Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження моделей та засобів збереження та обробки конфиденційної інформації у хмарі
(тема)

Виконав:
Випускник 2 курсу, групи ІПЗм-19-2
Сіренко О. Є.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Вечур О. В.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« 26 » березня 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Сіренко Олександра Євгеньовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження моделей та засобів збереження та обробки конфиденційної інформації у хмарі»

затверджена наказом університету від 26.03.2021 № 385 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «08» травня 2021 р.

3. Вихідні дані до роботи *моделі та засоби зберігання секретних даних у хмарі*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, методи та засоби зберігання секретних даних у хмарі, передача секретних ключів у системі*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів,

ілюстрацій (слайдів) *мета завдання, обґрунтування доцільності розроблення, постановка задачі, методи і архітектурні підходи, структурно-логічна схема взаємодії даних, опис отриманих результатів, демонстраційні матеріали*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	Вечур О. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	25.01.21 – 19.02.21	*Виконано*
2	Огляд існуючих моделей та засобів	20.02.21 – 10.03.21	*Виконано*
3	Підготовка пояснювальної записки	26.03.21 – 15.04.21	*Виконано*
4	Спецчастина	15.04.21 – 17.04.21	*Виконано*
5	Підготовка презентації та доповіді	18.04.21 – 19.04.21	*Виконано*
6	Нормоконтроль, рецензування	24.05.21 – 25.05.21	*Виконано*
7	Занесення диплома в електронний архів	25.05.21	*Виконано*
8	Попередній захист	25.05.21	*Виконано*
9	Допуск до захисту у зав. кафедри	25.05.21	*Виконано*

Дата видачі завдання 25 січня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Вечур О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ ABSTRACT

Кваліфікаційна робота магістра містить: 79 с., 14 рис., 11 джерел.

CLOUD STORAGE, WEB SECURITY, SYSTEM SECURITY, SECURE DATA, SENSETIVE DATA, ENTRYPTION KEY EXCHANGE, ENTRYPTION.

Об’єктами дослідження є моделі зберігання, обробки та передачі конфиденційних даних.

Метою роботи є дослідження архітектури та методів системи, що дозволяє зберігати та обмінюватися конфиденційною інформацією без можливості доступу до неї 3-х осіб.

Результатом роботи є рекомендації щодо забезмечення безпеки програмної системи, модель, що дозволяє зберігати дані у хмарі, без можливості доступу до даних 3-м сторонам.

PWA, JAVASCRIPT, INDEXEDDB, TYPESCRIPT, REACT, ARCHITECTURE, PUSH API, SERVICE WORKER, NOTIFICATION API, BROWSER, TYPESCRIPT.

The objects of the study are model materials, samples and transmission of confidential data.

The aim of the work is to study the archive and methods of systems to allow the selection and exchange of confidential information without the possibility of access to it by 3 people.

The result of the work is recommendations for scanning secure software systems, models, for dosing data on Khmer, without the possibility of access to devices on 3 sides.

Я, *Сіренко Олександр Євгеньович,* студент групи ІПЗм-19-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження моделей та засобів збереження а обробки конфиденційної інформації у хмарі», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

FullPP - Full Private Protection;

AWS - Amazon Web Service;

S3 - Simple Storage Service;

WSS - WebSocket Secure;

VDO - Vanish Data Object;

DHT - Distributed Hash Table;

SDO - Secure Data Object;

SSDD - Secure Self Destructed Data.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ 10
1.1 Типи секретної інформації 10
1.2 Види автентифікації 11
1.3 Паролі 13
1.4 Аналіз існуючих методів зберігання конфіденційних даних в хмарі 15
1.4.1 Механізми поділу даних 16
1.4.2 Методи анонімізації даних 17
1.4.3 Криптографічні методи 18
1.5 Провайдери хмар 20
1.6 Постановка задачі 21
2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ 22
2.1 Безпека хмар 22
2.1 Проблеми безпеки в хмарах 23
2.1 Безпека даних у AWS S3 25
2.1 Критерії сховища даних 26
2.1 Авторизація та автентифікація 27
2.1 Канал передачі даних 28
2.1 Сховище даних 29
2.1 Підпис документів 30
2.1 Розподілене зберігання шматків документів 32
2.1 Зберігання ключів шифрування 33
2.1 Розділення функцій, відповідальності, рівнів доступу 34
3 ОГЛЯД АРХІТЕКТУРНИХ РІШЕНЬ 35
2.1 Vanish архітектура 36
2. SafeVanish архітектура 39
2.1 Архітектура Single Secure 39
2.1 Архітектура FullPP 41
2.1 Розділення функцій, відповідальності, рівнів доступу 44
4 ОПИС ЄКСПЕРЕМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ 53
ВИСНОВКИ 58
ДОДАТОК А. Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії 59

ДОДАТОК Б Звіт результатів перевірки кваліфікаційної роботи на унікальність тексту……..**Ошибка! Закладка не определена.**

ДОДАТОК В Звіт результатів перевірки кваліфікаційної роботи на унікальність тексту……..**Ошибка! Закладка не определена.**

ДОДАТОК Г. Слайди презентації 69

ДОДАТОК Д Лістинг модуля программи ……..77

ДОДАТОК Е. Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015……..78

ВСТУП

За останні роки все більше людей стають користувачами мережі інтернет. Майже всі люди користуються тими чи іншими застосунками, соціальними мережами, сайтами, мають аккаунти у різних мобільних додатках, зберігаючи ту чи іншу інформацію на віддаленому сховищі.

Перед розробниками додатків завжди стоять велика кількість вимог. Одна з головних – зберігання даних, особливо зберігання конфиденційнії інформації. Нерідко можна почути про втрату чи витік даних. Тому питання збереження та обробки конфиденційної інформіції насьогодні стоїть дуже гостро.

Сьогодні кількість інформації, що зберігається в хмарі швидко збільшується, багато держав мають закони, щодо належного зберігання та високий рівень захищеності інформації.

Є правило, що система безпечна настільки, наскільки безпечне її саме слабке місце. Оскільки сьогодні велика кількість інформації зберігається на віддаленому сервері, тому є важливим правильне зберігання інформації в хмарі.

До секретної інформації належить:

— паролі;

— паролі-фрази:

— ключі шифрування;

— авторизаційні токени;

— номери кредитних карток;

— особиста інформація (імена, номери телефонів, адреси електронної пошти, фізичні адреса та ін.);

— MAC-адреси, IP-адреси.

Конфіденційними даними також називають інформацію, що ідентифікує особу або дані про значний вплив на бізнес. Те, що вважається конфіденційною інформацією, сильно варіюється в залежності від штату та країни. Різні стандарти

відповідності, такі як стандарт дотримання вимог індустрії платіжних карток (PCI), вимагають спеціальних кроків під час збору конфіденційних даних, щоб залишатися у відповідності.

Актуальність теми атестаційної робити обумовлена наявністю проблеми зберігання секретної інформації на сервері. Розробнику треба реалізувати всі аспекти зберігання даних та обробки даних – від збереження даних у базі даних, шифрування, до реалізації належної авторизації та доступу до даних.

Метою роботи є дослідження моделей збереження та засобів збереження та обробки секретної інформації на стороні сервера.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Типи секретної інформації

На сьогодні велика кількість даних зберігається та обробляється у хмарах. Важливим моментом є коректне та безпечне зберігання секретної інформації. В залежності від типу секретної інформації використовують ті чи інші прооколи збереження та обробки інформації. Наприклад, фінансова інформація, банківські системи, особиста інформація, паролі, ключі паролі, авторизаційні токени.

Кожен вид секретної інформації має свої методи її збереження та обробки, протоколи захисту та обміну, айтентифиікаційну та авторизаційну систему.

В залежності від типу секретної інформації існують правила збереження та обробки інформації.

Стандартна схема збереження даних в системах зображена на рисунку 1.

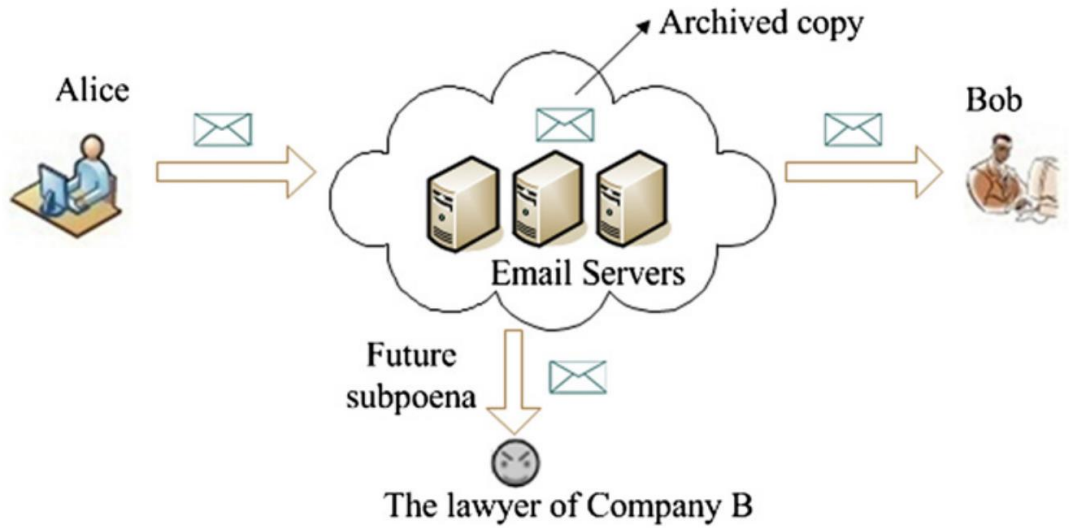

Рисунок 1 – Загальна схема збереження даних у хмарі

Оскільки дані лежать на віддалених серверах та ключі шифрування зберігаються на тому ж самому сервері. Через це сервіс має доступ до даних

користувача, що є вразливістю з точки зору безпеки даних користувачів.

1.2 Види автентифікації

Аутентифікація стала дуже важливим механізмом безпеки. Останнім часом багато зловмисників намагаються атакувати бази даних паролів організацій, оскільки саме вони є джерелами, які зловмисники потенційно можуть використовувати для отримання несанкціонованого доступу до інформації, мережі та ресурсів. Тому важливо зрозуміти, як безпечно вибирати та зберігати паролі таким чином, щоб це могло перешкодити зловмисникам дізнатися, що таке паролі користувачів.

Перший метод автентифікації називається методом “something-you-know”. Це метод, за допомогою якого користувач використовує те, що він пам’ятає, для підтвердження своєї особистості. Іншими словами, користувач використовував би свої знання, щоб довести, що він насправді є тим, ким він або вона претендує. Хорошим прикладом цього методу автентифікації є пароль або пін-код банкомата.

Другий метод автентифікації називається методом “something-you-have”. Цей другий спосіб автентифікації стосується того, що користувач має у своєму розпорядженні. Сюди входять смарт-картки та маркери автентифікації. Токен може бути як синхронним, так і асинхронним. Синхронний маркер синхронізується із сервером. Вони використовують час, щоб сформувати число, яке буде введено на етапі входу користувача. Коли користувач бажає увійти в систему, він або вона вводить ім’я користувача разом із номером, що відображається на маркері. Ця кількість змінюється відповідно до поточного часу. Для RSA SecurID Token номер у маркері змінюється щохвилини, а також синхронізується із сервером автентифікації. З іншого боку, асинхронні маркери використовують систему виклику і відповіді. Сервер автентифікації та маркер не потрібно синхронізувати.

Тобто, коли користувач бажає увійти, сервер представляє користувачеві числову послідовність. Тому він замінює цю відповідь у системі, щоб отримати доступ. Основною перевагою використання токена автентифікації, синхронного чи асинхронного, є те, що пароль користувача змінюється кожного разу, коли він або вона входить. що знижує ризик вгадування паролів зловмисниками.

Третій спосіб автентифікації стосується характеристик користувача. Це процес використання вимірювань тіла, відомий як біометрія. Популярні біометричні дані включають відбитки пальців, відбитки долоні, геометрію рук, розпізнавання обличчя, малюнок райдужки та відбиток сітківки. З усіх можливих біометричних даних було сказано, що лише три вважаються справді унікальними. Це відбитки пальців, сітківка ока і райдужна оболонка ока. Біометрія працює, порівнюючи фактичні характеристики користувачів із збереженими даними, щоб визначити, чи справді користувач заявляє, що він є. Проблема цього методу полягає в тому, що характеристики людини можуть змінюватися з часом внаслідок нормального розвитку, хвороби або травми. Тому на будь-який випадок слід створити резервні або безпечні механізми автентифікації.

Четверта область автентифікації - це те, що користувач виконує або виробляє. До цієї категорії належать два популярні методи - розпізнавання підпису та голосу. За допомогою розпізнавання підписів користувач підписує цифрову клавіатуру стилусом, який фіксує підпис. Потім підпис зберігається і порівнюється з підписом у базі даних для перевірки. Розпізнавання голосу працює подібним чином, коли користувач вимовляє фразу в мікрофон. Голос фіксується та зберігається у базі даних. Пізніше, під час аутентифікації, користувач промовляє ту саму фразу, щоб її можна було порівняти з голосом, що зберігається в базі даних. Зараз ми побачили чотири методи автентифікації. Питання в тому, який із них нам насправді використовувати? Стверджувалося, що для досягнення більш міцної автентифікації, отже, кращої безпеки, слід використовувати два методи автентифікації поряд. Це називається двофакторною аутентифікацією. Тобто, ми могли б використовувати смарт-карту разом із паролем, який може бути у вас із чимось, що ви знаєте. Прикладом є той факт, коли користувач знімає гроші з

готівкового пункту, йому або їй доведеться ввести свій пін-код (метод "щось знаєш"), а також вставити кредитну картку (метод "щось у вас є"). Незважаючи на те, що двофакторна автентифікація сильніша, ніж використання лише одного методу, деякі дослідники все ще стверджують, що цього недостатньо. Сказавши, що сьогодні існують усі чотири методи автентифікації, у цьому розглядатиметься лише перший метод, зокрема паролі, папір.

1.3 Паролі

Пароль - найпопулярніший метод автентифікації на сьогоднішній день. Ось чому ми відчуваємо, що потрібно детально обговорити паролі. Перш ніж іти далі, давайте коротко встановимо тут, яким повинен бути ідеальний пароль. Загалом, було запропоновано, що ідеальним паролем має бути те, що користувач може запам'ятати, те, що може перевірити комп'ютер, і те, про що ніхто інший не може здогадатися. Це звучить легко, але важко досягти. Проблема паролів сьогодні полягає в тому, що люди, як правило, вибирають "погані" паролі. Це паролі, які легко “зламати”. Яке рішення цього? Одним із рішень є використання випадково сформованих криптографічних ключів. Це зробить роботу з взяття пароля рівноцінною роботі грубої сили або вичерпного пошуку ключів.

Збереження паролей. Останнім часом низка гучних компаній бачили, як їх паролі потрапляли в мережу, навіть незважаючи на те, що було докладено багато зусиль для їх захисту. На жаль, розкриття баз даних паролів є однією з головних цілей спільноти хакерів. Тому важливо розуміти, як можна зберігати паролі, і що кожен спосіб зберігання означає для безпеки паролів. Давайте розглянемо та проаналізуємо основні методи.

Найпростіший спосіб збереження пароля - це відкритий текст. Це означає, що у файлі паролів або базі даних паролів імена користувачів та паролі зберігаються у зручній для читання формі. Тобто, якщо пароль якщо testpassword, він також

зберігається в базі даних як testpassword. Коли користувач вводить своє ім’я користувача та пароль, система перевіряє їх у відповідності до бази даних, щоб перевірити, чи збігаються вони. Це найгірший із можливих методів зберігання паролів в контексті безпеки. Більшість авторитетних систем та веб-сайти не зберігають паролі у відкритому тексті. Це пов’язано з тим, що якщо зловмисник отримує базу даних паролів, то всі паролі негайно отримують доступ, відомі та скомпрометовані.

Для того, щоб зменшити ризик викриття паролів як відкритого тексту, деякі веб-сайти прийняли рішення для шифрування. Шифрування, як нагадування, використовує секретний ключ для перетворення відкритого тексту пароля у випадковий рядок тексту. Це означає, що якби противник захопив базу даних паролів, він або вона не змогли б побачити, якими є справжні паролі. Будуть бачити лише паролі у формі зашифрованого тексту. Супротивник повинен мати секретний ключ для їх розшифровки. Це звучить не так погано, правда? Проблема цього методу полягає в тому, що секретний ключ часто зберігається на тій самій машині чи сервері, на якому паролі. Що станеться, якщо сервер зламають? Хакеру не доведеться робити багато роботи, щоб отримати секретний ключ, який дозволить йому або їй розшифрувати всі паролі. Це означає, що цей метод вважається небезпечним.

Хешування схоже на шифрування в тому сенсі, що воно перетворює пароль у випадковий рядок букв і цифр. Однак, як ми вже знаємо, хешування, таке як MD5 та SHA-1, забезпечує принаймні дві речі. По-перше, неможливо зробити зворотний хеш. Ми не можемо взяти хешований пароль і запустити алгоритм хешування назад, щоб отримати оригінальний пароль. По-друге, дуже малоймовірно, що два різних паролі видаватимуть однакові випадкові рядки букв і цифр. Хешування є найбільш широко використовуваним методом зберігання паролів як у локальній мережі, так і в Інтернеті. Хеш-функція працює, приймаючи за вхідний пароль пароль і скремблюючи його, щоб отримати на перший погляд випадковий результат. Дві популярні хеш-функції, що використовуються для зберігання паролів, - це MD5 і SHA-1.Хеш-функції часто використовуються в системах

паролів, оскільки замість зберігання паролів у чистому тексті зберігаються лише хеш-значення паролів. Оскільки змінити хеш практично неможливо, метод хешування здається безпечним способом збереження паролів. Коли користувач входить у систему, обчислюється хеш-значення введеного пароля та порівнюється з хешем у базі даних паролів. Якщо вони збігаються, пароль правильний. Якщо ні, користувачеві доведеться зробити ще одну спробу. Більше того, оскільки кажуть, що хеші є унікальними, дуже ймовірно, що користувач зможе увійти лише з правильним паролем. На жаль, протягом останніх місяців на момент написання статті ряд інтернет-компаній виявили, що їхні бази даних паролів були зламані, навіть незважаючи на те, що їх паролі були хешовані. Причина того, що це сталося, полягає в тому, що у цього методу є мінус. Тобто, хоча супротивник не може повернути хеш-значення назад до початкового пароля, він або вона може спробувати безліч різних паролів, поки один не збігається з необхідним хешем. хакери можуть попередньо обчислити всі хеші всіх можливих паролів і зберегти їх у базах даних.

Головною перевагою bcrypt та scrypt є те, що з часом вартість може бути збільшена, щоб йти в ногу зі все швидшими та швидшими комп'ютерами, а також захищати паролі, роблячи хеш-функцію повільнішою та повільнішою.

1.4 Аналіз існуючих методів зберігання конфіденційних даних в хмарі

Існуючу методи можно поділити на 3 типи [1]:

— механізми розподілу даних;

— метоти анонімізації даних;

— методи криптографії;

— методи захисту передачы ынформації.

1.4.1 Механізми поділу даних

Механізми поділу даних працюють наступним чином. Розбиття даних - це метод захисту, заснований на фрагментації конфіденційних даних і зберігання фрагментів у чіткому вигляді в окремих місцях. Фрагменти мають бути такими, щоб окремий фрагмент не дозволяв ідентифікувати предмет кому він відповідає, а також не розголошує конфіденційну інформацію, яка може бути пов’язана до певного предмета. Наприклад, якщо фрагмент складається із значень атрибута «Діагностика», то очевидно, що просто знання списку діагнозів марно для зловмисник, оскільки він не може пов’язати їх із відповідними предметами. За хмарним сценарієм дані можуть передаватися в аутсорсинг локальним проксі-сервером, виконуючи розділення даних на окремі хмарні облікові записи в межах одного і того ж CSP, або для різних хмар, кожна з яких управляється різною CSP, що забезпечує однаковий тип сервісу (тобто, багатохмарна). Для захисту даних на основі розбиття ефективні місця зберігання повинні залишатися незалежними та не пов'язаними, так що CSP не зможуть здогадатись, щоб згрупувати часткові дані, сподіваючись зламати захист даних (див. рис. 2).

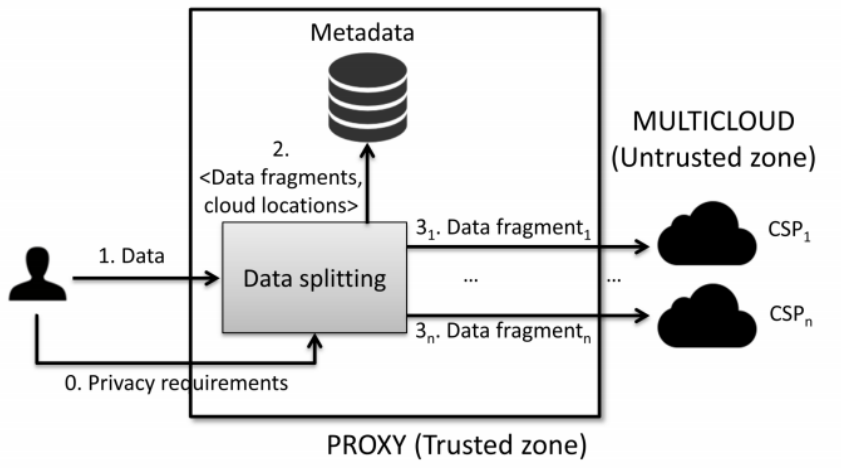

Рисунок 2 - Робочий процес зберігання даних та розподіл на сховище.

1.4.2 Методи анонімізації даних

На відміну від розбиття даних (а також шифрування даних), анонімізація даних безповоротно маскує дані, зберігаючи конфіденційність. Методи анонімізації були розроблені в галузі статистичного контролю за розкриттям інформації [2] як засіб звільнення конфіденційних даних дані третій стороні, якій не довіряють, щоб вони залишались аналітично корисними для вторинного використання, але не розголошували інформацію, яка може бути пов’язана з конкретною фізичною особою.

У наших умовах ненадійною стороною є CSP, а вторинними - аутсорсингові обчислення. Тоді як зашифровані дані марні для тих, хто не може розшифрувати їх, анонімні дані зберігають певну корисність для всіх. Зокрема, ДСП, які пропонують свої послуги безкоштовно в обмін на монетизацію аналізу інформації, яку вони зберігають, можуть бути проти зашифрованих даних (за винятком випадків, коли шифрування виконують самі), але не проти анонімних даних (у більшості випадків їх інструменти для автоматизованого аналізу навіть не можуть виявити, що відбулася анонімізація).

І все-таки головна перевага анонімних даних перед зашифрованими та даними розбиття стосується простоти першої обробки. З анонімністю даних, маскування виконується лише на етапі зберігання і може бути здійснено за допомогою лінійних або квазілінійних алгоритмів; після цього будь-який запит щодо анонімних даних (пошук, пошук, обчислення) є прозорим і не вимагає жодних накладних витрат або для проксі-сервера, або для CSP. Через незворотність анонімізації даних, отримані дані та результати обчислень не потрібно демонтувати (насправді вони не можуть), що означає, що локальний проксі-сервер потрібен лише на етапі зберігання, і не потрібно зберігати чи керувати ключовими матеріалами. Недоліком анонімних даних є те, що загалом результати обчислень на них приблизні, а не точні.

Другий мінус - це ключове - словопошук анонімних атрибутів не підтримується. Фактично, оскільки анонімізація даних змінює окремі значення, але

намагається зберегти корисність набору даних у цілому, вона підходить для сукупних розрахунків (наприклад, статистичні дані) щодо наборів даних, що передаються, але не для запитів чи аналізів щодо конкретних фізичніих осіб. Було б мало конфіденційності, якщо робити висновки щодо конкретних осіб було б легко з анонімних даних. На відміну від них, деякі статистичні дані можеть бути збережені певними методами анонімізації, такими як середнє значення або дисперсія.

1.4.3 Криптографічні методи

У цьому розділі ми розглядаємо сучасні криптографічні методи, орієнтовані на хмарні обчислення. Справді, криптографію можна розглядати як одну з найсильніших та найважливіших будівельних блоків в багатьох планах безпеки, розгорнуті для вирішення проблем конфіденційності в хмарі. Сучасна криптографія перетворилася на задоволення дуже різноманітних реальних потреб у безпеці, багато з яких стосуються хмарних обчислень. Як галузь, криптографія розглядає широкий спектр цілей інформаційної безпеки, архітектури користувачів та функціональні можливості. Реальні сценарії хмарних обчислень можуть вимагати різних цілей безпеки, такі як конфіденційність даних або цілісність даних. Крім того, можна захотіти виконувати функціональні можливості, кращі за просте безпечне зберігання в хмарі, наприклад пошук та отримання даних із хмари, або обчислення на захищених даних. Через втрату даних, що спостерігалася протягом останніх кількох років, багато хмар сьогодні використовують криптографічні рішення для шифрування даних при збереженні, зокрема, використовуючи симетричні схеми шифрування, такі як AES256. Однак у цих службах CSP часто керує всіма введеннями матеріалів, а отже, конфіденційність переданих даних, в кінцевому рахунку, покладається на довірчі відносини між користувачами та CSP. Природним криптографічним підходом, який допомагає зберегти конфіденційність

даних від ненадійних CSP, є шифрування даних за допомогою традиційних схем шифруванняїх аутсорсингу. Тим не менше, такий підхід заважає CSP виконувати будь-які складні функції та змушує користувачів завантажувати цілі дані, якщо вони хочуть використовувати передані дані. Отже, традиційні схеми шифрування не підтримують делегування більшості функціональних можливостей через передані зашифрованим даним в хмару, і це явно погіршує продуктивність та економічні переваги хмарних архітектур. Щоб вирішити це питання, нещодавно криптографічне дослідження фокусується на розробці методів шифрування, які дозволяють клієнту надійно делегувати хмарі певні функціональні можливості передані зашифровані дані на сторонніх джерелах. Більшість сучасних хмарних служб захищають дані за допомогою криптографічних методів (див. рис. 3) [3].

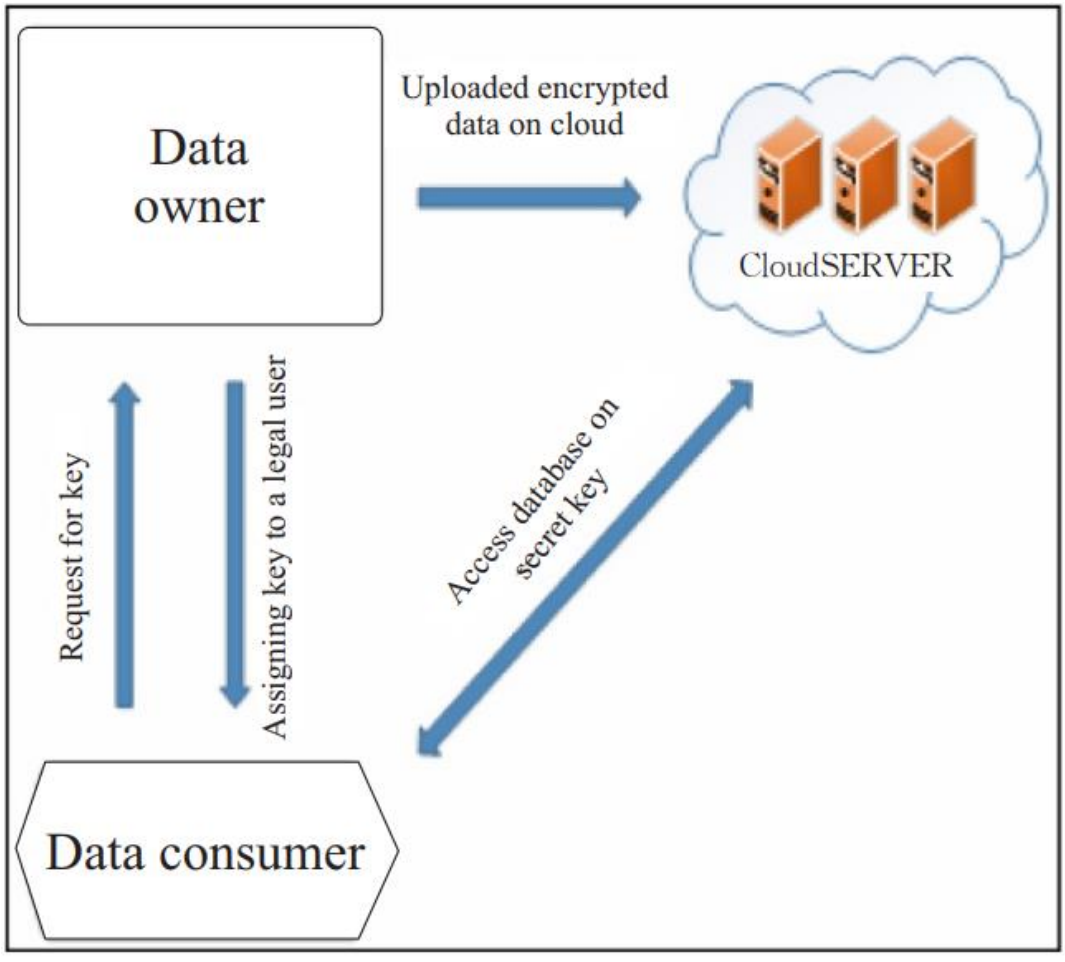

Рисунок 3 - Захищений доступ до даних у хмарі

Загальнодоступні хмари, такі як Google Drive або Dropbox, шифрують

збережені дані за допомогоюсиметричні схеми шифрування, такі як AES, але, як правило, криптографічні ключі зберігаються постачальником хмарних послуг. Отже, дані шифруються, але користувачі делегують управління ключами хмарі, а отже, неявно довіряють CSP та його політикам управління даними. Існує інший підхід для шифрування даних – посилати у хмару вже зашифровані дані, таким чиином процедура шифруваня/дешифрування проходить на окромому вузлі.

1.5 Провайдери хмар

Провайдери хмар мають доступ до даних хмари, та мають можливість слухали дані, що проходять через них. Таким чиином вони мають доступ до даних. Щоб уникнути цього потрібно гарантувати безпечний протокол передачі даних та авторизаційну стратегію (див. рис. 4).

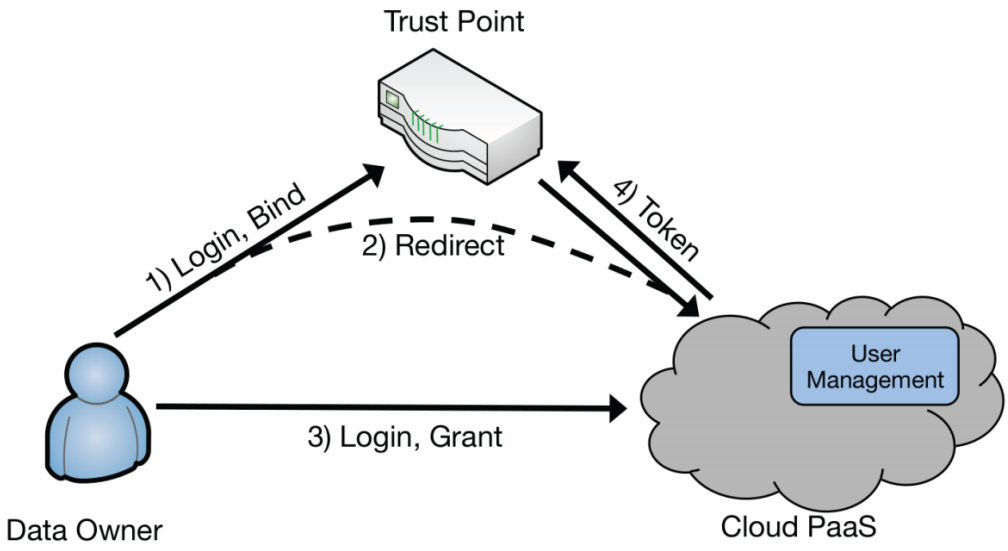

Рисунок 4 – З метою встановлення безпечного та взаємно автентифікованого зв’язку між Довірчим фондом Point і Cloud, власник даних запускає процедуру прив’язки на основі OAuth.

Такий підхід реалізує Google Drive та Dropbox.

1.6 Постановка задачі

Базуючись на проведеному аналізі предметної галузі метою роботи стає визначення впливу різних засобів та методів збереження та обробки конфиденцційної інформації у хмарі, створення моделі сисеми, що включає в себе кращі практики існуючих моделей, таким чином створення ще бульш безпечнішої моделі.

Поставлена задача може бути розділена на наступні частини:

— дослідження існуючих засобів та методів зберігання та обробки секретної інформації у хмарі;

— виділення найбільш цікавих засобів та методів для їх подальшого дослідження;

— аналіз можливих проблем при обробці та збереженні конфиденційних даних;

— визначення поняття ефективності та критеріїв її оцінювання;

— формулювання рекомендацій щодо використання різних засобів та методів збереження конфиденційної інформації в хмарі;

— створення безпечної моделі збереження та доступу до конфиденційних документів.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

Зберігання конфиденційних даних у хмарі потребує чіткої процедури захисту даних та захисту доступу до цих даних. Система повинна бути безпечка на усіх ділянках – від авторизації до збереження даних.

2.1 Безпека хмар

Існує гостра необхідність в надійному зберіганні, управлінні, спільному використанні та аналізі величезних обсягів складних (наприклад, напівструктурованих і неструктурованих) даних для визначення закономірностей і тенденцій з метою підвищення якості охорони здоров'я, кращого захисту нації та вивчення альтернативних джерел енергії. Через критичний характер додатків важливо, щоб хмари були безпечними. Основна проблема безпеки в хмарах полягає в тому, що власник даних може не контролювати, де вони розміщатися. Це пов'язано з тим, що якщо хтось хоче використовувати переваги використання хмарних обчислень, необхідно також використовувати розподіл ресурсів і планування, що надаються хмарами. Тож нам необхідно захистити дані в розпал ненадійних процесів. Нова модель хмарних обчислень намагається впоратися з вибуховим зростанням підключених до Інтернету пристроїв і обробляти величезні обсяги даних. Тепер Google представила фреймворк MapReduce для обробки великих обсягів даних на товарному обладнанні. Розподілена файлова система Hadoop від Apache (HDFS) стає чудовим програмним забезпеченням компонентом для хмарних обчислень в поєднанні з інтегрованими компонентами, такими як MapReduce.

Потреба в розширенні людських здібностей до міркування, інтерпретації та прийняття рішень призвела до появи семантичної мережі, яка представляє собою ініціативу, спрямовану на перетворення мережі з її нинішньої, просто читаної людиною форми в форму, що піддається машинній обробці. Це, в свою чергу,

призвело до появи численних сайтів соціальних мереж з величезними обсягами даних, якими можна обмінюватися і управляти. Тому нам терміново потрібна система, яка може масштабуватися для обробки великого кількості сайтів і обробки величезних обсягів даний. Однак сучасні системи, що використовують HDFS і MapReduce недостатні з-через те, що вони не забезпечують адекватних механізмів безпеки для захисту конфіденційних даних. Ми проводимо дослідження в області безпечних хмарних обчислення. Через велику складність хмари ми стверджуємо, що в даний час буде важко забезпечити цілісне рішення для забезпечення безпеки хмара. Тому наша мета полягає в тому, щоб внести додаткові поліпшення в забезпечення безпеки хмари, які в кінцевому підсумку призведуть до створення безпечної хмари. В зокрема, ми розробляємо безпечну хмару, що складається з апаратного забезпечення (включає в себе 800 ТБ зберігання даних на механічному диску, 2400 ГБ пам'яті і кілька комп'ютерів), програмне забезпечення (включаючи Hadoop) і дані (включаючи сховище даних semantic web).

2.2 Проблеми безпеки в хмарах

Існує безліч проблем безпеки хмарних обчислень, оскільки вони охоплюють безліч технологій, включаючи мережі, бази даних, операційні системи, віртуалізацію, планування ресурсів, управління транзакціями, балансування навантаження, управління паралелізмом і управління пам'яттю. Тому проблеми безпеки для багатьох з цих систем і технологій застосовні до хмарним обчисленням. Наприклад, мережа, з'єднує системи в хмарі, повинна бути безпечний. Крім того, парадигма віртуалізації у хмарних обчисленнях призводить до декількох проблемам безпеки. Наприклад, зіставлення віртуального машини до фізичних машин повинні виконуватися надійно. Безпека даних включає в себе себе шифрування даних, а також забезпечення дотримання відповідних політик для обміну даними. Крім того, алгоритми розподілу ресурсів і управління пам'яттю

повинні бути безпечний. Нарешті, методи інтелектуального аналізу можутьбути застосовні для виявлення шкідливих програм в хмарах. Ми розширили технології та концепції, розроблені нами для secure grid, до безпечної Хмари. Ми визначили багаторівневу структуру для гарантованих хмарних обчислень, що складається з захищеного рівня віртуальної машини, безпечного рівень хмарного сховища, рівень захищених хмарних даних і рівень моніторингу захищеної віртуальної мережі (Рис. 1). Наскрізні Послуги надаються рівнем політики, рівнем хмарного моніторингу, рівнем надійності та рівнем аналізу ризиків. Для захищеної віртуальної машини (VM) Монітор ми об'єднуємо апаратні і програмні рішення у віртуальних машинах для вирішення таких проблем, як реєстратор ключів, що вивчає XEN, розроблений в Кембриджському університеті, і досліджуємо безпеку для задоволення потреб наших додатків (наприклад, безпечне розподілене сховище управління даними). Для безпечного управління хмарними сховищами ми розробляємо інфраструктуру зберігання, яка об'єднує ресурси декількох постачальників для формування масивної віртуальної системи зберігання. Коли вузол зберігання розміщує дані з декількох доменів, для кожного домену буде створена віртуальна машина для ізоляції інформації і відповідної обробки даних. Оскільки дані можуть динамічно створюватися і розподілятися по вузлах зберігання, ми вивчаємо служби безпечного управління віртуальними машинами, включаючи управління пулом віртуальних машин, управління диверсифікацією віртуальних машин і Управління доступом до віртуальних машин. Hadoop і MapReduce-це використовувані технології. Для безпечного управління хмарними даними ми розробили безпечні алгоритми обробки запитів для RDF (Resource Description Framework) і дані SQL (HIVE) в хмарах за допомогою XACML (розширюваний контроль доступу Мова розмітки) менеджер політик, що використовує платформу Hadoop/MapReduce. Для безпечного управління хмарними мережами наша мета- впровадити монітор захищеної віртуальної мережі (VNM), який буде створювати наскрізні віртуальні зв'язку з необхідною пропускною здатністю, а також контролювати обчислювальні ресурси. На рис. 2 Показані технології, які ми використовуємо для кожного з шарів. Цей проект

виконується тісно співпраця з проектом AFOSR MURI щодо гарантованого обміну інформацією та EOARD фінансується дослідницький проект з управління політикоюдля обміну інформацією. Ми завершили потужна демонстрація безпечної обробки запитів. Ми також розробили безпечне сховище алгоритмів та завершено розробку XACML для Hadoop.

2.3 Безпека даних у AWS S3

S3 - це єдиний сервіс зберігання об'єктів з можливістю блокування публічного доступу до всіх об'єктів у кошику або на рівні навчальних записів за допомогою функцій S3 Block Public Access. S3 відповідає нормативам таких стандартів, як PCI-DSS, HIPAA / HITECH, FedRAMP, директиви ЕС за захистом даних та FISMA, що дозволяє виконати законодавчі вимоги. AWS також підтримує різноманітні можливості аудиторії, щоб відстежити запрошення на доступ до вашого ресурсу в S3.

Можливості, що надає AWS S3 сховище:

- тимчасові доступи до данних (тимчасові посилання);
- елементи управління правами та доступами;
- AWS Trusted Advisor;
- Amazon S3 Object Lock по версіям (див. рис. 5);
- можливість Server-Side шифрування;
- реплікація данних.

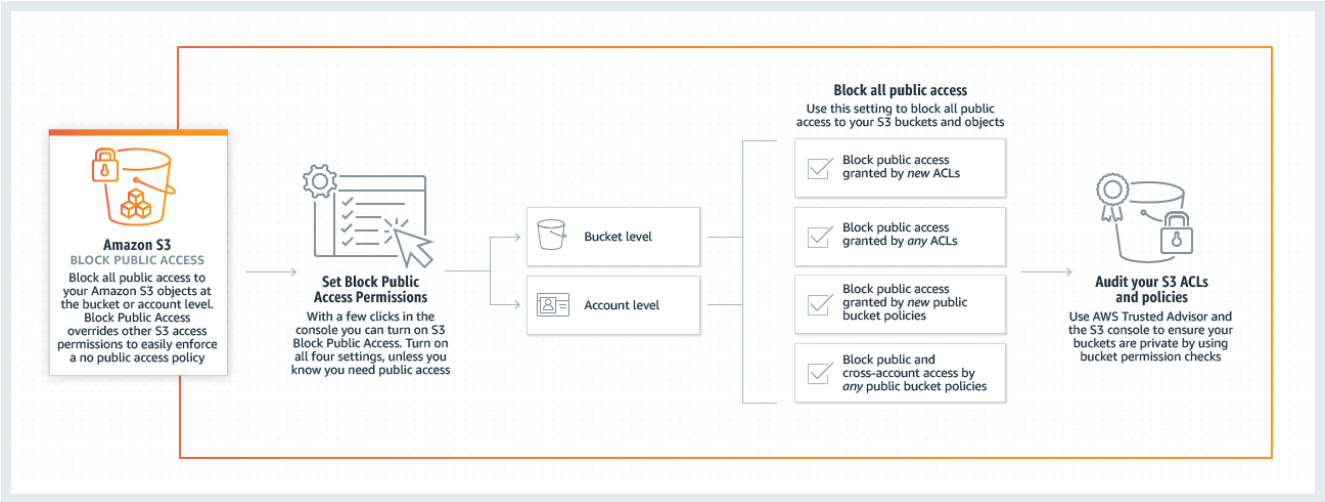

Рисунок 5 – Схема блокування публічного доступу в S3

Приблизно такі ж характеристики мають Azure Blob Storage та Google Cloud Storage. Ці сервіси мають великі можливості для береження та обробки конфиденційної інформації. Але шифрування та розшифрування даних відбувається на стороні сховища, тому можно вважати, що сховище має доступ до інформації, що є потенційною загрозою для конфиденційних даних.

2.4 Критерії сховища даних

Загальні принципи, які використовуються для визначення стану безпеки сховища - конфіденційність, цілісність та доступність.

Конфіденциальність - принцип мінімальних привілегій. Доступ до інформації належить лише особам, які явно пропонують розрішення. Ця інформація включає захист паролів користувачів, сертифікати віддаленого доступу та вміст повідомлення про електронну пошту.

Цілісність - запобігання несанкціонованому змінам неактивних або передаваних даних. Загальний підхід, що використовується при передачі даних, входить у том, що для корекції створюється унікальний відпечаток даних із

використанням одностороннього хеш-алгоритму. Хеш відправляється отримуєлюте разом з данними. Хеш даних повторно рассчитывается і порівнюється одержувачем з оригіналом, щоб дані не втрачались і не підвергалися змінам у процесах передачі.

Доступність - забезпечення доступності служби для авторизованих користувачів. Атаки типу «показ у обслуговуванні» є найбільш поширеною причиною доступності послуг для користувачів. Крім того, на випадок стихійних погіршень системи розробляють таким чином, щоб уникнути одних точок звільнення та розробити кілька екземплярів додатків у географічно віддаленому другому місці.

2.5 Авторизація та автентифікація

Багатофакторна аутентифікація (МФА, англ. Multi-factor authentication, MFA) - розширена аутентифікація, метод контролю доступу до комп’ютера, в якому користувач для отримання доступу до інформації необхідний для попередження більшого одного «довідкового механізму аутентифікації». К категоріям таких доказів відносин:

— знання - інформація, яку знає суб’єкт, наприклад пароль, ПІН-код;

— володіня- вещь, який володіє суб'єктом, наприклад електронна або магнітна карта, токен, флеш-пам'ять;

— ознака, яким влаштовує суб'єкт, наприклад біометрія, природні унікальні відліки: літо, відпечатки пальців, радужна оболочка глаз, капілярні узори, послідовність ДНК.

Фактор знання - щось, що ми знаємо - пароль. Це тайні свідоцтва, які повинні мати лише авторизований суб'єкт. Паролем може бути визначеним словом, текстовим словом, поєднанням замку або лічним ідентифікаційним номером (PIN). Парольний механізм може бути досить легко реалізований і має низьку вартість.

Однак він має суттєві недоліки: зберегти пароль у тайне зачастує биває сложно, зловмисники постійно придумують нові способи кражі, взломи та підбору пароля (см. Бандитський криптоаналіз, метод грубої сили). Це робить парольний механізм слабозащищеним. Багато секретних питань, таких як «Где ви родились?», Основні показники фактору знання, оскільки вони можуть бути відомими широкій груповій організації людей або бути дослідниками.

Фактор влади - що-то, що ми маємо - пристрій аутентифікації. Тут важливо обставительство володіння суб'єктом каким-то неповторним предметом. Це може бути лічна друк, ключ від замка, для комп'ютера це файл даних, що містить всі характеристики. Характеристика часто вбудовується в окреме пристрій аутентифікації, наприклад, пластикову карту, смарт-карту. Для зловмисника запобігання такому пристрою більш сложно, якщо ввімкнути пароль, суб'єкт може за умови, що його повідомлятимуть у разі використання пристроїв. Це робить данний метод більш захищеним, чим парольний механізм, однак вартість такої системи більш висока.

Фактор властивості - що-то, що є частиною нас - біометрія. Характеристикою є фізична особливість субъекта. Це може бути портрет, відпечаток пальця або ладоні, голос або особливість глази. З точки зору суб'єкта, данний вміст є найпростішим: не потрібно ні заповнювати пароль, ні переглядати з ним пристрій аутентифікації. Однак біометрична система повинна забезпечити високу чутливість, щоб підтвердити авторизованого користувача, а не виявити зловмисника, схожий на біометричні параметри. Також вартість такої системи досить велика. Час життя токену не повинене перевищувати 2-х хвилин.

2.6 Канал передачі даних

Для безпечного перебігу данних від користувача до хмари або від хмари до користувача потрібно організувати беспечний канал передачі даних.

HTTPS - це розширення протоколу HTTP, що підтримує шифрування за допомогою криптографічних протоколів SSL і TLS.

WSS - протокол зв'язку поверх TCP-з'єднання, призначеного для обміну повідомленнями між браузером і веб-сервером в реальному часі.

Можливо використання незалежного шифрування данних, що можна реалізувати над HTTPS таким чином, що всі данні, що будуть передаватись будуть шифруватись. Це дає змогу створити подвійне шифрування, що надає більшу конфеденційність.

2.7 Сховище даних

Сховище даних або база даних повинні відповідати таким критеріям:

— передача данних по безпечному каналу TLS/SSL;

— авторизація та автертифікація;

— рольовий доступ до даних;

— шифрування даних;

— реплікація даних.

Для цього можуть підходити такі сховища як AWS S3, Google Cloud Storage, Azure Storage.

Amazon Simple Storage Service (Amazon S3) - це послуга зберігання об'єктів, що пропонує найкращі в відображенні показники продуктивності, масштабності, доступності та безпеки даних. Це означає, що нашими клієнтами можуть бути компанії будь-яких розмірів та будь-яких обласних підприємств. Вони можуть використовувати наш сервіс для зберігання та захисту будь-яких об'єктів даних у різних ситуаціях, наприклад для забезпечення роботи озера даних, сайтів, мобільних додатків, для резервного копіювання та відновлення, архівації, корпоративних додатків, пристроїв IoT та аналізу більших даних. Amazon S3

пропонує просте використання інструментів адміністрування, що дозволяє організувати дані та точно налаштувати обмеження доступу відповідно до потреб вашого бізнесу або законодавчих вимог. Amazon S3 забезпечує надійність 99,999999999% (тут 11 днів) і зберігає дані мільйонів додатків в інтересах компаній у цілому світі.

2.8 Підпис документів

Підпис або шифрування документів, що відправляються до сховища даних дає додатковий рівень безпеки. Документи або файли, що будуть фізично знаходитить у сховищі, будудь зашифровані тому доступ до фізичного сховища не буде мати ніяких результатів, оскільки файл не буде нести ніякої інформації, тому що буде у зашифрованому вигляді.

На даному етапі можуть використовуватися деякі алгоритми шифрування.

Шифрування - оборотне перетворення інформації з метою приховування від неавторизованих осіб, з наданням, в цей же час, авторизованим користувачам доступу до неї. Головним чином, шифрування служить завданням дотримання конфіденційності інформації, що передається. Важливою особливістю будь-якого алгоритму шифрування є використання ключа, який стверджує вибір конкретного перетворення з сукупності можливих для даного алгоритму. Аутентифікація. Шифрування відкритим ключем доводить, що вихідний сервер веб-сайту належить приватному ключу, і тому йому законно присвоєно сертифікат SSL. У світі, де існує стільки шахрайських веб-сайтів, це важлива особливість.

Конфіденційність. Шифрування гарантує, що ніхто не може читати повідомлення або отримувати доступ до даних, крім законного одержувача або власника даних. Цей захід заважає кіберзлочинцям, хакерам, постачальникам послуг Інтернету, спамерам і навіть державним установам отримати доступ та прочитати персональні дані.

Відповідність нормативним актам. У багатьох галузях промисловості та урядових відомствах діють правила, які вимагають від організацій, які працюють з персональною інформацією користувачів, зберігати ці дані в зашифрованому вигляді. Вибірка стандартів регулювання та відповідності, що забезпечують шифрування, включає HIPAA, PCI-DSS та GDPR.

Безпека. Шифрування допомагає захистити інформацію від порушень даних, незалежно від того, перебувають вони в стані спокою або в процесі передачі. Наприклад, навіть якщо корпоративний пристрій втрачено або викрадено, дані, що зберігаються на ньому, швидше за все, будуть захищені, якщо жорсткий диск належним чином зашифрований. Шифрування також допомагає захистити дані від зловмисних дій, таких як атаки "людина посередині", і дозволяє сторонам спілкуватися, не боячись витоку даних.

Розглянемо популярні алгоритми шифрування.

AES. Стандарт розширеного шифрування (AES) - це надійний стандартний алгоритм, що використовується урядом США, а також іншими організаціями. Незважаючи на надзвичайну ефективність у 128-бітовій формі, AES також використовує 192- та 256-бітні ключі для дуже вимогливих цілей шифрування. AES широко вважається невразливим для всіх атак, за винятком грубої сили. Незважаючи на це, багато експертів з безпеки в Інтернеті вважають, що AES врешті буде розглядатися як стандартний стандарт шифрування даних у приватному секторі.

Потрійний DES. Triple DES є наступником оригінального алгоритму Data Encryption Standard (DES), створеного у відповідь хакерам, які придумали, як порушити DES. Це симетричне шифрування, яке колись було найпоширенішим симетричним алгоритмом у галузі, хоча воно поступово припиняється. TripleDES застосовує алгоритм DES тричі до кожного блоку даних і зазвичай використовується для шифрування паролів UNIX та PIN-кодів банкоматів.

RSA. RSA - це асиметричний алгоритм шифрування з відкритим ключем і стандарт для шифрування інформації, що передається через Інтернет. Шифрування RSA є надійним і надійним, оскільки створює величезну купу безглуздість, яка

перешкоджає потенційним хакерам, змушуючи їх витрачати багато часу та енергії, щоб зламати системи.

Blowfish. Blowfish - ще один алгоритм, який був розроблений для заміни DES. Цей симетричний інструмент розбиває повідомлення на 64-розрядні блоки та шифрує їх окремо. Blowfish завоював репутацію швидкості, гнучкості та незламності. Це загальнодоступне, тож це робить його безкоштовним, додаючи ще більше привабливості. Blowfish зазвичай можна знайти на платформах електронної комерції, захисті платежів та в інструментах управління паролями.

Дві риби. Twofish - наступник Blowfish. Це безліцензійне симетричне шифрування, яке розшифровує 128-розрядні блоки даних. Крім того, Twofish завжди шифрує дані в 16 раундів, незалежно від розміру ключа. Twofish ідеально підходить як для програмного, так і для апаратного середовища і вважається одним із найшвидших у своєму роді. Багато сучасних програмних рішень для шифрування файлів і папок використовують цей метод.

Рівест-Шамір-Адлеман (RSA). Rivest-Shamir-Adleman - це асиметричний алгоритм шифрування, який працює на розкладання на множники добутку двох великих простих чисел. Лише користувач, який знає ці два номери, може успішно декодувати повідомлення. Цифрові підписи зазвичай використовують RSA, але алгоритм сповільнюється, коли він шифрує великі обсяги даних.

2.9 Розподілене зберігання шматків документу

Після підпишу (шифрування) файлу, файл може бутфи розбитий на шматки, які в свою чергу повинні бути збережені на різних фізичних серверах (див. рис. 6).

Таким чином окремі шматки файлу не несуть ніякої цінності. Якщо зловмистник отримає доступ до фізичного сховища, то шматок файлу не може бути використаний та прочитай без інших шматків.

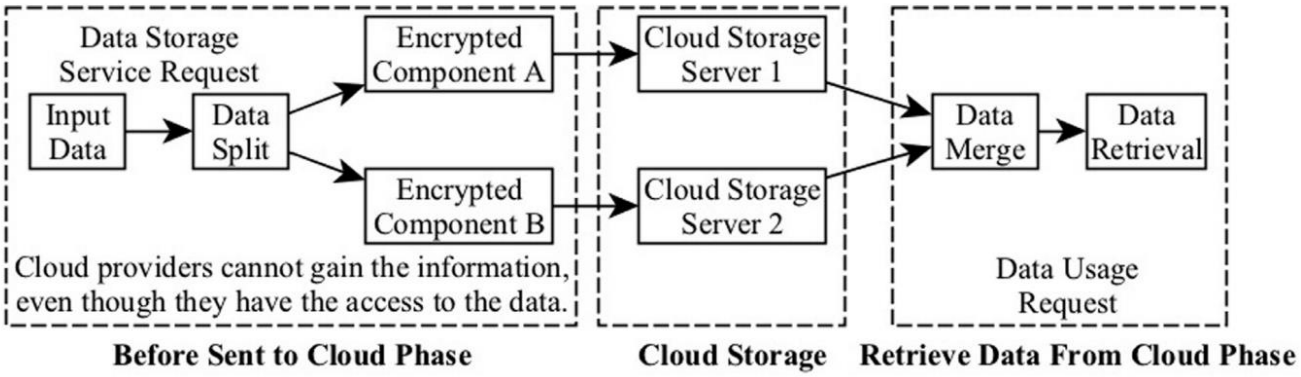

Рисунок 6 – Розподілення шматків файлів

Для розділення шматків повинен бути використаний алгоритм змішування та розділення файлу. Той самий алгоритм повинен бути використаний і для зворотнього циклу. Таким чином документ буде надійно захищений від вктручання у фізичні носії.

2.10 Зберігання ключів шифрування

Ключі шифрування повинні лежати в окремому сховищі та надійно захищені. Зберігання змінних середовища повинно бути в окремо захищеному місці. Таким місцем може бути AWS Secters Manager.

AWS Secrets Manager дозволяє захищати конфіденційні дані, які використовуються для доступу до додатків, сервісів і IT-ресурсів. Сервіс надає можливість простий ротації і вилучення даних для доступу до БД, ключів API та інших конфіденційних даних, а також управління ними на протязі всього життєвого циклу. Користувачі і додатки витягають дані для доступу за допомогою виклику API сервісу AWS Secrets Manager, що позбавляє від необхідності вказувати в коді конфіденційні дані в текстовому форматі. З метою ротації конфіденційних даних для доступу Secrets Manager підтримує вбудовану інтеграцію з Amazon RDS,

Amazon Redshift і Amazon DocumentDB. Крім того, сервіс передбачає можливість розширення для роботи з іншими типами конфіденційних даних, включаючи ключі API і маркери OAuth. AWS Secrets Manager також дозволяє керувати доступом до конфіденційних даних за рахунок точної настройки дозволів і здійснювати централізований контроль ротації даних для доступу до ресурсів, розташованим в хмарі AWS, в сервісах сторонніх постачальників або локально.

Ці паролі повинні бути оновлєні за певний період часу. Доступ до паролей повинен бути у вузького кола людей з двухфакторної автентификацією.

2.11 Розділення функції, відповідальності, рівнів доступу

В розглянутій моделі системи можна виділити різні відповідальності, різний незалежній функціонал с різним рівнем доступу. Для такої системи краще за всі архітектури підійде микросервесна архітектура.

Мікросервісная архітектура - варіант сервіс-орієнтованої архітектури програмного забезпечення, спрямований на взаємодію наскільки це можливо невеликих, слабо пов'язаних і легко змінюваних модулів – мікросервісов.

Властивості, характерні для мікросервісной архітектури:

— модулі можна легко замінити в будь-який час: акцент на простоту, незалежність розгортання та оновлення кожного з мікросервісов;

— модулі організовані навколо функцій: мікросервіс по можливості виконує тільки одну досить елементарну функцію;

— модулі можуть бути реалізовані з використанням різних мов програмування, фреймворків, сполучного програмного забезпечення, виконуватися в різних середовищах контейнеризації, віртуалізації, під керуванням різних операційних систем на різних апаратних платформах: пріоритет віддається на користь найбільшої ефективності для кожної конкретної функції, ніж стандартизації засобів розробки і виконання;

— архітектура симетрична, а не ієрархічна: залежності між мікросервісамі однорангові.

Мікросервіси можуть розроблятися і розвиватися незалежно один від одного. Кожен мікросервіс може має свою власну базу даних, щоб бути відокремленою від інших служб. Мікросервіси інкапсулюють певний функціонал системи і створюються виходячи з особливостей предметної області корпоративного веб-додатки.

Основні переваги мікросервісной архітектури:

— використання різних мов програмування і програмних засобів;

— оптимальних для реалізації кожного мікросервіса;

— взаємозамінність мікросервісов;

— незалежність мікросервісов один від одного, кожен мікросервіс може бути розгорнутими незалежно від інших служб;

— спрощення процесу масштабування розробляється веб-програми;

— організація мікросервісов як модулів навколо окремих функцій.

Кожен мікросервіс - це невелика монолітна програма, яка виконує свою функцію. У корпоративних веб-додатках з мікросервісной архітектурою можна додавати будь-яку кількість нових мікросервісов, розширюючи функціональність системи (див. рис. 7).

Таким чином можна виділити такі складові розробленої моделі:

— мікросервіс авторизації;

— мікросервіс підпису/шифрування та розшифрусання документів;

— сервер gateway, що буде додатково шифрувати дані;

— мікросервіс для збереження користувацької інформації;

— мікросервіс, що маніпулює документами та розділяє файли на шматки, зберігає шматки на різних фізичних сховищах, та записує інформацію про ці шматки в базу даних;

— для кожного мікросервіса знадобиться своя база даних та автентифікаційний флоу;

— всі паролі доступів повинні бути розміщені в спеціальних сховищах.

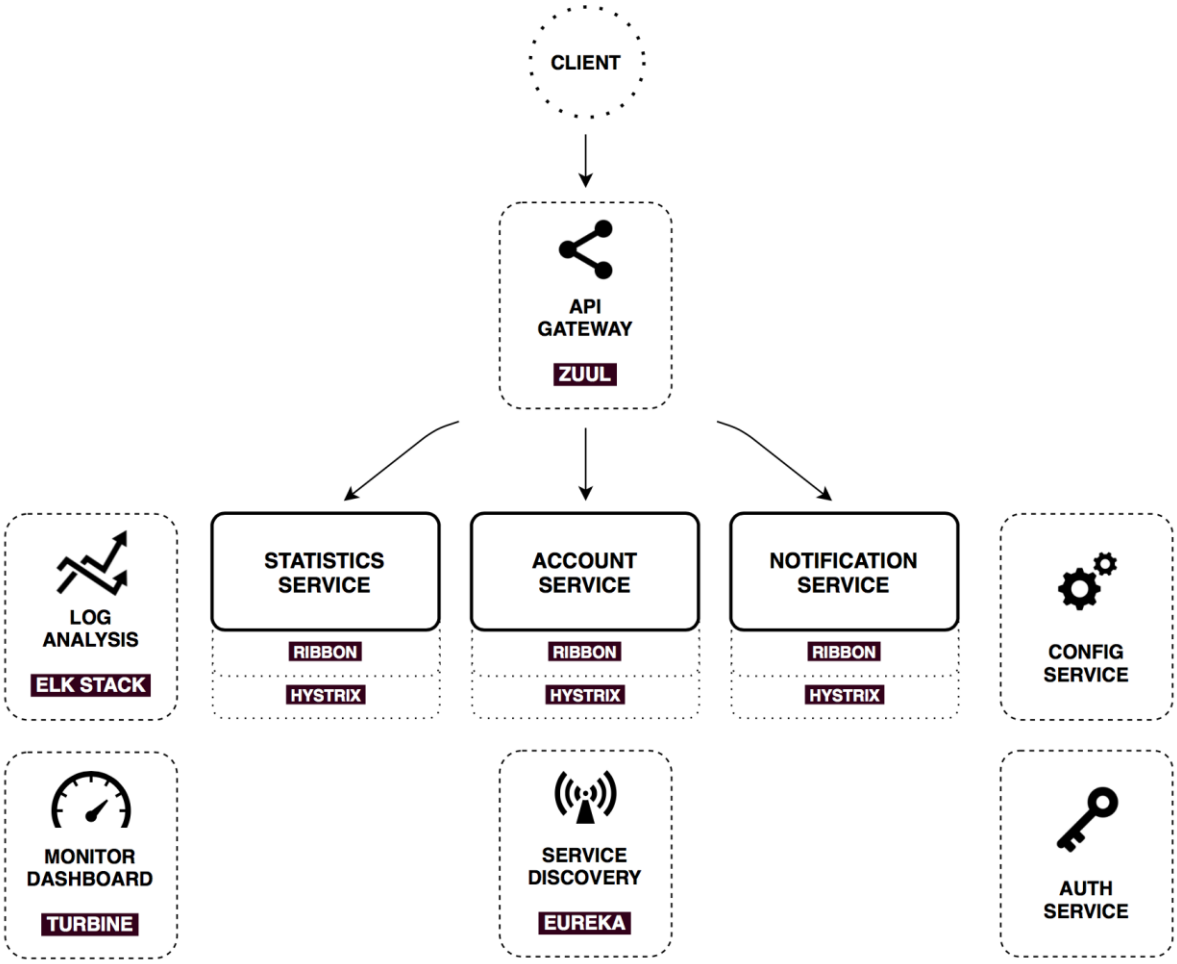

Рисунок 7 - Схема мікросервісної архітектури

3 ОГЛЯД АРХІТЕКТУРНИХ РІШЕНЬ

3.1 Vanish архітектура.

Система має такі вимоги (див. рис. 8):

— навіть якщо зловмисник може заднім числом отримати первозданну копію цих даних та будь-які відповідні постійні криптографічні ключі та парольні фрази до цього часу очікування, можливо, із збережених або заархівованих копій;

— без використання явної дії видалення користувачем або сторонами, що зберігають ці дані;

— без необхідності модифікувати будь-яку із збережених або заархівованих копій цих даних;

— без використання захищеного обладнання;

— не покладаючись на впровадження будь-яких нових зовнішніх служб, які потрібно було б використовувати (незалежно від того, довіряють вони чи ні).

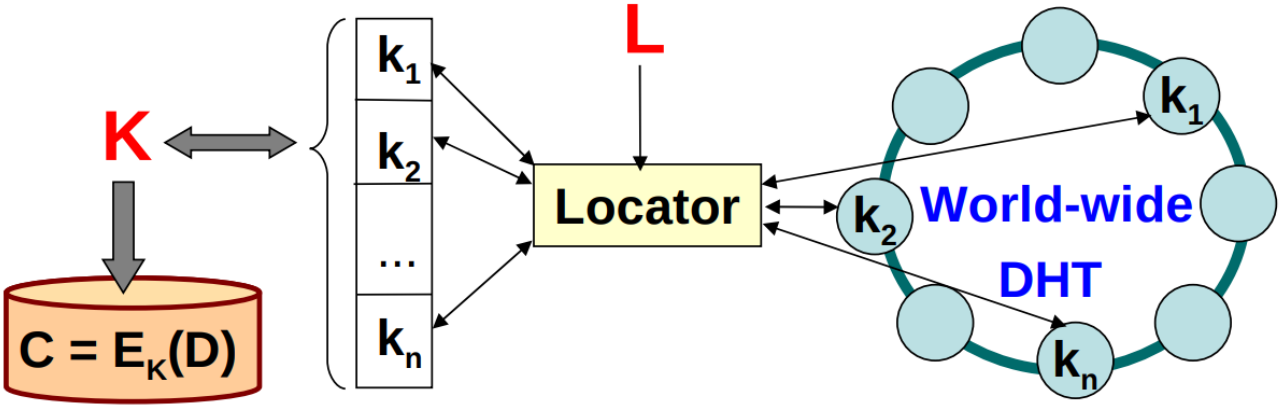

Рисунок 8 - Архітектура системи Vanish

Ключовим розумінням підходу та відповідних систем, яка називається

Vanish, є використання послуг, які надають децентралізовану глобальну інфраструктуру P2P і, зокрема, розподіленими хеш-таблицями (DHT). Як випливає з назви, DHT виготовлений 18-м симпозіумом із системою безпеки USENIX 301 для асоціації USENIX для реалізації надійної бази даних даних про індексу значень на колекції вузлів P2P [64]. Інтуїтивно Vanish зашифровує дані користувачів локально за допомогою випадкового ключа шифрування, невідомого учасника, знищує локальну копію ключа, а потім розбризкує біти (секретні папки Шамір [49]) ключа через випадкові індекси (відже, випадкові вузли) у DHT. Наш вибір DHT як системи зберігання для Vanish походить від трьох унікальних характерних можливостей DHT, що робить їх приватними для наших цілей зниження даних. По-перше, їх величезний масштаб (понад 1 млн вузлів для Vuze DHT [28]), географічне розповсюдження вузлів у багатьох країнах та повна децентралізація роблять їх надійними для потужних та юридично впливових супротивників (див. рис. 9).

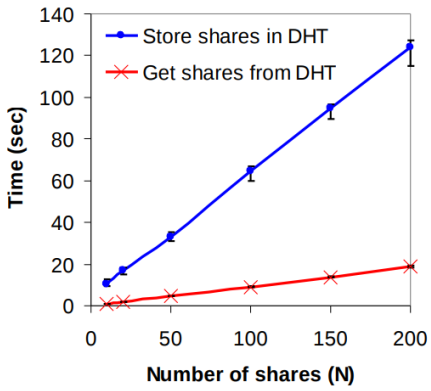

Рисунок 9 – Маштабування DHT операцій

По-друге, DHT призначені для забезпечення надійного розподіленого сховища [35, 56, 64]; ми використовуємо цю можливість системи для того, щоб захищати дані залишаються доступними для користувачів, бажаних проміжок часу. Нарешті, але не зважаючи на те, що DHT має таку властивість, якою ми можемо користуватися унікальним і нестандартним способом: той факт, що DHT постійно

змінюється, означає, що розповсюджена інформація про природні особи зникає (зникає), коли спільні DHT перетворюються або внутрішньо очищаються, через що пропонують захищені дані постійно недоступні з часом.

3.2 SafeVanish архітектура

Існуючі ультрасучасні схеми самознищення даних (Vanish) демонструють крихкість для хопинг атаки та сніфинг атаки в реалістичному застосуванні. Нова схема SafeVanish, яка базується на розширенні діапазону довжини ключових ресурсів та застосуванні криптосистеми з відкритим ключем, щоб помножити вартість хопинг атаки, включаючи вимоги до сховища та пропускну здатність мережі, а також уникнути сніфинг атаки.

3.3 Архітектура Single Secure

Данна модель створена для того, щоб забезпечити майже невразливість системи, оскільки дані будуть шифруватись на стороні клієнта (див. рис. 10). Модель представлена декілька важливих складових.

Модель має микросервіс генерування ключів шифрування. Тобто користувач буде генерувати ключ за допомогою віддаленого мікросервісу. Та зберігає цей ключ локально.

Надалі користувач повинен пройти процес авторизації через авторизаційний мікросервіс, що поверне користувачу токен авторизації.

Наступним процессом для зберігання даних є шифрування даних симетричним алгоритмом на боці клієнта, після чого дані відправляються на головний сервер вже в зашифрованому вигляді.

Ще одним кроком до надійної системи є поділ переданого до серверу файлу на шматки спеціальним симетричним алгоритмом.

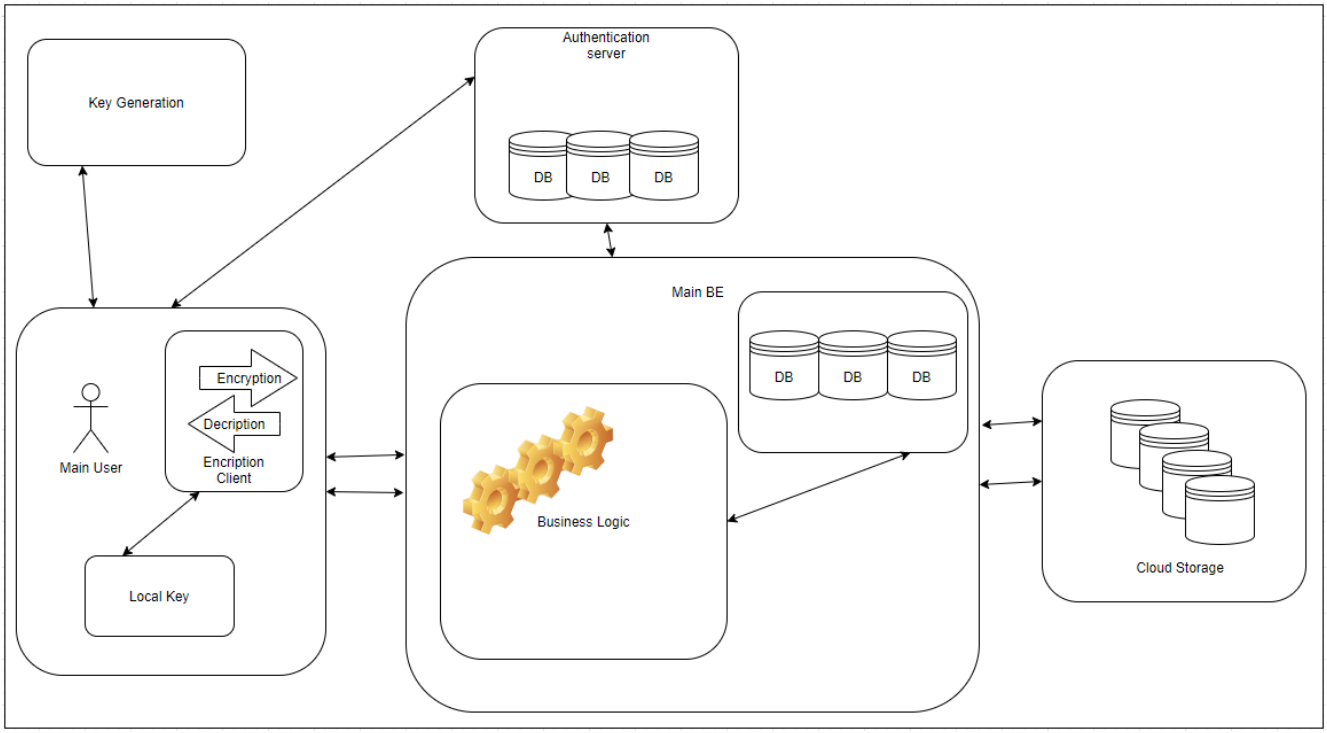

Рисунок 10 – Модель даних з доступом лише для одного користувача

В данній системі достатньо чотирьох шматків та алгоритм не потребує додаткових паролей. Після поділу файлів створюється кожному файлу унікальний ідентификатор, що відправляється на запис до хмарного сховища.

Процесс перегляду файлу має той самий алгоритм дій. Авторизований користувач запрошує дані з серверу. Після перевірки прав головний сервер витягує з бази даних ідентифікатори шматків файлу, після чого створює один файл за тим самим алгоритмом, що й був створений файл. Далі зашифрований файл відправлється на клієнт, де розшифровується використовуючи симетричний алгоритм та пароль до нього.

Звичайно увесь потік данних проходить по захищеному каналу HTTPS/WSS.

Дана модель має такі недоліки:

— зашифрований файл залишається назавжди зашифрованим одним ключем;

— втрата ключа шифрування призведе до втрати доступу до файлів, оскільки не буде можливості його розшифрувати;

— неможливість оновлення ключей шифрування;

— потребує потужностей для шифрування/дешифрування документів;

— доступ до файлу є тільки в одного користувача.

Дана модель має такі переваги:

— ніхто по мережі не має доступу до розшифрованого файлу;

— доступ до файлу є тільки в одного користувача.

3.4 Архітектура SSDD

Щоб скласти основу нашої схеми вводимо поняття об’єкта зникаючих даних (VDO) [3]. VDO інкапсулює дані користувача (наприклад, файл або повідомлення), щоб запобігти необмеженому збереженню їх вмісту у третіх сторін та стати джерелом витоку інформації із зворотним зв'язком. Незалежно від того, копіюється, передається чи зберігається VDO в Інтернеті, він стає нечитабельним через заздалегідь визначений проміжок часу (див. рис. 11).

Система має кілька ключових припущень:

— обмежена у часі цінність. Схема самознищення використовується для захисту безпеки даних користувача, що є цінним для користувача лише протягом обмеженого періоду часу;

— попередньо визначений тайм-аут. Коли користувач інкапсує свою інформацію в VDO, він знає приблизний час життя VDO;

— підключення до Інтернету. Користувач може підключитися до Інтернету під час надсилання спільних ресурсів ключа, які генеруються відповідно

до ключа розшифровки, витягнутого шифротексту та відповідної витягнутої позиції, до мережі DHT та декапсуляції VDO до його закінчення;

— жодних атак на VDO до його закінчення. Метою нашої схеми є захист конфіденційних даних користувачів від майбутніх атак. Зловмисник не знає, які конкретно дані атакувати, поки вони не закінчаться. Це означає, що VDO не буде атакований, поки термін його дії не закінчиться;

— довірені авторизовані користувачі. Уповноважені користувачі, які мають доступ до одного і того ж VDO, повинні довіряти один одному і не зберігатимуть жодної копії відкритого тексту, відновленого з VDO. Це припущення є простим, інакше для нашої системи було б неможливо захистити від конфіденційних даних користувача, якщо авторизовані користувачі витікають або постійно зберігають відкритий текст, який відновлюється з VDO;

Система має наступні цілі проектування:

— автоматичне знищення через зазначений час. Дані користувача повинні бути знищені автоматично без явних дій користувача або будь-якої третьої сторони, що зберігає VDO. Він забезпечує тип прямої таємниці, коли об'єкти із закінченим терміном дії надійно захищені, навіть якщо VDO скомпрометований противниками після закінчення часу його дії.

— доступний до закінчення терміну дії. До закінчення терміну дії VDO його вміст повинен бути доступним для авторизованих користувачів;

— ніякої спеціальної інфраструктури та безпечного обладнання. Схема може бути реалізована з наявною інфраструктурою. Він не повинен покладатися на зовнішню або спеціальну інфраструктуру і не вимагати використання будь-якого виділеного захищеного обладнання;

— немає змін на традиційну схему симетричного шифрування. Схема розширює ключовий простір традиційної симетричної схеми шифрування без будь-якого чергування із самою схемою шифрування, витягуючи та розподіляючи частину зашифрованого тексту в мережі DHT;

— немає довірених третіх осіб. Схема не покладається на будь-які

довірені треті сторони;

— стійкість до атак. Схема може протистояти не тільки традиційному криптоаналізу та атаці грубої сили, але й деяким атакам у мережі DHT, таким як атака нюху магазину, атака пошуку нюху та стандартні атаки DHT.

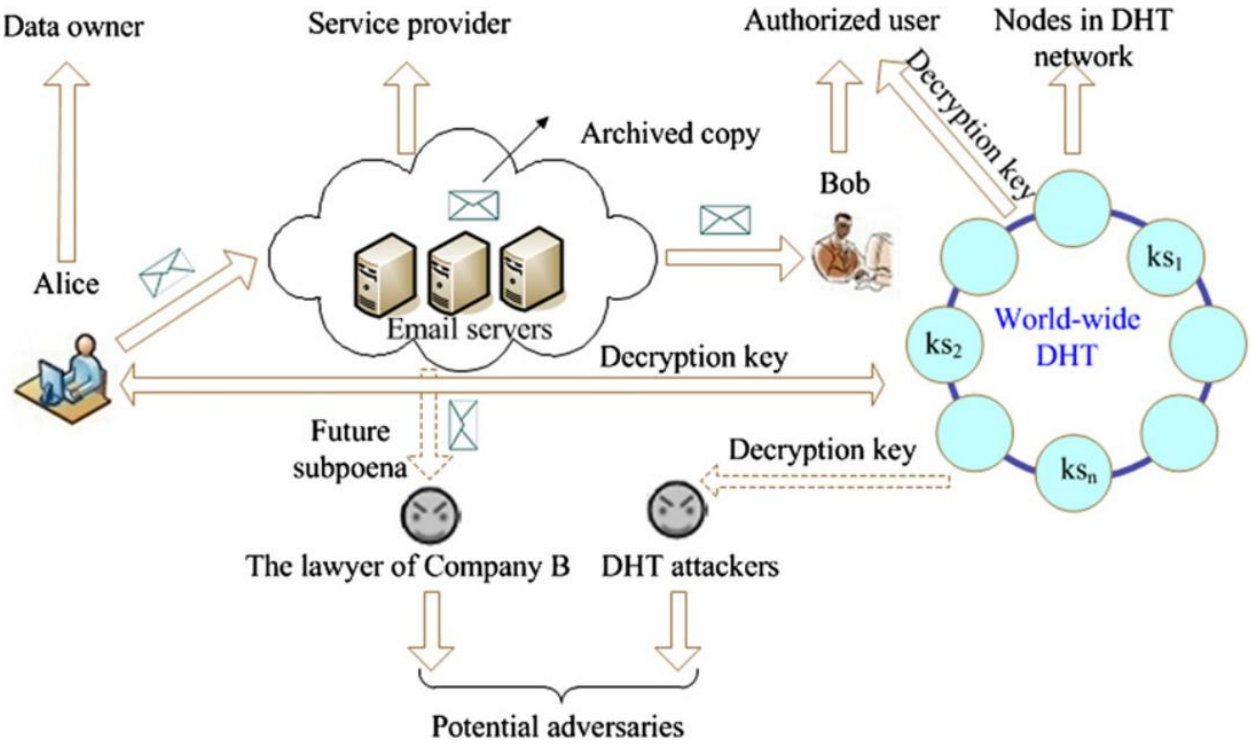

Рисунок 11 – Архітектура SSDD системи

Цільова система складається з таких сторін: власники даних, уповноважені користувачі, постачальники послуг, вузли в мережі DHT та потенційні супротивники. Власник даних надсилає свій VDO постачальнику послуг, а постачальник послуг несе відповідальність за зберігання VDO та забезпечення отримання лише конфіденційних даних лише уповноваженими користувачами. Вузли в мережі DHT відповідають за зберігання ключових ресурсів. Згідно з четвертим припущенням, потенційні супротивники атакують VDO після закінчення терміну його дії. Крім того, потенційні супротивники атакують мережу

DHT до закінчення часу, оскільки всі ключові папки знищуються після закінчення часу.

Метою є захист від розкриття даних із зворотною силою, наприклад, у відповідь на повістку в суд, наказ суду, зловмисний компроміс архівованих даних та випадковий витік даних. Як описано вище, у нашій схемі авторизовані користувачі довіряють один одному, і потенційні супротивники атакують VDO після закінчення терміну його дії або атакують мережу DHT до закінчення часу. У нашій моделі безпеки потенційні супротивники можна класифікувати на два типи: Супротивники, що ініціюють атаки на VDO після закінчення часу, і противники, що ініціюють атаки на мережу DHT до закінчення часу. Наприклад, до першого виду належать як постачальник послуг, який може надавати VDO для допомоги майбутнім викликам до суду, так і адвокат компанії B, який може отримати вміст даних користувачів законними способами; Тоді як супротивники, які можуть отримати ключові частки в мережі DHT до закінчення часу, потрапляють до другого виду.

3.5 Архітектура FullPP

Для того, щоб скласти основу схеми FullPP, були введені такі поняття:

— Бажаний час випуску: це момент часу, визначений власником, який бажає, щоб певний фрагмент конфіденційних даних був відкритий або прочитаний уповноваженим користувачем у цей момент.

— Час дії: це пороговий час, визначений власником. Конфіденційні дані можуть бути прочитані або використані уповноваженим користувачем до цього моменту часу і більше не мають значення для авторизованого користувача після цього моменту часу. Тому конфіденційні дані слід надійно видаляти або самознищувати після закінчення терміну дії. Період авторизації: це період часу, починаючи з бажаний час випуску і закінчується часом закінчення.

— Період авторизації: це проміжок часу, починаючи з бажаного часу випуску і закінчуючи часом закінчення.

— Повний життєвий цикл: це часовий інтервал, що містить бажаний час випуску, період авторизації та термін дії. Метою даної статті є забезпечення повного захисту конфіденційності життєвого циклу для конфіденційних даних у хмарних обчисленнях.

— Об'єкт самознищення даних (SDO): Це структура даних, подібна до VDO, описаної в [27]. SDO інкапсулює конфіденційні дані та запобігає витоку їх вмісту до будь-якого неавторизованого користувача.

Дана архітектура має наступні системні припущення.

Ефективність періоду дозволу. Схема FullPP використовується для захисту обмеженої у часі безпеки конфіденційних даних власника. Захист діє лише протягом періоду авторизації конфіденційних даних, попередньо визначених власником.

Підключення до Інтернету. Власники та авторизовані користувачі можуть підключатися до Інтернету, щоб вони могли взаємодіяти з KGC (центр генерації ключів IBE), сервером часу та хмарними серверами для інкапсуляції та декапсуляції SDO протягом періоду авторизації.

Довірені авторизовані користувачі та KGC. Ми повинні довіряти авторизованим користувачам, які мають дозвіл на доступ до SDO, і не створюватимуть резервні копії жодної копії чутливого відкритого тексту, відновленого з SDO. Аналогічно, уповноважені користувачі довіряють один одному. Крім того, ми повинні довіряти KGC, який відповідає за підтримку алгоритму ID-TRE. Він створює приватний ключ для кожного авторизованого користувача.

Дана архітектура має наступні системні вимогиУ цьому розділі ми представляємо системні вимоги Схема FullPP.

Конфіденційні дані повинні бути доступними з бажаного часу випуску. Конфіденційні дані, інкапсульовані як SDO, повинні бути доступними для читання протягом періоду авторизації; ніхто не може отримати до нього доступ до цього

моменту.

Самостійне активування відбувається після закінчення часу. SDO може самостійно шукати будь-яких інших дій із боку власника або іншої сторінки. Забезпечуючи пряму безпеку, у якій SDO є безпечною та недоступною для будь-якого користувача, який отримує копію SDO після закінчення терміну дії.

Стійкість до атак. З одного боку, схема FullPP не створює нових ризиків для безпеки та конфіденційності для користувачів; з іншого боку, вона може протистояти одночасно атакам грубої сили та атакам Сибілі.

Дотримуючись схеми SSDD [1], системна модель схеми FullPP складається з наступних семи типів сутності: власники чутливих даних, KGC, сервер часу, хмарні сервери, авторизовані користувачі, вузли в мережі DHT та потенційні зловмисники, як показано на рисунку 12.

Обов'язки власників конфіденційних даних, авторизованих користувачів, хмарних серверів, вузлів у мережі DHT та потенційних зловмисників такі ж або подібні, як у схемі SSDD [1]. Сервер часу - це пасивний контрольний сервер часу без будь-якої взаємодії з іншими сутностями в системі. Він надає лише точну специфікацію часу випуску у вигляді оновлених ключів з обмеженим часом [4]. Доданий KGC відповідає за підтримку генерації та управління ключами IBE.

Потенційні атаки на схему FullPP можна класифікувати на три категорії: 1 атака на SDO після закінчення терміну дії, 2 атаки на мережу DHT протягом періоду авторизації SDO та 3 атаки як на SDO, так і на мережу DHT до закінчився час. Перші два типи атак такі ж, як і в існуючих схемах самознищення [1, 2].

Схема FullPP використовує симетричний алгоритм шифрування для шифрування конфіденційних даних і використовує алгоритм ID-TRE для шифрування ключа дешифрування. Обидва зашифровані тексти генерують спільні дані зашифрованого тексту (CS), CS та SDO за допомогою ряду алгоритмів. Потім CS розпорошується у велику мережу DHT, що дозволяє CS зникнути після закінчення часу, щоб досягти самознищення конфіденційних даних без втручання людини. Схема FullPP складається з двох основних фаз: фази інкапсуляції та фази декапсуляції відповідно до існуючих процесів саморуйнування. На рисунку 12

показано робочий процес схеми FullPP.

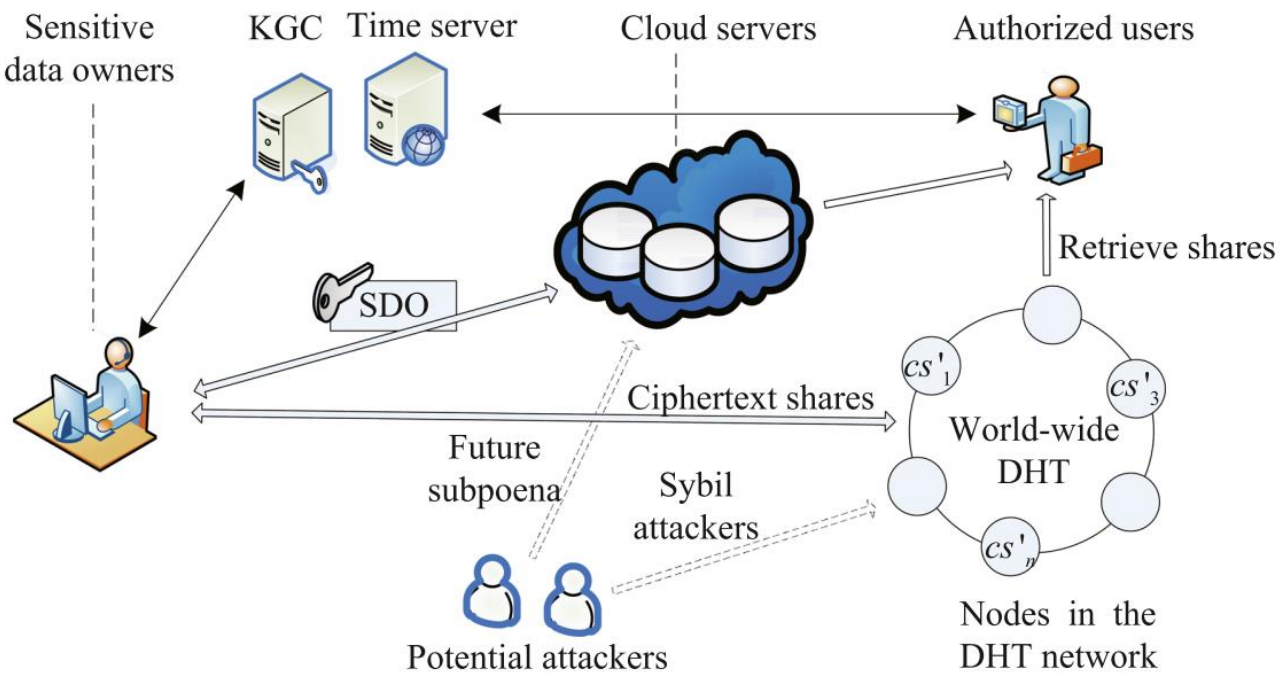

Рисунок 12 – Архітектура FullPP системи

Ми представимо схему FullPP на двох рівнях: Рівень системи та Рівень алгоритму. Перший описує реалізації верхніх операцій, тоді як другий фокусується на реалізаціях базових алгоритмів, що викликаються операціями системного рівня. Деталі цих двох рівнів детально розроблені в далі.

Налаштування системи. На етапі ініціалізації системи власник даних вибирає великий параметр захисту к і викликає налаштування алгоритму (к), що належить до рівня алгоритму, для генерації параметрів системних параметрів, відкритого ключа системи та головного ключа.

Управління зашифрованим текстом конфіденційних даних. На цьому етапі власник даних повинен зашифрувати конфіденційну інформацію та отримати зашифрований текст за допомогою ряду алгоритмів, щоб надати основу схем самозмінення. По-перше, власник даних створює алгоритм Encrypt для шифрування конфіденційних даних симетричним ключем K для отримання

вихідного зашифрованого тексту C, а потім створює алгоритм DataExtract для генерації витягнутих зашифрованих CE та інкапсульованого CD-зашифрованого тексту, де CE створений для генерації CS інкапсуляції SDO. Нарешті, SDO зберігається на хмарних серверах.

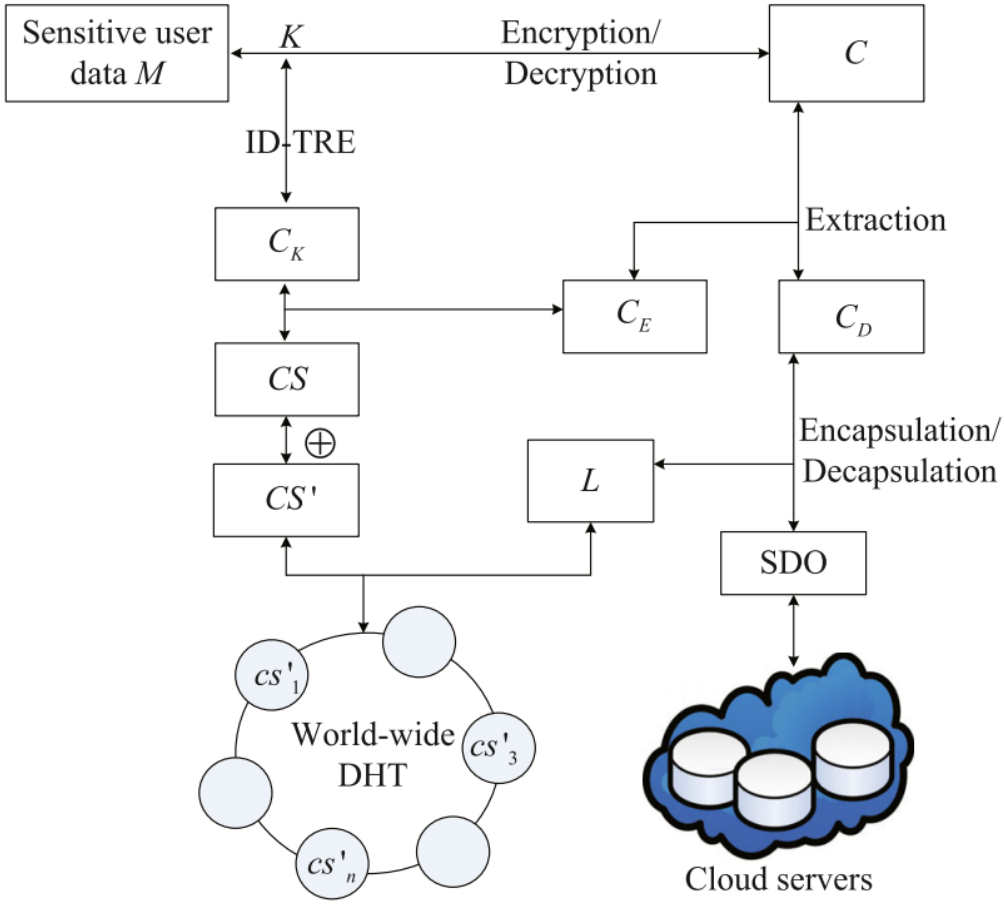

Рисунок 13 - Принцип роботи схеми FullPP

Шифрування ключа дешифрування K. Перш ніж розподіляти K в мережі DHT, власник даних повинен обробити ключ наступним чином: По-перше, власник даних отримує ідентифікатор ідентифікатора уповноваженого користувача i, з яким він хоче надати конфіденційні дані. Потім він встановлює час випуску миттєвого Ti і викликає алгоритм IdTimeReEncrypt для шифрування K, щоб отримати ключовий шифротекст CK.

Управління КС. У цій операції власник даних використовує схему секретного обміну Shamir [35] для створення CS наступним чином. По-перше, власник даних отримує як CE, так і CK і викликає алгоритм PolynomialGenerate для генерації поліномів Лагранжа F1 (x), F2 (x), ..., Fq + 1 (x). По-друге, він використовує алгоритм CSharesGenerate для отримання CS у вигляді CS = cs1, cs2, ..., csi, .., csn. Крім того, він використовує алгоритм CSharesTransform для перетворення CS в CS у вигляді CS = cs1, cs 2, ..., cs i, .., cs n. І нарешті, він вибирає ключ довільного доступу L і викликає алгоритм CSharesDistribute для посипання n спільних ресурсів CS до мережі DHT, який зникне після закінчення часу. Тим часом власник даних видаляє ключ дешифрування K.

Інкапсуляція SDO. Перш ніж зберігати зашифрований текст конфіденційних даних на хмарних серверах, власник даних повинен інкапсулювати інкапсульований зашифрований текст у SDO. Отримавши витягнутий компакт-диск із зашифрованим текстом, індекс довільного доступу L та параметри системних параметрів, власник даних викликає алгоритм SDOGenerate для отримання SDO, який потім надсилається на хмарні сервери.

Гнучкий контроль доступу під час періоду авторизації SDO. Коли хтось із друзів власника хоче отримати доступ до SDO протягом періоду його авторизації, він повинен пройти ідентифікацію особи, щоб стати авторизованим користувачем. Він повинен виконати деякі процеси.

По-перше, уповноважений користувач отримує SDO із хмарних серверів і викликає алгоритм Decapsulate, щоб отримати L і CD. Він засіває PRNG з L, щоб отримати більше t - 1 індексів у l1, ..., li, ..., ln та збирає акції cs i з цими індексами з мережі DHT. По-друге, авторизований користувач використовує cs i для отримання csi за допомогою свого IDi. Він використовує багаточлен інтерполяції Лагранжа для відновлення поліномів q +1 і реконструює CE і CK. По-третє, уповноважений користувач використовує CE і CD для відновлення C. Крім того, він отримує як закритий ключ, так і обмежений за часом ключ оновлення від надійного сервера і викликає алгоритм IdTimeReDecrypt для дешифрування CK для отримання оригінального ключа дешифрування K. Нарешті, уповноважений

користувач розшифровує C за допомогою K, щоб отримати відкритий текст конфіденційних даних M. Якщо уповноважений користувач хоче отримати доступ до SDO до бажаного часу випуску, він не може правильно розшифрувати CK, щоб отримати K, оскільки у нього немає правильного обмеження часу оновлення ключа. Завдяки вищезазначеним операційним процесам алгоритми можуть контролювати авторизованим користувачам доступ до різних конфіденційних даних у хмарних обчисленнях. Отже, схема FullPP здатна забезпечити гнучкий контроль доступу під час періоду авторизації SDO.

Самознищення SDO після закінчення часу. Схема FullPP повністю використовує функції самооновлення даних мережі DHT після закінчення терміну дії. Тобто кожен вузол у мережі DHT періодично самовідкидає спільні ресурси CS, щоб звільнити місце для зберігання, щоб зберегти нові дані без втручання вручну. Таким чином, CS надійно видаляється після закінчення терміну дії SDO. Оскільки ми втратили CS, ми не можемо отримати CS для реконструкції CK і CE, а також отримати ключ розшифровки K. Крім того, через відсутність CE, навіть якщо ми можемо отримати SDO з хмарних серверів після закінчення терміну дії, ми не можемо відновити оригінал зашифруйте текст C та отримайте відкритий текст даних про конфіденційність. Тому схема FullPP здатна здійснити самознищення конфіденційних даних.

Аналіз безпеки. У схемі FullPP спільні файли зашифрованого тексту, включаючи зашифрований текст ключа та частину зашифрованого тексту конфіденційних даних, розподіляються в мережу DHT після перетворення і будуть самостійно видалені через унікальну властивість мережі DHT. Ключ розшифровки та вихідний зашифрований текст конфіденційних даних не підлягають відновленню. Отже, є три шляхи порушення нашої схеми: використання традиційних атак для порушення SDO на хмарних серверах після закінчення терміну дії, використання атак Sybil для компрометації мереж DHT протягом періоду авторизації та використання одночасних атак для порушення системи в будь-який час. Перші два також трапляються в існуючих схемах самознищення, таких як Vanish [27] та SSDD [1]. Третій - використовувати як традиційний

криптоаналіз, так і грубу атаку на хмарні сервери, а також атаки Sybil на мережу DHT.

Безпеку схеми FullPP можна проаналізувати з наступних трьох питань: 1 Якщо алгоритм шифрування може протистояти традиційним атакам? 2 Якщо мережа DHT може протистояти атакам Sybil? 3 Якщо система може протистояти як традиційним атакам, так і атакам Sybil?

У наступному підрозділі ми аналізуємо безпеку схеми FullPP на основі трьох вищезазначених питань. У таблиці 2 порівнюється схема FullPP, запропонована в цьому дослідженні, із системою Vanish [27], схемою SafeVanish [36], схемою SSDD [1] та схемою Single Secure[2].

У системі Vanish [27] випадковим симетричним ключем шифрується лише відкритий текст. Крім цього, схема SSDD [1] та схема МКС [2] також розділяють, асоціюють та витягують зашифрований текст таким чином, щоб кожен фрагмент зашифрованого тексту був пов'язаний між собою. FullPP не тільки витягує зашифрований текст, як це роблять у SSDD та ISS, але шифрує ключ дешифрування за допомогою алгоритму ID-TRE. Відкритий текст ніколи не буде відновлений без повного оригінального зашифрованого тексту та оригінального ключа розшифровки, розшифрованого алгоритмом ID-TRE. Оскільки безпека алгоритму ID-TRE добре проаналізована в Chan & Blake [4], деталі процесів перевірки опущені в цій роботі через обмеження простору. При грубій атаці зловмисник намагається розшифрувати повний зашифрований текст усіма можливими ключами дешифрування. Vanish і SafeVanish вразливі до цього типу атак, оскільки повний зашифрований текст інкапсульований у VDO. Однак SSDD, ISS та FullPP можуть викликати алгоритм DataExtract, щоб зробити зашифрований текст неповним. Вони здатні ефективно протистояти атаці грубої сили. Передумова традиційної атаки криптоаналізу полягає в тому, що зловмисник може отримати повний зашифрований текст. Подібно до вищезазначеного аналізу, Vanish та SafeVanish також вразливі до цього типу атак. SSDD, ISS та FullPP можуть протистояти цьому типу атак. Крім того, FullPP шифрує ключ дешифрування за допомогою алгоритму ID-TRE, тому зловмисник не може отримати оригінальний

ключ дешифрування. Тому запропонована нами схема FullPP є більш безпечною, ніж SSDD, у протистоянні цим двом типам атак.

Як описано у схемі Vanish [27], специфічні властивості мережі DHT викликають значні труднощі у зловмисників, які хочуть зібрати достатню кількість ключових акцій до закінчення терміну дії. Однак Волчок та співавт. [34] та Zeng та співавт. [36] зробив детальні експерименти та аналіз атак в мережі DHT, і результати показали, що Vanish вразливий до атак Sybil, які постійно сканують мережу DHT і зберігають кожну збережену спільну спільну інформацію під час періоду авторизації. Вони можуть ефективно відновити ключі для понад 99% повідомлень Vanish за доступною загальною вартістю.

Вищезазначений аналіз показує, що Vanish, SSDD, SafeVanish та ISS розглядають лише перші два типи атак. Далі ми розглянемо найсильніший тип атаки на FullPP: зловмисник може атакувати як хмарні сервери, так і мережу DHT у будь-який час після збереження SDO на серверах. Давайте розглянемо випадок, коли деякий час зловмисник запускає атаки Sybil, щоб зібрати достатньо трансформованих акцій з мережі DHT, і одночасно запускає традиційний криптоаналіз або грубу силу для отримання SDO з хмарних серверів. Через відсутність алгоритму CSharesTransform та ключа розшифровки ID-TRE зловмисник не може отримати спільні файли зашифрованого тексту із трансформованих спільних ресурсів, щоб відокремити вихідний ключ розшифровки та частину зашифрованого тексту, а також отримати повний зашифрований текст для підтримки традиційний криптоаналіз. Нарешті, зловмиснику не вдається отримати оригінальні конфіденційні дані користувача. Таким чином, FullPP здатний протистояти цим двом типам атак одночасно для досягнення загальної безпеки системи. Вищезазначений аналіз показує, що FullPP здатний протистояти вищезазначеним трьом типам атак без ідеального припущення: "Немає атак на SDO до його закінчення". Ще однією відмінною рисою FullPP є підтримка функції заздалегідь визначеного часу випуску, якої немає в інших схемах.

4 ОПИС ЕКСПЕРЕМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

Для даної теми були обрані декілька основних архітектур систем для безпечного збереження та передачі секретних даних.

У схемі FullPP спільні файли зашифрованого тексту, включаючи зашифрований текст ключа та частину зашифрованого тексту конфіденційних даних, розподіляються в мережу DHT після перетворення і будуть самостійно видалені через унікальну властивість мережі DHT. Ключ розшифровки та оригінальний зашифрований текст конфіденційних даних не підлягають відновленню. Отже, є три способи порушити нашу схему. Перший - використання традиційних атак для порушення SDO на хмарних серверах після закінчення терміну дії, використання атак Sybil для компрометації мереж DHT протягом періоду авторизації та використання одночасних атак для порушення системи в будь-який час. Перші два також трапляються в існуючих схемах самознищення, таких як Vanish [27] та SSDD [1]. Третій - використовувати як традиційний криптоаналіз, так і грубу атаку на хмарні сервери та атаки Sybil на мережу DHT.

Безпеку архітектур можна проаналізувати з наступних трьох питань: 1 Якщо алгоритм шифрування може протистояти традиційним атакам? 2 Якщо мережа DHT може протистояти атакам Sybil? 3 Якщо система може протистояти як традиційним атакам, так і атакам Sybil?

Порівняння систем в сфері безпеки приведено у таблиці 2.

Таблиця 2

Характеристика безпеки	Vanish	SafeVanish	SSDD	FullPP	Single Secure
Жодних атак на VDO до його закінчення	+	+	+	-	-

Продовження таблиці 2

Характеристика безпеки	Vanish	SafeVanish	SSDD	FullPP	Single Secure
Незалежний зашифрований текст чи ні?	-	-	+	-	-
Чи зашифрований ключ чи ні?	-	+	-	+	+
Чи видаляється зашифрований текст чи ні?	-	-	+	+	-
Стійкість до традиційного криптоаналізу?	-	-	+	+	+
Стійкість до атаки brute-force	-	-	+	+	+
Стійкість до хопинг атаки	-	+	+	+	+
Стійкість до снифинг атаки	-	+	-	+	+
Загальна безпека системи?	-	-	-	+	+
Підтримує бажаний час випуску?	-	-	-	+	-
Розрахована на більш ніж одного користувача	+	+	+	+	-

Аналізуємо ефективність запропонованої схеми FullPP і порівнюємо її із схемою SSDD [1] та Single Secure, що представляють сучасний стан. Ефективність

трьох схем в основному проявляється в обчислювальних накладних витратах порівняно з накладними витратами на зв'язок, тому ми зосереджуємося на аналізі та вимірюванні обчислювальних накладних витрат.

Обчислювальні накладні витрати - це час роботи для шифрування конфіденційних даних, шифрування ключа дешифрування, генерації та трансформації спільних ресурсів шифротексту. Обчислювальні накладні витрати на SSDD та FullPP можна порівняти у дві фази: фаза інкапсуляції та фаза декапсуляції.

Обчислювальні накладні витрати фази інкапсуляції проявляються в наступних п’яти основних операціях: шифрування інформації, асоціація оригінального зашифрованого тексту, шифрування ключа розшифровки, генерація зашифрованих шматків тексту, трансформація файлів зашифрованого тексту.

Обчислювальні накладні витрати цієї фази декапсуляції порівнюються в наступних операціях: детрансформація файлів зашифрованого тексту, розшифровка зашифрованого тексту ключа, від’єднання шифротексту, розшифровка вихідного зашифрованого тексту конфіденційних даних.

З вищенаведеного аналізу різниці в обчислювальних накладних витратах між SSDD та FullPP полягають у тому, що FullPP потрібно запустити ще два алгоритми в обидві фази, але це зменшує загальні обчислювальні накладні витрати, уникаючи алгоритмів асоціації та роз’єднання в обидві фази. Ми вимірюємо ці обчислювальні накладні витрати у наступному підрозділі.

Вимірювання обчислювальні накладні витрати на схему FullPP, використовуючи бібліотеку CryptoJS. Експеримент реалізований на тестовому комп’ютері з наступними конфігураціями: Intel Core i7, ОЗУ 16Гб, CPU 1.5Ггц.

Вимірюються обчислювальні накладні витрати FullPP з точки зору часу виконання основних криптографічних операцій. Всі обчислення виконуються 1000 разів, і вибирається середнє значення. Графік вимірювання часу фази інкапсуляції навединий на рисунку.

FullPP додає до обчислювальних накладних витрат при шифруванні ключа дешифрування за допомогою алгоритму ID-TRE та трансформації спільних

ресурсів зашифрованого тексту. На відміну від цього, як ISS, так і SSDD збільшують обчислювальні накладні витрати при асоціюванні вихідного зашифрованого тексту, а ISS збільшує обчислювальні накладні витрати при шифруванні ключа дешифрування за допомогою алгоритму IBE порівняно з SSDD. Результат вимірювання вказує на те, що обчислювальні накладні витрати FullPP нижчі, ніж у SSDD та ISS. Таким чином, FullPP є більш ефективним і результативним, ніж існуючі схеми.

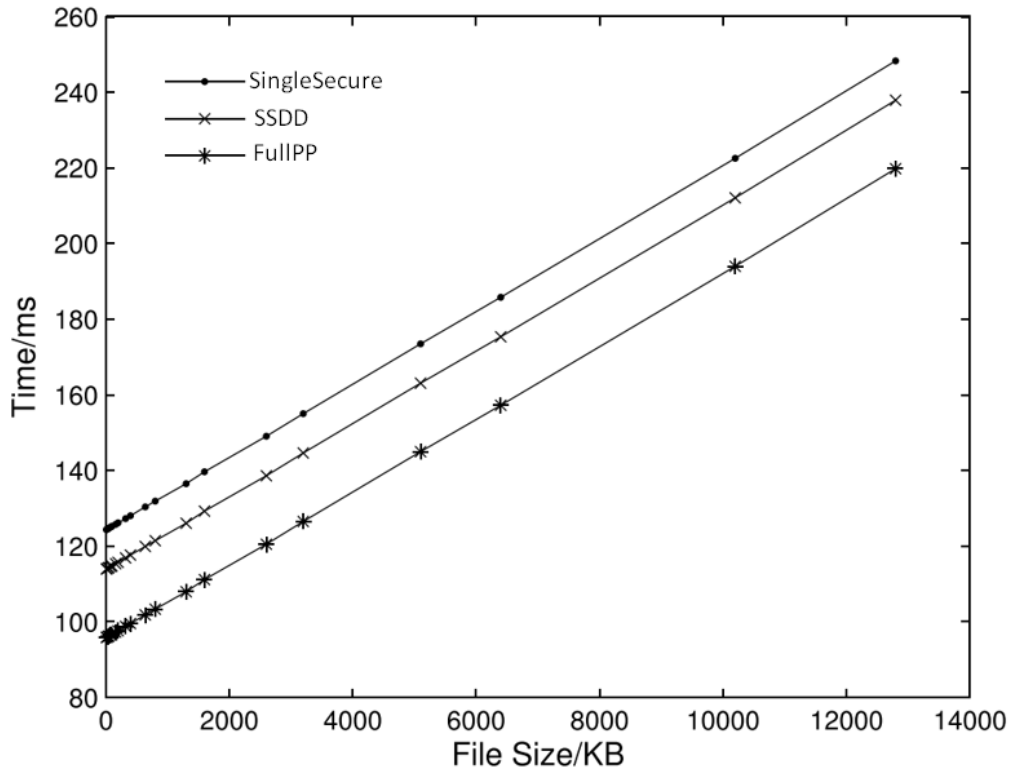

Рисунок 14 - Обчислювальні накладні витрати Full у порівнянні з SSDD та Single Secure

ВИСНОВКИ

У результаті роботи було проведений докладний аналіз предметної області, проведено дослідження методів та засобів сбереження та обробки конфиденційної інформації у хмарі.

Був проведений аналіз різних методів та підходів для сбереження та обміну конфіденційних документів у хмарі. Були представлені характеристики, яким повинна відповідати система аби бути максимально беспечною. Був проведений аналіз взаємодії всії частин системи та поділено відповідальність та фанкції між блоками системи.

В результаті було обрано мікросервісну архітектуру моделі для збереження данних для одного користувача, що гарно підлягає під концепцію моделі – всі складові системи незалежні один від одного, мають свої певні функції та має різний рівень досупу та захищеності. Були розгляуті концепції зберігання секретних даних та створена модель системи для зберігання конфиденційних документів у хмарі.

Для передачі конфиденційних даних була обрана модель Single Secure та розглянуті основні існуючі моделі, що мають надійний захист даний, який предумовлений зберігання даних на певний проміжок часу, після якого дані видаляються. Важнивою частиною системи є передача ключів шифрування від одного користувача до іншого.

Також були надані рекомендації щодо збереження паролів або ключів, їх оновлення та збереження. Була надана схема взяємодії між клієнтом та сервером, включаючи захищені канали передачі даних, а саме подвійне шифрування даних над HTTPS або WSS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Wang G, Yue F, Liu Q (2013) A secure self-destructing scheme for electronic data. J Comput Syst Sci 79(2):279–290 (дата звернення: 28.03.2021)

2. Xiong J, Yao Z, Ma J, Liu X, Li Q (2013) A secure document selfdestruction scheme: an abe approach. In: Proceedings of the 15th IEEE international conference on high performance computing and communications. IEEE, pp 59–64 (дата звернення: 24.03.2021).

3. Josep Domingo-Ferrer, Oriol Farr`as, Jordi Ribes-Gonz´alez and David S´anchez Universitat Rovira i Virgili, UNESCO Chair in Data Privacy, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Pa¨ısos Catalans 26, E-43007 Tarragona, Catalonia / Elsevier Preprint. - Privacy-preserving Cloud Computing on Sensitive Data: a Survey of Methods, Products and Challenges, 2019. (дата звернення: 16.04.2021)

4. International Journal of Grid and High Performance Computing, 5(4), 97-112, October-December 2013 URL: https://www.martinhenze.de/wp-content/papercite-data/pdf/hhm+13.pdf (дата звернення: 28.03.2021)

5. Liang K, Huang Q, Schlegel R, Wong DS, Tang C (2013) A conditional proxy broadcast re-encryption scheme supporting timedrelease. In: Information security practice and experience. Springer, pp 132–146(дата звернення: 29.03.2021)

6. Tang Y, Lee PPC, Lui JCS, Perlman R (2012) Secure overlay cloud storage with access control and assured deletion. IEEE Trans Dependable Secure Comput 9(6):903–916(дата звернення: 10.03.2021)

7. Zeng L, Chen S, Wei Q, Feng D (2013) Sedas: a self-destructing data system based on active storage framework. IEEE Trans Magn 49(6):2548–2554 (дата звернення: 15.04.2021)

8. Xiong J, Yao Z, Ma J, Liu X, Li Q (2013) A secure document selfdestruction scheme: an abe approach. In: Proceedings of the 15th IEEE international conference on

high performance computing and communications. IEEE, pp 59–64(дата звернення: 04.04.2021)

9. Wang G, Liu Q, Wu J, Guo M (2011) Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. Comput Secur 30(5):320–331 (дата звернення: 01.05.2021)

10. Zeng L, Shi Z, Xu S, Feng D (2010) Safevanish: an improved data self-destruction for protecting data privacy. In: Proceedings of the second international conference on cloud computing technology and science. IEEE, pp 521–528 (дата звернення: 17.04.2021)

11. Gheorghe G, Lo Cigno R, Montresor A (2011) Security and privacy issues in p2p streaming systems: a survey. Peer-to-Peer Netw Appl 4(2):75–91 (дата звернення: 20.04.2021).

12. Kachko O., N. Bilous , Semerkov V. Research on methods for secure web applications development Information Technolo-gies in Innovation Busi-ness (ITIB), 7-9 October, 2015, Kharkiv, Ukraine Proceedings of ITIB, p.26-27, ISBN 978-966-659-214-2.

13. . Sergiy Zagorodnyuk, Bohdan Sus, Oleksandr Bauzha, Ilona Revenchuk. Information Security of Users Rights Assignment via the Software Solutions Based on LDAP. Problem of Infocommunications. Science and Technolpgy (PIC S&T′2020).Kharkiv, Ukraine- 6-9 October 2020.

14. . Andrey Arsenov, Igor Ruban, Kyrylo Smelyakov, Anastasiya Chupryna Evolution of Convolutional Neural Network Architecture in Image Classification Problems Selected Papers of the XVIII International Scientific and Practical Conference on IT and Security (ITS 2018).–CEUR Workshop Processing 11 Andrey Arsenov, Igor Ruban, Kyrylo Smelyakov, Anastasiya Chupryna.