моделі і підвищити безпеку інформації з меншим впливом на зручність її використання. Наприклад, шифрування з можливістю пошуку - це одна з областей, де дослідники розробляють можливість пошуку в зашифрованих даних без їх дешифрування. Лише авторизовані користувачі можуть запитувати і отримувати зашифровані дані, не відкриваючи ніяку приватну інформацію ні про дані, ні про запит, який може містити інформацію про дані. Іншим прикладом є методи шифрування, які дозволяють виконувати обчислення зашифрованих даних без їх дешифрування. В таких методах також захищаються результати обчислення, які зазвичай містять інформацію про захищені дані. Прикладом цих методів є гомоморфне шифрування. Існують і інші методи криптографії, такі як шифрування на основі ідентифікаційної інформації (IBE) і шифрування на основі атрибутів (ABE), які можуть поліпшити управління ключами і контроль доступу до хмарних систем. Чоу, Голле (Chow, Golle) та співавтори вважають, що ці нові криптографічні рішення можуть бути найбільш підходящими інструментами для вирішення декількох проблем безпеки і надання хмарним клієнтам більш ефективного контролю над своїми даними в хмарі, розглядаючи, як ці методи можуть бути поліпшені, щоб адаптувати їх до практичного хмарного середовища [10].

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

TCG починає будувати свої рішення на основі створення стандартного апаратного модуля - Trusted Platform Module (TPM), який виконує основні функції криптографії, такі як хеш-функція і RSA. Ці криптографічні функції необхідні для встановлення надійності в обчислювальному обладнанні. Іншими словами, TCG спрямована на надання стандартних апаратних і програмних технологій для надійних обчислень, включаючи хмарні обчислення.

Тому деякі з рішень безпеки, зокрема ті, які пропонуються для забезпечення віртуальної ізоляції між віртуальними машинами і віртуальною машиною від провайдера сервера, засновані на технологіях ТС. Ці технології як і раніше стикаються з декількома проблемами і не можуть виключно використовуватися для забезпечення універсального рішення всіх проблем з хмарною безпекою. Наприклад, якщо TPM, який є основним компонентом ТС, скомпрометовано, всі рішення буде порушено, як вказав Крістофер Тарновський (Christopher Tarnovsky) в 2010 році. Крім того, концепція ТС не надає користувачам хмарних обчислень достатній контроль над своїми даними з точки зору конфіденційності та політики безпеки.

1.13 Підхід орієнтований на безпеку інформації

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

1.14 Висновки до розділу 1

У першому розділі роботи проведено дослідження та порівняльний аналіз моделей розміщення програмного забезпечення у хмарних середовищах. Наведені переваги, недоліки та варіанти використання моделей відомими світовими лідерами IT-індустрії.

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

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

– надійні обчислення;
– орієнтація на безпеку інформації (ОБІ).

Подальші дослідження направлені на розроблення веб-сервісу з забезпечення безпеки зберігання даних у хмарному середовищі.

1.15 Перелік посилань до розділу 1

1. Что такое модель SaaS, ее преимущества и примеры. [Електронний ресурс]. – Режим доступу: https://www.kasper.by/blog/model-saas/ (дата звернення 19.10.2019).

2. Платформа как услуга. [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/ru-ru/overview/what-is-paas/ (дата звернення 19.10.2019).

3. Хмарна піраміда: SaaS, PaaS, IaaS. [Електронний ресурс]. – Режим доступу: https://gigacloud.ua/blog/navchannja/hmarna-piramida-iaas-paas-i-saas (дата звернення 23.10.2019).

4. The SaaS Business Model Explained Greg Elfrink June 22, 2016. [Електронний ресурс]. – Режим доступу: https://empireflippers.com/saas-business-model-explained/ (дата звернення 24.10.2019).

5. Brian Mackley. SaaS 101: Starting a Software as a Service Business. [Електронний ресурс]. – Режим доступу: https://articles.bplans.com/software-as-a-service-or-saas-101/ (дата звернення 24.10.2019).

6. Базиленко Анна. Усі хмарні сервіси Google об'єднали під спільним брендом Google Cloud. [Електронний ресурс]. – Режим доступу: http://watcher.com.ua/2016/09/30/usi-hmarni-servisy-google-ob-yednaly-pid-spilnym-brendom-google-cloud/ (дата звернення 24.10.2019).

7. Сергій Депутат. Гігабайти в хмарі. Чотири кращих хмарних сервіса для зберігання даних. [Електронний ресурс]. – Режим доступу: https://techno.nv.ua/ukr/it-industry/hihabajti-v-khmari-chotiri-krashchikh-khmarnikh-servisu-dlja-zberihannja-danikh-2448855.html (дата звернення 24.10.2019).

8. Аулов І.Ф. Дослідження моделі загроз ключових систем хмари та пропозиції захисту від них. Східноєвропейський журнал передових технологій. 5/2 (77), 2015. С.4-13.

9. Каблуков О.А. Основні загрози безпеці інформації у віртуальних середовищах і хмарних платформах. [Електронний ресурс]. – Режим доступу: http://stratcom.co.ua/osnovni-zagrozi-bezpetsi-informatsiyi-u-virtualnih-seredovishhah-i-hmarnih-platformah/ (дата звернення 24.10.2019).

10. Популярні хмарні сервіси: особливості роботи та важливі налаштування. [Електронний ресурс]. – Режим доступу: http://gsm-ka.com.ua/ua/populyarnye-oblachnye-servisy-osobennosti-raboty-i-vazhnye-nastroyki/ (дата звернення 24.10.2019).

РОЗДІЛ 2
ІНФОРМАЦІЙНО-ОРІЄНТОВАНИЙ ПІДХІД ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ІНФОРМАЦІЇ У ХМАРНОМУ СЕРЕДОВИЩІ

2.1 Хмарні середовища та безпека зберігання даних

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

Відповідно, з перспективи користувача хмарного сервісу, найбільш цінним активом у хмарному середовищі є його дані, особливо ті дані, що містять інформацію делікатного характеру і вимагають особливого ставлення, а саме: дані урядового характеру, охорони здоров'я та фінансового спектру.

Переваги, що надаються за використання хмарних обчислень, дають користувачам, що раніше розміщували делікатні дані на своїх ПК, більш привабливі перспективи до аутсорсингу своїх даних в хмару. Як і будь-які інші послуги в мережі Інтернет, хмарні сервіси також зазнають атак на системи безпеки.

Компрометація доступності хмарних послуг, як правило, призводить до короткострокових ефектів і пошкодження можуть бути відновлені. Компрометація конфіденційності та приватності даних споживачів хмарних послуг, може привести в свою чергу до довгострокових ефектів і будь-які втрати можуть бути достатньо важким для відновлення.

Наприклад, коли кілька паролів з адміністративних облікових записів UK's National Healthcare System (NHS) було взламано в червні 2011 року, система NHS була закрита органами охорони здоров'я для захисту записів пацієнтів. Це показує, що для таких випадків конфіденційність даних є більш важливою ніж нормальне функціонування системи як такої.

Внутрішні ризики можуть бути від користувачів-зловмисників, які використовують той самий хмарний сервіс, і пом'якшення ризиків, в такому випадку, залежить повністю від хмарного провайдера і виходить з-під контролю власника даних. З точки зору

власників даних, хмарне середовище невидиме і власник даних не впевнений в тому як його/її дані захищені від ризиків пов’язаних з безпекою.

Наприклад, виходячи з природи файлів, їх може бути переміщено через сервіс провайдерів, невідомих для власників даних, або в різних країнах, з різною юрисдикцією щодо конфіденційності даних. Як наслідок з цих проблем, клієнти хмари відчувають обмежений контроль над своїми даними і їм бракує впевненості щодо безпеки даних. З іншої точки зору, провайдери хмарних послуг має надмірний контроль над даними клієнтів, особливо щодо їх безпеки та приватності.

Отже, власники даних стурбовані безпекою та приватністю їх даних і мають бажання зберегти свої дані в безпеці, навіть від провайдерів послег зберігання даних в хмарі. Крім того, вони надають перевагу власноруч управляти політикою безпеки своїх даних в безпечному режимі, як ніби ці дані зберігалися на їх ПК.

Традиційна концепція безпеки зазвичай зосереджується навколо технологій і пристроїв, що використовуються для зберігання та обробки даних. Ця концепція може бути важко адаптованою для забезпечення необхідного відповідного рівня безпеки, особливо для делікатних та конфіденційних даних.

Наукова спільнота нещодавно звернула увагу на питання безпеки та конфеденійності даних в хмарах, запропоновані рішення в основному спрямовані на забезпечення безпеки операційних систем (ОС), що лежать в основі та віртуальних машин (VM), що хостять хмарні сервіси.

Таким чином, більшість рішень як і раніше засновані на традиційній концепції безпеки і в основному фокусуються на будь-який ОС-орієнтованій або VM-орієнтованій безпеці. Деякі з цих рішень засновані на концепті надійних обчислень, який було запропоновано і розроблено групою Trusted Computing Group (TCG).

TCG прагне розробити набір стандартів і технологій, таких як Trusted Platform Module (TPM), які можуть зберігати клієнтські дані і додатки, оброблювані в межах хмарної інфраструктури, який убезпечений навіть від системних адміністраторів хмари. TCG, на основі їх технологій, фокусується на наданні набору інструментів, які можуть бути використані для надання допомоги клієнтам, щоб оцінити надійність провайдерів, стежити за дотриманням політики, а також створювати можливість прозорості щодо фізичного розташування даних в хмарі.

Проте, клієнти хмари повинні спочатку довіряти TCG технологіям з точки зору оцінки надійності провайдерів та якщо TPM, який є основним компонентом щоб побудувати цю довіру, буде скомпрометовано, постраждає все рішення. У 2010 році, Крістофер Тарновський стверджував, що йому вдалося скомпрометувати TPM. Отже,

дослідження і запропоновані рішення на основі на TPM, можливо, варто піддати переоцінці. Навіть якщо TPM та інші інструменти TCG є гарантією безпеки, фокус цих інструментів не на тому щоб надати користувачам бажаний контроль за безпекою та приватністю їх даних. Замість цього, з точки зору клієнта, реалізація концепції ТС дозволяє здійснювати клієнтам моніторинг або аудит операцій, в тому числі над політикою контролю доступу, для серверу через довіренні інструменти, які можуть забезпечити докази відповідності концепції ТС для користувачів.

Нова концепція була запропонована декількома дослідниками для вирішення конкретних питань безпеки хмарних обчислень, переміщаючи фокус з забезпечення безпеки для даних клієнтів на дані, як такі, і вони називають це інформаційно орієнтована безпека. Ця концепція все ще розвивається, і існують різні думки щодо її застосування до моделі хмари.

2.2 Класифікація існуючих рішень

Хмарні обчислення та рішення для зберігання даних надають користувачам і підприємствам різні можливості для зберігання і обробки їх даних у центрах обробки даних сторонніх виробників. Є ряд питань / проблеми, пов'язані з безпекою хмарних обчислень, але ці питання діляться на дві великі категорії: питання безпеки, з якими стикаються під час використання хмарних послуг, які надають програмне забезпечення, платформи, чи інфраструктуру як послуги через використання хмарних технологій і питання безпеки, з якими стикаються їх клієнтів.

У даній роботі, так і з точки зору розуміння концепції ОБІ, існуючі рішення можуть бути класифіковані за двома критеріями [2]:

– класифікація заснована на тому, на якому рівні забезпечується безпека;

– 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 класифікуються як системно-центровані. Якщо рішення сфокусовано на окремому визначеному рівні, його класифіковано відповідно до цього специфічного рівня. Наприклад, рішення спрямовані на поліпшення безпечної ізоляції між віртуальною машиною та гіпервізором можуть бути класифіковані як VM-орієнтовані рішення в області безпеки. З іншого боку, рішення, спрямовані на забезпечення безпеки даних усередині самих даних, як показано на лівій частині рисунку 2.1, класифікуються як інформаційно-орієнтовані підходи.

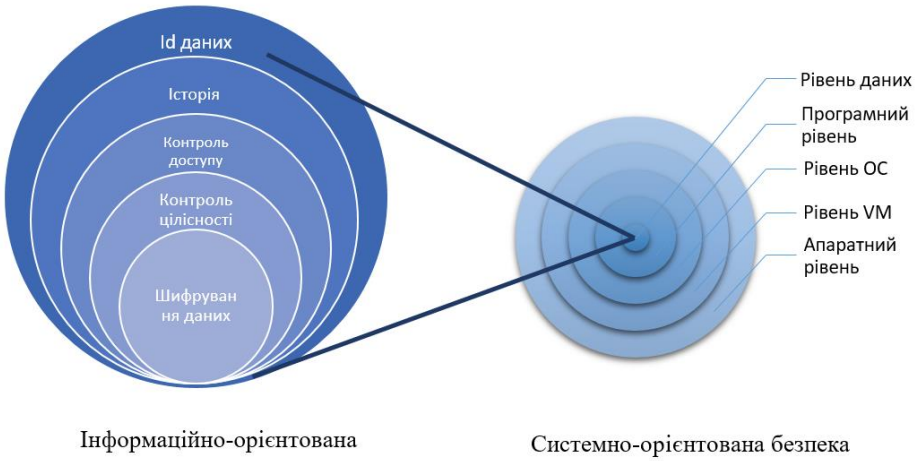


Рисунок 2.1 – Інформаційно-орієнтована та системно-орієнтована моделі

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.

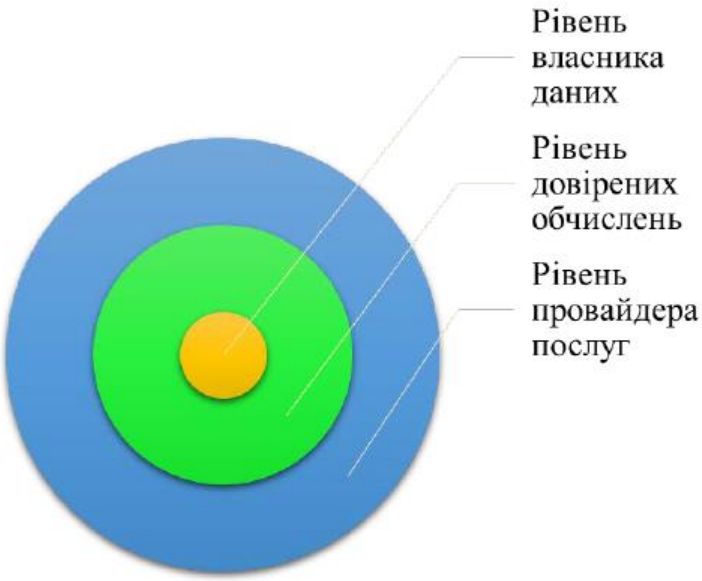


Рисунок 2.2 – Можливі рівні безпеки

Існує три рівні відповідальності безпеки:

− рівень сервіс провайдерів, де забезпечується безпека та здійснюється підтримка хмарних провайдерів;

− рівень довірених обчислень, де забезпечується безпека та організатором виступає третя сторона;

− рівень даних, на якому забезпечується безпека та організаторами виступають власники даних.

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

З точки зору вивчення різноманітності поглядів в розумінні концепції ОБІ (або ОБІ), дослідники погоджуються зосередитися на інформації при розробці необхідного раціонального рівня безпеки. Однак у дослідників різні погляди на те, як забезпечити необхідний рівень. Чи надавати його за межами даних (додатки, обладнання і т. д., як показано в правій частині рисунка 2.1. протягом його життєвого циклу) або всередині даних (як показано в лівій частині того ж рисунка)?

Перші два посилання в списку орієнтовані на забезпечення безпеки за межами даних, оскільки концепція ОБІ застосовується до бізнес-процесів з використанням традиційних обчислювальних систем. Їх мета полягає в тому, що якщо система має кілька рівнів політик безпеки і механізмів, які розробляються незалежно, то система зможе застосувати належний рівень безпеки до інформації відповідно до її класифікації.

При застосуванні концепції ОБІ до хмарної обчислювальної системи мова йде про захист даних від самої хмарної системи. Отже, з одного боку, в деяких дослідженнях концепція ОБІ застосовується до хмари, все ще фокусуючись на забезпеченні належного рівня безпеки за межами даних, і вони намагаються захистити свої механізми безпеки і захищати дані від хмарної системи, яка розміщує ОБІ залежить від технологій ТС (НО) для оцінки надійності середовища, тому для забезпечення безпеки даних потрібен зовнішній захист даних.

В орієнтація на безпеку інформації 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 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 роботі на концептуальному рівні.

2.3 Огляд існуючих ОБІ рішень

На даний момент концепція ОБІ знаходиться на етапі зародження, проте стрімко розвивається. Систематичний перегляд літератури в напрямку пошуку реалізованих систем інформаційно-орієнтованої безпеки не дало плідних результатів. Проте можна відзначити роботу колег з Індії, які практично дослідили інформаційно-орієнтовану систему безпеки на прикладі розподіленої системи охорони здоров'я. Значність результатів дослідження підкріплюється авторитетом міжнародної конференції "Розподілених обчислень та мереж" [3].

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

Далі використовується асиметрична техніка розподілу ключів. Для генерації симетричного ключа використовуючи алгоритм KDF.

Традиційна концепція безпеки зазвичай зосереджується навколо технологій і пристроїв, що використовуються для зберігання та обробки даних. Ця концепція може бути важко адаптованою для забезпечення необхідного відповідного рівня безпеки, особливо для делікатних та конфіденційних даних.

2.4 Характеристики ОБІ

В цьому пункті підхід ОБІ обговорюється більш детально. Деякі з його характеристик виділені та описано для більш чіткого розуміння концептуальних вимог ОБІ. Всі ці характеристики пов’язані з проблемами конфіденційності та безпеки щодо захисту даних клієнтів у хмарних обчисленнях. Традиційно безпека в основному відноситься до вирішення трьох основних проблем: конфіденційності, цілісності та доступності. Як правило, модель CIA використовується в якості скорочення для цих трьох важливих вимог безпеки. З точки зору безпеки даних, вимоги CIA стосуються конфіденційності даних, цілісності даних і доступності даних. У хмарних обчисленнях або будь-яких інших обчислювальних системах, вимоги CIA також застосовуються до системи щодо конфіденційності системи, цілісності системи і доступності системи [4].

2.5 Класифікація даних

Власник даних зазвичай є найкращим суб’єктом для оцінки вимог безпеки своїх власних даних. Наприклад, якщо дані підключені до бізнес-моделі, вимоги до безпеки даних повинні бути результатом аналізу їх використання в бізнес-процесі. Як правило, дані повинні оброблятися відповідно до їх значенням, оскільки наступний принцип безпеки проголошує: «цінність того, що захищається, впливає на заходи, прийняті для його захисту». Отже, дані класифікуються на основі оцінки власника даних та вимог до безпеки. Наприклад, дані можуть бути просто класифіковані як цілком таємні, секретні або конфіденційні [5]. Виходячи з цього, політики контролю доступу та властивості безпеки, необхідні для обробки даних, надаються відповідно до цих вимог безпеки. Класифікація даних на концептуальному рівні може бути представлена як інформація, приєднана до даних, або представлена на вимогу необхідних заходів безпеки, які застосовуються до даних.

2.6 Політики доступу до даних і їх застосування

Політика контролю доступу - це основна функція безпеки, в якій вимоги безпеки задаються для кожного набору даних відповідно до політик безпеки, які можуть включати будь-які потенційні юридичні зобов’язання. Ці політики включають правила і обмеження, які контролюють доступ, використання і потік даних. Основа концепції полягає в контролі того, хто або що може отримати доступ до даних і з якими наданими правами, наприклад,

читати, писати, які відповідають традиційній концепції політики контролю доступу. Питання полягає не тільки у визначенні цих політик, але і в забезпеченні дотримання цих політик безпечним чином, який зберігає конфіденційність користувачів при їх доступі або спільному використанні даних. У підході ОБІ політики контролю доступу та механізм їх застосування визначається власником даних, а хмарна система несе відповідальність за дотримання механізму примусового виконання. Завдання полягає в тому, як політики контролю доступу та їх примусове застосування можуть бути поширеними і легко модифікованими для адаптації до динамічних змін вимог до використання і спільного використання даних в хмарному обчислювальному середовищі. Оскільки у кожного набору даних є свої політики доступу, виникають інші складні проблеми щодо взаємодії різних політик контролю доступу та права власності на новий набір даних, який створюється в хмарі від обробки декількох наборів даних з різними політиками і власниками даних.

2.7 Самозахист

Конфіденційність вмісту набору даних захищена від будь-яких неавторизованих об’єктів, навіть хмарних провайдерів. Щоб самозахистити конфіденційність даних, вони шифруються за допомогою достатньої технології шифрування, і вони залишаються зашифрованими до тих пір, поки вони не будуть переведені в повністю надійний домен і не будуть розшифровані авторизованим користувачем. Тільки авторизовані користувачі можуть мати доступ до секретного ключа, який прикріплений до набору даних, а також захищений. Зашифровані дані упаковуються з їх параметрами безпеки, такими як захищений секретний ключ і політика використання даних [6].

Кожен із самозахищених наборів даних має свої політики доступу, і ці політики консультуються, коли авторизований користувач запитує доступ до захищених даних. За допомогою самозахисту, де політики доступу впроваджені в дані, доступ до захищених даних може бути виконаний в будь-якому місці, якщо існує механізм для виконання політик вбудованого доступу. Механізм контролю доступу буде рудиментарним в його реалізації в локації. Тільки власник даних може керувати політиками контролю доступу та іншими параметрами і функціями, пов’язаними з самозахистом, характерним для кожного набору даних. Для того, щоб зберегти ресурси, скоротити витрати, та зберегти ефективність, провайдери хмарних послуг часто зберігають більше одного разу дані клієнта на тому ж сервері. Для вирішення таких складних ситуаціях, постачальники хмарних послуг повинні забезпечувати правильну ізоляцію даних і логічні сегрегації зберігання.

2.8 Перевірка цілісності даних

Дуже важливо, щоб цілісність набору даних могла бути перевірена на будь-якому етапі. При зберіганні в загальнодоступному домені набір даних піддається модифікації неавторизованими об’єктами, яким може бути навіть сам хмарний провайдер, випадково або навмисно протягом життєвого циклу набору. Тому власники даних і авторизовані користувачі вимагають, щоб ця функція гарантувала, що дані не будуть некоректно змінені. У підході ОБІ інформація для перевірки цілісності даних вбудована в дані і також захищена. Перевірка не залежить від інформації, наданої поза самими даними. Отже, пряме підтвердження цілісності даних в хмарі без необхідності завантаження, це великий виклик, особливо якщо дані динамічно змінюються в хмарі [7].

2.9 Приватність та конфіденційність

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

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

Наприклад, ім’я або місцезнаходження звичайної особи зазвичай не вважається конфіденційним, але в деяких випадках воно може вважатися приватним. Тому захист приватності йде далі, ніж захист фактичних даних від несанкціонованого доступу. Він також запобігає витокам будь-якої інформації, яка розглядається як приватна інформація під час обробки даних. Наприклад, прикріплені політики контролю доступу вважаються приватною інформацією, і провайдер хмари не повинен розкривати їх [8].

Застосування такого виду захисту до процесу управління доступом призводить до підходу до доступу до конфіденційності, який може використовуватися для запобігання витоку навіть часткової інформації про дані під час процесу доступу. Тому будь-яке рішення безпеки має враховувати конфіденційність даних і приватність хмарних клієнтів у відповідності з різними правилами та вимогами.

2.10 Доступність

У хмарних обчисленнях доступність відноситься до послуг хмарних обчислень, наявних і доступних для авторизованих користувачів, коли вони їм потрібні. Крім того, ці служби можуть бути пов’язані з обробкою критично важливих даних і запуском важливих додатків, які повинні бути доступні весь час і здатні обслуговувати велику кількість користувачів. Доступність послуг хмарних обчислень залежить від постійної безперервної здорової роботи інфраструктур і мережевих ресурсів [9] .

Тому хмарний провайдер повинен мати надійні і надлишкові стратегії для забезпечення доступності системи. Хмарний провайдер також несе відповідальність за обслуговування своїх сервісів, навіть якщо є внутрішня або зовнішня загроза, націлена на доступність системи. Іншою проблемою доступності є доступність даних клієнта для доступу власником даних або авторизованими користувачами. Авторизований користувач повинен мати можливість отримувати дані з хмари в будь-який час. Як згадувалося раніше, на етапах життєвого циклу даних, постачальник хмари повинен мати ефективну стратегію резервного копіювання та архівування для захисту доступності даних. З точки зору власника даних, дані є найважливішим активом, яким він володіє в хмарі. Пропонований ОБІ підхід в цій роботі дає власнику даних більш істотну відповідальність і контроль над захистом своїх даних, які повинні зберігатися в середовищі хмарних обчислень.

Кожен із самозахищених наборів даних має свої політики доступу, і ці політики консультуються, коли авторизований користувач запитує доступ до захищених даних.

2.11 Концептуальна основа ОБІ

В цьому пункті створено цілісну концептуальна основа ОБІ. Ця основа базується на огляді різних попередніх дослідницьких робіт щодо концепцій ОБІ, застосовуваних до хмарної моделі, і на основі характеристик ОБІ, визначених раніше у пункті 1.1.2. В рамках основи, що базується на концепції, дані захищені перш ніж покинути довірений домен власника даних, а їх параметри безпеки прив'язані до даних як частина їх метаданих. В ідеальній ситуації тільки облікові дані, які використовуються для відміни або доступу через захист, зберігаються за межами даних, а також власником даних і авторизованими користувачами відповідно до їх прав доступу. Будь-які інші параметри безпеки даних повинні бути прив'язані до даних. Наприклад, в тих випадках, коли дані динамічно оновлюються, користувач, який звертається до даних, повинен перевірити, чи дані є найсучаснішими. Ця вимога має бути надана з фактичних даних або метаданих, прикріплених до даних, наприклад, функція історії є під-рівнем від рівня даних, як показано на рисунку 2.1 [10].

Як інший приклад, доказ достовірності джерела даних також може бути додано разом з доказом цілісності, що зазвичай включає цифровий підпис власника даних. Будь-яка додаткова функція може бути додана як новий підрівень під рівнем даних або включена в якості одного зі зразкових підрівнів, проілюстрованих в лівій частині рисунку 2.1. Крім того, всі ці метадані безпеки надаються на рівні даних і створюються власником даних. Вони також залишаються захищеними протягом усього життєвого циклу даних. У окремого набору даних є свої метадані безпеки, незалежні від інших наборів даних. У наступному списку наведено критерії, які повинні задовольняти будь-яке надане рішення безпеки на основі підходу ОБІ:

− кожен набір даних є самоописовим, самозахищеним і самоохороняємим (тобто захищеним набором даних). Отже, вимоги та функції безпеки кожного набору даних надаються зсередини і не залежать від установок поза набором даних, крім деяких базових процесів обробки даних;

− захист даних не залежить від провайдера хмарних обчислень або довіреної третьої сторони (TTP);

− тільки власник даних відповідає за створення і управління цими вимогами і функціями безпеки для кожного набору даних з моменту його створення до кінця життєвого циклу набору даних;

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

Приклад концептуальної основи ОБІ для моделі хмарних обчислень показаний на рисунку 2.3.

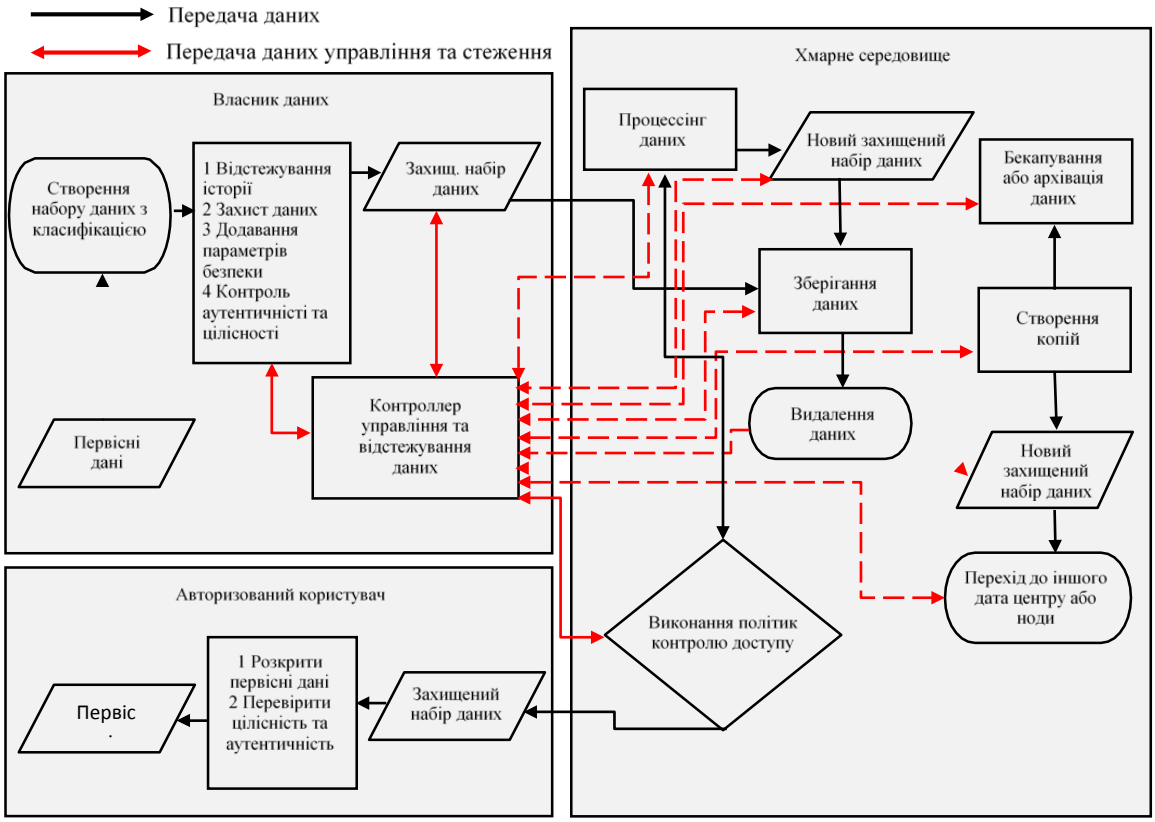


Рисунок 2.3 – Основа ОБІ

Створення даних виконується в надійному домені з боку власника даних. На етапі створення дані класифікуються на основі вимог безпеки і, відповідно, метадані безпеки прив’язані до даних, наприклад, їх політика контролю доступу, перевірка цілісності, запис їх історії і можливості відстеження. На етапі створення, створюється безпечний набір даних і він інкапсулює вміст захищених даних і відповідні метадані. Контролер управління даними і відстеження дозволяє власнику даних управляти і відстежувати набір даних, наприклад, розташування, використання та історію доступу, з моменту створення, поки він не буде видалений з хмарного середовища [11].

Крім того, передача між надійним доменом, в якому знаходиться користувач, і хмарним доменом зазвичай здійснюється через Інтернет і захищена основним механізмом

захисту, забезпечуваним підходом ОБІ або протоколом інтернет-безпеки. У захищеного набору даних протягом його життєвого циклу в хмарі є можливість, виконавши додані до нього коди, інформувати власника даних, якщо є будь-які зміни його стану. У той же час захищений набір даних включає в себе виконуваний код, готовий для прийому і відповіді на запити власника даних для відстеження інформації і команд управління. Навіть якщо зв'язок між хмарним доменом і надійним доменом не працює, оскільки набір даних містить всю необхідну інформацію для виконання основних політик безпеки, пов'язаних з даними, захищений набір даних може бути постійно доступний авторизованим користувачам без участі власника даних.

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

Коли захищений набір даних повинен бути отриманий авторизованим користувачем, хмарний сервер в хмарному домені перевірятиме права доступу користувачів до даних з використанням пов'язаних параметрів безпеки. Авторизованим користувачам дозволено завантажувати захищений набір даних. Потім авторизовані користувачі можуть перевірити цілісність і оригінальність завантажених даних, використовуючи додані докази цілісності та аутентичності. Ця концептуальна основа ОБІ дозволяє, з одного боку, оптимізувати конфігурацію безпеки параметрів безпеки для кожного набору даних, а з іншого - знижує складність загального управління безпекою. Це особливо важливо, коли відповідальність за хмарну безпеку також оптимально розподілена між потенційними сторонами хмарної системи (тобто провайдерами хмарних обчислень, користувачами і ТТР) відповідно до трьох рівнів відповідальності безпеки (див. рис. 2.3).

Наприклад, хмарні провайдери несуть відповідальність за захист своїх систем, ТТР, відповідальних за активи, заходи безпеки, що надаються провайдерами, і власники даних несуть відповідальність за захист своїх даних в хмарі. Інші очікувані переваги застосування ОБІ до хмарної моделі обговорюються в наступному пункті [12].

2.12 Висновки до розділу 2

У другому розділі розглянуто підхід до ОБІ і його застосування до хмарної моделі.

Підхід ОБІ ґрунтується на трьох фундаментальних концепціях:

– вимоги безпеки забезпечуються всередині даних.

– власник даних відповідає за ці вимоги безпеки протягом усього терміну служби даних;

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

Ці концепції відрізняють підхід ОБІ від інших підходів для забезпечення безпеки і приватності даних у хмарі. Інші підходи забезпечують заходи безпеки на рівні обладнання, платформи або програми, спираючись на хмарного провайдера і/або третю сторону, включаючи технології надійних обчислень.

На противагу цьому, підхід ОБІ забезпечує рішення для забезпечення безпеки і приватності даних для покриття всього життєвого циклу даних. Відповідно, кожен набір даних є самоописовим, самозахисним і з усіма доданими до нього специфікаціями безпеки, і виконання цих специфікацій здійснюється безпечним чином.

Очікується, що застосування такого підходу до хмарної моделі поліпшить приватність даних і безпеку в хмарі, щоб відповідати її масштабованості і еластичності. Однак повне застосування ідеальної концептуальної основи ОБІ, як і раніше, ускладнено декількома проблемами. Сфера застосування підходу ОБІ в цьому дослідженні визначається для обмеження таких проблем. Хоча область охоплення обмежена неструктурованими даними, що зберігаються і розділяються в публічному хмарному сховищі, пропоноване рішення охоплює найбільш поширені ситуації в хмарних сервісах. Більш того, рішення може бути змінено для застосування в інших формах даних і в інших хмарних моделях. Основні вимоги безпеки для пропонованого рішення перераховані в пункті 2.1.10, які, в основному, пов’язані із захистом приватності і цілісності при доступі і пошуку зашифрованих даних, що зберігаються на хмарних серверах.

На основі концепцій ОБІ розглядаються методи безпеки, які можуть задовольняти ці вимоги. Методи, що відповідають вимогам безпеки, адаптовані і використані в запропонованому рішенні ефективним і практичним способом.

РОЗДІЛ 3
ІНФОРМАЦІЙНО-ОРІЄНТОВАНЕ РІШЕННЯ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПРИВАТНОСТІ ТА БЕЗПЕКИ У ХМАРНОМУ СЕРЕДОВИЩІ

У третьому розділі наведено запропоноване рішення щодо основних дослідницьких запитань даної роботи. Це рішення має за основу ОБІ підхід описаний у розділі 2.

Мета даного розділу полягає в розроблені рішення, яке робить оригінальний дослідницький внесок в концепцію ОБІ, що вважається адекватним підходом до вирішення питань безпеки та конфіденційності в хмарних обчисленнях [13] .

Рішення призначене для задоволення наступного переліку бажаних вимог до додатків в середовищі хмарних обчислень, які:

– зашифровані та доступні лише для авторизованих користувачів;
– доступні для пошуку без загрози для їх конфіденційності;
– відповідають вимогам самозахисту та необхідним параметрам безпеки.

Параметри контролю доступу приховані від провайдерів хмарних серверів та інших користувачів. Провайдер серверу не знає кількість або особистість користувачів, що авторизовані та мають право доступу до даних.

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

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

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

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

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

3.1 Теорема про залишки та її застосування

Теорема про залишки як одна із математичних теорем для аналізу арифметичних задач [14]. Теорему розвивали математики по всьому світу, поки вона не досягла сучасного свого вигляду формули.

Суть теореми про залишки [11] полягає в наступному.

Для будь-яких заданих цілих чисел *a1, a2, ..., ak,* наступна система одночасної конгруенції має унікальне рішення *X*, яке лежить в межах $0 \leq X < n = n_1,n_2,...nk$, за умови, що невід'ємні цілі числа $n_1, n_2, \ldots n_k$ взаємно прості.

$$\begin{aligned} X &\equiv a_1 \bmod n_1 , \\ X &\equiv a_2 \bmod n_2 , \\ X &\equiv a_{\mathrm{n}} \bmod n_{\mathrm{m}} \end{aligned} \tag{3.1}$$

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

$$a_{\mathrm{i}} = X \bmod n_1 \ , \tag{3.2}$$

для $i = 1,2, \ldots, k.$

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

$$x = \sum_{i-1}^{k} a_i M_i \, M_i^{-1} (mod\ M) \tag{3.3}$$

де $M = n_1, n_2 \ldots n_k = M/n_1,$

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

Через те що M_i відносно первинне до n_i, є унікальний мультиплікативний зворот *mod* a_1.

Отже, розрахунок мультиплікативного зворотного в модулі є істотною частиною визначення рішення теореми.

Розширений алгоритм Евкліда [12] може бути використаний для обчислення мультиплікативного звороту та є істотним компонентом алгоритму Гарнера [13], який зазвичай використовується для визначення рішення цієї теореми.

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

Вхід: додатні цілі числа $n = \prod_{i=1}^{k} > 1$, 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:

Для $i = 2$ до k справедливе:

1.1.$C_1 \leftarrow 1$.

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

1.2.1. $u \leftarrow nj-1 \ mod \ n_i$.

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

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

Для $i = 2$ до k справедливе:

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

4. Повернути (X).

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

Однією із корисних функцій цієї форми є те, що мінімальна зміна для поточної відповідності це додавання нового модулю до конгруентності. Наприклад, якщо рішення X для рівняння (3.1) вже було знайдено та якщо новий модуль $a_{k=1} \ mod \ n_{k+1}$ додається до конгруентності, нехай рішення для нової конгруенції буде X'.

Нове рішення може бути знайдене шляхом обчислення рішення наступних двох конгруенцій $X' \equiv X \ mod \ n_1 n_2 \ldots n_k$ та $X' \equiv a_{k+1} mod \ n_{k+1}$.

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

Простіше, якщо існуючу конгруентність видалити з конгруентностей, нове рішення може бути отримане простим розрахунком. Наприклад, нехай нове рішення отримає позначення X'' після останньої конгруентності $X \equiv a_k \ mod \ n_k$ у рівнянні (3.1) буде виключено. Нове рішення може бути вирахуване дією з одним модулем [15]

$$X'' \equiv X \ mod \ n_1 n_2 \ldots n_{k-1} \tag{3.4}$$

Ці функції можуть бути використані для мінімізації додаткових обчислень розрахунку рішення в таких випадках. Наприклад, ця функція використовується в запропонованому рішенні в цьому пункті при видачі прав доступу новому користувачу до певних ресурсів.

Даний підхід використовує кілька додатків в криптографії та зв'язок захищеними мережами через свою арифметичну природу, що не споживає високих обчислювальних ресурсів [16].

Таким чином, він підходить для обчислювальних і комунікаційних додатків з обмеженим ресурсами. Розділення ключів є однією із основних функцій безпеки з використанням в криптографії. Це дозволяє виконувати поділ виводячи з нього декілька спільних, які можуть бути відновлені тільки з певним заздалегідь підготовленим набором значень. Багато додатків засновані на цій схемі розділення ключів, такі як порогова криптографія та електронне голосування.

У бездротових мережах, де пристрої можуть мати обмежені ресурси, особливо обчислювальні ресурси і ресурси зберігання, запропоноване рішення може бути використане в декількох схемах. Зокрема, безпосередньо при аутентифікації та управлінні доступом до схем.

3.2 Управління доступом та розділення ключів

У даній роботі, запропоноване рішення використовує криптосистему з публічним ключем для безпечного обміну даними захищених користувачів, що зберігаються в середовищі хмарних обчислень серед авторизованих користувачів [16]. Дані, які іноді згадуються як ресурси або файли, захищені симетричним методом шифрування, де як секретний ключ використовується *Ks* [17].

У даній роботі, ресурсом може бути набір даних або файл, який містить дані будь-якого типу, в тому числі текст, аудіо, зображення або відео. Для поширення секретного ключа для авторизованих користувачів, його зашифровано з використанням методу публічного шифрування відкритого ключа користувача.

Шифрування також включає в себе інше значення, *Cr*, яка буде використовуватися в якості відповіді на запит сервера, коли користувач робить запит щодо доступу до ресурсу. Лише авторизовані користувачі, які мають відповідний приватний ключ, можуть розшифрувати шифротекст *Cr* та *Kr* для отримання *Cr*, щоб отримати доступ до ресурсу *r*. Параметри *Cr* та *Ks* зчеплені для формування *Cr||Ks* та розглядаються як єдине значення. Це значення шифрується за допомогою публічного ключа $E_{k\ pubi}$ кожного авторизованого

користувача u_i, в результаті шифротекст ($E_{k\ pubi}$ (Cr || Ks)) для цього користувача u_i, де $i=1, 2, 3, \ldots, k,$ та k кількість авторизованих користувачів, що мають доступ до ресурсу r.

Щоб застосувати запропоноване рішення, для користувачів u_1, u_2, ..., u_k, кожен авторизований користувач пов'язаний з унікальним відносним простим числом $n_i = n_1, n_2, \ldots, n_k$, де k це кількість авторизованих користувачів. Всі n_i, $1 \leq i \leq k$, є відносні прості числа. Потім, шифротекст Cr || Ks генерується для кожного користувача, тобто ($E_{k\ pubi}$ (Cr || Ks)), використовується для заміни a_i у рівнянні (3.1) для виведення наступної спільної конгруентності:

$$\begin{aligned} Xr &\equiv (E_{k\ pubi}\ (Cr \,||\, Ks))\ mod\ n_1 \\ Xr &\equiv (E_{k\ pubi}\ (Cr \,||\, Ks))\ mod\ n_1 \\ Xr &\equiv (E_{k\ pubi}\ (Cr \,||\, Ks))\ mod\ n_k \end{aligned} \tag{3.5}$$

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

$$\text{Cr||Ks} = D_{k\ prvi}\ (x_r\ mod\ n_i) \tag{3.6}$$

де $D_{k\ prvi}$ це операція дешифрування з використанням приватного ключа користувача u_i, та n_i як відносного простого числа специфічного для даного користувача.

Значення Cr відправляється назад до серверу користувачем, щоб довести володіння правом доступу до ресурсу. Потім сервер надсилає ресурс для користувача ключ KS, який використовується користувачем для щоб розшифрувати файл та отримати його вміст.

3.3. Підвищення безпеки ключів та файлів

З точки зору підвищення безпеки в запропонованому рішенні, власник даних повинен використовувати унікальний симетричний ключ, *KS*, для шифрування кожного файлу перед відправкою його на зберігання в хмару. Відповідно до ОБІ підходу, всі вимоги до безпеки прикріплені до фактичних даних, отже, кожен симетричний ключ прикріплюється до його файлу. З Рівняння (3.3), симетричний ключ *KS* захищений двома рівнями: спочатку шифруванням його авторизованим користувачем, а потім за допомогою КТЗ, щоб знайти загальне значення *Xr*. Лише авторизовані користувачі можуть обчислити *KS* з *Xr* за допомогою рівняння (3.4), якому потрібен відповідний ni та приватний ключ авторизованого користувача. Для ефективного управління ключами, власник даних додає себе в якості користувача при розрахунку *Xr* для кожного файлу за допомогою Рівняння (3.3). Отже, ані власник даних, ані користувачі не зобов'язані зберігати *KS*, але їм потрібен тільки їх приватний ключ і призначене значення *ni*. Таким чином, ключ *KS* надійно та ефективно, спільно з авторизованими користувачами, захищено від несанкціонованого доступу, включаючи постачальника послуг хостингу файлу [18].

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

3.4 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 *KS*, який було використано для шифрування даних. Лише авторизовані користувачі можуть виявити *KS* 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 [19].

Для полегшення необхідних розрахунків, можна відзначити, що, коли авторизований користувач u_i має доступ до файлу *r*, власник даних вже повинен був надіслати його відносне просте ni 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, *ni* повинен бути надісланий користувачеві надійно, наприклад, за допомогою шифрування його з відкритим ключем користувача [26].

Таким чином, зловмисникам буде важче виявити шифротекст ($E_{k\ pubi}$ (*Cr* || *Ks*)*)* 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 n_i.

Повідомлення, якими обмінюються авторизовані користувачі і хостинг хмарним сервером загальних файлів показано на рисунку 34.1. У пропонованому рішенні, коли користувачу ui потрібен доступ до файлу r, користувач надсилає запит на сервер, що містить ID файлу, IDi власника файлу, нонс (будь-яке випадкове число), і сесійний ключ Kt. Все шифрується за допомогою відкритого ключа сервера $E_{k\ pubs}$. Результат можна представити у вигляді $E_{k\ pubs}$ *(IDr, Idi, Kt*).

Нонс (англ. nonce) — довільне число використовне під час криптографічного зв'язку лише один раз. Нонс часто випадкове або псевдовипадкове число утворене протоколом автентифікації, для гарантування унеможливлення використання старих сеансів зв'язку в атаці повторного відтворення.

Наприклад, нонс використовує дайджест автентифікація HTTP для обчислення MD5 гешу/дайджесту пароля. З кожною відповіддю 401 сервер породжує новий нонс і відсилає його як частину відповіді клієнту, той у свою чергу може відправити серверу його відповідь, включно з нонсом, і також свої ім'я і пароль. Такий підхід практично унеможливлює атаки відтворенням. Нонс можна використати для гарантування безпеки потокових шифрів. Якщо один і той самий ключ використовують для шифрування більш як одного повідомлення, тоді використовують нонс для забезпечення того, що потік ключа буде відмінним для всіх повідомлень зашифрованих за допомогою цього ключа. Часто використовуються номер повідомлення. Наприклад, нонс використовується при утворенні потокового шифру з блочного в режимі CTR, завдяки чому вдається отримати можливість розпаралелювання [27].

Приклад використання клієнт-серверного сеансу зв'язку автентифікації з нонсом наведено на рисунку 3.1 [20].

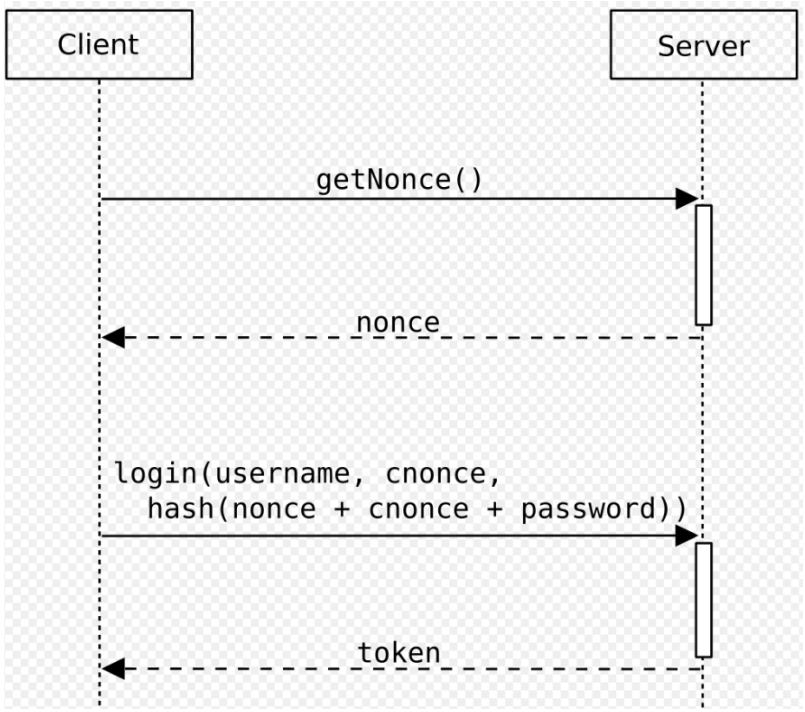


Рисунок 3.1 – Клієнт-серверний сеанс зв'язку автентифікації з нонсом

Сесійний ключ *Kt* створюється користувачем для безпечного обміну інформацією між користувачем і сервером. Сервер знаходить файл з *IDr*, читає його спільне значення *Xr* і шифрує їх за допомогою сесійного ключа *Kt*. Результат, можна представити як $E_{k\,t}(Xr)$, нонс, який повертається користувачу.

На рисунку 3.2 представлена процедура контролю доступу до файлу.

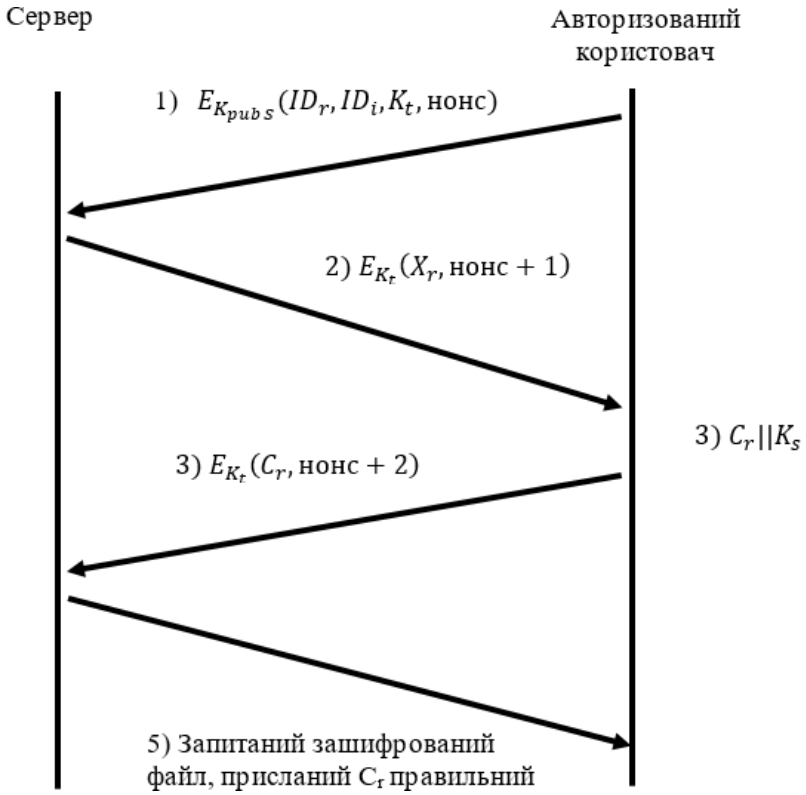


Рисунок 3.2 – Процедура контролю доступу до файлу

Після отримання інформації, представленої в повідомленні номер два на рисунку 3.2, з сервера, користувач обчислює секретний параметр *Cr*, використовуючи рівняння (3.4) і повертає *Cr* на сервер і шифрується сесійний ключ *Kt*. Сервер перевіряє *Cr* і, якщо він збігається з тим, що прикріплений до файлу на який було надіслано запит, сервер надсилає зашифрований файл користувачеві [21].

Користувач може розшифрувати файл за допомогою ключа *KS*, 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 з метою запобігання традиційних атак, такі як Man-In-The Middle (MIM).

Як було згадано вище, для конфеденційного доступа, особистість авторизованого користувача приховано від сервера, отже, процедури уникає використання публічного ключа користувача, щоб зберегти його особистість в таємниці. На противагу цьому, користувач починає сессію з сервером з повідомленням зашифрованим за допомогою публічного ключа сервера для перевірки аутентичності сервера. Таким чином, якщо користувач здійснює зв’язок з певним конкретним сервером, тільки цей сервер може розшифрувати повідомлення

для того щоб розкрити сесійний ключ *Kt* і нонс, а потім реагувати зашифрованим повідомленням яке містить нонс + 1.

Процедура доступу показує, що виконання політики контролю доступу та забезпечення авторизованого користувача секретний ключ для дешифрування даних здійснюється в одному механізмі [25].

Даний метод зменшує додаткове навантаження як на хмарному сервері, так і користувача, з точки зору обчислення та управління ключами. До того ж, використання його в механізмі, є ще одним фактором для зменшення додаткових витрат, оскільки операції є простими модульними арифметичними операціями; зокрема, наприклад, не має модульних експонентних операцій. Ще одна важлива особливість що досягається за допомогою використання даного методу, є те, що він дозволяє власнику даних приховувати число авторизованих користувачів.

Шифротекст кожного авторизованого користувача u_i *для* $i = 1, 2, \ldots k$ може бути представлений з використанням даного методу з одним значенням *Xr*. Сервер не може дізнатися скільки користувачів представлені в цьому *Xr*, але без використання КТЗ власник даних повинен прикріпити всі шифротексти, щоб розкрити серверу кількість користувачів.

3.5 Процедура надання та скасування доступу

Щоб надати доступ до файлу r новому користувачу u_{k+1}, до даного методу додається нова конгруентність, як показано в рівнянні (3.5). У такому випадку, власнику необхідно перерахувати загальне значення *Xr′*.

$K_{pub\ k+1}$ +1 є відкритим ключем нового користувача (див. п 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 [22]:

1. Власник даних надсилає нове загальне значення $X'r$ з ID файлу та ID власника даних до серверу для заміни старого *Xr*. Механізм додавання нового користувача ефективний з двох причин; необхідна лише одна асиметричний операція шифрування для фіксованої довжини інформації, тобто, $Cr||Ks$, та він знаходить рішення лише для двох конгруенцій,

див. рівняння (3.6). Хоча загальні значення *X′r* базуються на основі секретних значень, лише авторизовані користувачі мають доступ до них.

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

$$\begin{aligned} x''_r &= (E_{k\ pub1}(c' \,||\, k_s) \bmod n_1 \\ x''_r &= (E_{k\ pub2}(c' \,||\, k_s) \bmod n_2 \\ x''_r &= (E_{k\ pub\ k-1}(c' \,||\, k_s) \bmod n_{k-1} \end{aligned} \qquad (3.7)$$

2. Власник даних надсилає на сервер нові значення *C′r* та *X''r* з ID -файлу та ID власника файлу щоб замінити відповідні старі значення для файлу. Якщо користувач отримав доступ до даних перед скасуванням старих значень, то можливо, що користувач і сервер можуть змовитися для доступу до даних після скасування.

Коли авторизований користувач отримав доступ до файлу, ключ цього файлу стає відомим для нього. Пізніше, якщо для цього користувача буде анульовано доступ до файлу і власник даних змінив лише *Cr* для цього файлу, користувач може домовлятися з сервером, так що сервер дозволяє йому отримати доступ до файлу, і користувач надає серверу старий ключ *Ks*, що досі розшифровує файл.

Схема не розкриває особистість користувачів для сервера і може зменшити імовірність такого роду collusion attack, тому рекомендується змінити ключ *Ks* і повторно зашифрувати файл, особливо якщо відбулася зміна змісту. Потім замінити старий файл на новий, який було зашифровано з новим ключем і який має нові *C′r* та *X''r* прикріплені до нього.

Таким чином, сервер і користувач, позбавлені права доступу, не можуть вступити в змову з метою отримання доступу до нового вмісту файлу.

Як було згадано вище, для конфеденційного доступа, особистість авторизованого користувача приховано від сервера, отже, процедури уникає використання публічного ключа користувача, щоб зберегти його особистість в таємниці. На противагу цьому, користувач починає сессію з сервером з повідомленням зашифрованим за допомогою публічного ключа сервера для перевірки аутентичності сервера.

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

3.6 Можливість зашифрованого пошуку

У даній роботі використані методи для пошуку зашифрованих даних, які сумісні з ОБІ підходом. Власник даних повинен вказати один секретний ключ *Kw*, 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 вхідним повідомленням *m* [15].

Припустимо, w_i – *i*-ий ключ в списку ключових слів, R – випадкове число, прапорець – постійне бітове знаення з фіксованою довжиною 1 біт і зміщенням випадковим значенням бітів.

$$a_i = H_{Kw}(w_i),\ b_i = H_{ai}(R),\ c_i = b_i \oplus (\text{прапорець}|\text{зміщення}) \qquad (3.8)$$

Від кожного ключового слова *wi*, 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 *ci*. Наступний пункт описує, як користувачі можуть безпечно здійснювати пошук їхніми зашифрованими файлами за допомогою використання зашифрованих ключових слів.

3.7 Процедура доступу з підтримкою можливості безпечного пошуку

У запропонованому рішенні (рисунок 3.3) для пошуку зашифрованих даних, що зберігаються в хмарному середовищі, коли користувач ui здійснює пошук ключового слова wi, користувач має надіслати запит власнику даних для отримання можливості пошуку, що містить ключове слово wi зашифроване за допомогою публічного ключа власника (*Kpub O*) і підписаного приватного ключа користувача (*Kprv i*). Власник даних потім відповідає з *Tw= HKw(wi),* зашифрованому за допомогою публічного ключа користувача (*Kpub i*). Користувач розшифровує повідомлення за допомогою свого приватного ключа та потім зашифровує *Tw* з публічним ключом сервера *(Kpub S),* перед відправкою його на сервер в якості пошукового запиту з ID власника, нонс і сесійний ключ *Kt*, як показано в рівнянні (3.9).

$$E_{K\text{pub }s}(Tw, I\text{D власника}, Kt, \text{нонс}) \quad (3.9)$$

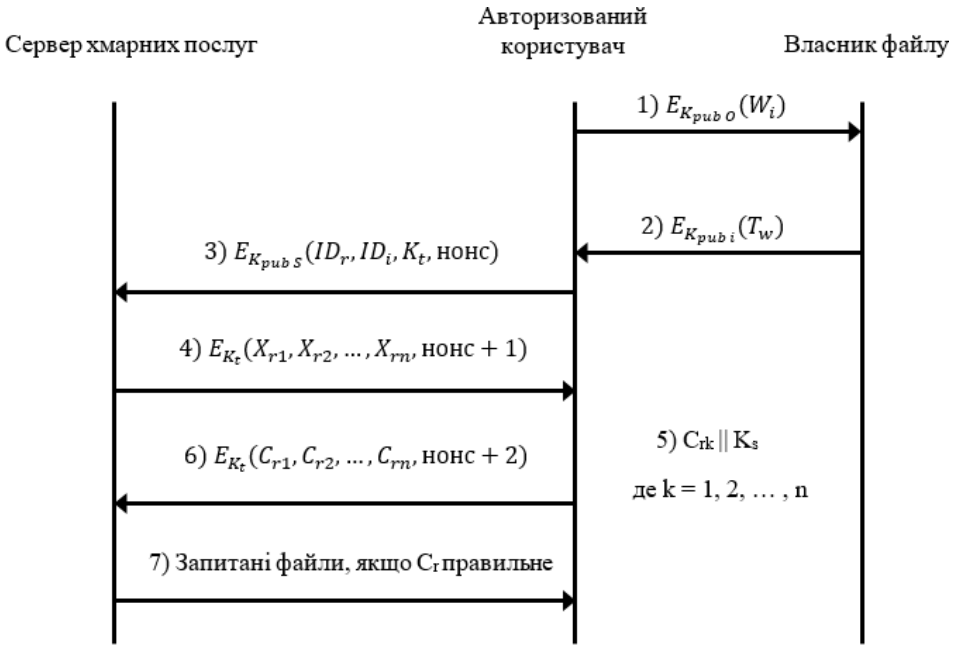


Рисунок 3.3 – Процедура безпечного пошуку

Сервер повинен шукати тільки дані, пов’язані з ID власника, використовуючи Tw для обчислення $P = H_{Tw}(R)$, а потім для кожного сі сервер перевіряє результат $P \oplus c_i$,, щоб побачити, якщо перший біт дорівнює прапорцю. Потім ID файлу, пов’язане з цим c_i, додається до списку результатів пошуку разом із власними двома параметрами *Xr* та *Cr*.

$$P = HTw(R), \text{ if } P \oplus c_i = \text{флаг зміщення} \quad (3.10)$$

Після пошуку всіх файлів, сервер надсилає лише те значення Xr_n , де n - кількість файлів, які відповідають критеріям пошуку, та нонс+1 і всі зашифровані за допомогою сесійний ключ, Kt, назад клієнту.

$$E_{Kt}(X_{r1}, X_{r2}, \ldots X_{rr}, \text{нонс} + 1) \quad (3.11)$$

Користувач отримує секретне значення Cr_i для кожного загального значення Xr_i , де $i = 1, 2, \ldots, n$, за допомогою рівняння (3.4), а потім повертає секретні значення і нонс + 2 на сервер зашифроване за допомогою сесійного ключа *Kt*.

$$E_{Kt}(Cr_1,Cr_2,\dots Cr_n, \text{нонс+2}), \quad (3.12)$$

Сервер перевіряє секретні значення ($Cr_1,Cr_2,\dots Cr_n$) та надсилає клієнту лише ті файли, секретні значення яких збігається з тими, які були надіслані користувачем. Тепер користувач може розшифрувати файли за допомогою секретного ключа *Ks*, витягнутого з Xr_i для кожного файлу. Рисунок 3.2 ілюструє безпечну процедуру пошуку.

3.8 Побудова ОБІ-файлу з перевірками цілесності та аутентичності

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

Перед створенням ОБІ-файлу є чотири основні операції [23]:

– шифрування вихідного вмісту файлу за допомогою алгоритму симетричного шифрування;

– створення, за допомогою теореми про залишки значення *Xr;*

–генерування секретних ключових слів для можливостей пошуку;

– створення перевірок цілісності та аутентичності для зашифрованого вихідного файлу даних.

Зашифрований файл захешований, а потім отриманий дайджест значення має цифровий підпис за допомогою його шифрування використовуючи приватний клбч *Kprv O* власника файлу.

Потім ОБІ-файл створюється на основі цих чотирьох операцій.

Рисунок 3.4 показує вихідну структуру ОБІ-файлу. Лише авторизовані користувачі мають можливість отримати вихідний файл з ОБІ-файлу.

Всі зашифровані ключові слова, де *ci* для *i=1,2, ..., z* та *z,* як кількість ключових слів, прикріплюються до ОБІ-файлу разом з асоційованими рандомними числами *Ri*. ОБІ-файл тепер може бути відправлений для зберігання в хмарне середовище.

Приватність та цілісність вихідного файлу надійно зберігається і не залежать від хмарного серверу для забезпечення вимог безпеки, за винятком деяких операцій для їх виконання.

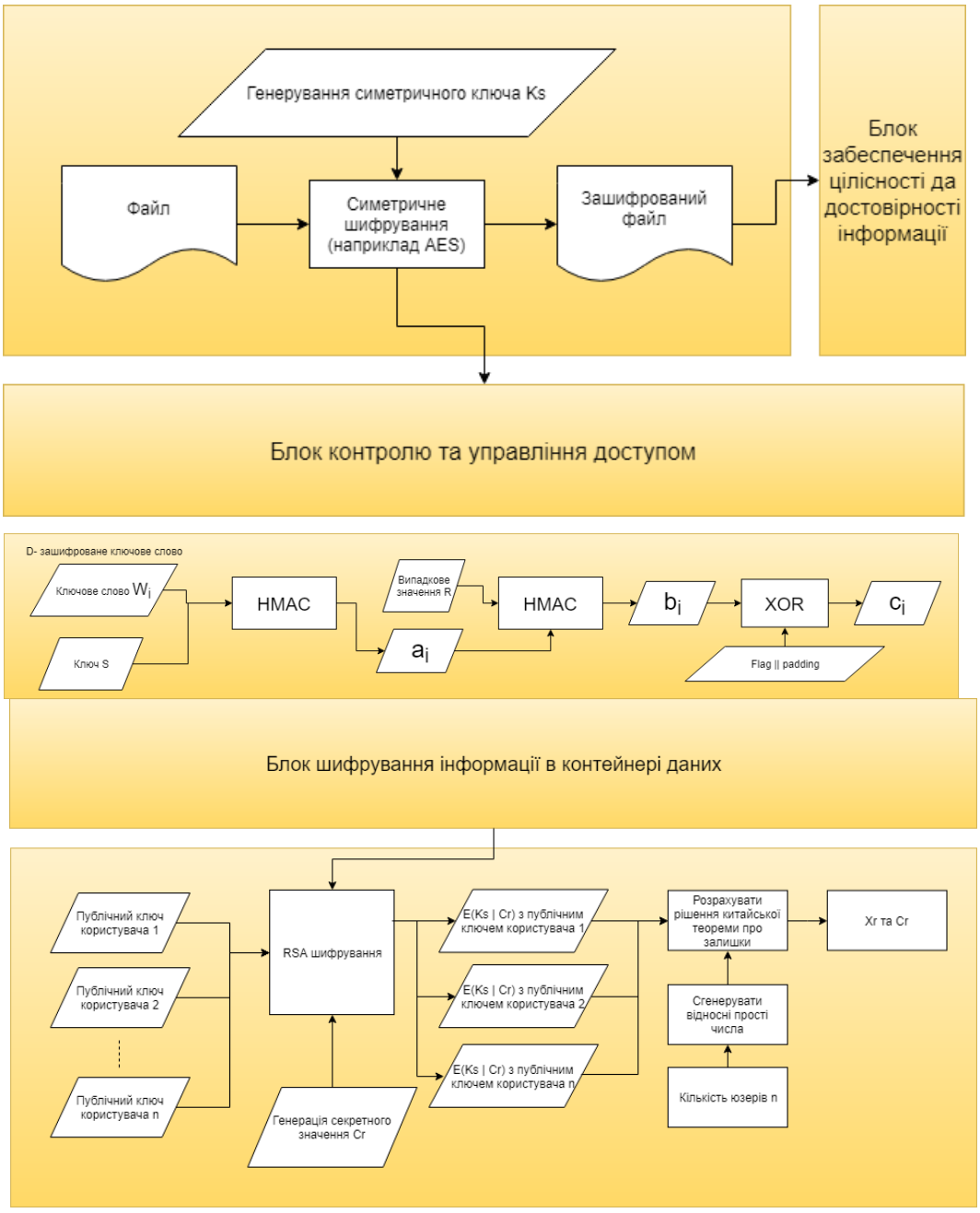


Рисунок 3.4 – Структура ОБІ-файлу

3.9 Збереження конфіденційності та цілісність запропонованого рішення

У порівнянні з попередніми роботами, запропоноване рішення є більш безпечним, так як використовується відмінний симетричний ключ для шифрування кожного файлу та інше

секретне значення для забезпечення дотримання параметрів контролю доступу для кожного файлу.

Це поліпшення знаходить своє відображення в кількох аспектах безпеки та приватності, які наведено нижче:

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

– особистість користувача завжди прихована;

– якщо симетричний ключ *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;

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

Оскільки кожен файл шифрується з іншим симетричним ключем, сервер не може вступити в змову з користувачем, позбавленим права доступу, для того щоб розкрити вміст даних, якщо такий користувач не мав доступу перед позбавленням його таких прав. Особливо, якщо такий користувач отримав доступ до ОБІ-файлу і показав симетричний ключ *Ks*, то власник даних може видалити файл і використовувати новий ключ Ks для створення нового ОБІ-файлу [24].

Таким чином, якщо сервер запобігає збереженню копії цього ОБІ- файлу, то він не може вступити в змову з користувачами, які були позбавлені права доступу.

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

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

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

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

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

3.10 Висновки до розділу 3

У третьому розділі представлено рішення, яке використовує ОБІ підхід для хмарних обчислень. Рішення розроблене на основі модулів, кожен з яких забезпечує набір сервісів безпеки, які в основному знаходяться в управлінні власника даних.

Перший модуль призначений для шифрування даних перед їх аутсорсингом.

Другий модуль створює можливості для більш ефективного та дієвого способу управління політикою контролю доступу, які виражені секретним та спільним значенням, які призначені для спільного використання і доступу до контролю даних.

Третій модуль використовується для забезпечення можливості безпечного пошуку для зашифрованих даних через генерацію зашифрованих ключових слів.

Четвертий модуль виділяє перевірки цілісності та аутентичності.

Результати всіх попередніх модулів використовуються для створення ОБІ-файлу. Використання ОБІ-файлу є результатом пошуку рішення що підвищує безпеку і конфіденційність та відповідає середовищу хмарних обчислень. Використання ОБІ-файлу, як частини запропонованого рішення, здатне зберегти приватність, цілісність, аутентичність даних які було розміщено в хмарі, навіть від самого хмарного провайдеру з мінімальним впливом на функціональність зашифрованих даних, які відповідають можливостям пошуку та поширення для авторизованих користувачів. Функції безпеки, запропоновані в ОБІ-файлі, залежать від криптографічних алгоритмів, які використовуються в даному рішенні і знаходяться під управлінням власника даних.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ МОДЕЛІ ТА ЇЇ ДОСЛІДЖЕННЯ

В цьому розділі розглядаються питання реалізації запропонованого рішення з метою оцінки впливу використання рішення на стороні клієнта і сервера з точки зору ресурсів обчислень і зберігання.

4.1 Інструменти реалізації і середовище експерименту

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

- процесор: i5-7400 3.0GHz/8GT/s/6MB;
- оперативна пам’ять: 16 ГБ;
- загальна сховище на жорсткому диску: 256 ГБ (SSD).

Для створення віртуального сервера використана платформа VMware VSphere 6.

VMware є одним зі світових лідерів в області віртуалізації і хмарної інфраструктури. Віртуальний сервер був налаштований з такими специфікаціями:

- ОС: Windows Server 2012 R2 Enterprise 64-розрядної версії;
- віртуальна пам’ять: 8 ГБ;
- віртуальний простір для зберігання: 100 ГБ;
- процесор: 2 ядра Intel i5-7400 3.0 ГГц.

Інші віртуальні машини були встановлені на тому ж виділеному апаратному сервері для створення віртуального середовища, що імітує середу віртуалізації, яка використовується в публічній хмарі.

В публічній хмарі кілька клієнтів використовують один і той же апаратний сервер, використовуючи свої віртуальні машини, що працюють на одному і тому ж апаратному сервері. Ці віртуальні машини ізольовані віртуально з використанням платформи віртуалізації.

В експерименті використовувався комп’ютер з наступними характеристиками:

- процесор: Intel CORE i3 CPU 2.2 ГГц;
- пам’ять: 16 ГБ;
- ОС: Windows 10 64 біт.

Комп’ютер використовувався для моделювання як сторони власника даних, так і авторизованого з користувальницького боку.

З боку власника даних було реалізовано і протестовано операції зі створення ОБІ-файлу перед його аутсорсингом на хмарний сервер.

Для авторизованого користувача реалізація та тестування проводилися для операцій шляхом обчислення значення Cr||Ks із загального значення Xr, використовуючи рівняння (3.4), і шляхом дешифрування даних з файлу ОБІ, що містить зашифровані дані.

На стороні сервера було виконано пошук ключових слів у ОБІ-файлі.

На рисунку 4.1 показано налаштування середовища експерименту, повідомлень, переданих між об’єктами, і основних операцій з кожного боку. На етапі реалізації одним із завдань був вимір службових даних сховища і накладних витрат на обчислення для виконання операцій ОБІ.

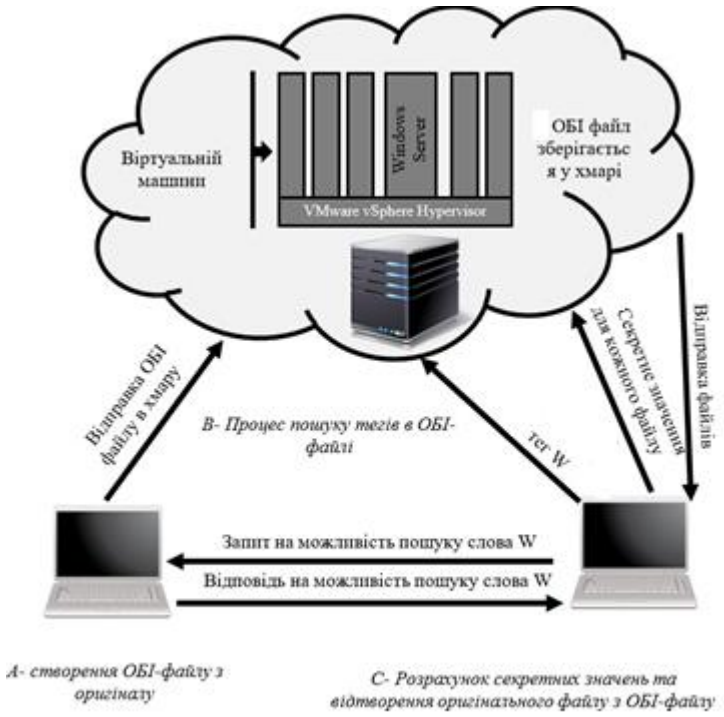


Рисунок 4.1 – Експериментальне середовище для запропонованого рішення

4.2 Реалізація програми на стороні клієнта

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

авторизованим користувачем. Пошук рішення і шифрування ключових слів виконується тільки власником даних.

У наступних пунктах описується реалізація необхідних функцій для клієнтської сторони.

4.3 Симетричне шифрування вихідного файлу

Шифрування даних за допомогою симетричного алгоритму шифрування є найбільш важливою операцією. Для надійного захисту дані повинні бути зашифровані, навіть якщо вони залишаться у власності власника даних. Тому перша дія після класифікації даних як конфіденційних даних – це шифрування даних з використанням адекватного алгоритму шифрування і сили ключа. У загальному випадку алгоритм симетричного шифрування використовується замість асиметричного шифрування для шифрування великої кількості даних, оскільки він більш ефективний.

У даній реалізації файли зашифровуються за допомогою Advanced Encryption Standard (AES), що забезпечується програмою AES Crypt з відкритим вихідним кодом для платформ Windows з використанням 256-бітної довжини ключа, сильнішої довжини ключа для цього алгоритму. У наступному пункті описується реалізація алгоритму, використаного для спільного використання Ks з секретним значенням Cr.

4.4 Реалізація розрахункового рішення

Рішення використовується для розподілу двох спільних секретних значень для кожного файлу серед авторизованих користувачів; симетричний ключ Ks, який використовується для шифрування певного файлу і секретного значення Cr, використовуваного для забезпечення дотримання політик контролю доступу для цього певного файлу. У реалізації довжина значення Cr повинна бути зафіксована для всіх файлів, тому легко відрізнити її від симетричного ключа Ks, коли два значення об’єднані як одне значення Cr||Ks. Ці два значення зашифровуються одним з асиметричних алгоритмів шифрування з публічним ключем кожного авторизованого користувача.

В результаті шифротекст для $i=1, 2, \ldots, k$, де k – кількість авторизованих користувачів для певного файлу. Отриманий в результаті шифротекст для кожного користувача, якого було авторизовано для доступу до файлу, буде вихідним для операції КТЗ для обчислення загального значення Xr для цього файлу, використовуючи рівняння (3.3) з пункту 3.2.

Потім для реалізації потрібно знайти рішення одночасної конгруентності КТЗ, представленої в рівнянні (3.3).

Наприклад, при використанні алгоритму RSA для шифрування значення Cr||Ks, мінімальна довжина результату шифротексту буде становити не менше 1024 біт. Для реалізації платформа .NET надає бібліотеку класів System.Numerics.BigInteger для обробки довгих цілих значень довжиною понад 64 біти. Крім того, для КТЗ потрібна функція для генерації відносних простих чисел $n_i = n_1, n_2, \ldots n_k$, де k - кількість авторизованих користувачів. Генерація відносних простих чисел може бути виконана один раз для всіх користувачів, а потім визначена унікально для кожного користувача.

Програма, яка використовується для вирішення одночасної конгруентності теореми про залишки, базується на алгоритмі Гарнера та розширеному алгоритмі Евкліда (РАЕ) і використовується для обчислення модульного мультиплікативного звороту. Вихідні коди для цих двох алгоритмів доступні у вигляді програми з відкритим вихідним кодом.

Однак оригінальні вихідні коди реалізовані для підтримки цілих чисел з довжиною в 64 біти. Для реалізації запропонованого рішення вихідні коди модифікуються на основі бібліотеки класів System.Numerics.BigInteger для підтримки цілих чисел більше 64 бітів, класифікованих на C# як Biglntegers.

Отже, результуюча програма для вирішення теореми про залишки може обробляти цілі числа довжиною понад 64 біти. CalculateKTZ.cs і EuclidHelper.cs – це два файли класів C#, які містять оновлені програмні коди для пошуку рішення КТЗ і для обчислення модульного мультиплікативного звороту. Вихідні коди цих двох файлів класів C# описані в Додатку Б. Функція CalculateKTZ.Execute (Biglnteger [] input_a, Biglnteger [] input_n) використовується для знаходження рішення КТЗ, де *a* і *n* – вхідні масиви Biglnteger значень ai і ni.

На рисунку 4.2 показано вихідний код для вирішення КТЗ, що підтримує Biglntegers. У функції Execute, модульний мультиплікативний зворот обчислюється з використанням функції EuclidHelper.ExecuteInverse (Biglnteger input_p, Biglnteger input_g), як показано на рисунку 4.3. Поверненим значенням зворотної функції є g-1 (mod p). Найбільший спільний дільник (GCD) двох цілих чисел можна обчислити, використовуючи функцію Biglnteger p.gcd (Biglnteger g), включену в бібліотеку класу Biglnteger.

```
using System;

public class CalculateKTZ
{
    // Розрахунок рішення для китайської теореми про залишки
    public static System.Numerics.BigInteger Execute(System.Numerics.BigInteger[] input_a, System.Numerics.BigInteger[]
    {
        if (input_a == null || input_n == null) {
            Console.WriteLine("Жоден з аргументів не може бути пустим");
            return null;
        }
        if (input_a.Length < 2 || input_n.Length < 2) {
            Console.WriteLine("Довжина кожного елемента повинна бути більше двох");
            return null;
        }
        if (input_a.Length != input_n.Length) {
            Console.WriteLine("Довжина обох аргументів повинна дорівнювати");
            return null;
        }
    System.Numerics.BigInteger x = input_a[0];
    System.Numerics.BigInteger nInverse, y, z;
    System.Numerics.BigInteger ni = System.Numerics.BigInteger.One;
     int i = 0;
     while (i < input_a.Length - 1) {
         ni = ni * input_n[i];
         //перевіримо чи n відносно просте
         if (!System.Numerics.BigInteger.One.Equals(input_n[i + 1].gcd(ni))) {
             Console.WriteLine("The n's are not relatively prime->" + input_n[i + 1] + "," + ni);
             return null;
         }
         nInverse = cryptoBig.inverse(n[i + 1], ni);
         if (nInverse.compareTo(0) == -1) {
             nInverse = nInverse + input_n[i + 1];
         }
         z = input_a[i + 1] - x;
         z = z * nInverse;
         y = z.remainder(input_n[i + 1]);
         if (y.compareTo(0) == -1) {
             y = y + n[i + 1];
         }
         x = x + ni * y;
         i++;
     }
     return x; // рішення КТЗ
     }
}
```

Рисунок 4.2 – Код для знаходження рішення

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

        return result[2];
    }
```

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

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

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

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

        return result;
    }
}
```

Рисунок 4.3 – Код для обчислення модульного мультиплікативного звороту

4.5 Шифрування асиметричного ключа

В реалізації використано алгоритм RSAв якості асиметричного алгоритму шифрування ключа для шифрування значення Cr||Ks для кожного авторизованого користувача. Довжина Ks може бути 128, 192 або 256 біти і це для секретного значення Cr має бути обрано так, що загальна довжина Cr||Ks буде менше 1024 бітів.

Це гарантує, що зашифроване значення Cr||Ks обмежене 1024 бітами. Більш того, обмеження довжини значення Cr||Ks призведе до зменшення обчислювальних витрат операцій шифрування і дешифрування RSA.

Платформа .NET надає пакет який містить кілька бібліотек класів, які включають в себе функції, які використовуються для генерації пар ключів RSA і RSA-операцій шифрування/дешифрування. З цих функцій було запрограмовано два файли C#-класів: клас GenerateRSAkey.cs і клас RSAfurBytes.cs.

Клас GenerateRSAkey.cs призначений для створення пари ключів RSA і зберігання їх у файлі. Клас RSAforBytes.cs містить дві функції: rsaEncrypt (byte [ ] data, String PubKeyFile)

для шифрування масиву байтів і rsaDecrypt (byte [ ] data, String PrivKeyFile) для дешифрування цього масиву байтів. Коди двох класів перераховані в Додатку А.

На рисунку 4.4 показано деякі результати генерації пар ключів RSA і шифрування значення Cr||Ks з використанням згенерованих ключів.

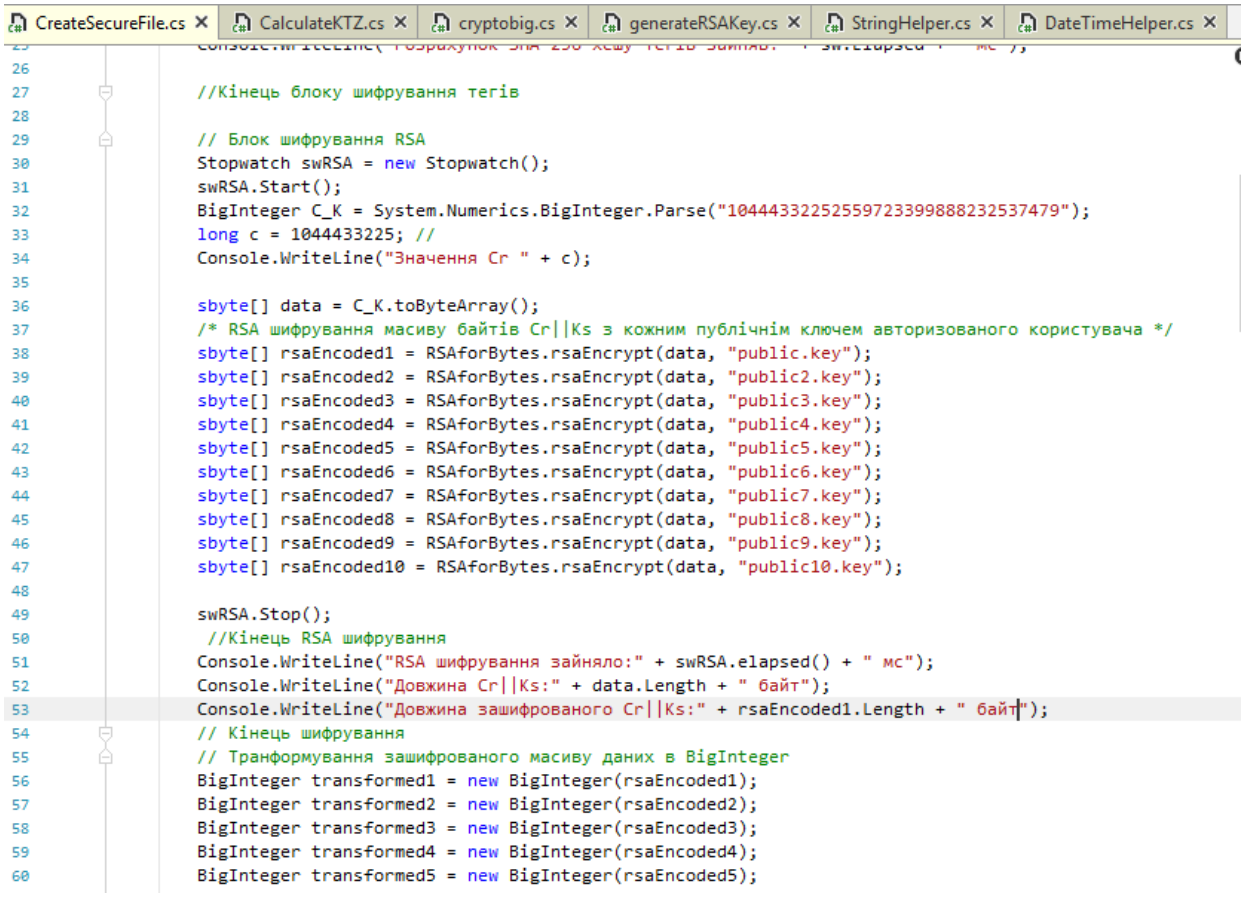


Рисунок 4.4 – Шифруванням RSA для 10 користувачів

Значення Cr||Ks вибране рівне «10444332252559723399888232537479». Це ціле значення перетворене з BigInteger в масив байтів, тому що функція rsaEncrypt запрограмована для шифрування даних в форматі масиву байтів.

Перетворення результувало у 14 байтах, а шифрування цього значення для десяти користувачів зайняло 15 мілісекунд, як показано в знімку екрана на рисунку 4.4.

Отримане зашифроване значення для кожного користувача має довжину 128 байтів (1024 біти), оскільки використаний відкритий ключ генерується з довжиною 1024 біти. Кожен зашифрований вихід повинен бути перетворений назад в BigInteger. Потім отримані зашифровані значення BigInteger будуть використовуватися в якості вхідних даних для

функції CRTforBiglnteger.findSolution (Biglnteger [ ] a, Biglnteger [ ] n), де масив a [ ] містить значення ai і n [ ], містить значення ni, як представлено в рівнянні (3.1).

Потім отримане рішення КТЗ для Xr з findSolution буде прикріплене до захищених даних як частина ОБІ файлу.

4.6 Шифрування ключових слів

В реалізації ключові слова для пошуку зашифровуються з використанням коду повідомлень на основі Hash (HMAC) з алгоритмом хешування SHA256. Клас C# HMACSHA256.C# містить функцію encode (String key, String data), яка обчислює значення HMAC-SHA256 для строкових вхідних даних з ключем, який вводиться у вигляді рядка.

Функція encode повертає зашифрований рядок у форматі байтового масиву довжиною 256 біти.

На рисунку 4.5 показано код функції кодування

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

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

Рисунок 4.5 – Код для вирахування HMAC для ключового слова

Ключ, що використовується для цієї функції, однаковий для всіх ключових слів.

На рисунку 4.6 наведено фрагмент програмного коду для шифрування п’яти ключових слів з використанням функції кодування. Шифрування зайняло 187 мс.

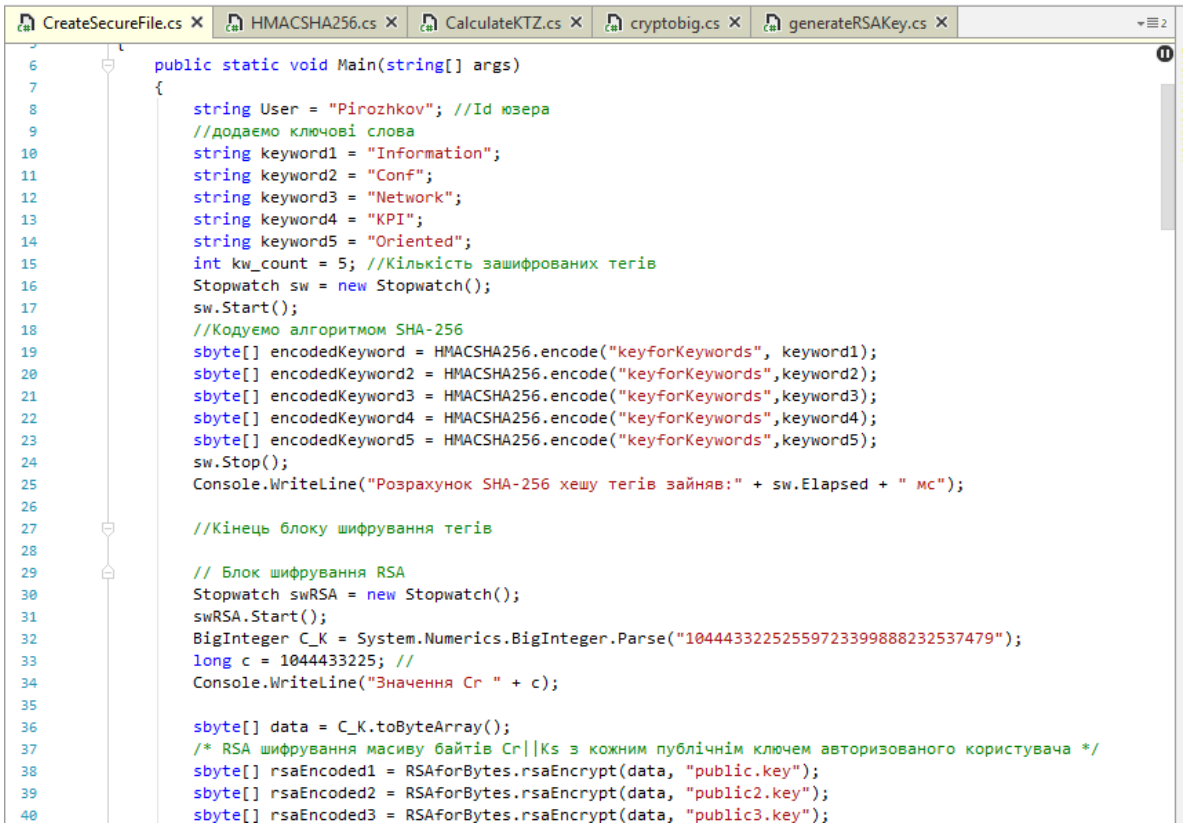


Рисунок 4.6 – Знімок екрану з вирахуванням HMAC-SHA256 для п’яти ключових слів

4.7 Обчислення цілісності та достовірності

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

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

Наприклад, при використанні SHA-256 підсумкова довжина дайджесту становить 256 біти. Потім отриманий дайджест зашифровано з використанням функції шифрування RSA rsaEncrypt (байт [ ] data, String PrivKeyFile), де ключ є закритим ключем власника даних (тобто цифровим підписом власника даних). Значення цифрового підпису буде 1024 біти при використанні 1024-бітного закритого ключа і 2048 біт при використанні 2048-бітного ключа.

Рисунок 4.7 містить програмний код для обчислення значення, FSignature в програмному коді, який використовується для перевірки цілісності та достовірності.

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

Рисунок 4.7 – Код для обчислення доказів цілісності та достовірності

4.8 Створення захищеного ОБІ-файлу власником даних

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

1. Визначене наперед ім’я власника або ID. string User = "Tsyganok Julia "; //Id юзера

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

//додаємо ключові слова

string keyword1 = "Information"; string keyword2 = "Conf";

int kw_count = 2; //Кількість зашифрованих тегів

//Кодуємо алгоритмом SHA-256

sbyte[] encodedKeyword = HMACSHA256.encode("keyforKeywords", keyword1);

sbyte[] encodedKeyword2 = HMACSHA256.encode("keyforKeywords",keyword2);

3. Визначені наперед окремо значення Cr||Ks і Cr. BigInteger C_K =

System.Numerics.BigInteger.Parse("1044433225255972339988823253747 9"); long c = 1044433225;

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

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

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

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

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

sbyte[] rsaEncoded2 = RSAforBytes.rsaEncrypt(data, "public2.key"); // Користувач2

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

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

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

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

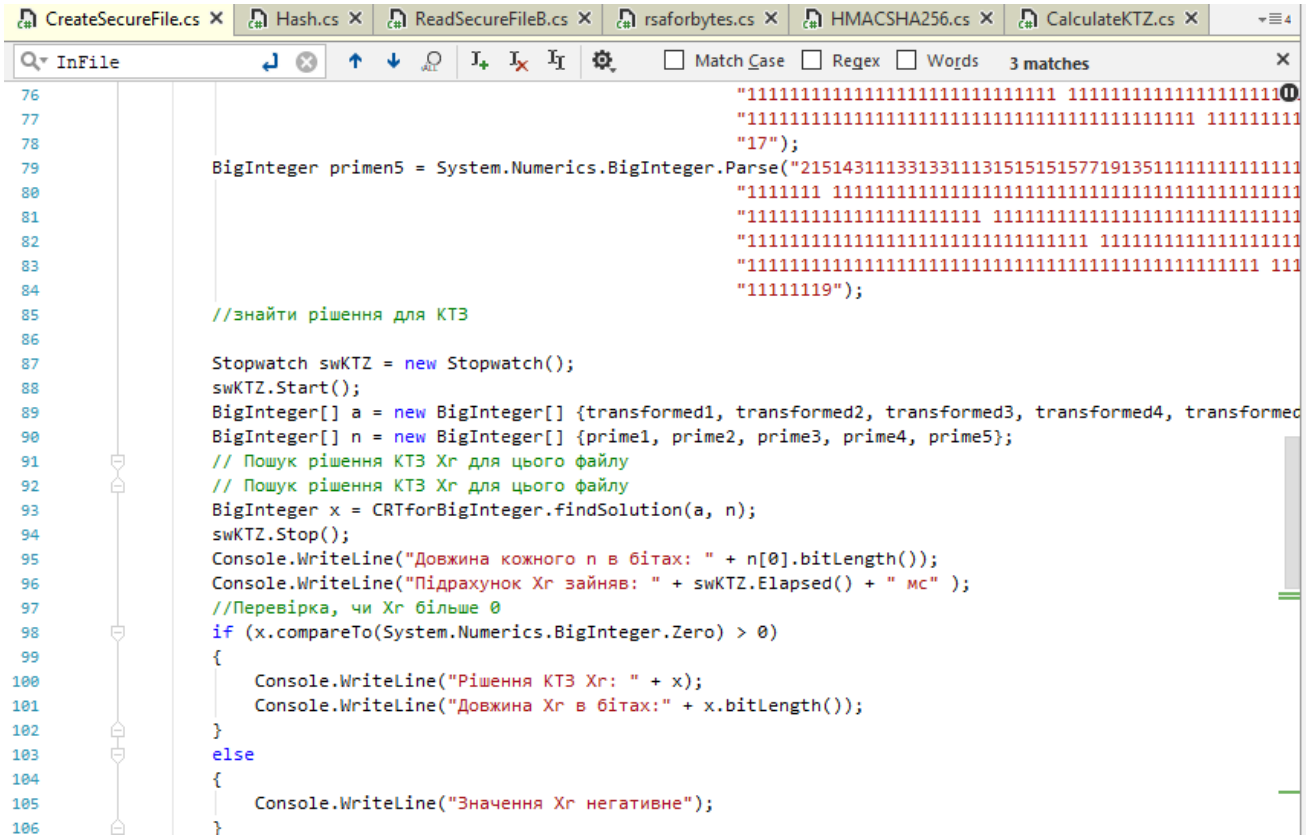


Рисунок 4.8 – Знімок екрану з вирахуванням КТЗ рішення для п’яти користувачів

Після визначення вищевказаних параметрів програма готова до виконання. Виконання спочатку обчислює HMAC-SHA256 ключових слів, як показано на Рисунку 4.6 вище. Потім

вона шифрує значення Cr||Ks з відкритим ключем кожного користувача, приклад якого показано на рисунку 4.8.

Отримані зашифровані значення з відносними цілими числами користувачів використовуються для знаходження загального значення за допомогою функції CRTforBiglnteger.findSolution.

Наприклад, обчислення рішення для п’яти користувачів, де довжина їх відносних простих чисел становить 1025 біти, а значення *Cr||Ks* становило 1024 біта, зайняло близько 5,1 мілісекунди, а отриманий *Xr* був 5122 біти, що отримано кодом на рисунку 4.8.

Потім програма запропонує користувачеві ввести зашифрований файл, який буде перетворений в файл ОБІ з прикріпленими параметрами безпеки. Програма обчислює дайджест вхідного файлу і підписує дайджест особистим ключем власника даних.

На рисунку 4.9 показаний приклад використання розробленої програми для обчислення дайджесту файлу розміром 40,7 МБ і створення цифрового підпису файлу. Це зайняло 780 мілісекунд, коли алгоритм хеша SHA-256 і 1024-бітний закритий ключ RSA були використані для цього конкретного файлу.

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

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

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

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

4.9 Операції на стороні авторизованого користувача

Авторизований користувач вимагає, в основному, чотирьох операцій, тобто обчислення значення Cr||Ks із загального значення Xr, відтворення вихідного зашифрованого файлу з файлу ОБІ, перевірки цілісності та перевірки аутентичності і, нарешті, дешифрування зашифрованого файлу для створення файлу у вигляді нешифрованого тексту.

Всі ці операції реалізовані в одному файлі класу ReadSecureFileB.cs. Користувач повинен визначити наступні параметри. Приклади присвоєних значень параметру наведені для кожного з параметрів:

Унікальний відносно простий користувач

BigInteger n = new BigInteger(“3812938192391”);

Файл, що містить закритий ключ користувача.

string pr_key_file = “pr.key”;

Файл, що містить відкритий ключ власника даних.

String publ_key_file = “publicUser.key”;

Прості і закриті ключі користувача використовуються для обчислення Cr||Ks із значення Xr, а відкритий ключ власника даних використовується для перевірки цілісності та аутентичності файлу ОБІ.

Вирахування секретного значення Cr||Ks із загального значення Xr

Після визначення значення Cr||Ks авторизований користувач ui тепер може використовувати Cr для завершення процедури контролю доступу та Ks для дешифрування вихідного зашифрованого файлу після його завантаження. Оскільки комунікаційна частина процедури контролю доступу не розглядається в цій реалізації, то код читає Xr безпосередньо з файлу ОБІ, поки сервер повинен це зробити і відправити його користувачеві під час процедури контролю доступу. Основна увага в цій частині реалізації була присвячена

вимірюванню накладних витрат обчислень, викликаних обчисленням Cr||Ks на стороні користувача.

Як показано в Таблиці 4.2 вище програмний код виконує в основному дві функції. Перша функція пов'язана з алгоритмом КТЗ, який заснований на простій модульної функції (тобто Xr mod ni), і, отже, розрахунок може бути виконаний дуже швидко.

Наприклад, використовуючи цю функцію, обчислюючи значення Xr з довжиною 5121 біти, для п'яти користувачів, та n з довжиною 1024 біти займає близько 96,565 мікросекунд. Друга функція - це операція розшифровки КТЗ для отриманого зашифрованого значення з першої функції. Ця операція дешифрування зайняла близько 516 мілісекунд.

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

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

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

4.10 Перевірка цілісності та достовірності

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

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

Розшифровка зашифрованого файлу

Після перевірки цілісності та аутентичності зашифрованого файлу даних зашифрований файл даних може бути дешифрований за допомогою програмного забезпечення AES Crypto з ключем Ks для цього файлу. Велика частина обчислювальних

витрат пов’язана з розшифровкою зашифрованого файлу, особливо для великих файлів, як в результатах, показані в пункті 4.5.2.

4.11 Реалізація та експерименти на стороні сервера

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

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

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

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

String path = “C:\\some_fold”;

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

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

4.12 Результати та проблеми вповадженя на стороні власника даних

Накладні витрати визначаються як час, витрачений на обчислення. Результати засновані на припущенні, що всі значення параметрів вже згенеровані, такі як пари ключів RSA, відносні прості числа, значення *Cr* і *Ks*. В цьому пункті основна увага приділяється стороні власника даних і авторизованій користувальницькій стороні, оскільки серверна сторона вже розглянута в попередньому пункті 4.3.

Спочатку файл даних на машині власника буде зашифрований з використанням алгоритму AES і отриманий файл зашифрованих даних стане основною частиною файлу ОБІ.

В таблиці 4.1 показані накладні витрати на зберігання і час виконання шифрування файлів різних типів даних з використанням алгоритму AES з розміром ключа 256 біти.

Таблиця 4.1 – Зберігання та часові накладні витрати для шифрування AES

Назва файлу	Розмір в байтах	Розмір після шифрування в байтах	Збільшення розміру в байтах	AES шифрування в секундах
Sample.docx	14,175	14,485	310	<1
visio.vsd	45,029	45,341	312	<l
photo.jpg	64,819	65,130	318	<1
video1.wmv	25,327,026	25,327,338	312	1.6
video2.mov	41,670,503	41,670,812	309	2.3
video3.avi	136,318,116	136,318,429	313	14.8
video4.mov	598,787,322	598,787,634	311	86.7

Накладні витрати на зберігання складають близько 300 байт для всіх файлів. Збільшується обчислювальне навантаження зі збільшенням розміру файлу, щоб досягти більш ніж однієї хвилини для файлу з розміром 571 МБ. Ця операція є основою для будь-якого методу, заснованого на шифруванні даних, перш ніж дані будуть відправлені в хмару. Крім того, важливо вибрати сильний алгоритм шифрування і довший ключ для досягнення більш високої безпеки. Однак власники даних мають гнучкість, відповідно до їх вимог безпеки, щоб зменшити обчислювальні накладні витрати, використовуючи більш короткий ключ або слабший алгоритм шифрування з меншими обчислювальними витратами. Загалом, процес шифрування несе в собі найбільші обчислювальні витрати на стороні власника даних в запропонованому в цій роботі методі.

На рисунку 4.10 наведено залежність часу AES шифрування від розміру файлу

Рисунок 4.10 – Залежність часу AES шифрування від розміру файлу

Обчислення рішення для визначення значення Xr є суттєвою частиною запропонованого рішення. На основі цієї цінності засновані політики обміну ключами і контролю доступу. На розрахунок КТЗ не впливає розмір файлу, але на нього впливає кількість користувачів і алгоритм асиметричного шифрування, використовуваний для шифрування значення *Cr||Ks*.

В таблиці 4.1 показані витрати часу на зберігання і час при обчисленні КТЗ-рішення для різних користувачів з двох до шести. Ці обчислення виконувалися, коли значення Cr||Ks

було зашифровано з 1024-бітовим RSA і кожне відносне просте число було 1022-бітним. Результати показують, що сама операція КТЗ вимагає незначного часу виконання.

Наприклад, для шести користувачів обчислення КТЗ-рішення зайняло близько 6 мілісекунд. Час виконання обчислення значення обумовлено головним чином процесом шифрування КТЗ і збільшується зі збільшенням кількості користувачів. Однак, оскільки процес шифрування КТЗ виконується для фіксованого значення довжини Cr||Ks, загальний час виконання пошуку все ще незначний і задовільний. Наприклад, загальний час становив 31 мілісекунду для шести користувачів, а загальний час збільшувалася приблизно на 2 мілісекунди для кожного додаткового користувача. Більшість витрат, викликаних операцією КТЗ - це накладні витрати на зберігання, які збільшуються зі збільшенням числа користувачів.

В таблиці 4.2 наведні результати, щодо часових витрат на зберігання та розрахунку загального значення Xr

Таблиця 4.2 – Часові витрат на зберігання та розрахунку загального значення Xr

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

На рисунку 4.11 показано, що на підставі результатів таблиці 4.2 розмір Xr лінійно збільшується приблизно на 1000 біт для кожного додаткового користувача, коли довжина ключа КТЗ дорівнює 1024 біти. Наприклад, для шести користувачів розмір дорівнював 6142 біт.

Рисунок 4.11 – Розмір Xr в бітах в залежності від кількості користувачів

4.13 Додавання зашифрованих ключових слів з можливістю пошуку

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

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

Наприклад, при використанні HMAC-SHA256 для п’яти ключових слів загальна необхідна пам’ять становить 1280 біт. При використанні HMAC-SHA512 для п’яти ключових

слів необхідне сховище буде 2560 біт. Власник даних повинен віднайти компроміс між рівнем безпеки, необхідним для ключових слів і накладними витратами на сховище і час виконання. Також необхідно ретельно визначити кількість необхідних ключових слів, щоб зменшити накладні витрати, не втрачаючи зручність пошуку.

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

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

Додавання перевірки цілісності та аутентичності та створення ОБІ файлу Додавання доказів цілісності та аутентичності в файл ОБІ має важливе значення, але це призводить до додаткових витрат обчислення і зберігання. У таблиці 4.4 показані накладні витрати на зберігання і час виконання при розрахунку цілісності та аутентичності файлів різних типів і розмірів. Хеш-функція, яка використовується для дайджесту зашифрованих файлів була SHA-256, а ключ RSA для шифрування дайджесту був 1024-біт. Через збільшення розміру файлу збільшується обчислювальне навантаження, а накладні витрати на зберігання фіксуються на 1024 біт, що є результатом шифрування 256-бітного дайджесту файлу RSA з довжиною в 1024 біт.

SHA-256 являє собою односпрямовану функцію для створення цифрових відбитків фіксованої довжини (256 біт, 32 байт) з вхідних даних розміром до 2,31 ексабайт (2^{64} біт) і є окремим випадком алгоритму з сімейства криптографічних алгоритмів SHA-2 (Secure Hash Algorithm Version 2) опублікованим АНБ США в 2002 році.

HMAC можна використовувати для визначення того, чи було повідомлення, передане по незахищеному каналу, було змінено за умови, що відправник і одержувач спільно використовують секретний ключ. Відправник розраховує хеш-значення для вихідних даних і відправляє вихідні дані і хеш-значення як одне повідомлення. Одержувач повторно обчислює хеш-значення в отриманому повідомленні і перевіряє, чи відповідає обчислений код HMAC переданому коду HMAC.

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

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

4.14 Створення ОБІ-файлу

Для створення ОБІ-файлу необхідно, щоб всі параметри були вже обчислені. Загальні накладні витрати на зберігання залежать від кількості користувачів, яких авторизовано для доступу до файлу, кількості прикріплених ключових слів і специфікацій використовуваних алгоритмів. У таблиці 4.5 показано, що загальні накладні витрати при створенні ОБІ-файлу складають 952 байта для файлів різного розміру.

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

Вхідне ім’я файлу	Розмір в байтах	Розмір ОБІ-файлу в байтах	Накладні витрати на зберігання в байтах
Sample.docx.aes	14,485	15,437	952
visio.vsd.aes	45,341	46,293	952
photo.jpg.aes	65,130	66,082	952
video1.wmv.aes	25,327,338	25,328,290	952
video2.mov.aes	41,670,812	41,671,764	952
video3.avi.aes	136,318,429	136,319,381	952
video4.mov.aes	598,787,634	598,788,586	952

Файли ОБІ були створені з п’ятьма прикріпленими зашифрованими ключовими словами і повинні використовуватися п’ятьма авторизованими користувачами. Алгоритми використовувалися зі специфікаціями, визначеними в реалізаціях, тобто RSA-1024, AES-256, SHA-256. У разі великих файлів цими накладними витратами на зберігання може бути знехтувано.

Процес створення файлу ОБІ вимагає копіювання вмісту зашифрованого файлу в новий файл ОБІ. Ця операція вимагає додаткового час, як показано в таблиці 4.6. Цього часу можна уникнути, якщо одночасно шифрування AES і обчислення зашифрованого файлового дайджесту виконуються одночасно при створенні ОБІ-файлу.

Іншими словами, вихідний файл даних зашифрований в блоках, а потім кожен блок систематизується і використовується для створення файлу ОБІ за один раунд. Тому на цей раз це пов’язано з реалізацією і не розглядається як накладні витрати щодо запропонованого рішення.

Таблиця 4.6 – Часові витрати копіювання вмісту зашифрованого файлу в ОБІ-файл

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

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

Таблиця 4.7 та рисунок 4.12 показують загальний час виконання цих операцій для різних розмірів файлів. Значення Xr було розраховано для п’яти користувачів і було додано п’ять зашифрованих ключових слів.

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

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

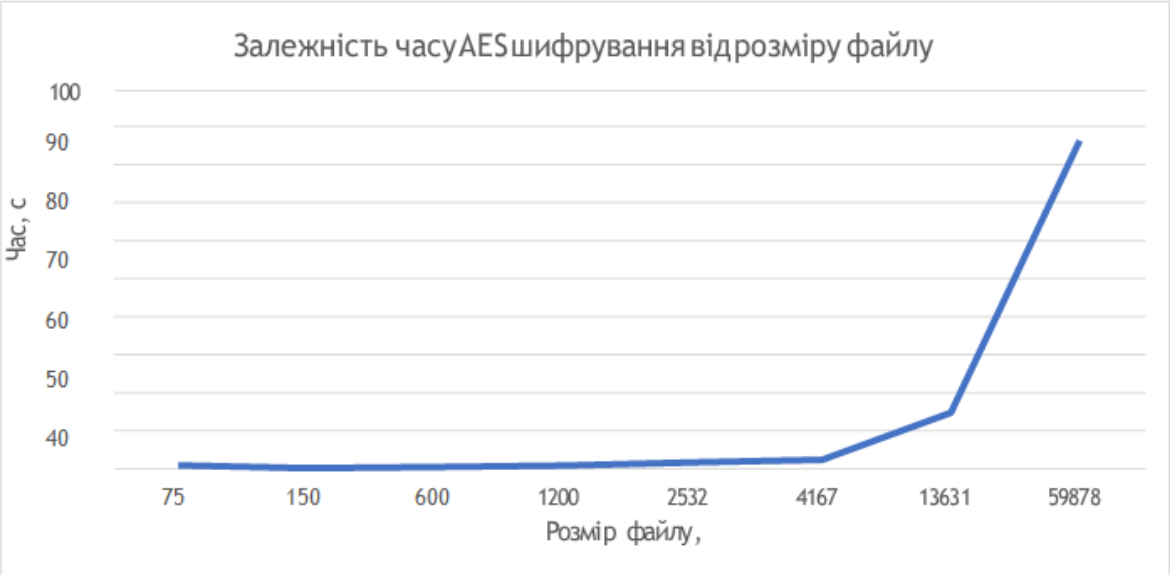


Рисунок 4.12 – Залежність сумарного часу від розміру файла

Загальний час виконання становид менше 4 секунд для файлів розміром менше 50 МБ і близько 20 секунд для файлу розміром 130 МБ. Для файлу в 571 МБ загальний час виконання становив близько 1,7 хвилини. З результатів, показаних в Таблиці 4.7, ясно, що обчислення параметрів, необхідних для контролю доступу, спільного використання ключів і пошуку, має менший вплив на обчислення. На противагу цьому, шифрування AES, за яким слід обчислювати цілісність та достовірність, має найбільший вплив на обчислення. Тому в

запропонованому рішенні операція симетричного шифрування бере на себе більшість обчислювальних накладних витрат, особливо для великих файлів. Однак будь-яке рішення, засноване на шифруванні даних перед відправкою даних в хмару, вимагає принаймні одного раунду шифрування даних.

На стороні авторизованого користувача користувач повинен провести три операції: обчислення значення Cr||Ks із значення Xr, перевірку цілісності та аутентичності зашифрованого файлу даних і дешифрування файлу зашифрованих даних.

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

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

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

4.15 Порівняльний аналіз отриманих результатів

На рисунку 4.13 – Наведено результати дослідження залежності часу AES дешифрування від розміру файлу.

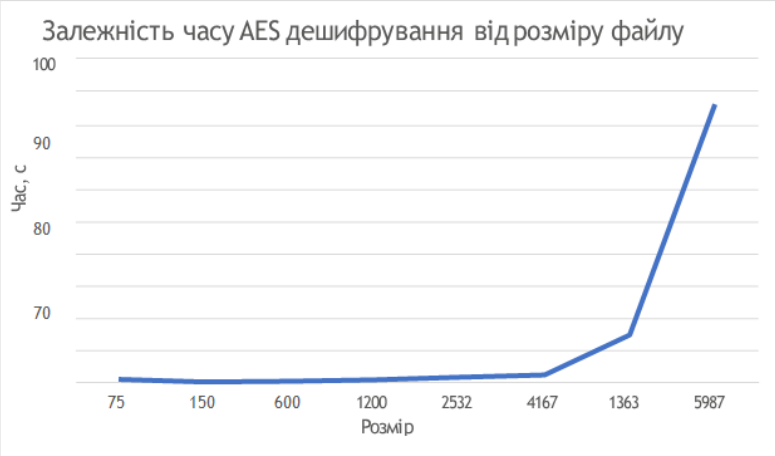


Рисунок 4.13 – Залежність часу AES дешифрування від розміру файлу

В таблиці 4.9 та на рисунку 4.14 наведені результати роботи інформаційно-орієнтованої системи.

Таблиця 4.9 – Результати роботи інформаційно-орієнтована системи

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

Рисунок 4.14 – Швидкість роботи системи

Далі для дослідження розмір двох файлів доводяться до приблизно такого ж розміру, а потім апроксимуються до точок 750 КБ, 1500 КБ. Результати порівняння предствлені на рисунку 4.15.

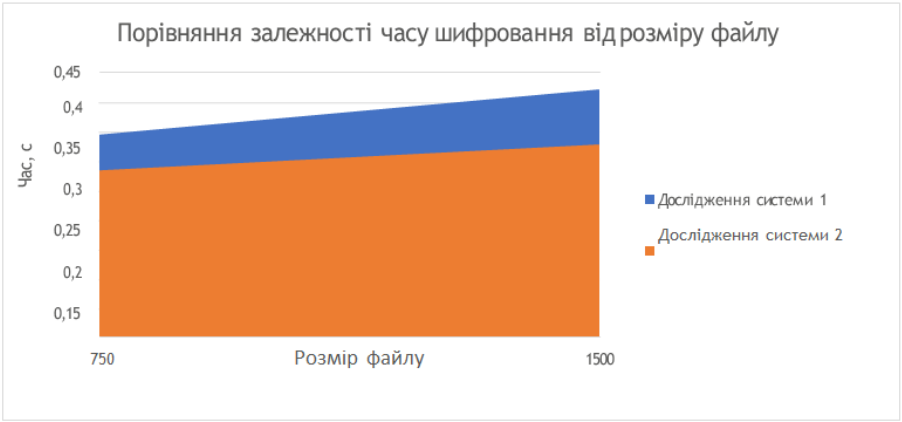


Рисунок 4.15– Порівняння швидкості рботи для малих файлів

4.16 Висновки до розділу 4

У четвертому розділі проведено дослідження та реалізація запропонованого рішення. Основна увага приділялася навантаженню на зберігання і обчислення, викликаним використанням запропонованого рішення з боку клієнта і сервера. Для обох сторін була встановлена експериментальна платформа, і для реалізації основних операцій запропонованого рішення була використана платформа Rider від JetBrains для мови програмування C#. Для операцій власника даних, основний код для створення файлу ОБІ з зашифрованого файлу був записаний у файлі класу C# з ім'ям CreatSecureFile.cs. Цей клас обчислює загальний занчення Xr для будь-яких користувачів, викликавши функцію findSolution з CRTfurBiginteger.cs, який було реалізовано для обчислення теорем про залишки при використанні цілих чисел розміром більше 64 бітів.

Крім того, було створено CreatSecureFile для обчислення інших параметрів безпеки, а саме зашифрованих ключових слів для можливості пошуку, перевірки цілісності та достовірності. Потім він прив'язав всі ці параметри до файлу ОБІ таким чином, який можна легко прочитати пізніше. ReadSecureFile.cs був реалізований для читання параметрів безпеки в ОБІ-файлі і відтворення вихідної зашифрованої файлової форми. Файл також містить операції, необхідні на стороні авторизованого користувача, а саме, виявлення секретних значень *Cr,Ks* з *Xr,* перевірки цілісності та аутентичності зашифрованого файлу.

Відкрите вихідне програмне забезпечення AES Crypto використовувалося для шифрування вихідного файлу на стороні власника даних, а потім для його дешифрування на стороні користувача.

Файл класу Search.cs реалізований на сервері для пошуку заданого ключового слова в файлах ОБІ. Для підвищення продуктивності при створенні ОБІ- файлу була додана інформація про те, скільки ключових слів і яка їх довжина. Отже, функція пошуку може обчислювати діапазон всередині ОБІ-файлу, який містить зашифровані ключові слова, тому процес пошуку буде обмежений цим діапазоном.

Таким чином, час пошуку не буде залежати від довжини файлу. Крім того, використовувався алгоритм пошуку KMP для підвищення швидкості пошуку. Результати проведених операцій показують, що запропоноване рішення є простим і не вимагає складних операцій. Крім того, накладні витрати на зберігання при створенні файлу ОБІ є низькими в порівнянні з наданими функціями. Цими функціями є: створення ОБІ-файлу з можливостями пошуку, політикою прихованого контролю доступу та цілісністю і достовірністю, незалежно від того, де він зберігається в хмарі. Запропоноване рішення може бути практично реалізовано з мінімальними витратами на обчислення і зберігання. Запропоноване рішення є ефективним, так як воно не вимагає складних методів розподілення ключів і файл даних не потрібно шифрувати більше одного разу.

4.17 Перелік посилань до розділів 2-4

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

2. Mell P. The NIST Definition of Cloud Computing [Електронний ресурс] / P. Mell, T. Grance // National Institute of Standards and Technology. – 2018. – Режим доступу до ресурсу: https://csrc.nist.gov/publications/detail/sp/800-145/final (дата звернення 05.12.2019)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

17. big-O notation [Електронний ресурс] // U.S. National Institute of Standards and Technology. – 2004. – Режим доступу до ресурсу: https://xlinux.nist.gov/dads/HTML/bigOnotation.html (дата звернення 01.12.2019)

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

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

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

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

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

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

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

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

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

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

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної магістерської роботи є розроблення методів та засобів забезпечення безпеки даних у хмарному середовищі.

Для організації роботи над проектом створено таблицю 5.1 – розміри приміщення, таблицю 5.2 - характеристики робочого місця, таблицю 5.3 – аналіз небезпечних і шкідливих виробничих факторів, таблицю 5.4 – норми мікроклімату робочої зони об’єкту та проведені необхідні розрахунки.

5.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці»[4] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

5.1.1 Правові та організаційні основи охорони праці

Законодавство України про охорону праці є системою взаємозв'язаних нормативних актів, що регулюють відносини у галузі реалізації державної політики щодо правових, соціально-економічних, організаційно-технічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

Конституція України – основний закон держави, який регламентує найважливіші з погляду держави суспільні відносини.

5.1.2 Організаційно-технічні заходи з безпеки праці

Технічні заходи - технічні засоби, що забезпечують безпечні і нешкідливі умови праці, та пов'язані з впровадженням нового обладнання, пристроїв і приладів безпеки і безпечною експлуатацією засобів виробництва.

Організаційні заходи:

Контроль за технічним станом обладнання, інструментів, будівель і споруд; контроль за дотриманням вимог нормативних документів з охорони праці; нагляд за обладнанням підвищеної небезпеки; організація навчання.

Санітарно-гігієнічні заходи:

контроль за впливом виробничих факторів на здоров'я працівників;забезпечення санітарно-побутових умов згідно з діючими нормами;планування заходів щодо поліпшення санітарно-гігієнічних умов праці.

Соціально-економічні заходи:

надання пільг і компенсацій працівникам, які працюють зі шкідливими і небезпечними умовами праці; створення умов для економічної зацікавленості роботодавця і працівника у поліпшенні умов і підвищенні безпеки праці.

5.2 Аналіз стану умов праці

Робота над створенням автоматизованої системи перевірки якості JavaScript-коду проходитиме в приміщенні компанії. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

5.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 5.1.

Таблиця 5.1 – Розміри приміщення

Найменування	Значення
Довжина, м	6
Ширина, м	4
Висота, м	2,7
Площа, $м^2$	24
Об’єм, $м^3$	64,8

Розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

5.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за . «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2] і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 5.2 - Характеристики робочого місця

Найменування параметра	Наймен. параметра	Нормативне значення
Висота робочої поверхні, мм	690	680 ÷800
Висота простору для ніг, мм	600	не менше 600
Ширина простору для ніг, мм	580	не менше 500
Глибина простору для ніг, мм	650	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	450	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Приміщення кабінету знаходиться на третьому поверсі чотирьох поверхової будівлі і має об'єм 64,8 м^3, площу – 24 м^2. У цьому кабінеті обладнано два місця праці, укомплектовані ПК.Температура в приміщенні протягом року коливається у межах 25-32°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум в

лабораторії знаходиться на рівні 45 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

Розміщення вікон забезпечує природне освітлення з коефіцієнтом природного освітлення не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою шести люмінесцентних ламп, забезпечує рівень освітленості не менше 200 Лк.У кабінеті є електрична мережа з напругою 220 В, яка створює небезпеку ураження електричним струмом. ПК та периферійні пристрої можуть бути джерелами електромагнітних випромінювань, аерозолів та шкідливих речовин (часток тонеру, оксидів нітрогену та озону).За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5 .

5.2.3 Навантаження та напруженість процесу праці

Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи. Роботу над дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви: для операторів персональних комп’ютерів тривалістю 15 хв через дві години роботи.

5.3 Виробнича санітарія

Виробнича санітарія - це система організаційних заходів і технічних засобів, що запобігають або зменшують вплив на працюючих шкідливих виробничих факторів, які в певних умовах можуть привести до травм або професійних захворювань. Основною метою є зменшення або повне усунення впливу несприятливих 1 шкідливих виробничих факторів на організм людини. Оскільки головним у діяльності з охорони праці є профілактика травматизму, заходів щодо поліпшення умов праці й побуту працюючих.

5.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з електронними пристроями» [3], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ПК з ВДТ і ПП. Основними робочими характеристиками ПК є наступні:

- робоча напруга U = +220В ± 5%;
- робочий струм I = 2А;
- потужність споживання P = 350 Вт.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 5.3).

Таблиця 5.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ПК, принтерів, сканерів чи/або серверного обладнання для роботи	2	ДСН 3.3.6.042-99[1]
- підвищена яскравість світла	порушення умов праці (організації місця прац і-налагодження моніторів)	2	ДСанПіН 3.3.2.007-98[2]
- понижена контрастність	-//-	1	ДСанПіН 3.3.2.007-98[2]
психофізіологічні			
- нервово-психічне перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз	3	НПАОП 0.00-7.15-18[3] ДСанПіН 3.3.2.007-98[2]

	алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи		
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці - сидіння користувача,) та організації робочого часу - безперервна робота)	2	НПАОП 0.00-7.15-18 [3] ДСанПіН 3.3.2.007-98[2]

5.3.2 Пожежна безпека

Електронно-обчислювальна машина (ЕОМ; далі — комп'ютер) — обладнання з необов'язковими додатковими пристроями (пристрої для друку, сканери, модеми, блоки безперервного живлення та інші спеціальні периферійні пристрої). Відеодисплейний термінал (ВДТ; далі — монітор) — частина електронно-обчислювальної машини, що містить пристрій для візуального відображення інформації; Периферійні пристрої (далі — ПП) — сукупність необов'язкових додаткових пристроїв, які використовуються в процесі діяльності оператора ЕОМ (клавіатура, маніпулятор «миша», дискова система, звукова система, модем, мікрофон, принтер, сканер тощо).

На підприємстві, де експлуатується комп'ютерна техніка, створюється служба охорони праці згідно з «Типовим положенням про службу охорони праці»[9].

Нормативна база

Перелік нормативно-правових актів, які регулюють це питання, досить широкий. Наприклад, ст. 21 Кодексу законів про працю України визначає обов'язки роботодавця щодо забезпечення працівникам комфортних та безпечних умов праці, а ст. 13Закону України «Про охорону праці» закріплює це право з позиції охорони праці.

Особливості охорони праці при роботі з комп'ютером.

Комп'ютерне обладнання повинні підключатися до електромережі лише за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їх конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним.

Не допускається підключати комп’ютерну техніку до звичайної двопровідної електромережі, зокрема з використанням перехідних пристроїв.

5.3.3 Електробезпека

Основні шкідливі та небезпечні фактори, що можуть впливати на організм людини під час роботи з персональним комп'ютером (ПК), такі, як підвищений рівень електромагнітних випромінювань, підвищений рівень іонізуючих випромінювань, підвищена чи знижена іонізація повітря, підвищена яскравість світла, пряма і відбита блискітливість.

5.4 Гігієнічні вимоги до параметрів виробничого середовища

5.4.1 Параметри мікроклімату

Мікроклімат виробничих приміщень - це сукупність параметрів повітря у виробничому приміщенні, які діють на людину у процесі праці, на його робочому місці, у роб зоні. Робоче місце - територія постійного або тимчасового знаходження людини у процесі праці. Робоча зона - частина простору робочого місця, обмежене по висоті 2 м від рівня підлоги. Параметри мікроклімату це температура повітря Т, 0° С; відносна вологість Y, % та швидкість руху повітря V, м/с.

Значні коливання параметром мікроклімату можуть привести до порушення терморегуляції організму (здатність організму утримувати постійну температуру), що приводить до порушення системи кровообіг, загальної слабкості. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають «Санітарні норми мікроклімату виробничих приміщень» [1] і наведені в табл. 5.4.

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура С0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 2А	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються

оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря .

5.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2+b^2}\cdot S_b = (1/8 \div 1/10)\cdot S_n \qquad (5.1)$$

де S_b – площа віконних прорізів, м2;

s– площа підлоги, м2.

$$s_n = a \times b = 4 \times 6 = 24 \text{ м}^2$$

$$S_{вік} = \frac{1}{8} \times 24 = 3 \text{ м}^2$$

Приймаємо 2 вікна площею 6 м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 6 м, ширина 4 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників п виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{5.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 24 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400 лм (для ЛБ-80).

Підставивши числові значення у формулу (5.2), отримуємо:

$$\mathrm{n} = \frac{300 \times 24 \times 1{,}1 \times 1{,}5}{5400 \times 0{,}575 \times 2} \approx 2$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

5.4.3 Шум та вібрація, електромагнітне випромінювання

Найшкідливішим фактором виробничого і побутового середовищ є шум. Особливо небезпечним є виробничий шум, дія якого є тривалою, тобто постійно супроводжує виробничий процес.

Виробничий шум – сукупність різноманітних за силою і частою звуків. Джерелом шуму можуть бути двигуни, насоси, вентиляційні пристрої, холодильне обладнання, компресори, деякі технологічні процеси – обрубка металу, його клепання, карбування, штамповка, робота на ткацьких верстатах, випробування двигунів, будівельні роботи. Постійним джерелом шуму є транспорт, особливо міський (залізниця, метро, автомобілі) і на сьогодні на вулицях великих міст рівень шуму досягає 80-90 Дб.

Вібрація та її вплив на організм.

Джерела вібрації на виробництві – це пневмо - та електроінструменти ударної або обертальної дій, машини, які установлені на основі без достатньої амортизаційної прокладки, а також транспортні і сільськогосподарські машини.

Вплив вібрації на організм людини залежить від локальної інтенсивності вібраційних хвиль, що викликає зміни стану тканин і органів (стиснення й розтягнення, скручування й згин, утруднення кровопостачання, посилення або послаблення згортальних властивостей крові та ін). Послабити дію вібрації на людину можна засобами віброгасіння, вібропоглинання та віброізоляції.

Електромагнітні хвилі та їх вплив на організм.

Електромагнітні хвилі різного діапазону частот широко використовують в радіолокації, телебаченні, радіозв'язку, фізіотерапії, для термічної обробки металів, приготування їжі, сушки деревини тощо. Їх джерелами є також високовольтні лінії електропередач, електротранспорт. Електромагнітні поля мають певну потужність, енергію і поширюються у вигляді електромагнітних хвиль. Основним параметрами електромагнітних коливань є:

- довжина хвилі;
- частота коливань;
- швидкість розповсюдження.

За частотою антропогенні електромагнітні випромінювання поділяють на:

- низькочастотні (НЧ, 0,003 Гц-30 кГц);
- радіохвилі високочастотного (ВЧ) діапазону (30кГц-300 мГц);
- радіохвилі ультрависокочастотного (УВЧ) діапазону (30мГц-300 мГц);
- надвисокочастотні (НВЧ) хвилі (30мГц-300 гГц).

5.4.4 Вентилювання

Види вентиляції промислових приміщень

Залежно від способу переміщення повітря промислова вентиляція, як і вентиляція приватного будинку, може бути природною і механічною.

Для вентиляції промислових приміщень застосовують припливну, витяжну або припливно-витяжну механічні системи. На великих виробництвах користуються тільки останнім варіантом. У приватних майстернях і невеликих цехах зазвичай встановлюють витяжну механічну вентиляцію. Приплив свіжого повітря у такому випадку відбувається шляхом аерації або інфільтрації, тобто різновидів природної вентиляції. У невеликому виробничому приміщенні витяжної механічної вентиляції зазвичай достатньо.

5.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Заходи безпеки під час експлуатації персонального комп’ютера та периферійних пристроїв передбачають:

- дотримання заходів електробезпеки;

- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення).

Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів та постійно стежити за справністю ізоляції електромережі та мережевих кабелів.

Вимоги безпеки при надзвичайних ситуаціях:

1) При раптовому припиненні подачі електричної енергії вимкнути всі пристрої ПК в такій послідовності.Витягнути вилки з розеток. При наявності ознак горіння (дим, запах горілого) необхідно вимкнути всі пристрої ПК, знайти місце загоряння і виконати всі можливі заходи для його ліквідації.

2) При замиканні, перевантаженні електричного струму на електричному обладнанні, внаслідок ураження грозової блискавки та ймовірної небезпеки ураженням електричним струмом, приймають наступне:

- попередження замикання здійснюється правильним вибором, монтажем експлуатації мереж;

- застосування захисту схем у вигляді швидкодіючих реле, а також вимикачів, плавких запобіжників, автоматичних вимикачів.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п} \tag{5.3}$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

Rш - опір опори, яке з'єднує заземлювачі, Ом;

п - кількість заземлювачів;

ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 - 0,9; ηз = 0,5

ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 - 0,7; ηш = 0,3;

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right) \quad (5.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 - 700 Ом·м; приймаємо ρ = 500 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 2,5 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,04 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 3 м.

$$\mathrm{R_z} = \frac{500}{2 \times 3{,}14 \times 2{,}5}\left(\ln \frac{2 \times 2{,}5}{0{,}04} + \frac{1}{2} \ln \frac{4 \times 3 + 2{,}5}{4 \times 3 - 2{,}5}\right) = 31{,}8 \times (4{,}8 + 0{,}2) = 159\text{ , Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_ш = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1} \quad (5.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: Пбуд. = 32·2 + 25·2 =114 м; L = 114 м;

b - ширина смуги, м; b = 0,03 м;

t1 - глибина заземлення від рівня землі, м; t1 = 0,4 м.

$$\mathrm{R_n} = \frac{500}{2 \times 3{,}14 \times 114} \times \ln \frac{2 \times 114^2}{0{,}03 \times 0{,}4} = 0{,}69 \times 14{,}36 = 9{,}9\text{ , Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з} \quad (5.6)$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{zzn} = \frac{159 \times 9{,}9}{9{,}9 \times 79 \times 0{,}6 + 159 \times 0{,}4} = \frac{1574{,}1}{532{,}8} = 2{,}9 \text{ Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: Rззп < 4 Ом.

5.6 Охорона навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: розробка автоматизованої системи перевірки якості JavaScript-коду, в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [5], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [6].Законом України «Про відходи» [7], Законом України «Про охорону атмосферного повітря»[10], Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру»[11], Водний кодекс України [8].

В процесі діяльності за комп,ютером виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- побутові відходи - IV клас небезпеки
- матеріали пакувальні, що вміщують п/ет, п/пр - IV клас небезпеки
- відходи друкуючих пристроїв - IV клас небезпеки

Загалом відходи сфер виробництва і сфери споживання залежно від фізичних, хімічних і біологічних характеристик усієї маси відходу або окремих його інгредієнтів поділяються на чотири класи небезпеки:

I-й клас — речовини (відходи) надзвичайно небезпечні;

II-й клас — речовини (відходи) високо небезпечні;

III-й клас — речовини (відходи) помірно небезпечні;

IV-й клас — речовини (відходи) мало небезпечні.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари .

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» [7] повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів.

5.7 Висновки до розділу 5

В даному розділі розроблені рекомендації з охорони праці, техніки безпеки при роботі на комп'ютері. Проведений аналіз умов праці, вплив шкідливих та небезпечних чинників на здоров'я людини. Визначено параметри і характеристики приміщення. Приведені рекомендації щодо організації робочого місця, електробезпеки та пожежної безпеки. Наведені розміри приміщення та значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики та зроблені висновки щодо екології навколишнього середовища.

5.8 Перелік посилань до розділу 5

1. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. Постанова N 42 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

2. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин Затверджено Постановою Головного державного санітарного лікаря України 10 грудня 1998 р. N 7. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з електронними пристроями Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за № 508/31960. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

4. Закон України «Про охорону праці» Вводиться в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669. - Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/2694-12

5. Закон України «Про охорону навколишнього природного середовища».Вводиться в дію Постановою ВР № 4005-XII від 24.02.94, ВВР, 1994, № 27, ст.219. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/4004-12

6. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення» Відомості Верховної Ради України (ВВР), 1994, № 27, ст.218){Вводиться в дію Постановою ВР № 4005-XII від 24.02.94, ВВР, 1994, № 27, ст.219

7. Закон України «Про відходи» Відомості Верховної Ради України (ВВР), 1998, № 36-37, ст.242. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/187/98-вр

8. Кодекс водний України Вводиться в дію Постановою ВР № 214/95-ВР від 06.06.95, ВВР, 1995, № 24, ст.190 Режим доступу: https://zakon1.rada.gov.ua/laws/show/213/95-%D0%B2%D1%80

9. НПАОП 0.00-4.35-04 Типове положення про службу охорони праці . Із змінами, внесеними згідно з Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду N 236 (z1191-07) від 02.10.2007 Наказом Міністерства соціальної політики N 148 (z0236-17) від 31.01.2017 Режим доступу: https://zakon.rada.gov.ua/laws/show/z1526-04

10. Закон України «Про охорону атмосферного повітря» Відомості Верховної Ради України (ВВР), 1992, № 50, ст.678 Режим доступу: https://zakon.rada.gov.ua/laws/show/2707-12

11. Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру». Відомості Верховної Ради України (ВВР), 2000, N 40, ст.337 Режим доступу: https://zakon.rada.gov.ua/laws/show/1809-14#o8

ВИСНОВКИ

В роботі розглянуто та досліджено концепцію орієнтованої на безпеку інформації (ОБІ) як важливий підхід до підвищення безпеки та приватності даних, що зберігаються клієнтами в хмарних обчислювальних системах, особливо в середовищі публічних хмарних обчислень, де дані можуть часто переміщатися з одного хмарного сервера на інший і можуть бути доступні іншим кінцевим об’єктам.

Основна увага приділялась захисту фактичних даних від можливих ризиків безпеки в хмарному середовищі.

Запропонована концептуальна основа для застосування підходу ОБІ, метою якого є підвищення безпеки і приватності даних в хмарі.

Запропонована структура заснована на визначенні вимог безпеки безпосередньо у фактичних даних, тому дані є самоописовими, самозахищеними та самоохороняємими протягом усього їх життєвого циклу. Власник даних зберігає повну відповідальність за визначення і управління безпекою та приватністю своїх даних, переданих в хмару. Крім того, приватність даних і приватність користувачів, які звертаються до даних, залишаються захищеними навіть від хмарних провайдерів. Захист даних не залежить від надійності провайдера хмарних обчислень і не вимагає надійності третьої сторони. Очікується, що пропонована концептуальна структура додасть надійний рівень безпеки для даних у хмарі, і, отже, потенційні користувачі можуть мати більше впевненості в переміщенні своїх даних в хмарні сховища.

В результаті підвищення рівня безпеки та приватності, що досягається за допомогою запропонованої структури, переваги використання хмарних сервісів можуть бути досягнуті без втрати контролю над безпекою та приватністю даних власником даних або оприлюднення даних за можливих порушень безпеки і приватності, напротивагу зберіганню даних у власному приватному обчислювальному середовищі користувача.

Проведені в роботі дослідження технології захисту даних у хмарному середовищі забезпечують більш чітку і ширшу концептуальну структуру ОБІ для застосування підходу ОБІ до хмарної моделі з трьома основними критеріями:

− всі заходи безпеки і вимоги надаються з самого набору даних;

− власник даних відповідає за конфігурацію, управління і моніторинг даних та їх характеристик безпеки протягом усього терміну служби даних;

− безпека і приватність даних і користувачів, які мають доступ до даних, не покладаються на провайдерів хмарних обчислень або довірених третіх осіб.

Запропоновано рішення, що засновано на створенніОБІ-файлу для підвищення безпеки і приватності будь-якого типу файлів даних, таких як документи, зображення і відео, що зберігаються і спільно використовуваних в публічному хмарному сховище.

Кожен файл ОБІ містить зашифрований вихідний файл даних і набір параметрів безпеки, які використовуються для забезпечення дотримання політик контролю доступу, пошуку ключових слів і перевірки цілісності та аутентичності файлу даних.

Запропоноване рішення забезпечує платформу, незалежну від обчислювального середовища, і підходить для складної і динамічної природи хмарної середовища, особливо у випадку публічної хмари.

Файл ОБІ має свій зашифрований файл даних, а вимоги безпеки до нього прив’язані як параметри безпеки, які також відсилають до метаданих безпеки. Кожен ОБІ-файл має свої незалежні параметри безпеки, включаючи докази цілісності та аутентичності, прикріплені до нього, і може управлятися індивідуально власником даних. Тому власники даних можуть бути впевнені в безпеці своїх файлів ОБІ, навіть якщо файли передаються між різними хмарними провайдерами та різними географічними локаціями. Незалежно від хмарного сервера, в якому знаходиться ОБІ-файл в хмарному середовищі, авторизовані користувачі можуть здійснювати пошук, доступ та перевірку цілісності, оскільки вони вбудовані в файл і не залежать від обчислювального середовища. Більш того, коли провайдер хмари створює копії для резервного копіювання або поліпшення доступу для декількох користувачів або з яких-небудь інших причин, параметри безпеки вихідного файлу ОБІ підтримуються скопійованими версіями.

Результати, отримані при тестуванні реалізованих операцій запропонованого рішення, показують, що реалізація, а отже і відповідне рішення, проста і не вимагає складних операцій, що вимагають великих обчислювальних ресурсів, а також високого обслуговування.

Крім того, запропоноване рішення може бути практично використане для будь-якого типу файлу даних з помірними накладними витратами з точки зору зберігання і обчислювальних ресурсів. Крім того, запропоноване рішення дозволяє власнику даних гнучко вибирати алгоритми шифрування і хешування на основі необхідних сильних сторін безпеки і додатків.

Результати роботи та запропоновані рішення можуть бути використані у навчальному процесі кафедри комп’ютерних наук та інженерії при вивченні дисципліни «Захист інформації в комп’ютерних системах».

ДОДАТОК А

Лістинг програми

```
using System;
using System.Diagnostics;
public class CreateSecureFile
{
public static void Main(string[] args)
{
string User = "Tsyganok Julia"; //Id юзера
//додаємо ключові слова
string keyword1 = "Information";
string keyword2 = "Conf";
int kw_count = 2; //Кількість зашифрованих тегів
Stopwatch sw = new Stopwatch();
sw.Start();
//Кодуємо алгоритмом SHA-256
sbyte[] encodedKeyword = HMACSHA256.encode("keyforKeywords", keyword1);
sbyte[] encodedKeyword2 = HMACSHA256.encode("keyforKeywords",keyword2);
sw.Stop();
Console.WriteLine("Розрахунок SHA-256 хешу тегів зайняв:" + sw.Elapsed + " мс");
//Кінець блоку шифрування тегів
// Блок шифрування RSA
Stopwatch swRSA = new Stopwatch();
swRSA.Start();
BigInteger C_K = System.Numerics.BigInteger.Parse("10444332252559723399888232537479");
long c = 1044433225;
Console.WriteLine("Значення Cr " + c);
sbyte[] data = C_K.toByteArray();
/* RSA шифрування масиву байтів Cr||Ks з кожним публічнім ключем авторизованого
користувача */
sbyte[] rsaEncoded1 = RSAforBytes.rsaEncrypt(data, "public.key");
sbyte[] rsaEncoded2 = RSAforBytes.rsaEncrypt(data, "public2.key");
sbyte[] rsaEncoded3 = RSAforBytes.rsaEncrypt(data, "public3.key");
sbyte[] rsaEncoded4 = RSAforBytes.rsaEncrypt(data, "public4.key");
sbyte[] rsaEncoded5 = RSAforBytes .rsaEncrypt(data, "public5.key");
sbyte[] rsaEncoded6 = RSAforBytes.rsaEncrypt(data, "public6.key");
sbyte[] rsaEncoded7 = RSAforBytes.rsaEncrypt(data, "public7.key");
sbyte[] rsaEncoded8 = RSAforBytes.rsaEncrypt(data, "public8.key");
sbyte[] rsaEncoded9 = RSAforBytes.rsaEncrypt(data, "public9.key");
sbyte[] rsaEncoded10 = RSAforBytes.rsaEncrypt(data, "public10.key");
swRSA.Stop();
//Кінець RSA шифрування
Console.WriteLine("RSA шифрування зайняло:" + swRSA.elapsed() + " мс");
Console.WriteLine("Довжина Cr||Ks:" + data.Length + " байт");
Console.WriteLine("Довжина зашифрованого Cr||Ks:" + rsaEncoded1.Length + " байт");
// Кінець шифрування
// Транформування зашифрованого масиву даних в BigInteger
BigInteger transformed1 = new BigInteger(rsaEncoded1);
BigInteger transformed2 = new BigInteger(rsaEncoded2);
```

```
BigInteger transformed3 = new BigInteger(rsaEncoded3);
BigInteger transformed4 = new BigInteger(rsaEncoded4);
BigInteger transformed5 = new BigInteger(rsaEncoded5);
// Відносні прості числа
BigInteger prime1=
System.Numerics.BigInteger.Parse("215143111331331113151515157719135111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111");
BigInteger prime2 =
System.Numerics.BigInteger.Parse("215143111331331113151515157719135111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111113");
BigInteger prime3 =
System.Numerics.BigInteger.Parse("215143111331331113151515157719135111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111115");
BigInteger prime4 =
System.Numerics.BigInteger.Parse("215143111331331113151515157719135111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111117");
BigInteger primen5 =
System.Numerics.BigInteger.Parse("215143111331331113151515157719135111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111119");
//знайти рішення для КТЗ
Stopwatch swKTZ = new Stopwatch();
swKTZ.Start();
BigInteger[] a = new BigInteger[] {transformed1, transformed2, transformed3, transformed4,
transformed5};
BigInteger[] n = new BigInteger[] {prime1, prime2, prime3, prime4, prime5};
// Пошук рішення КТЗ Xr для цього файлу
// Пошук рішення КТЗ Xr для цього файлу
BigInteger x = CRTforBigInteger.findSolution(a, n);
swKTZ.Stop();
Console.WriteLine("Довжина кожного n в бітах: " + n[0].bitLength());
Console.WriteLine("Підрахунок Xr зайняв: " + swKTZ.Elapsed() + " мс" );
//Перевірка, чи Xr більше 0
if (x.compareTo(System.Numerics.BigInteger.Zero) > 0)
{
Console.WriteLine("Рішення КТЗ Xr: " + x);
Console.WriteLine("Довжина Xr в бітах:" + x.bitLength());
}
```

```
else
{
Console.WriteLine("Значення Xr негативне");
}
//вкажіть зашифрований файл для створення ОБІ-файлу
Console.Write("Введіть ім'я файла з розширенням: ");
string InputFile;
Console.ReadLine(InputFile);
File InFile = new File(InputFile); //input file
if (!InFile.exists())
{
Console.WriteLine("File does not exist.");
Environment.Exit(0);
}
// Розрахунок вхідного дайджесту файла і сигнатури власника даних
string private_key_file = "prkey_dataowner.key"; //приватний ключ власника даних
Stopwatch swDigest = new Stopwatch();
swDigest.Start();
//Підрахунок дайджесту файла
sbyte[] fileContainingDigest = Hash.createFileHash(InputFile, "SHA-256");
// Підпис файлу
sbyte[] fileSignature = RSAforBytes.rsaEncrypt(fileContainingDigest, private_key_file);
Console.WriteLine("Довжіна підпису файла у байтах: " + FSignature.Length);
//Зупинка таймеру обчислювання дайджеста та сігнатури
swDigest.Stop();
Console.WriteLine("Підрахунок дайджеста та сігнатури зайняв:" + swDigest.Elapsed() + " ms");
// Кінець розрахунків дайджеста та сігнатури
//геенрування ОБІ-файла
Console.Write("Введіть назву та росширення вихідного ОБІ файла: ");
BufferedReader outc = new System.IO.StreamReader(System.in);
string outfileExtension = outc.ReadLine();
File file = new File(outfileExtension); //output file
Stopwatch swOutput = new Stopwatch();
swOutput.Start();
RandomAccessFile outputFile = new RandomAccessFile(file, "rw");
outputFile.writeInt(kw_count); //кількість ключових слів
outputFile.write(encodedKeyword);
outputFile.write(encodedKeyword2);
outputFile.write(encodedKeyword3);
outputFile.write(encodedKeyword4);
outputFile.write(encodedKeyword5);
outputFile.writeUTF(User); // запис Id користувача
sbyte[] xr = x.toByteArray(); //Конвертація BigInteger Xr в масив байтів
// calculating the length of the xr so we can read it later
int xlength = xr.Length;
outputFile.writeLong(Cr); // Cr
outputFile.writeInt(xlength); // довжина Xr
outputFile.write(xr); //спільне Xr
outputFile.write(FSignature); //додавання цифрового підпису
//зчитування
RandomAccessFile in = new RandomAccessFile(InFile, "r");
// Перезапис байтів з вхідного у вихідний
```

```
sbyte[] buf = new sbyte[1024];
int len;
while ((len = in.read(buf)) > 0)
{
f.write(buf, 0, len);
}
Console.WriteLine("Нова довжина ОБІ-файлу:" + f.length());
f.seek(0);
f.close();
in.close();
Console.WriteLine("ОБІ-файл зроблено");
swOutput.Stop();
Console.WriteLine("ОБІ-файл сгенеровано за:" +swOutput.Elapsed() + " мс");
Console.WriteLine(DateTime); // show end date and time
}
private static string DateTime
{
get
{
DateFormat df = new SimpleDateFormat("yyyy-MMdd_ hh:mm:ss");
df.TimeZone = TimeZone.getTimeZone("PST");
return df.format(DateTime.Now);
}
}
}
public class HMACSHA256
{
public static sbyte[] ExecuteEncode(string key_input, string data_input)
{
Mac hash_algo = Mac.getInstance("HmacSHA256"); // Вибір хеш алгоритму
SecretKeySpec secret_key = new SecretKeySpec(key_input.GetBytes(), "HmacSHA256");
hash_algo.init(secret_key); //Генерація на основі переданого ключа
return hash_algo.doFinal(data_input.GetBytes()); //Складання хеша
}
}
using System;
public class CalculateKTZ
{
// Розрахунок рішення для теореми про залишки
public static System.Numerics.BigInteger Execute(System.Numerics.BigInteger[] input_a,
System.Numerics.BigInteger[] input_n)
{
if (input_a == null || input_n == null)
{
Console.WriteLine("Жоден з аргументів не може бути пустим");
return null;
}
if (input_a.Length < 2 || input_n.Length < 2)
{
Console.WriteLine("Довжина кожного елемента повинна бути більше двох");
return null;
}
```

```
if (input_a.Length != input_n.Length) {
Console.WriteLine("Довжина обох аргументів повинна дорівнювати");
return null;
}
System.Numerics.BigInteger x = input_a[0];
System.Numerics.BigInteger nInverse, y, z;
System.Numerics.BigInteger ni = System.Numerics.BigInteger.One;
int i = 0;
while (i < input_a.Length - 1) {
ni = ni * input_n[i];
//перевіримо чи n відносно просте
if (!System.Numerics.BigInteger.One.Equals(input_n[i + 1].gcd(ni))) {
Console.WriteLine("The n's are not relatively prime->" + input_n[i + 1] + "," + ni);
return null;
}
nInverse = cryptoBig.inverse(n[i + 1], ni);
if (nInverse.compareTo(0) == -1) {
nInverse = nInverse + input_n[i + 1];
}
z = input_a[i + 1] - x;
z = z * nInverse;
y = z.remainder(input_n[i + 1]);
if (y.compareTo(0) == -1) {
y = y + n[i + 1];
}
x = x + ni * y;
i++;
}
return x; // рішення КТЗ
}
}
public class EuclidHelper
{
//повертає g в ступены (-1) (mod p)
public static System.Numerics.BigInteger
ExecuteInverse(System.Numerics.BigInteger input_p, System.Numerics.BigInteger input_g)
{
System.Numerics.BigInteger[] result = ExtendedEuclid(input_p, input_g);
if (result[2].compareTo(0 as System.Numerics.BigInteger) != -1) {
return input_p + result[2];
}
return result[2];
}
//Ця функція виконає розширений алгоритм Евкліда, щоб знайти GCD для значень input_a та
input_b
public static System.Numerics.BigInteger[]
ExecuteEEA(System.Numerics.BigInteger input_a, System.Numerics.BigInteger input_b)
{
System.Numerics.BigInteger[] result = new BigInteger[3];
System.Numerics.BigInteger q;
if (input_b.Equals(0 as System.Numerics.BigInteger)) {
result[0] = input_a;
```

```
result[1] = 1 as System.Numerics.BigInteger;
result[2] = 0 as System.Numerics.BigInteger;
}
else
{
// В іншому випадку зробити рекурсивний виклик
q = input_a / input_b;
result = ExecuteEEA(input_b, a.remainder(input_a));
System.Numerics.BigInteger s = result[2] * q;
System.Numerics.BigInteger temp = result[1] - s; // - ans[2]*q;
result[1] = result[2];
result[2] = temp;
}
return result;
}
}
```

Додаток Б

Презентація

Міністерство освіти і науки України
Східноукраїнський національний університет імені Володимира Даля
Факультет інформаційних технологій та електроніки
кафедра комп'ютерних наук та інженерії

«Інформаційно-орієнтований підхід забезпечення безпеки даних у хмарному середовищі»

Студент гр. КІ-18 зм **Циганок Ю. С.**
Керівник проекту **к.т.н. Нестеров М. В.**

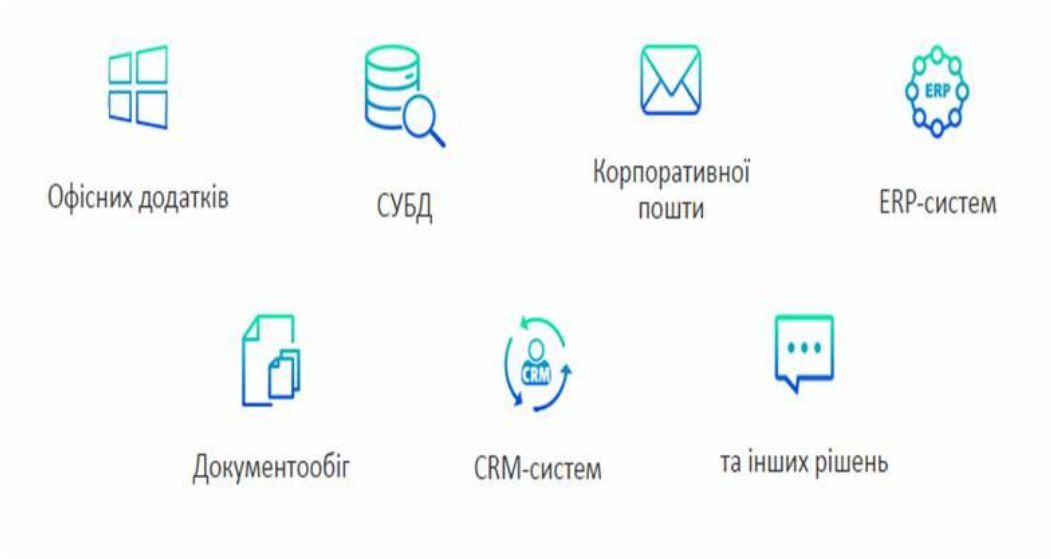
- **Актуальність** – впровадження хмарних сервісів у інформаційні технології супроводжуються проблемами забезпечення безпеки та конфіденційності даних користувачів через віртуалізацію хмар та характер своєї багаторівневої природи.
- **Метою роботи** є підвищення ефективності безпеки даних у хмарному середовищі.
- **Об'єкт дослідження** – хмарні середовища.
- **Предмет дослідження** – методи захисту даних у хмарних середовищах.
- **Методи дослідження** – аналіз існуючих традиційних підходів методів захисту веб-сервісів та концептуальних складових інформаційної безпеки. При формалізації задачі дослідження використано модель **«Програмне забезпечення як послуга» (Software as a Service (SaaS)).**

Основні задачі магістерської роботи

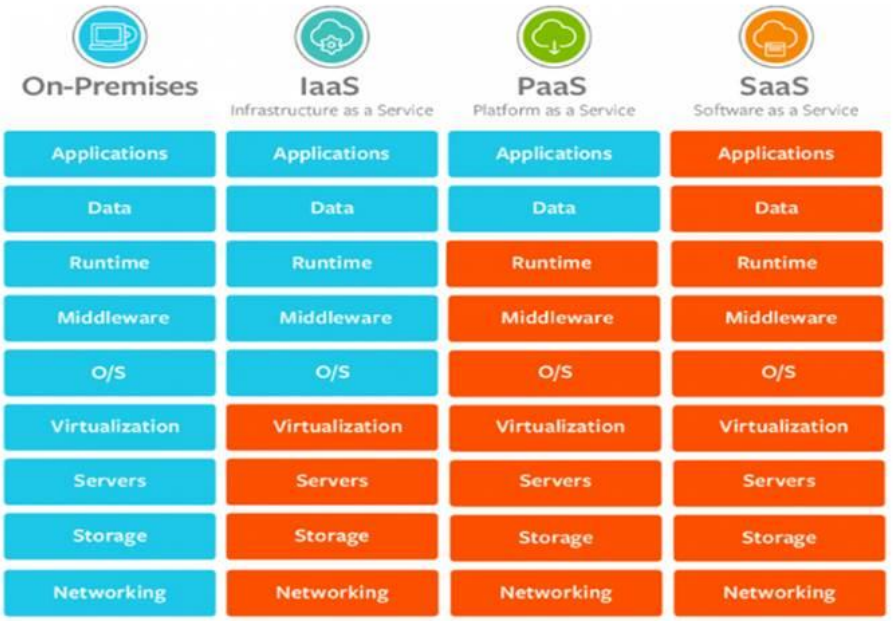
- – дослідження традиційних методів забезпечення безпеки у хмарному середовищі;
- – дослідження існуючих концепцій орієнтованих на безпеку інформації, визначення характеристик та критеріїв підходу для досягнення максимальної ефективності;
- – розроблення концептуальної основи інформаційно–орієнтованого підходу;
- – дослідження та вибір алгоритмів шифрування та підтримки цілісності даних;
- – дослідження та вибір алгоритмів забезпечення контролю доступу та перевірки аутентифікації;
- – дослідження та розроблення складової частини інформаційно–орієнтовного підходу, що забезпечує безпечний пошук у зашифрованих даних;
- – тестування клієнт–серверної моделі, що імітує хмарне середовище.

АНАЛІЗ ЗАХИСТУ ДАНИХ У ХМАРНИХ ТЕХНОЛОГІЯХ

варіанти розміщення програмних продуктів у хмарних ресурсах.



Основні відмінності технологій розміщення інформації у хмарних технологіях



Програмне забезпечення як послуга

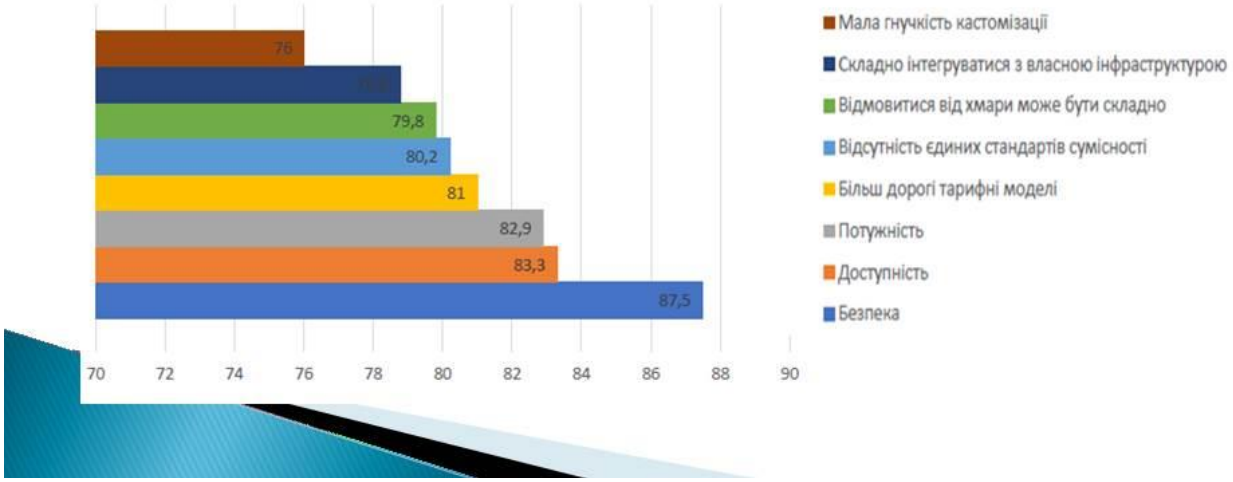
Платформа як послуга

Інфраструктура як послуга

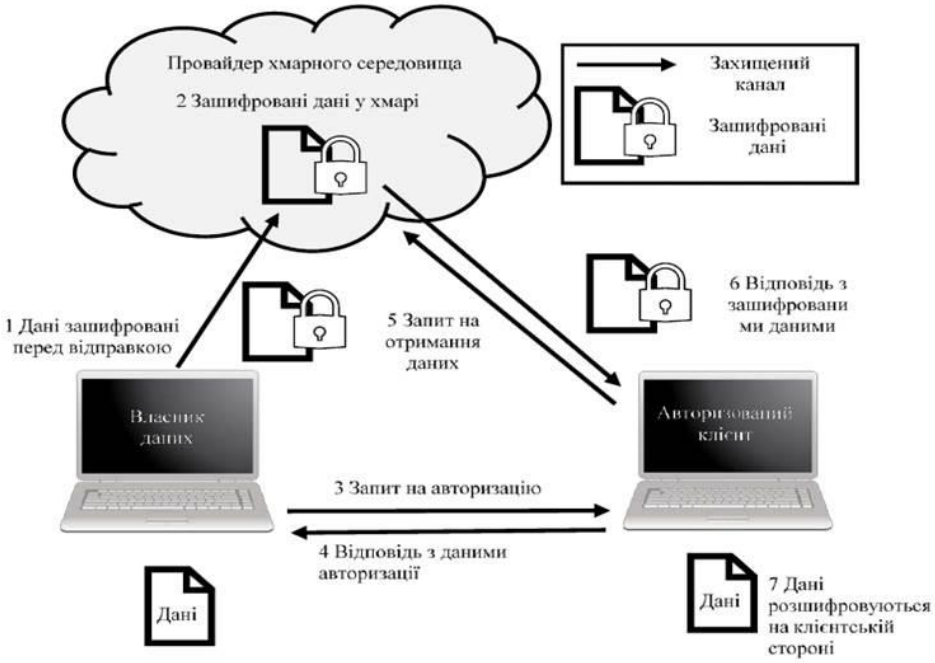
Приклади використання моделей SaaS, PaaS та IaaS

Platform Type	Common Examples
SaaS	Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting
PaaS	AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift
IaaS	DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

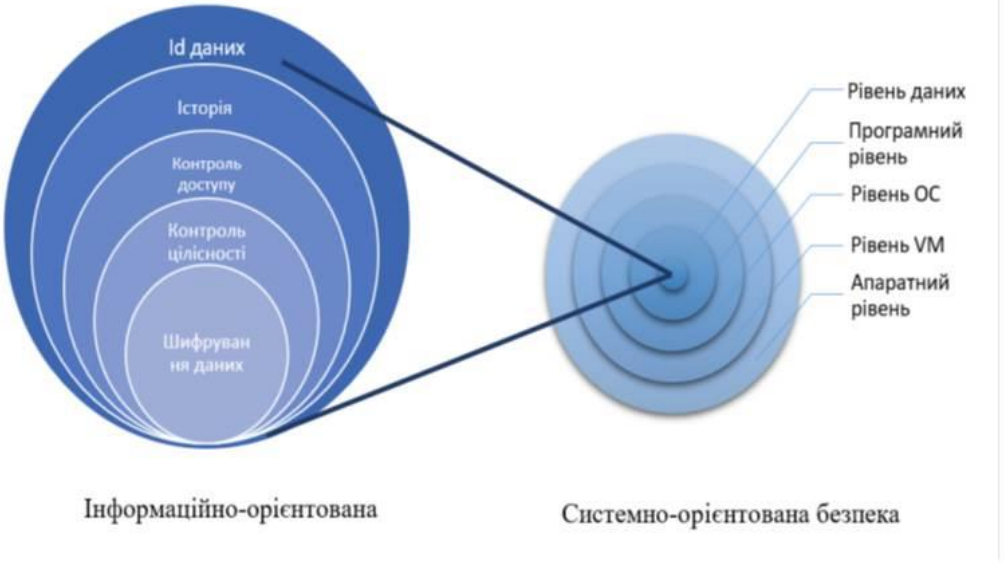
Діаграма проблем та викликів хмарних технологій



Базова архітектура для збереження конфіденційності даних у хмарі

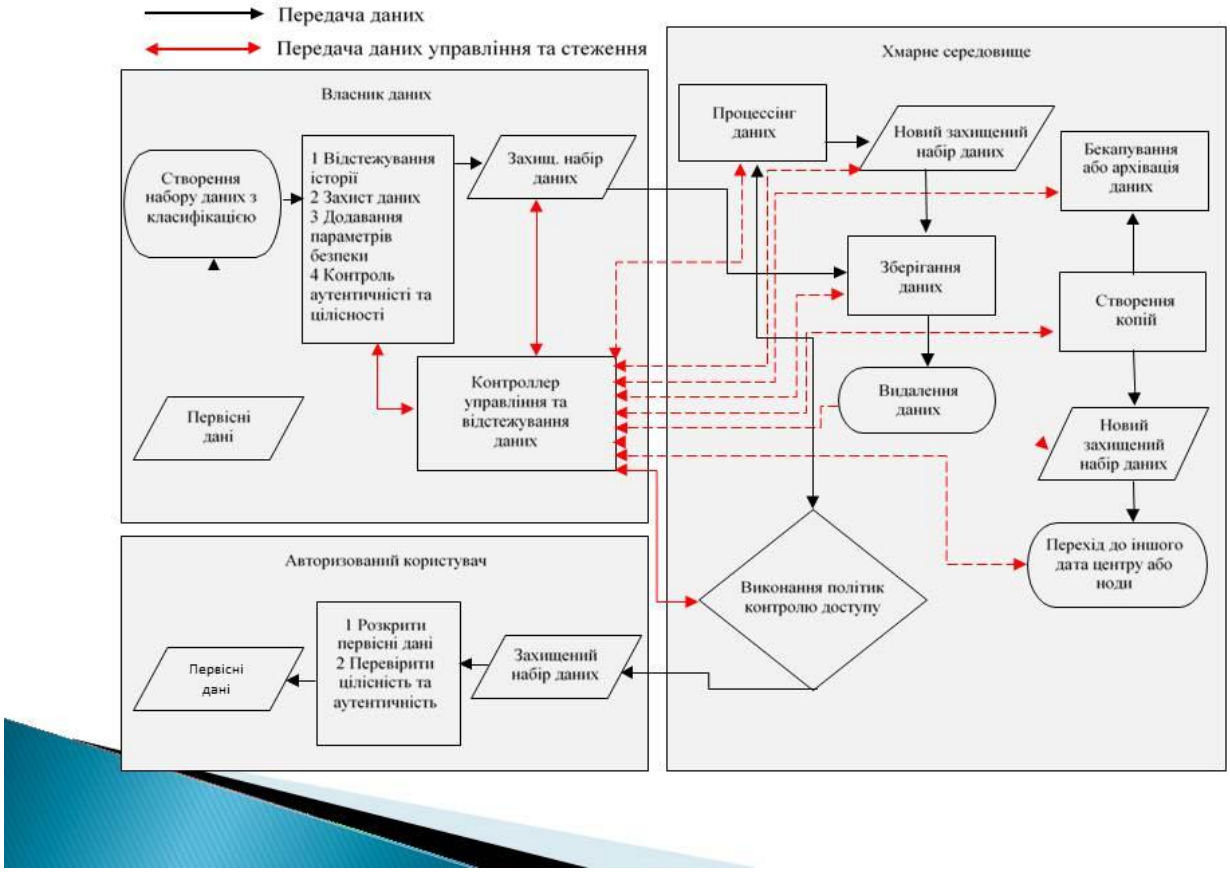


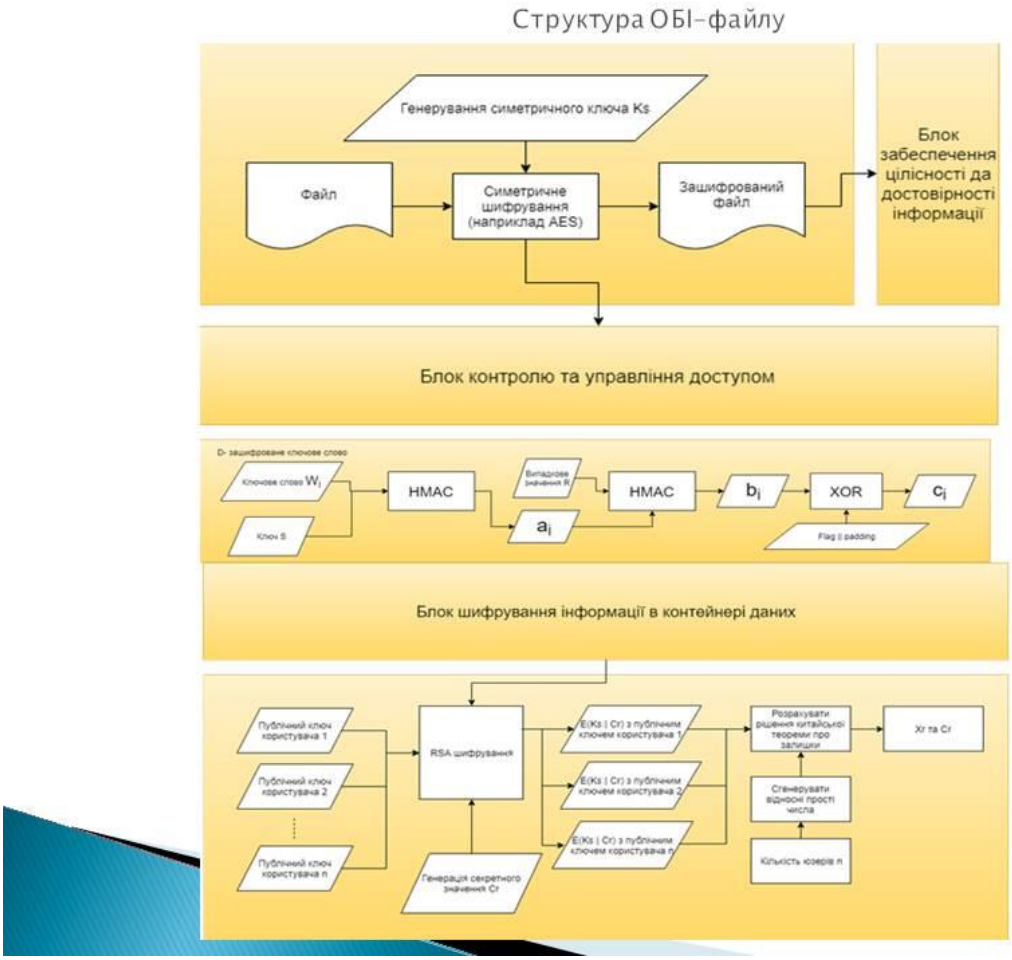
Моделі, спрямовані на поліпшення безпекової ситуації у хмарному середовищі



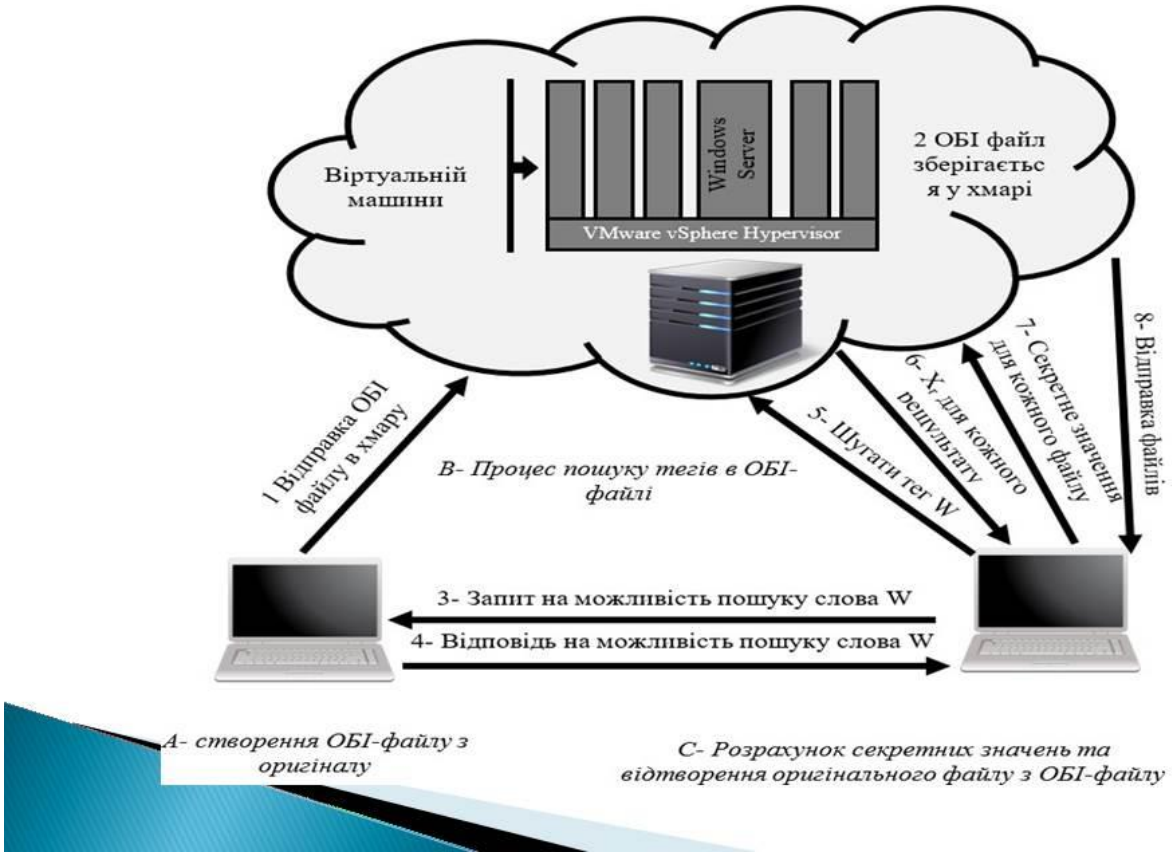

Рівні відповідальності безпеки
Рівень власника даних
Рівень довірених обчислень
Рівень провайдера послуг

Структура моделі, орієнтованої на безпеку інформації (ОБІ)

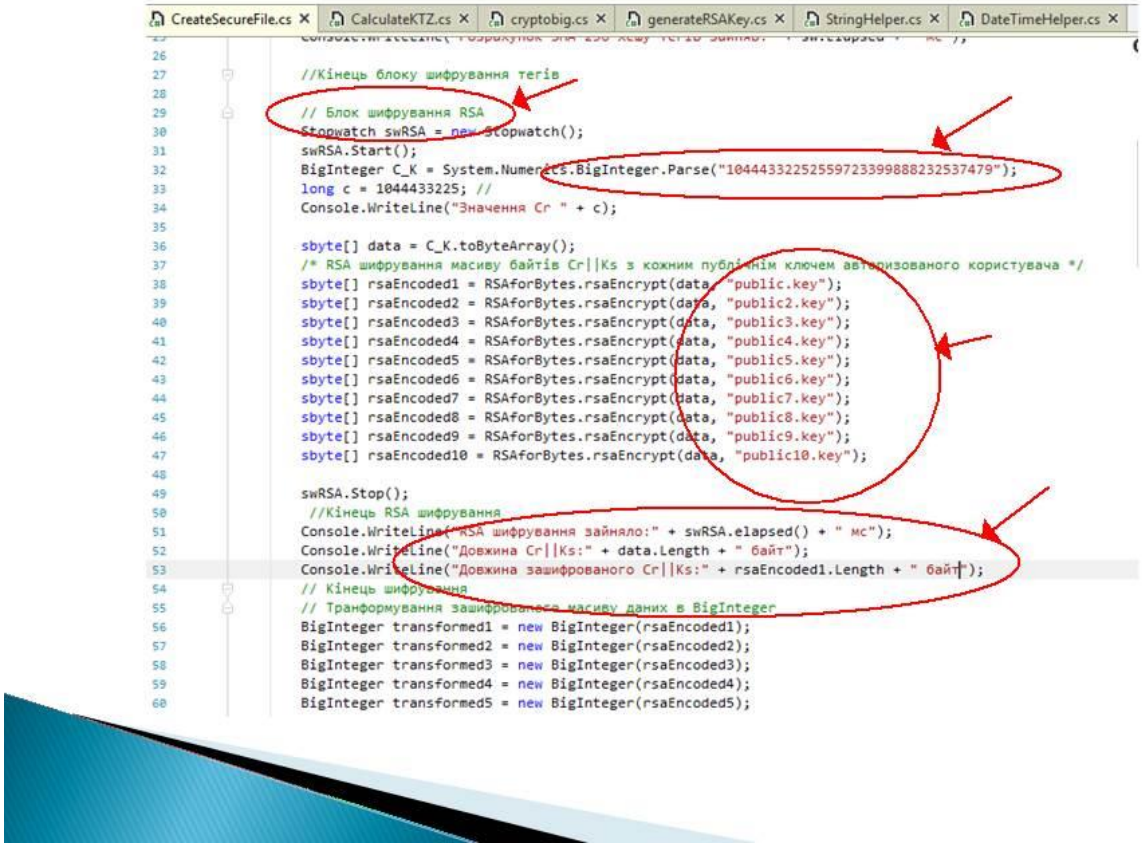



Структура ОБІ-файлу
Генерування симетричного ключа Ks
Файл
Симетричне шифрування (наприклад AES)
Зашифрований файл
Блок забезпечення цілісності да достовірності інформації
Блок контролю та управління доступом
HMAC
HMAC
XOR
Блок шифрування інформації в контейнері даних
RSA шифрування

Експериментальне середовище для запропонованого рішення

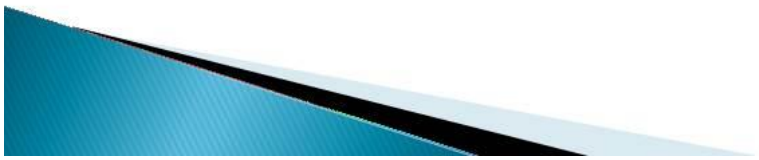


Результати генерації пар ключів RSA і шифрування з використанням згенерованих ключів



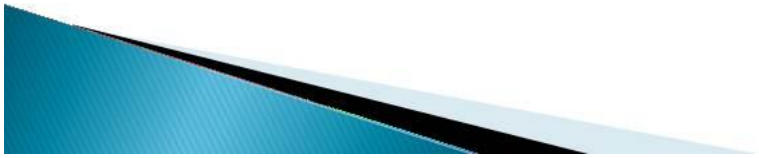
Зберігання та часові накладні витрати для шифрування AES

Назва файлу	Розмір в байтах	Розмір після шифрування в байтах	Збільшення розміру в байтах	AES шифрування в секундах
Sample.docx	14,175	14,485	310	<1
visio.vsd	45,029	45,341	312	<1
photo.jpg	64,819	65,130	318	<1
video1.wmv	25,327,026	25,327,338	312	1.6
video2.mov	41,670,503	41,670,812	309	2.3
video3.avi	136,318,116	136,318,429	313	14.8
video4.mov	598,787,322	598,787,634	311	86.7



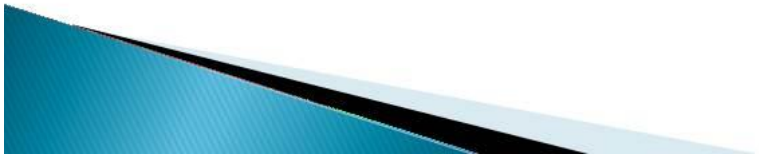
Часові витрат на зберігання та розрахунку загального значення повідомлення (Xr)

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



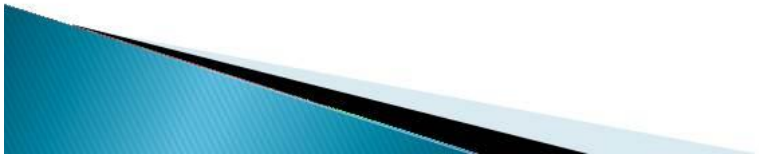
Накладні витрати на сховище і час виконання для додавання зашифрованих ключових слів

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



Загальний час, необхідний для створення ОБІ-файлу

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



Висновки, наукова новизна та практичне використання

Запропонована концептуальна основа для застосування підходу ОБІ, метою якого є підвищення безпеки і приватності даних в хмарі.

Запропонована структура заснована на визначенні вимог безпеки безпосередньо у фактичних даних протягом усього терміну зберігання у хмарі.

Результати, отримані при тестуванні реалізованих операцій запропонованого рішення, показують, що реалізація захисту даних не вимагає складних математичних операцій та великих обчислювальних ресурсів.

Проведені в роботі дослідження технології захисту даних у хмарному середовищі забезпечують більш чітку і ширшу концептуальну структуру ОБІ.

Практичне використання. Результати дослідження, запропоновані рішення дозволять підвищити безпеку зберігання даних у хмарному середовищі від несанкціонованого доступу до них та можуть бути використані у навчальному процесі кафедри комп'ютерних наук та інженерії при вивченні дисципліни «Захист інформації в комп'ютерних системах».