МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ «ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»

Галузь знань 12 «Інформаційні технології»

Спеціальність 125 «Кібербезпека»

Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему **«Організація гомоморфного шифрування в хмарних обчисленнях»**

здобувача Чикильова Арсенія Віталійовича _____

Науковий керівник: к.е.н., професор Бєгун Анатолій Володимирович

_____

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти (ЕК)

Завідувач кафедри: д.ф.-м.н., професор Джалладова І.А.

_____

Київ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВАДИМА ГЕТЬМАНА

ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»

Галузь знань 12 «Інформаційні технології»

Спеціальність 125 «Кібербезпека»

ПОГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проєктної групи (гарант) Освітньо-професійної програми «Кібербезпека»	Завідувач кафедри системного аналізу та кібербезпеки
доцент Г.В. Мамонова	д.ф.-м.н., проф. І.А. Джалладова
_____ 2024р.	_____ 2024р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Чикильова Арсенія Віталійовича

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи

на тему «Організація гомоморфного шифрування в хмарних обчисленнях»

Тему затверджено наказом ректора Університету від «___»_____20__р.№__

Кваліфікаційна бакалаврська робота виконується на матеріалах літературних та інформаційних джерел, статей, моделей загроз, моделей побудови мережевої архітектури.

План кваліфікаційної бакалаврської роботи

Розділ 1	Теоретичні принципи захисту інформації та хмарного середовища
Розділ 2	Аналіз гомоморфного шифрування як засобу забезпечення конфіденційності інформації
Розділ 3	Впровадження програмної реалізації гомоморного шифрування в хмарних обчисленнях

Об'єкт дослідження:	Процес перетворення даних за допомогою гомоморфного шифрування
Предмет дослідження:	Методи захисту даних в хмарних обчисленнях із впровадженням гомоморфного шифрування
Мета кваліфікаційної бакалаврської роботи:	Програмна реалізація гомоморфного шифрування для забезпечення безпеки в хмарному середовищі

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі 1 «Теоретичні принципи захисту інформації та хмарного середовища»: дослідити особливості провадження інформаційної безпеки в організаціях; проаналізувати хмарні обчислення та їх використання компаніями; визначити види хмарних обчислень та типи моделей розгортання й надання послуг; розглянути загрози хмарного середовища та методи захисту його даних.

У розділі 2 «Аналіз гомоморфного шифрування як засобу забезпечення конфіденційності інформації»: проаналізувати стратегії використання шифрування; визначити види шифрування, їх особливостей; розглянути можливості гомоморфного шифрування; дослідити види гомомофрного шифрування; проаналізувати алгоритм шифрування Пайє.

У розділі 3 «Впровадження програмної реалізації гомоморного шифрування в хмарних обчисленнях»: розробити програмний застосунок для гомоморфного шифрування даних; реалізувати інфраструктуру в хмарному середовищі для обчислення гомоморфних криптосистем.

Завдання підготував
науковий керівник _____ А.В. Бєгун
«____»_____2024 р.

Завдання
одержав здобувач _____ А.В. Чикильов
«____»_____2024 р.

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Організація гомоморфного шифрування в хмарних обчисленнях» складається із списку умовних позначень та скорочень, зі вступу, 3 розділів, загальних висновків, списку використаних джерел, 5 додатків. Загальний обсяг роботи – 66 сторінки. Робота містить 24 рисунків. Список використаних джерел включає 50 посилання.

Об'єкт дослідження – процес перетворення даних за допомогою гомоморфного шифрування.

Предмет дослідження – методи захисту даних в хмарних обчисленнях із впровадженням гомоморфного шифрування.

Мета кваліфікаційної роботи – програмна реалізація гомоморфного шифрування для забезпечення безпеки в хмарному середовищі.

Метод дослідження – аналіз системи захисту даних в хмарному середовищі.

Практична цінність – реалізовано застосунок, що обчислює зашифровані дані.

Ключові слова: захист інформації, конфіденційність, хмарне середовище, асиметричні криптосистеми, гомоморфне шифрування, алгоритм Пайє.

Відгук

на дипломну роботу освітньо-кваліфікаційного рівня бакалавр зі спеціальності «Кібербезпека» КНЕУ імені Вадима Гетьмана на тему **«*Організація гомоморфного шифрування в хмарних обчисленях*"**

Чикильова Арсенія Віталійовича

Актуальність теми. Кількість рішень про інтеграцію між хмарними середовищами та інформаційними системами компаній постійно зростає. А це свідчить про збільшення атак на інфраструктуру з підвищеним ризиком та конфіденційну інформацію, що зберігається та обробляється в хмарі. Така динаміка робить великий акцент на негайному впровадженні заходів і технологій, які б забезпечили конфіденційність обробки даних у хмарних середовищах.

Серед найбільш ефективних методів реалізації захисту даних є використання гомоморфного шифрування. Впровадження гомоморфного шифрування дозволяє зберігати конфіденційність даних у віддалених хмарних обчислювальних середовищах. Головна особливість цього способу перетворення даних полягає в тому, що він надає можливість виконувати математичні операції над зашифрованими даними та отримати такі результати обчислень, які після розшифрування, будуть ідентичними з результатами виконання таких же операцій над двома відкритими текстами.

Повнота розкриття теми в дипломній роботі відповідає поставленим меті, задачам, обсягу і змісту. Робота складається з трьох розділів, кожний з яких має свою специфіку і особливості.

Теоретичний рівень запропонованої роботи свідчить про вміння здобувача опрацьовувати та аналізувати наукові видання, нормативні документи, технічну документацію, проводити аналіз, використовувати методи максимального правдоподобія та синтезу.

Практична значущість дипломної роботи полягає у реалізації застосунків, які зашифровують повідомлення та обчислюють отримані криптосистеми, згідно з конфігурацією користувача. Впровадження таких застосунків продемонструвало їх

ефективність у забезпеченні конфіденційності даних під час обробки у хмарній платформі.

Самостійність виконання роботи визначається здатністю здобувача самостійно працювати з науково-технічною літературою, послідовно знаходити і обробляти статистичну, публічну і наукову інформацію, будувати вектор дослідження. Робота перевірена на плагіат і складає коефіцієнт схожості - 1,5%.

Якість оформлення, загальна та спеціальна грамотність відповідає вимогам кафедри до написання пояснювальної записки дипломної роботи бакалавра, написана якісно і в доступній для ознайомлення формі.

Наведена дипломна робота та її результати свідчать про важливість і актуальність гомоморфних алгоритмів шифрування в хмарних обчисленнях, впливає на створення нових, більш ефективних та безпечних від несанкціонованого доступу хмарних моделей, які забезпечать конфіденційність, цілісність та доступність інформаційних активів Цей факт дозволяє зробити висновок про те, що дипломна робота Чикильова А. В. має теоретичне і практичне значення, відповідає вимогам кваліфікаційної роботи, виконана на відповідному рівні і заслуговує позитивної оцінки, а здобувачу може бути присвоєно освітній рівень бакалавра за спеціальністю «Кібербезпека».

Науковий керівник
професор кафедри системного
аналізу та кібербезпеки, к.е.н., професор А.В. Бєгун

РЕЦЕНЗІЯ

на кваліфікаційну бакалаврську роботу

студента _____Чикильова А.В._____гр. ІК-401

(прізвище, ініціали)

На тему:_____Організація_гомоморфного_шифрування_в_хмарних обчисленнях_____
_____

Актуальність теми: використання хмарного обчислення забезпечує переваги у підвищенні доступності сервісів та легкої масштабованості ресурсів. Однак існують недоліки, що пов'язані з тим, що зловмисники, попередньо отримавши доступ до хмарного середовища, можуть легко скористатися персональними даними, що розміщені там. Для усунення цієї загрози, впроваджуються системи гомоморфного шифрування, що дозволяють приховати вміст інформації і виконувати обчислення над шифротекстами, без їх розшифрування, та безпечно зберігати результати на стороні хмарного середовища. При правильній організації цих систем, забезпечується приватність та конфіденційність даних клієнта хмарних послуг від несанкціонованого доступу._____

Наукова новизна: полягає у детальному аналізі та дослідженні особливостей впровадження сучасних алгоритмів гомоморфного шифрування в хмарних обчисленнях. _____

Якість проведеного аналізу: кваліфікаційна робота демонструє досить високий ступінь знань студента з обраної теми. Усі розділи роботи структуровані та логічно пов'язані._____

Уміння користуватися літературними джерелами: було проаналізовано значний обсяг інформаційних джерел, статей та досліджень, що показало високий рівень студента у вмінні аналізувати та структурувати зібрану інформацію для використання у своїй роботі. _____

Практична цінність висновків і рекомендацій: полягає в розробці рекомендації по організації систем гомоморфного шифрування даних. Результати роботи можуть бути впроваджені організаціями, що виконують обчислення та зберігають чутливі дані у хмарному середовищі, для забезпечення їх конфіденційності._____

Переваги та недоліки: у кваліфікаційній роботі послідовно та детально подано матеріал, надано рекомендації щодо покращення захисту даних під час хмарного обчислення. Недоліком є те, що недостатньо уваги приділено можливим обмеженням реалізації систем гомоморфного шифрування в реальних умовах. Зазначений недолік не суттєво впливає на загальний рівень роботи._____

Загальний висновок і оцінка роботи: в процесі виконання дипломної роботи студент продемострував себе компетентним фахівцем у обраній темі. Дана робота є актуальною, адже кількість компаній, що використовують хмарне середовище для обробки та зберігання даних, постійно зростає. Вважаю, що кваліфікаційна робота заслуговує на оцінку “відмінно”.

Рецензент

___ТОВ “ІТ СПЕЦІАЛІСТ”_____
(Назва організації)

Аудитор з інформаційної безпеки
(посада) _____ ___Цапро Д.О._____
(підпис) (ініціали, прізвище)

Печатка “___”_____2024р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕННЬ 4
ВСТУП 5
РОЗДІЛ 1 ТЕОРЕТИЧНІ ПРИНЦИПИ ЗАХИСТУ ІНФОРМАЦІЇ ТА ХМАРНОГО СЕРЕДОВИЩА 7
1.1 Імплементація інформаційної безпеки в організаціях 7
1.2 Хмарні обчислення в комерційній діяльності 10
1.3 Види хмарних обчислень 12
1.3.1 Типи моделей розгортання 13
1.3.2 Типи моделей надання послуг 16
1.4 Аналіз загроз хмарного середовища 22
1.5 Методи захисту даних в хмарному середовищі 26
Висновки до розділу 1 28
РОЗДІЛ 2 АНАЛІЗ ГОМОМОРФНОГО ШИФРУВАННЯ ЯК ЗАСОБУ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ІНФОРМАЦІЇ 30
2.1 Огляд стратегій використання шифрування 30
2.1.1 Види шифрування та їх особливості 32
2.2 Огляд особливостей гомоморфного шифрування 34
2.2.1 Види гомомофрного шифрування 36
2.3 Алгоритм шифрування Пайє 39
Висновки до розділу 2 41
РОЗДІЛ 3 ВПРОВАДЖЕННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ГОМОМОРНОГО ШИФРУВАННЯ В ХМАРНИХ ОБЧИСЛЕННЯХ 43
3.1 Структуризація алгоритмічних потреб для реалізації зашифрованих обчислень 43
3.2 Реалізація першого програмного застосунку для клієнта 44
3.3 Впровадження другого застосунку в хмарному сервісі Azure 48

3.3.1 Підготовка хмарного середовища до реалізацій застосунку 49
3.3.2 Огляд програмної реалізації застосунку для обчислень 52
3.3.3 Аналіз працездатності другого застосунку та його кооперації з першим 56
Висновки до розділу 3 58
ВИСНОВОК 60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62
ДОДАТКИ 67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕННЬ

ІС – інформаційна система;

IaaS – інфраструктура як сервіс

PaaS – платформа як сервіс

SaaS – програмне забезпечення як ссервіс

SSH – протокол безпечного доступу до сервера

SSL – протокол захищеного зв’язку

TSL – протокол безпеки передачі даних

RDP – протокол віддаленого доступу до робочого столу

NGFW – фаєрвол наступного покоління

CSPM – інструменти для керування безпекою в хмарі

FHE - повне гомоморфне шифрування

SHE – дещо гомоморфне шифрування

BHE – шифрування з можливістю початкового завантаження

DNS – система доменних імен

ВСТУП

Внаслідок повномасштабного вторгнення, виникла серйозна загроза безпеці даних для компаній, що в себе включає як можливість фізичного пошкодження чи знищення апаратного забезпечення через ворожі ракетні атаки, так і можливість компрементації інформаційних систем із-за кібератак. Крім того, існує нестабільність у постачанні електроенергії, що створює додаткові ризики, реалізація яких значно ускладнює постійну доступність користувачів до інфраструктури та ресурсів.

У відповідь на цю ситуацію підприємства, фінансові установи, урядові організації та інші суб'єкти, що вразливі до можливості імплементації загроз втрати своїх серверів для обробки даних, розпочали активно переносити свою інфраструктуру або створювати резервні копії у хмарних середовищах, для забезпечення неперервності доступу до їхніх даних та послуг, навіть у випадку можливих технологічних і фізичних загроз, що дозволяє організаціям забезпечити стійкість і надійність своєї інформаційної інфраструктури в умовах несприятливого зовнішнього впливу. Таким чином, згідно статті організації Nextgov, з початку війни до хмарного середовища мігрувало понад 161 державних реєстрів та 356 відомств, загальним розміром яких, є близько 15 петабайтів даних, які є критично важливими для функціонування держави [1].

Оскільки кількість рішень про інтеграцію між хмарними середовищами та інформаційними системами компаній зростають, це також означає збільшення атак на таку інфраструктуру з підвищеним ризиком для безпеки та конфіденційності інформації, що зберігається та обробляється в хмарі. Відповідно до дослідження компанії CrowdStrike, з 2022 по 2023 роки кількість вторгнень у хмарне середовище зросли аж на 75% [2, с.17]. Така динаміка робить великий акцент на негайному впровадженні заходів і технологій, які б забезпечили конфіденційність обробки даних у хмарних середовищах; Серед найбільш ефективних методів реалізації захисту даних є використання гомоморфного шифрування.

Впровадження гомоморфного шифрування дозволяє зберігати конфіденційність даних у віддалених хмарних обчислювальних середовищах. Головною особливістю цього способу перетворення даних полягає в тому, що він надає можливість виконувати математичні операції над зашифрованими даними та отримати такі результати обчислень, що після розшифрування, будуть ідентичними з результатами виконання таких же операцій над двома відкритими текстами. Для досягнення цього, необхідно, щоб існував зв'язок між відкритими та зашифрованими текстами, проте він повинен залишатися непомітним для будь-якого спостерігача [3].

Розвиток у сфері впровадження гомоморфних алгоритмів шифрування в хмарних обчисленнях вплине на створення нових, більш ефективних та безпечних від несанкціонованого доступу хмарних моделей у майбутньому, які забезпечать впевненість потенційних користувачів у безпеці їхніх даних під час обробкиа зберігання конфіденційної інформації без ризику її втрати.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ПРИНЦИПИ ЗАХИСТУ ІНФОРМАЦІЇ ТА ХМАРНОГО СЕРЕДОВИЩА

1.1 Імплементація інформаційної безпеки в організаціях

Зі зростанням швидкості розвитку технологій обробки даних, інформація стала цінним активом для бізнесу. Як наслідок, сучасні комерційні організації не можуть знехтувати впровадженням інформаційної безпеки, адже, зважаючи на роль даних у сучасному бізнесі, складно уявити компанію, що не має баз даних та не оперує інформаційними потоками.

Згідно із законом України “Про Засади Інформаційної безпеки”, інформаційна безпека це впровадження засобів, що захищають важливі інтереси людини, громадянина, суспільства і держави та забезпечують і запобігають шкоді через неповноту, несвоєчасність та недостовірність інформації, порушення цілісності та доступності інформації, незаконний обіг інформації з обмеженим доступом, а також через негативний вплив інформації на психологічний стан людини та умисне спричинення негативних наслідків застосування інформаційних технологій [4].

Для забезпечення інформаційної безпеки для ІС потрібно забезпечити 3 основні складові, а саме [5]:

Конфіденційність полягає в забезпеченні інформації від несанкціонованого доступу, тобто, лише ті особи, що пройшли авторизацію - певну послідовність дій для підтвердження того, що ця особа може мати доступ до цієї інформації. Для забезпечення конфіденційності, впроваджують алгоритми шифрування, контролю доступу та інших заходів безпеки.

Цілісність визначається, як захищеність інформації від несанціонованого втручання, руйнування або знищення. Для забезпечення цілісності впроваджують методи перевірки суцільності даних, стеження за змінами та запобігання втраті або пошкодженню інформації та інші.

Доступність полягає в забезпеченні інформації від несанкціонованого блокування, тобто забезпечення того, що інформація доступна для авторизованих користувачів у відповідний час і місце. Для забезпечення доступності використовують заходи з резервного копіювання, мережевої доступності і технічного обслуговування систем.

Для забезпечення цих основних вимог, впроваджуються методи та механізми захисту інформації, що охоплюють технологічні, організаційні та адміністративні заходи, спрямовані на забезпечення належного рівня безпеки та управління даними в умовах зростаючих кіберзагроз. На думку автора, найбільш поширеними методами є:

- впровадження фізичної безпеки, що забезпечується використанням заходів захисту фізичних пристроїв та засобів, що містять конфіденційну інформацію [6];
- покращення безпеки мереж, що включає використання засобів для блокування доступу, систем виявлення вторгнень для моніторингу та запобігання кібератакам та інші [7];
- впровадження розподіленого доступу користувачів до ресурсів, обмежує доступ до інформації лише авторизованим користувачам, що гарантує, що доступ до даних можуть мати лише ті, хто має належний дозвіл, зменшуючи ризик витоку даних і несанкціонованого доступу;
- шифрування, що перетворює інформацію в закодовану форму, що робить її нечитабельною для будь-кого без відповідного ключа розшифровки, тобто тільки авторизований користувач, який володіє ключем дешифрування, може розшифрувати та переглянути цю інформацію [8];
- навчання персоналу, що допомагає зменшити ризики порушення безпеки, збільшити загальний рівень захищеності організації та включає надання співробітникам необхідних знань, навичок для

мінімізації ризиків, пов'язаних із обробкою та зберіганням конфіденційної інформації [9];

- резервне копіювання даних, яке створюючи копії даних і зберігаючи їх у безпечному місці, забезпечує збереження даних у разі їх втрати або пошкодження, після чого їх буде легко відновити [10].

Однак впровадження повного комплексу заходів для забезпечення цілісності, конфіденційності та доступності може виявитися фінансово неефективним для компаній через високі витрати на введення систем захисту в експлуатацію, їх обслуговування та модернізацію. Тому підприємства у сфері комерції завжди ретельно виважують і зберігають рівновагу між впровадженням захисних систем і їх вартістю. Вони стежать за тим, щоб обрати такі рішення забезпечення безпеки, які відповідають їх бюджетним можливостям і водночас надають необхідний рівень захисту інформації.

Комерційні організації завжди оцінюють витрати на впровадження заходів захисту та забезпечують оптимальне співвідношення між ефективністю захисту інформації і вартістю цих заходів [11]. Вони прагнуть забезпечити необхідний рівень безпеки, уникати зайвих витрат і максимізувати ефективність використання ресурсів на забезпечення інформаційної безпеки. Тому, комерційні організації завжди беруть до уваги бюджетні обмеження і вирішують, які заходи захисту інформації є найбільш ефективними і доступними з погляду витрат.

Одним із варіантів вирішення цієї проблеми є перехід даних організацій у хмарне середовище, що може забезпечити не лише ефективний захист інформації, але й значно знизити витрати на її зберігання та обробку, адже частина ризиків буде перекладатися на постачальника хмарних послуг [12]. Значною перевагою даного способу є те, що хмарні сервіси надають можливість адаптувати їх середовище до системи замовника та надати ті сервіси та механізми захисту, які він потребує.

1.2 Хмарні обчислення в комерційній діяльності

Хмарне обчислення – використання віддалених послуг для обчислення даних та ресурсів, що доступні за певну плату. Тобто за своєю суттю, хмарні обчислення є моделлю надання масштабованих та гнучких засобів для проведення обчислень через Інтернет, що дозволяє користувачам отримувати доступ до ресурсів і додатків без необхідності розгалуженої локальної інфраструктури та апаратного забезпечення [13].

Завдяки використанню хмарних обчисленнь, користувачі можуть отримувати послуги на запит, такі як доступ до обчислювальних потужностей, сховищ та програмних додатків, не потребуючи керування фізичними серверами чи центрами обробки даних. Ця масштабованість дозволяє підприємствам швидко адаптуватися до мінливих вимог, незалежно від того, чи йдеться про збільшення масштабів у періоди пікових навантажень або зменшення масштабів для зменшення витрат у періоди непікового навантаження, тим самим оптимізуючи використання ресурсів і ефективність [14].

Хмарне середовище, зазвичай, створене як мережа віддалених центрів обробки даних, серверів і систем зберігання даних, якими володіють і управляють постачальники хмарних послуг, їх ще називають провайдерами. Вони несуть відповідальність за забезпечення обсягу зберігання, безпеки та обчислювальної потужності, необхідних для підтримки даних, які користувачі надсилають у хмару [15].

Клієнти можуть використовувати хмарові послуги для великої кількості різних цілей, отримуючи широкі можливості для вирішення різноманітних завдань та задач:

- Використовувати хмару для ефективного зберігання різних видів інформації в віддалених сховищах даних хмари, що дозволяє їм отримати доступ до даних з будь-якого місця та пристрою.

- Обробляти великі обсяги даних, використовуючи віртуалізовані ресурси, що спрощує масштабування обчислювальних потужностей відповідно до конкретних завдань та оптимізує продуктивність [16]. Тобто в залежності від попиту, користувачі можуть збільшити чи зменшити кількість ресурсів, що їм необхідні, для виконання певних задач, що дозволяє уникати надмірного забезпечення та платити лише за ті ресурси, які вони фактично використовують.
- Використовувати хмарну налаштовану мережу для забезпечення ефективного з'єднання та обміну даними між різними кінцевими девайсами.
- Забезпечувати можливості для проведення аналітичних операцій, що дає змогу клієнтам проаналізувати значення своїх даних і використовувати результат аналізу для прийняття стратегічних управлінських рішень [17].

Одною із переваг використання хмарних обчислень полягає в їхній значній економічній доцільності. Адже клієнти можуть уникнути початкових інвестицій у створення та підтримку власної апаратної та програмної інфраструктури, оплачуючи тільки те, що вони фактично використовують. У довгостроковій перспективі, такий підхід може призвести до істотної економії, особливо для малих підприємств [18].

Також, зазвичай хмарні провайдери інвестують значні ресурси в безпекові заходи для захисту даних клієнтів [19]. Серед таких заходів можуть бути шифрування, контроль доступу та моніторинг. Крім того, хмарні провайдери часто мають кращі засоби для протидії загрозам безпеці, ніж окремі користувачі або малі підприємства.

За бажанням, кожен користувач хмарних послуг може впровадити резервне копіювання і відновлення даних, автоматизовану інтеграцію програмного забезпечення та багато інших. Ці заходи зменшать ризики втрати важливих даних у випадку технічних збоїв чи кібератак.

При реалізації систем резервного копіювання та відновлення даних, створюються копії зрізів системи та наявної інформації, які в разі виникнення помилки, можна швидко відновити, що дозволяє значно знизити час простою ІС та зменшує втрати для бізнесу.

Автоматизована інтеграція аналізує наявне в системі ПЗ, перевіряє чи існують та чи доступні його оновлення і при підтвердженні автономно покращує ПЗ до найновішої версії, для підтримки актуальності ІС для захисту від загроз, якого не було в минулих версіях.

1.3 Види хмарних обчислень

Кожен тип хмарної інфраструктури об'єднує апаратні та програмні спроможності декількох серверів, систем зберігання даних та спільно розподіляє ці ресурси до користувачів через мережу Інтернет, дозволяє виконувати обчислювальні операції у всередині цієї платформи. Кожна хмарна система утворюється з унікального переліку технологій та інструментів, серед яких є: операційна система, платформа керування, інтерфейси програмування застосунків (API) та інші.

Існує кілька основних типів моделей хмарних обчислень, кожен з яких має свої особливості, переваги та недоліки. Розуміння особливостей різних типів моделей хмарних обчислень дозволяє організаціям приймати обґрунтовані рішення щодо впровадження технологій, адже вибір відповідної моделі залежить від потреб конкретної організації, її бізнес-цілей та технічних вимог. Впровадження правильної до потреб компанії хмарної моделі, значно покращює продуктивність імплементації, знижує витрати на інфраструктуру та забезпечує надійний доступ до ресурсів.

Існують два основні типи хмарних обчислень: моделі розгортання та моделі надання послуг. Моделі розгортання визначають, як і де будуть розміщуватися ресурси, тоді як моделі надання послуг описують, які саме послуги та в якому форматі будуть пропонуватися користувачам.

1.3.1 Типи моделей розгортання

Розрізняють три типи моделей розгортання, ними є: публічна, приватна та гібридна хмари.

Публічна хмара надає послуги та інфраструктуру, які керуються стороннім провайдером та доступні через Інтернет, що дозволяє користувачам входити в свої ресурси і програми з будь-якої локації по всьому світу [20]. Хмара повністю віртуалізована, створюючи середовище, де спільні ресурси, у разі потреби, використовуються.

Оскільки ці ресурси надаються через мережу Інтернет, то при впровадженні моделі розгортання публічної хмари, організаціям легше масштабуватись, адже потрібно оплачувати лише за ті хмарні ресурси та сервіси, якими користуєшся, що є величезною перевагою перед локальними серверами. Прикладами публічних хмарних провайдерів є: Microsoft Azure, Google Cloud Platform (GCP) та Amazon Web Services (AWS).

Публічна хмара має й свої недоліки [20]:

- Інформація, перебуваючи на серверах стороннього провайдера, може стати предметом ризику щодо її конфіденційності, оскільки існує можливість її розкриття або компрометації.
- Користувачі мають обмежений контроль над інфраструктурою та ресурсами, які використовуються для запуску їхніх програм, що ускладнює налаштування їх середовища.
- Для використання публічних хмарних сервісів необхідне надійне та стабільне підключення до Інтернету, щоб отримати доступ до ресурсів та програм, що розташовані у хмарі. У разі повільного чи нестабільного Інтернет-з’єднання знижується продуктивність та доступність цих сервісів.

При використанні приватної хмари, надається власне хмарне середовище, призначене для однієї організації, фізичні компоненти якої зберігаються локально

або в центрі обробки даних постачальника. Оскільки ця модель хмари доступна лише для однієї компанії, організація має повний контроль над хмарним сервісом, може налаштовати свою архітектуру, розширити протоколи безпеки та збільшити обчислювальні ресурси у віртуалізованому середовищі за потреби [21].

Організації переважно або на місці підтримують приватну хмарну інфраструктуру, надаючи послуги хмарних обчислень внутрішнім користувачам через внутрішньокорпоративну мережу або укладають контракт зі стороннім постачальником хмарних технологій для розміщення та обслуговування ексклюзивних серверів. Прикладами є: GigaCloud та IBM Cloud Private.

Недоліками приватної хмари є:

- Використання приватних хмар передбачає витрати на спеціалізоване обладнання, програмне забезпечення та мережеву інфраструктуру, придбання та обслуговування яких може бути дорогим, що обмежує використання приватної хмари для малих компаній або організацій з обмеженими фінансовими можливостями.
- Приватні хмари орієнтовані на обслуговування конкретної організації, що може призвести до меншої масштабованості порівняно зі публічними хмарними сервісами.

На додачу до основних моделей, ще існує і багатохмарна, коли використовуються дві чи більше платформи хмарних обчислень для виконання різноманітних завдань [22]. Організації, які не хочуть залежати від одного постачальника хмарних технологій, можуть використовувати ресурси кількох різних постачальників, щоб отримати найкращі переваги від кожної унікальної послуги. Така модель ефективна для розподілу обчислювальних навантажень між кількома хмарними платформами, для оптимізації продуктивності середовища, забезпечення гнучкості, щоб зменшити фінансовий ризик.

Багатохмарна модель використовує більше ніж одного публічного хмарного провайдера. Ця стратегія може знизити необхідність у хмарній міграції, оскільки частина даних може залишатися на серверах компанії. Організація може зберігати

дані користувачів, використовуючи одного постачальника для обслуговування інфраструктури та іншого для обслуговування програмного забезпечення. Багатохмарна модель може включати використання або декількох приватних, або декількох публічних, або гібридної хмар.

Гібридна це модель, що поєднує реалізації приватної та публічної хмар, що дає можливість організаціям використовувати переваги цих моделей. Гібридна хмарна модель дозволяє компаніям зберігати конфіденційні дані всередині та отримувати до них доступ через програми, що працюють у публічній хмарі. Наприклад, для дотримання правил конфіденційності, організація може зберігати конфіденційні дані користувачів у приватній хмарі та виконувати ресурсомісткі обчислення у публічній [22].

В порівнянні з використанням тільки одної хмарної моделі, організації, що впровадили гібридну, мають переваги в більшій гнучкості, масштабованості і ефективності, бо можуть керувати рівнями трафіку, особливо, в періоди його пікового використання. Прикладами гібридних хмар є: DataCore Software, Rackspace, Threat Stack,

Недоліками гібридної моделі є:

- Налаштування та управління гібридними хмарами вимагають інтеграції різних хмарних середовищ, що вимагає спеціалізованих технічних знань та ресурсів, адже треба забезпечити сумісність між різними програмами, службами та різних хмарних середовищ.
- Реалізація та управління гібридними хмарами може займати більше фінансових ресурсів порівняно з публічними чи приватними хмарами окремо, оскільки потребує додаткового обладнання, програмного забезпечення та мережевої інфраструктури.
- Гібридні хмари залежать від якості зв'язку між різними хмарними середовищами, що може призвести до затримок у мережі та проблем з продуктивністю.

Згідно дослідженню організації Flexera, вказано, яка саме модель розгортання є найчастіше вживаною компаніями наприкінці четвертого кварталу 2023 року [23]. Так на рисунку 1.1, видно, що перевагу надають саме багатохмарній моделі, що має аж 89%, з яких 73% є використання гібридних хмар. Таким чином, можна стверджувати, що організацій віддають перевагу комбінованому методу, щоб отримати та використати переваги як приватної, так і публічної хмар.

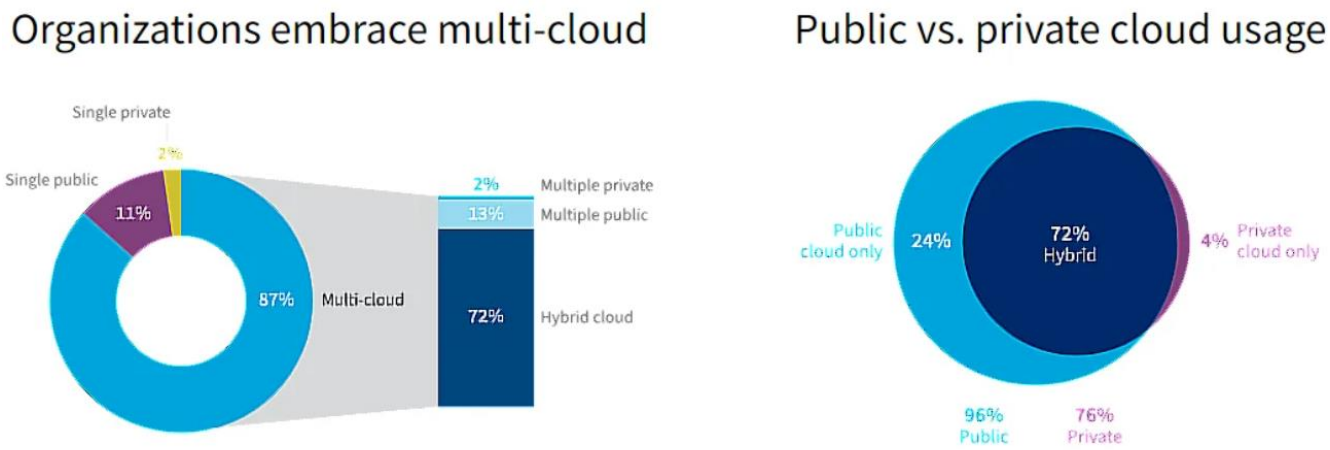


Рисунок 1.1 Відсоткове поширення методі розгортання [23]

Спостерігається тенденція до збільшення кількості використання гібридних хмар через постійні зміни технологій та розширення їх функціональних можливостей, що відповідає потребам сучасних організацій у покращенні продуктивності та безпеки [23].

1.3.2 Типи моделей надання послуг

Хмарні послуги представляють собою інфраструктуру, програмне забезпечення або платформи, які створені провайдерами та доступні для користувачів через мережу Інтернет. Вони є ефективним засобом взаємодії в хмарному середовищі, оскільки допомагають компаніям-клієнтам економити час і гроші, що полегшує процес розробки та контролю над інформаційними технологіями.

Існує 3 основні типи моделей надання послуг (рис.1.2): Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS) та Software-as-a-Service (SaaS).

Кожен з цих засобів сприяє пересиланню даних користувача між зовнішніми клієнтами та системами постачальника хмарних послуг, але їх відмінності полягають у функціональності та характеристиках, які вони пропонують. Крім того, кожна з цих послуг може мати свої унікальні інтеграційні можливості та протоколи для забезпечення сумісності з існуючими системами користувачів [24].

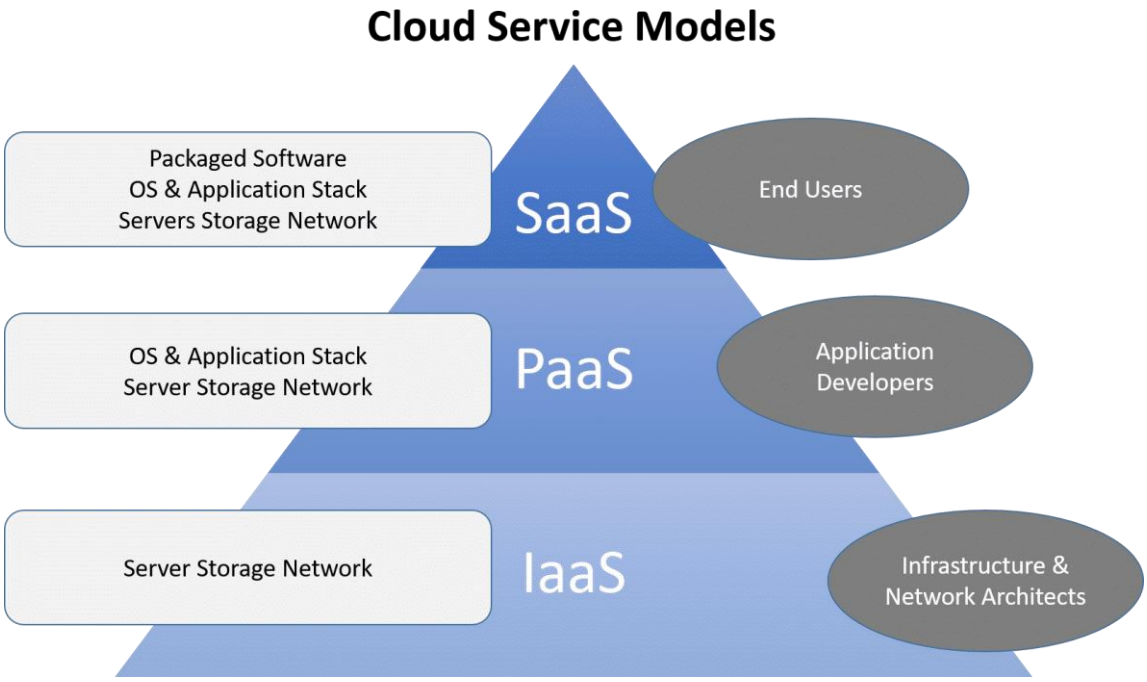


Рисунок 1.2 Моделі надання послуг [24]

IaaS використовує великі масштабовані та автоматизовані обчислювальні ресурси, надаючи повністю самостійне керування інфраструктурою для доступу, моніторингу комп'ютерів, мереж, сховищ та інших послуг, що дозволяє компаніям купувати ресурси за потребою, замість придбання апаратного забезпечення напередодні. Елементи інфраструктури можуть складатися з обчислювальних, мережевих обладнань та сховищ даних, а також інших компонентів та програмного забезпечення [25].

Цю модель часто впроваджують, для перенесення частини або повної локальної до хмарної інфраструктури, обов’зки розгротання якої належать провайдеру хмарних послуг. Переважною більшістю клієнтів, що використовують данний тип моделі надання послуг, є великі компанії, адже цей тип забезпечує їхню потребу в повному контролі над налаштуванням захисту інформації та над всією інфраструктурою і програмним забезпеченням. Також, IaaS обирають через його надійність і стабільність та, у порівнянні з локальними системами, гарантує більший час безперервної роботи, резервування на кожному рівні, вдосконалені безпекові параметри та захист від збоїв, інші заходи, що можуть налаштувати клієнти, а також гнучке масштабування, яке перевищує можливості локальних інфраструктур.

Основою моделі IaaS є те, що клієнт послуг має найбільший рівень відповідальності, з поміж інших моделей надання послуг, він повинен сам впроваджувати механізми захисту свої даних, операційних систем та стеки програмного забезпечення. Тимчасом хмарний провайдер лише відповідає за центри обробки даних, за володіння та керування апаратним і програмним забезпеченням [25].

Популярними провайдерами IaaS є: Microsoft Azure, Amazon Web Services, Rackspace and Google Compute Engine.

Наступною моделлю надання послуг є платформа як сервіс (PaaS), де провайдер надає компанії платформу для розгортання та розробки додатків. PaaS надає розробникам і програмістам спільну хмарну платформу для створення та керування додатками, не потребуючи від них створення та підтримки інфраструктури, що зазвичай пов'язана з цим процесом [26].

Ці ресурси дозволяють розробникам створювати, запускати та керувати всім, від простих невеликих додатків або веб-сайтів до складних і налаштованих систем. PaaS підтримує весь процес розробки, від створення та тестування до постійного обслуговування та оновлення програм. Це допомагає клієнтам уникати витрат, непластичності та важкої роботи, пов'язаних з налаштуванням і підтримкою ресурсів.

PaaS складається з трьох основних компонентів [27]:

- Хмарна інфраструктура, а також віртуальні машини, операційне програмне забезпечення, сховище, мережі та брандмауери.
- Програмне забезпечення для створення, розгортання та керування додатками.
- Графічний інтерфейс користувача, яким команди розробників користуються для виконанн певних завданнь продовж усього життєвого циклу програми.

Порівнянно з IaaS, в PaaS передається більше відповідальності саме до провайдера хмарних послуг. У цій моделі надання послуг, клієнт все ще розгортає і керує своїми програмами та пов’язаними даними, але саме провайдер забезпечує безпеку роботи основної інфраструктури та операційних систем у цілому [27].

Під час використання PaaS, клієнти можуть зосередитися на розробці та вдосконаленні своїх програм, оскільки вони можуть покладатися на провайдера стосовно надійності та безпеки інфраструктури. Цей підхід є ефективним способом використання часу та ресурсів для досягнення цілей розробки програмного забезпечення, не звертаючи уваги на сервери, мережі, безпеку тощо.

Прикладами PaaS є Facebook та пошукова система Google.

Останньою із трьох моделей надання послуг є SaaS, яка надає доступ до програмного забезпечення безпосередньо з браузера чи мобільного додатку. Провайдери розміщують застосунки на власних серверах та базах даних, також вони можуть використовувати сервери іншого хмарного провайдера, також вони відповідають за керування та підтримку платформ, інфраструктур, операційних систем, проміжним програмним забезпеченням, впроваджує потрібні алгоритми захисту, тобто повністю відповідає за налаштування та функціонування цієї системи [27].

SaaS моделі дозволяють клієнтам отримувати доступ до програми після авторизації, зосереджуючи їхню відповідальність виключно на управлінні даними, якими вони оперують в цьому застосунку, та контролі доступу до цих даних.

Але використання SaaS має певні недоліки, бо кожен клієнт не має контролю над його хмарною інфраструктурою, таким чином, можна стверджувати, що він сильно залежить від інфраструктури провайдера хмарних послуг, тому якщо відбудеться збій системи в провайдера, то клієнт не буде могти користуватися цим застосунком.

Досить відомим прикладом використання SaaS є веб-електронна пошта, де потенційний користувач може отримувати і надсилати електронні листи без необхідності керувати компонентами системи чи підтримувати сервери й операційні системи.

Таким чином, головна відмінність між SaaS, PaaS, IaaS є рівень котролю провайдера над системою клієнта. Кожний тип моделі надання послуг забезпечує різний ступінь контролю, тобто різний обсяг відповідальності зі сторони замовника (рис.1.3). IaaS пропонує віртуальні сервери і сховища. PaaS пропонує веб-інструменти для розробки програмного забезпечення. SaaS пропонує готові до використання програмні веб застосунки через мережу Інтернет.

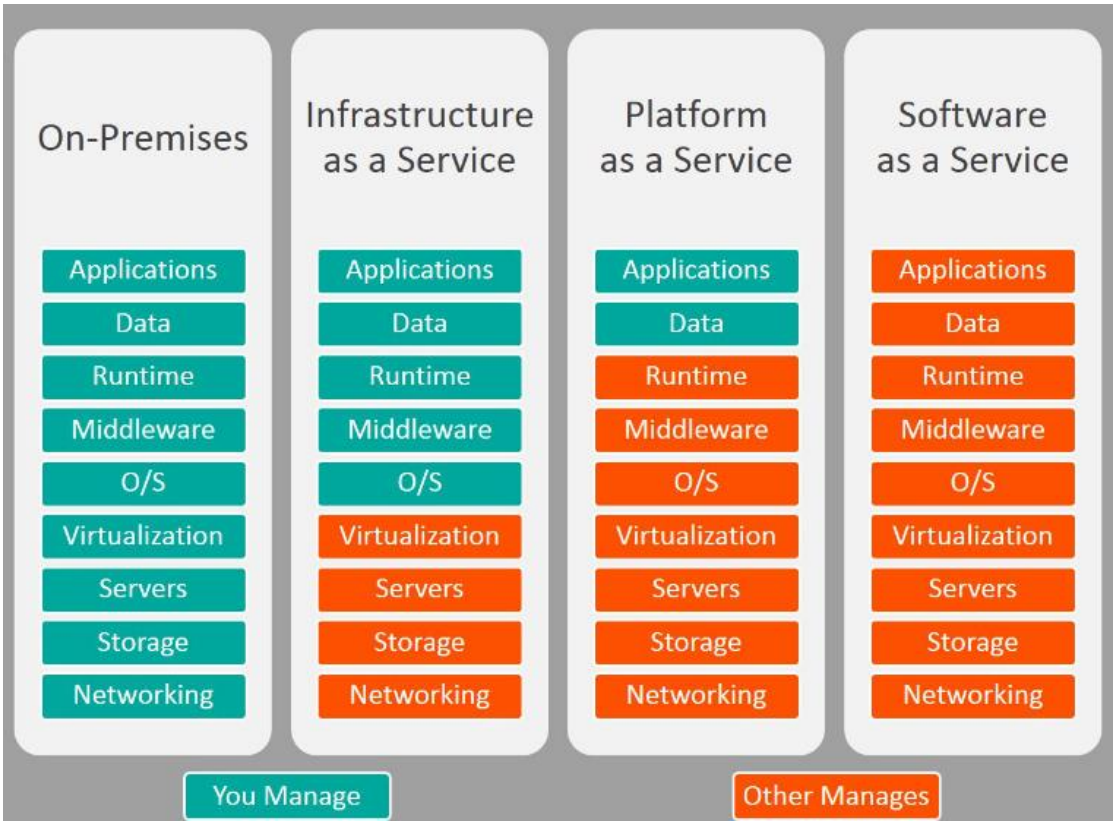


Рисунок 1.3 Опис розподілу відповідальності між клієнтом та провайдером в моделях надання послуг[28]

Вибір одної з наведених вище моделей залежить від потреб бізнесу та його готовності керувати різними аспектами інфраструктури. У довгостроковій перспективі, суб'єкти повинні обрати використання конкретної моделі, для їхніх систем, після проведення повного аналізу цілей, пов'язаних із ризиками та вигодами для кожної моделі, і досягти винятково відмінного балансу, який забезпечить ефективність, стабільність та інновації.

Згідно статистиці організація Statisto, рівень капіталізації кожного з методів надання хмарних послуг зростає, а отже збільшується кількість клієнтів, що ними користуються (рис 1.4). Найприбутковішим та найпоширенішим на даний момент методом надання послуг є SaaS, що зберігає перше місце впродовж років.

Цілком логічно, що SaaS є найпопулярнішим, адже він має перевагу в ціні з поміж інших моделей, із-за чого його частіше обирають звичайні користувачі та малі бізнеси. Також його легше налаштовувати, підтримувати та використовувати для звичайного клієнта.

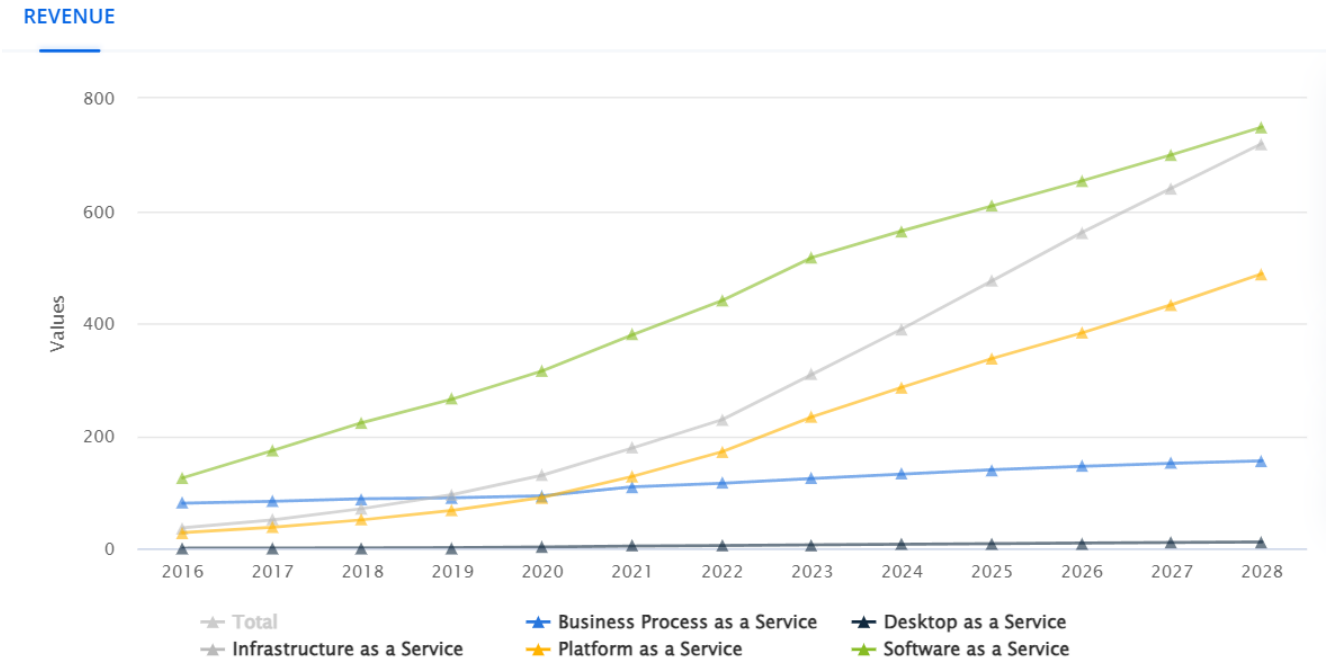


Рисунок 1.4 Графік капіталізації моделей надання послуг [29]

Але за допомогою SaaS, на відміну від IaaS і PaaS, користувач не має контролю над керуванням системою, оскільки провайдер послуг управляє всіма його аспектами. У зв'язку з цим, з 2020 року швидкими темпами розвиваються IaaS

та PaaS, що пов’язано з тим, що компанії вважають IaaS більш гнучким і адаптивним, ніж SaaS, оскільки дозволяє керувати своїми програмами, даними, проміжним програмним забезпеченням і операційною системою. З іншого боку, PaaS дозволяє керувати лише власними даними та програмами.

1.4 Аналіз загроз хмарного середовища

Згідно звіту Threat Horizons, який публікує організація Google Cloud, у 2023 році зросла кількість і складність загроз, націлених на всі IT-середовища, а особливо хмарних [30]. Це можна пов’яти з тим, що із збільшенням популярності впровадження чи міграції до хмари, підвищується рівень уваги зловмисників до цих систем, внаслідок чого, зросла кількість кібератак на хмарні середовища [2, с.17].

Серед проблем безпеки в хмарних обчисленнях, однією з головних є існування численних та різноманітних інструментів та послуг, об'єднаних в одну інфраструктуру, яка охоплює мережі, бази даних, операційні системи, віртуалізацію і багато іншого.

Під час проектування та впровадження програмного і апаратного забезпечення можуть виникати помилки, що призводять до порушення безпеки, чим згодом можуть скористуватися зловмисники. Вразливості локальних технологій актуальні і в хмарних середовищах, проте вплив кібератак тут значно вагоміший. Слід зазначити, що здійснення кібератаки на інформаційні системи компанії в хмарному середовищі породжує значні фінансові втрати, а також негативно впливає на її імідж в довгостроковій перспективі.

Основною метою здійснення кібератак на хмару є набуття доступу до системи з метою контролю та маніпуляції сервісами та системою, щоб завдати серйозної шкоди користувачам, які використовують цю хмарну платформи. Основними цілями таких атак включають використання вразливостей і потенційних "дір" в хмарній системі, що в результаті надає можливість здійснити крадіжку інтелектуальної власності користувача.

Зловмисники постійно вдосконалюють свої методи вторгнення, компроментації, використання та монетизації реалізованих загроз хмарної інфраструктури. Вони знаходять способи отримати первинний доступ до хмарних мереж, здійснюють атаки на ланцюги постачання та виконують шкідливі операції з використанням хмарних середовищ.

Однією з основних недоліком безпеки хмари, зазначає дослідження Threat Horizons, є питання облікових даних, особливо використання клієнтами слабких паролів [31, с.6]. Збір таких даних не потребує значних зусиль для зловмисників, організації, які не дотримуються базових стандартів безпеки, ймовірно, і надалі залишатимуться уразливими до цієї загрози.

Більша частина таких інцидентів вказує на те, що зловмисники отримують доступ до хмарних систем клієнтів, переважно через використання їх власниками слабких, ненадійних паролів або взагалі, що їх не мають. Підібравши слабкий пароль використовуючи словникові таблиці, зловмисники можуть, використати протоколи віддаленого доступу, такі як SSH та RDP, для несанкціонованого проникнення в хмарні системи.

Також досить частою проблемою безпеки є те, що багато випадків витоку даних спричинені неправильно налаштованою системою, що робить випадкові помилки користувачів великою загрозою для хмарних середовищ. Такими неправильними конфігураціями можуть бути як залишення стандартних паролів адміністратора, так і створення недоречних налаштувань конфіденційності так й інші способи.

Коли дані певної організації зберігаються в хмарі, важко відстежити, хто має до них доступ і як вони використовуються, що є наступною проблемою. Це відбувається через те, що багато хмарних сервісів працюють поза межами корпоративних мереж і підключаються через зовнішні, незалежні від провайдера та користувача, сторони.

В результаті, для класифікації способів реалізації противником цих проблем, організація Mitre створила базу знань під назвою MITRE ATT&CK, що

включає в себе тактику і методи діяльності зловмисників, що заснована на реальних спостереженнях.

ATT&CK є незамінним інструментом для аналізу загроз і розробки ефективних методів протидії їм, її застосовують розробники продуктів і послуг кібербезпеки для створення специфічних моделей, матриць, що потім використовуються для боротьби із загрозами, що можуть з’являтися у компаній різних галузей. Звісно, її реалізували для аналізу загроз хмарної архітектури, що зображений у додатоку А.

Така матриця охоплює широкий спектр тактик, технік та процедур, через використання яких, зловмисники можуть вплинути та компроментувати хмарні системи, в наслідок чого її використання дозволяє виявляти всі можливі типи атак на хмару [32].

Ці всі тактики умовно розподілені на групи, що відповідають меті цих атак та послідовності їх використання, наприклад, деякі групи зосереджуються на зборі інформації, а інші на порушенні фуніонування системи або викраденні її даних. Таке структурування дозволяє визначити наміри зловмисників, що, в свою чергу, в кінцевому результаті дозволить розробити більш ефективні заходи захисту. В матриці зазначені такі групи атак [32]:

1. Початковий доступ: атаки, спрямовані на отримання першого доступу до системи або мережі.
2. Виконання: атаки, що включають виконання зловмисного коду або команд на цільовій системі.
3. Наполегливість: атаки, які спрямовані на довготривалу та постійну активність у системі.
4. Підвищення привілеїв: атаки, спрямовані на підвищення рівня доступу та привілеїв на цільовій системі.
5. Ухилення від захисту: атаки, що спрямовані на ухилення від виявлення та захисту, включаючи атаки з маскуванням і змінюванням характеристик атак.

6. Обліковий доступ: атаки, спрямовані на отримання або використання облікових даних.
7. Відкриття: атаки, що використовуються для встановлення додаткових зв'язків або каналів зв'язку у мережі.
8. Бічний рух: атаки, що використовуються для руху або переміщення по мережі з метою пошуку цілей або розповсюдження атаки.
9. Колекція: атаки, спрямовані на збір конфіденційної інформації або розведку в системі або мережі.
10. Ексфільтрація: атаки, що спрямовані на незаконне виведення або втілення даних зі зламаної системи або мережі.
11. Вплив: атаки, що використовуються для нанесення шкоди або впливу на цільову систему, включаючи знищення або перешкоджання її роботі.

В залежності від того, який моделей надання хмарних послуг використовується в певній організації, зловмисники використовують різні типи атак на ці системи.

В залежності від того, яка з моделей надання хмарних послуг використовується в певній організації, матриця буде коригуватися, адже зловмисники використовують різні типи атак на ці системи, бо пені типи атак не будуть настільки дієвими або потрібнимми для IaaS, як для Saas, матриці для даних моделей надання послуг зображені у додатку Б та додатку В.

Так, наприклад, для IaaS, відносно інших моделей, характерні атаки, що пов’язані з експлуатацією вразливостей самої інфраструктури, яку налаштовує користувач, тобто віртуальні машини, мережі, сховища і тд.

Атаки на PaaS, зазвичай, охоплюють ті методи та заходи, що реалізація яких спрямована на сервіси та інструменти, які платформа надає для розробки, розгортання та управління додатками. Можуть використати такі атаки, як міжсайтовий скриптинг, SQL ін’єкція і тд.

Атаки, що націлені на SaaS, зловживають вразливостями додатків і способів зберігання даних в них. Зловмисники часто атакують облікові записи користувачів

експлуатуючи методи автентифікації та авторизації. Також використовують фішингові атаки, які можуть викрасти облікові дані, атаки на API, що дозволяють вплинути на інтегрування служб та процесів, які використовує програма та інші атаки.

1.5 Методи захисту даних в хмарному середовищі

Для компаній, що використовують хмару, реалізація загроз, що досліджені в підрозділі 1.4, може призвести до різних наслідків, що варіюються від шкоди його репутації до грошових витрат та збоїв у роботі інформаційної системи чи, навіть, її повної зупинки. Для подолання чи уникнення цих загроз, впроваджують спеціальні засоби хмарної безпеки в системах хмарного середовища.

Хмарна безпека охоплює комплекс заходів, програмних рішень, спрямованих на захист бізнесу від загроз як із середини, так із зовні середовища. Сучасні компанії мають тенденцію на впровадження стратегії цифрової трансформації, наслідком робити якої є, перенесення значної кількості документів, даних з паперового вигляду у цифровий та використання інтегрованих рішень хмарні сервісів й інформаційних систем своєї інфраструктури, що в результаті збільшує потребу в надійному захисті хмарного середовища [33].

Для забезпечення належного управління даними, якими оперую в хмарному сердовищі, необхідно гарантувати, що на будь-якому стадії даних у певний момент часу, передача, зберігання, обробка, дотримуються основні принципи інформаційної безпеки (конфіденційність, цілісність і доступність). Конфіденційність досягається через шифрування, контроль доступу та авторизацію, цілісність забезпечується механізмами захисту від змін даних, а доступність досягається за допомогою таких методів, як балансування навантаження та резервне обладнання [34].

При використанні підприємствами моделей PaaS та SaaS, провайдери хмарних послуг переймають на себе відповідальність за керування цією інфраструктурою і за безпеку активів даних. Більшість постачальників хмарних

послуг, зазвичай, мають найкращі заходи безпеки та впроваджують активні стратегії для забезпечення цілісності своїх серверів. Проте, організаціям потрібно звернути увагу на свої власні заходи безпеки для захисту даних, додатків і робочих навантажень, що функціонують у хмарному середовищі. Бо не вживаючи активних заходів, організації можуть стати перед значними ризиками пов’язаних з управлінням та оперуванням конфіденційної інформації клієнтів, незалежно від місця її зберігання.

Найважливішим аспектом для будь-якої компанії, що використовує хмарні обчислення, є активне зменшення рівня адміністративних привілеїв та використання адміністративних облікових записів лише за необхідності. Також важливt встановлення автоматизованих інструментів для ідентифікації всіх адміністративних облікових записів і перевірки, що кожен користувач, який має адміністративні привілеї на різних пристроях, є легітимним з точки зору вищого керівництва. Запровадження вимог до паролів, щоб усі адміністративні паролі були надійними, використовуючи комбінації цифр, літер і спеціальних символів, і не містили словникових слів, обов’язкове заміження паролів за замовчуванням для різних систем, перед їх впровадженням у мережевих середовищах. Облікові записи служб також повинні мати складні паролі, які періодично змінюються. Паролі в базі даних мають бути зашифровані або перетворені в хеш-форму [35].

Для забезпечення ефективної хмарної безпеки, реалізовують широкий спектр заходів, серед них найбільш доречними будуть [36, 37]:

- CSPM, впровадження якого допомагає зменшити ризики порушення відповідності, виявляючи та виправляючи неправильні налаштування у публічних хмарах. Вони забезпечують автоматизовану видимість налаштувань систем, постійний моніторинг і орагнізовані алгоритми дій для виправлення проблем у IaaS, PaaS, SaaS.
- Ведення журналів безпеки, для виявлення та розслідування інцидентів або порушень безпеки в системі, що забезпечують детальний опис

подій, здійснення проактивного моніторингу для виявлення підозрілої поведінки та пом’якшення потенційних загроз до їх ескалації.

- Регулярне оновлення засобів та програмного забезпечення, тобто визначення наявних та потрібних вдосконалень програм, що відповідають за захист системи, їх розміщення на відповідних серверах, а потім відстеження і верифікацію успішної інсталяції.
- Налаштування регулярного резервного копіювання або всієї інфраструктури організації, або часткової, що може включати критичні аспекти та зміни конфігурацій, компонентів системи. Впровадити кілька сховищ, що будуть зберігати певний час ці дані.
- Використання багатофакторної автентифікації для входу в систему, що поєднує використання кількох паролів, токенів, біометрії чи певних повідомлень до пристроя, що притаманний конкретному користувачу, що бажає здійснити вхід.
- Конфігурація елементів мережі, впровадження цифрових сертифікатів IPSec, SSL, TLS, для захисту передачі даних, налаштування IDS / IPS, мережевих екранів, інструментів захисту від DDOS та інші.
- Розподілений контроль доступу до системи, щоб певний користувач мав допуск тільки до тих апратних та програмних засобів, які йому потрібні для виконання певного завдання.
- Впровадження шифрування, для перетворення відкритого тексту в зашифрований перед тим, як дані будуть надіслані чи збережені в хмарі.

Висновки до розділу 1

Підсумовуючи, використання хмарних обчислень є одним з основних підходів для передачі ризику, для віддаленого виконання обчислень як для малого

бізнесу, так і для середніх та великих компаній. Адже хмарні технології реалізовують гнучкі та високомасштабовані технології, надаючи всі види ресурсів, щоб підприємства могли швидко адаптуватися та впоратися з змінами ринку, що виникають. Різноманітні типи надання послуг та моделей розгортання дають можливість обрати на налаштувати, згідно вимог організації, хмарні системи.

Використання хмарних сервісів сприяє зниженню витрат на інфраструктуру та її обслуговування, оскільки зменшується потреба у значних капіталовкладеннях у фізичні сервери та обладнання й користувачам хмарних послуг необхідно платити тільки за те, чим вони користуються.

Хмарні технології застосовують для зберігання значних обсягів даних, дозволяють виконувати складні обчислення, наприклад, обробку великих наборів операцій над певною інформацією, завдяки використанню масивів віртуалізованих ресурсів, що для клієнтів, створює можливості отримати віддалений доступ до програмного забезпечення, що дозволяє користувачам працювати з додатками та сервісами незалежно від їх місцезнаходження.

Крім того, хмарні технології можна поєднати із заходами безпеки що розташовані локально, тобто на серверах компанії, з можливістю управління розподіленої інфраструктури в одному програмному додатку.

Також, у хмарних системах, зокрема в IaaS, можна реалізувати функції резервного копіювання, налаштувати розподілений доступ до систем, файлів та особистої інформації, інтегрувати з системами управління та моніторингу подій для забезпечення покращеного рівня безпеки й ефективності в управлінні бізнес-процесами.

Тож впровадження хмарного обчислення покращить ефективність розподілу ресурсів, забезпечить інструментами реалізації доступності та цілісності систем клієнта цих послуг, що є перевагою перед використанням лише локальної інфраструктури.

РОЗДІЛ 2
АНАЛІЗ ГОМОМОРФНОГО ШИФРУВАННЯ ЯК ЗАСОБУ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ІНФОРМАЦІЇ

2.1 Огляд стратегій використання шифрування

Шифрування — це метод, який за допомогою визначеного алгоритму, перетворює інформацію з певного відкритого, незашифрованого повідомлення на організовану послідовність символів, що утворюють шифротекст та приховують справжнє значення інформації. Метою шифрування даних є забезпечення конфіденційної інформації, щоб сторонні користувачі, без згоди власника інформації, не могли з нею ознайомитись [38].

При реалізації алгоритму шифрування або шифру, використовується ключ, випадково згенерована послідовність символів, бітів, що відповідає потребам певному алгоритму. Чим довша довжина ключа, тим він надійніший.

Раунди шифрування – це певна послідовніть етапів обробки даних в алгоритмі та включає різні криптографічні операції, такі як підстановка, перестановка та змішування. Кількість раундів впливає на стійкість алгоритму до криптоаналізу, а також на швидкість шифрування та розшифрування даних [38].

Зазвичай, шифрування використовується для захисту даних, що знаходяться як у стані спокою, так і ті, що в русі. У стані спокою – це ті дані, які зберігаються на пристроях та, на момент їх розгляду, не використовуються в певному процесі чи обробці. В русі – це дані, що передаються між пристроями через мережу або використовуються в активному процесі.

Впровадження коректного шифрування грає важливу роль у функціонуванні, захисті різних категорій IT-активів і персональної інформації (PII). З цією метою шифрування виконує чотири основні ролі [39]:

- Шифрує дані, щоб перешкодити їх розумінню, якщо вони перехоплені.

- Підтверджує джерело зашифрованих даних.
- Підтверджує, що дані залишаються незмінними після шифрування.
- Перешкоджає передавачам відмовитися від передачі зашифрованих даних.

Існують певні стандарти, що вимагають компаній, яким він потрібен для діяльності, шифрування персональних даних. Наприклад, стандарт безпеки даних індустрії платіжних карток (PCI DSS) вимагає від компаній, що надають банківські послуги, шифрувати дані платіжних карток клієнтів як у стані спокою, так і під час передачі через мережу Інтернет.

Шифрування має ряд недоліків, воно також може запобігти власникам даних отримати доступ до їх власної інформації. Якщо ключі шифрування буде втрачено або знищено, власники даних можуть назавжди втратити доступ до цих даних. Тому для запобігання таких ситуацій, створили систему керування ключами, що автоматично, замість власника інформації, зберігає сам цей ключ, довжина ключа, дані про те, , в яких алгоритмах він використовується та в яких процесах.

Іншою проблемою з шифруванням є той факт, що зловмисники також можуть використовувати програми-вимагачі, після компрементації системи, після чого злочинці отримують доступ до конфіденційних даних, шифрують їх за допомогою власних алгоритмів, а потім вимагають у організації-жертви виплатити викуп, який може бути досить коштовним [40].

Використання шифрування також впливає на продуктивність системи, оскільки процес перетворення інформації у формат, який унеможливлює її зчитування, потребує певного часу. Залежно від алгоритму шифрування, що використовується для конкретного процесу, змінюється і час, необхідний для його реалізації, що може призводити до затримок, особливо при обробці великих обсягів даних.

Шифрування даних є важливою компонентою в інформаційній діяльності, бо використовується як для захисту особистих даних, так і для забезпечення безпеки корпоративних систем і комунікаційних каналів, також вона

впроваджується, як засіб захисту від несанкціонованого доступу до персональних даних користувача, реалізація якого є необхідною особливо фінансовій сфері, охороні здоров'я та для компаній, для яких втрата або порушення безпеки даних може призвести до серйозних наслідків.

2.1.1 Види шифрування та їх особливості

Шифрування поділяється на симетричне та асиметричне.

Симетричними є ті алгоритми шифрування, що використовують спільний ключ і для шифрування, і для дешифрування даних. Для його виконання, відправник та отримувач повідомлення повинні мати доступ до конкретного, спільного ключа. Переважна більшість алгоритмів шифрування з симетричним ключем, є набагато швидшими, ніж з асиметричним шифруванням. Найвідомішими прикладами симетричних алгоритмів є AES, DES, IDEA [41].

Симетричні алгоритми шифрування також можна поділити на категорії: потокові та блочні алгоритми шифрування.

Блоковий шифр — це метод шифрування даних, що поділяє вхідне, відкрите повідомлення на блоки фіксованого розміру та виконає над ними алгоритмічні дії одночасно. Більшість сучасних блокових шифрів призначені для шифрування даних у вигляді блоків фіксованого розміру 64 або 128 біт.

У потоковому шифруванні відкритий текст перетворюється за допомогою криптографічного ключа і алгоритму, застосовуючись поступово до кожного біту чи байту в потоці даних. Цей підхід забезпечує швидкий та відносно простий процес шифрування.

Реалізація симетричних алгоритмів має певні переваги над асиметричними, ними є переважна простота впровадження, швидкість обробки, що зумовлено тим, що операції шифрування та дешифрування в цих алгоритмах під час обчислень менше використовують ресурсів інфраструктури, що дозволяє швидше обробляти великі обсяги інформації.

Асиметричиними є ті алгоритми шифрування, що використовує пару ключів, які логічно та математично пов’язані, для шифрування та дешифрування даних. Для генерації ключів застосовуються прості числа, бо використовуючи їх можна створити досить складний ключ, якого досить важко взламати або підробити. Найвідомішими прикладами асиметричних алгоритмів є RSA, SHA, алгоритм Діффі-Хеллмана [42].

Для реалізації асиметричного шифрування отримувач та відправник повідомлення повинні виконати наступні дії:

1. Кожен повинен згенерувати публічний та приватний ключі. Публічний використовується для зашифрування, приватний – для розшифрування.
2. Потім отримувач та відправник обмінюються публічними ключами.
3. Використовуючи публічний ключ отримувача, відправник шифрує своє повідомлення і надсилає його отримувачу.
4. Одержавши повіомлення, отримувач розшифровує його за допомогою свого приватного ключа і для відправки та зашифрування відповіді, використовує публічник ключ відправника.

Значною перевагою асиметричних алгоритмів відносно симетричних, є безпека ключів, захист від підслуховування та незалежність ключів. Кожен з ключів використовуються тільки за своїм призначенням, приватний зберігається лише на стороні клієнта, що забезпечує те, що навіть при компроментації публічного ключа, повідомлення не стає доступним для зловмисника, що в результаті гарантує кращий рівень безпеки при передачі конфіденційних даних в порівняно з симетричними.

Симетричні алгоритми, зазвичай, використовуються для ефективного шифрування великих обсягів даних. Вони, зазвичай, використовується в застосунках, для зберіганні та передачі даних через електронну пошту, для віддаленого обмін інформацією SSH, шифрування файлів на диску та в сховищах.

Асиметричні алгоритми менш поширені, вони часто використовуються для завдань, де критично важливою є безпека, наприклад в цифрових підписах, у встановлення безпечних каналів зв’язку та інші.

Також для певних ситуацій, реалізовують поєднаний метод шифрування, коли використовується і симетричний, і асиметричний алгоритми. Прикладом такої реалізації є протоколи SSL/TSL, де асиметричне шифрування потрібне під час початкової фази, рукостискання для безпечного надсилання симетричних ключів. Після їх отримання, вже застосовується симетричне, щоб шифрувати подальші дані, якими обмінюються клієнт та сервер. В результаті такого поєднання, забезпечується як і достатня безпека, так і швидкість передачі повідомлення [43].

2.2 Огляд особливостей гомоморфного шифрування

Внаслідок постійного вдосконалення технологій обробки апаратного та програмного забезпечення, покращились методи та алгоритми шифрування, як наслідок, можливості обміну та зберігання даних стали більш безпечними, ніж будь-коли раніше. Однак у результаті розвитку хмарних технологій, класичні підходи до шифрування стають обмеженішими, і це зумовлює необхідність редагувати концепцію безпеки, враховуючи особливі проблем розподіленого обчислення та обміну даними в хмарній інфраструктурі, особливо публічної.

Існує тенденція компаній розгортати свої інформаційні системи на інфраструктурі сторонніх осіб, хмарних провайдерів, що вкрай ускладнює підтримку таких параметрів безпеки, як безпека даних, конфіденційність, цілісність та інших.

Тому користувачі хмарних технологій віддають перевагу зберіганню своїх даних в хмарному середовищі саме в зашифрованій формі, щоб зменшити ризик того, що зловмисник, отримавши доступ до системи, зможе скористатися цими даними. Однак, для виконання будь-якої операції з ними в хмарному середовищі потрібно спочатку їх розшифрувати, що створює проблеми з конфіденційністю та приватністю даних, що зберігаються в хмарі.

Для вирішення цих проблем Швидко розробляються та впроваджуються рішення, зокрема й технології гомоморфного шифрування.

Алгоритми шифрування, що мають властивості гомомофрності – це ті методи, реалізація яких, дозволяє обчислювати дані, які є попередньо перетворені у шифротекст, тобто виконувати адитивні та мультиплікативні операції над цими даними, не розшифровуючи їх, та аналізуючи цей текст напряму. Іншими словами, при впровадженні гомоморфних алгоритмів шифрування до систем, надається можливість оброблювати інформацію, що там циркулюється, в зашифрованому вигляді та надати потрібний результат, який потім треба буде розшифрувати [44].

Алгоритмічним прикладом може бути ситуація, де користувач, що працює з системою гомоморфного шифрування, може здійснити операції над даними, такі як знаходження суми двох чисел, в сервері, не розкриваючи при цьому значення самих чисел.

Отже при реалізації гомоморфного шифрування, можна виконувати обчислювальні функції на певних, попередньо зашифрованих даних, без знання їх приватного ключа, власником якого є окремий клієнт, що отримує послуги з обробки інформації. Щоразу, отриманий результат після цих операцій, для будь-якої обчислювальної функції, після розшифрування, буде подібним до того, результату, який би був отриманий при продовженні процедури обчислень на непроаналізованих і гомоморфно незашифрованих даних.

Алгоритм є гомоморфним, тільки в тому випадку, якщо над ним можливе виконання операцій такі, як додавання, віднімання та/або множення , ділення над зашифрованими даними, та отримання результату, що відповідає здійненню відповідних операцій щодо незашифрованих даних [45]. Вигляд рівнянь, що показують приклади реалізації цих властивостей буде відрізнятися залежно від розглянутого алгоритму, тому ,наприклад, для шифру Пайє, його можна описати формулами 2.1 для адитивних операцій та 2.2 для мультиплікативних.

$$D(En(t_1) * En(t_2)\ mod\ n^2) = t_1 + t_2\ mod\ n, \quad (2.1)$$

$$D\ (En\ (t_2\ )^{t_1} mod\ n^2) = t_1 * t_2\ mod\ n, \quad (2.2)$$

де t_1 та t_2 - це певні дані, числа;

$En(t_i)$ – зашифровані за певним алгоритмом дані

D(x) – розшифрування даних

2.2.1 Види гомомофрного шифрування

Гомоморфне шифрування існує у багатьох формах, щоб відповідати унікальним випадкам використання та вимогам безпеки. Основними видами гомоморфного шифрування є [46]:

- Часково гомоморфне шифрування, коли певний алгоритм може виконувати лише певний вид операцій над зашифрованою інформацією, без її розкриття. Воно також поділяється на адитивний та мультиплікативний гомоморфізми, тобто ті, які можуть виконувати лише операції або додавання та віднімання, або множення та ділення над шифротекстами відповідно
- Повністю гомоморфне шифрування (FHE), дозволяє виконувати операції додавання і множення над зашифрованими даними, а також складніші операції, такі як логічні (і, або, ні). Порівняно з іншими типами, його реалізація потребує значних операційних ресурсів у системах. FHE також поділяється наступні підтипи:
  - Дещо гомоморфне шифрування (SHE), що має тільки певну число операцій після закінчення яких, зникає властивість гомоморфізму.
  - Шифрування з можливістю початкового завантаження (BHE), є підтипом FHE і використовується для здійснення обчислень над зашифрованими даними, не обмежуючи кількість таких операцій. Однак для забезпечення ефективності і безпеки він потребує періодичного оновлення зашифрованого тексту, що може вимагати значних обчислювальних ресурсів і надійних алгоритмів.

- Приблизне гомоморфне шифрування, схеми гомоморфного шифрування, що мають приорітет на ефективності системи над точністю, тобто чим більша кількість операції проводиться за допомогою таких алгоритмів, тим збільшується шанс на виникнення помилок в обчисленні. Часто, їх реалізовують у сферах, де точність для обчислення не є критичною, наприклад, у машинному навчанні та в аналізі великих даних.

Попри те, що гомоморфне шифрування має значні переваги, але йому також притаманні й недоліки: обчислювальна складність та обчислювально-ресурсні витрати. Більшість гомоморфних схем шифрування потребують значних обчислювальних ресурсів та часу, особливо, при обробці великих обсягів даних або використання складних операцій, що робить їх менш ефективними для деяких конкретних застосувань, де швидкість та продуктивність є критичними факторами. Такі обчислення можуть бути особливо складними для реалізації в системах або в ресурсозатратних середовищах, де кожен цикл обчислень має значення.

2.2.2 Опис процесу підготовки зашифрованих даних для відправлення до хмари

Процес підготовки гомоморфно зашифрованих даних до передачі цих матеріалів у хмарну інфраструктуру складається із досить вагомої послідовності кроків, що виконуються поступово, для забезпечення коректності та цілісності даних, та гарантування безпеки цієї інформації для їх надсилання та зберігання на віддаленому сервері (рис 2.1).

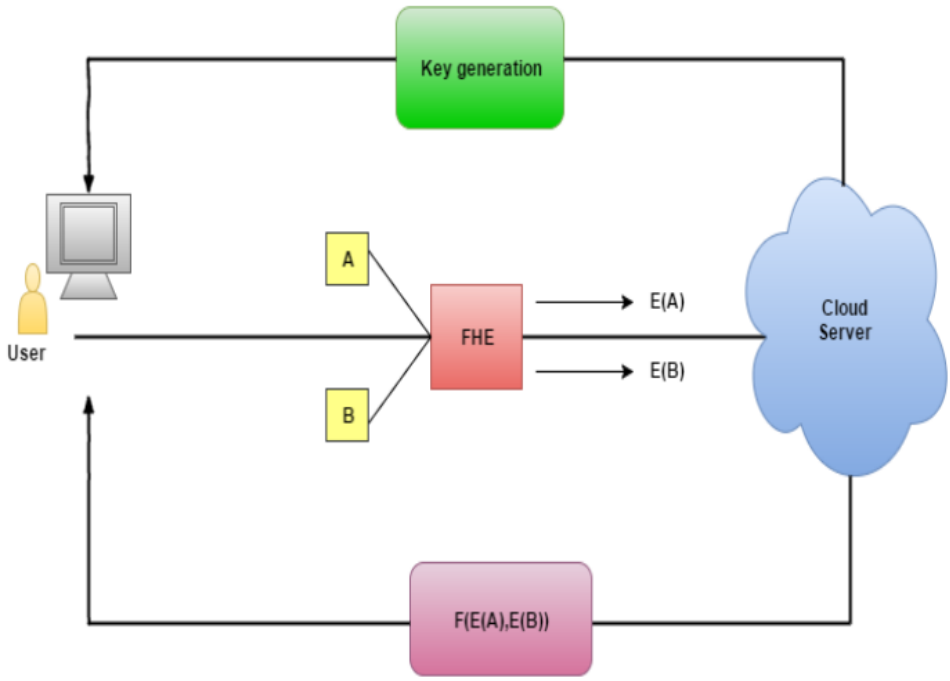


Рисунок 2.1 – Механізм відправлення даних до хмари [47]

Спочатку обирається алгоритм гомоморфного шифрування та довжина приватного ключа, що буде використовуватись для перетворення. Відповідно до вимог обраного алгоритму, генерується ключ, який важливо мати у вигляді випадкової послідовності символів.

Далі дані шифруються, для перетворення їх у формат, що незрозумілий для неавторизованих користувачів, що не мають ключа для їх подальшого розшифрування.

Потім, за допомогою безпечних протоколів зв’язку, дані надсилаються попередньо налаштованій, хмарній системі, що згідно з потребами користувача оброблює їх в зашифрованому вигляді.

Після завершення обчислень, хмарний сервер може відправити результат обробки клієнту для подальшого розшифрування та аналізу, або ж зберегти його до моменту, коли користувач вирішить його отримати.

2.3 Алгоритм шифрування Пайє

Криптосистема Пайє – це алгоритм шифрування, результатом реалізації якого перетворений в нечитабельний вигляд вхідне повідомлення. Цей алгоритм є асиметричним, отже для перетворень використовує два ключі. Публічний ключ потрібен для шифрування повідомлень, тоді як приватний— для їх розшифрування. Шифротексти, що утворені внаслідок перетворень цього алгоритму, мають властивості є часткової гомоморфності, що надає можливість використати безліч адитивних операцій над зашифрованими даними, при цьому отримуючи той самий результат, що й при обчисленні над відкритими даними. В рівнянні 2.2 видно, що мультиплікативні операції також можна реалізувати, але їх обчислення потрібно піднести значення шифротексту у степінь значення відкритого другого повідомлення, на яке хочеш помножити.

Для реалізації цього алгоритму, потрібно обрати 2 випадкових, великих простих числа p та q, вони не повинні бути пов’язані та повинні відповідати рівнянню 2.3:

$$\text{НСД}(pq,\ (p-1)(q-1)) = 1\ , \tag{2.3}$$

Далі встановлюються параметри n та λ:

$$n = p * q, \tag{2.4}$$

$$\lambda = \text{НСК}(p-1, q-1), \tag{2.5}$$

$$\mu = (L(g^{\lambda} mod\ n^2))^{-1}\ mod\ n, \tag{2.6}$$

де
$$L(x) = \frac{(x-1)}{n}, \tag{2.7}$$

Після того, як були сформовані ці параметри, можна вказати якими будуть приватний та публчний ключі.

Публічний ключ буде генеруватися, як комбінація параметрів n та g.

Приватний, буде сформований з параметрів λ та μ.

Для спрощення створення ключів, є варіант того, що змінні трохи по-іншому будуть отримувати значення. У випадку того, що p q матимуть однакову довжину, можна вказати, що:

$$g = n + 1, \lambda = \varphi(\mathrm{n}), \tag{2.8}$$

$$\mu = \varphi(\mathrm{n})^{-1}\ mod\ n, \tag{2.9}$$

$$\varphi(\mathrm{n}) = (\mathrm{p} - 1)(\mathrm{q} - 1), \tag{2.10}$$

Після генерації ключів, починається процес шифрування, в якому змінній m відповідає все повідомлення, що буде шифруватися, та воно повиннj відповідати умові:

$$0 \leq m < n\ , \tag{2.11}$$

Потім обирається випадкове r число, що відповідає наступним умовам:

$$0 < r < n, \tag{2.12}$$

$$\mathrm{НСД}(r, n) = 1\ , \tag{2.13}$$

В результаті, треба використати ці змінні для зашифрування повідомлення, за результат якого буде відповідати змінна c, що вираховується за формулою:

$$c = g^m r^n mod\ n^2, \tag{2.14}$$

Вже для розшифрування використовується пара змінних, що відповідаються приватному ключу та розраховується за наступною формулою:

$$m = \left(L\left(c^{\lambda} mod\ n^2\right)\right) * \mu\ \ mod\ n, \tag{2.15}$$

Тепер описавши алгоритм шифрування, спробую зашифрувати певний відкритий текст.

Маю повідомлення m значення його дорівнює 50.

Для генерації ключів надаю значення змінним, p=43, q=47, r=89.

Визначаю n=2021 та λ=966, g= 66.

Розраховую μ:

$$\mu = \left(\frac{66^{966}\ mod\ 2021^2 - 1}{2021}\right)^{-1}\ mod\ 2021, \tag{2.16}$$

$$\mu = 664, \tag{2.17}$$

$$c = 66^{50} * 89^{2021} mod\ 2021^2 = 2186974, \tag{2.18}$$

Тепер розшифровую:

$$m = \left(\frac{(2186974^{996} mod\ 2021^2) - 1}{2021}\right) * 664\ \ mod\ 2021, \tag{2.19}$$

$$m = 50 \quad (2.20)$$

Для перевірки на гомоморфність цього алгоритму треба обрахувати ще одне повідомлення, m=45, ключі також повинні бути схожими, тому p=43, q=47, r=90**,** n=2021 та λ=966, g= 67, $\mu = 664$.

Після зашифрування отримав c=163785.

Згідно з формулою 2.1, виконуватиму адитивну операцію над цими повідомленнями, як в зашифрованому вигляді, так і у відкритому.

$$2186974 * 163785 mod\ 2021^2 = 66^{m_1+m_2} * 89^{2021} mod\ 2021^2, \quad (2.20)$$

$$180528142778\ mod\ 2021^2 = 66^{95} * 89^{2021} mod\ 2021^2, \quad (2.21)$$

$$4019460 = 4019460, \quad (2.22)$$

Висновки до розділу 2

Впровадження шифрування в системах є досить важливим аспектом їх забезпечення захисту даних, оскільки може гарантувати конфіденційность й цілісность цієї інформації. З його допомогою можна було захистити дані від несанкціонованого доступу та зберегти їх конфіденційність навіть у разі порушення безпеки мережі та пристроїв, а також перехоплення даних, що є критично важливою в контексті зберігання та передачі чутливої інформації в хмарних системах, такої як особисті дані клієнтів або корпоративні секрети.

Шифрування також впливає на продуктивність системи, оскільки процес шифрування та дешифрування займає деякий час і ресурси. Оптимальний вибір алгоритмів шифрування та правильна конфігурація є дуже критичним завданням для компаній, впроваджують заходи засхисту, щоб досягти оптимальної продуктивності та безпеки інформаційних систем.

Правильна реалізація симетричних та асиметричних алгоритмів шифрування, дозволяє застосувати наявні переваги таких методик для збільшення ефективності використання ресурсів. Асиметричне шифрування застосовується для завдань, де безпека має вирішальне значення, наприклад, як початкові

рукостискання перед початком спілкування сервера з клієнтом. На противагу йому, симетричне потрібне, щоб швидко шифрувати великі об’єми даних.

Реалізація гомоморфного шифрування дозволяє виконувати обчислення на зашифрованих даних без їх дешифрування, результат цих операцій, для будь-якої обчислювальної операції, після дешифрування буде подібний до результату, отриманого від відкритого повідомлення, що забезпечує конфіденційність інформації під час обробки.

Криптосистема Пайє представляє собою алгоритм асиметричного шифрування, що базується на використанні публічного та приватного ключів для забезпечення безпеки обміну даними та має властивості часкової гомоморфності, що дає можливість виконувати адитивні операції над зашифроватими даними, що є дуже бажаним в середовищах, налаштування та підтримка, яких відбувається не за участі власника інформації.

РОЗДІЛ 3
ВПРОВАДЖЕННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ГОМОМОРНОГО ШИФРУВАННЯ В ХМАРНИХ ОБЧИСЛЕННЯХ

3.1 Структуризація алгоритмічних потреб для реалізації зашифрованих обчислень

Для впровадження надійної та оптимізованої системи гомоморфного шифрування в хмарних обчисленнях, програмну реалізацію потрібно розділити на дві частини, що будуть виконуватися для різних суб’єктів обчислення, а ними є:

- Клієнт;
- Хмарне середовище.

Потреба поділу виникає через необхідність реалізувати механізми, які на стороні клієнта буде перетворювати певні відкриті повідомлення на шифротексти за допомогою гомоморфного алгоритму шифрування Пайє, а потім, після їхнього надсилання, в хмарному середовищі буде обчислювати ці дані відповідно до налаштованої послідовності дій, які впливають на те, якими операціями будуть здійнюватись обчислення над отриманими шифрами.

Для цього було створено два програмних застосунків, що відповідають потребам, що вказані раніше. Повна програмна реалізація яких, вказана у додатках Д та Е. Їх було реалізовано так, щоб результати обчислень цих застосунків впливали на функціональність один одного та на коректність формату отриманого результату. Тому для розробки цих програмних додатків, було використано спільну мову програмування, а саме C#.

Оскільки C# є багаторівневою мовою програмування, що надає доступ до широкого спектру інструментів та бібліотек, використання яких, пришвидшує реалізацію криптографічних алгоритмів шифрування, зокрема і Пайє, та зменшує шанс виникнення помилок під час реалізації програмого застосунку, що написаний на цій мові [48]. Крім того, вона дозволяє працювати із класами, об'єктами, і саме

важливе, дозволяє керувати динамічною пам'яттю для ефективного використання ресурсів пристрою. Існують спеціальні шаблони пов’язані з C#, за допомогою яких, нпариклад, можна інтегрувати програмний застосунок з Azure, що полегшило подальшу реалізацію взаємодії додатку з хмарним середовищем.

Хоч створити реалізацію алгоритмів було б простіше з використанням Python, основним фактором вибору саме C# є те, що застосунок написаний на цій мові програмування, буде швидше обчислюватися відносно аналогічного застосунку, що створений на Python. Швидкодія є критичним фактором, оскільки для зашифрування та дешифрування відносно середніх та великих повідомлень, потрібно обрати приватний ключ, довжина якого повинна як мінімум бути не меншою за самий текст, в кращому випадку більшою, використання яких значно впливає на швидкість перетворення початкового значення.

3.2 Реалізація першого програмного застосунку для клієнта

Перший програмний застосунок, який впроваджений на стороні клієнта, реалізований у вигляді комадного рядка, тобто комунікація між користувачем та програмою здійснюється через CLI.

Застосунок має два режими роботи: шифрування та дешифрування. Для їх реалізації обов’язково потрібно згенерувати приватний та публічний ключі, які формуються з двох випадкових та двох простих чисел p та q. Особлива увага приділяється саме до останніх двух, адже вказавши числа, що не відповідають потребам алгоритму, можна отримати неправильні значення, що зіпсують функціонування програми.

Числа p та q обирає сам користувач застосунку, після того, як вказує власне повідомлення, що представлене в десятковому форматі. Слід зазначити, що для шифрування повідомлення, числа потрібно підібрати таким чином, щоб їх добуток був не меншим за значення повідомлення, приклад показано рисунку 3.1, де для повідомлення, значення якого є 100, було обрано прості числа 43 та 47, результатом добутку яких буде 2021, що значно більше за вхідне повідомлення.

Якщо результат шифрування буде потім використовуватись для обчислень без розкриття змісту повідомлення, то значення простих ключів повинні бути ще більшими, щоб результат обчислень в хмарному середовищі був вірний.

```
Do you want to encrypt or decrypt (type E or D)
e
ENCRYPTION
type your message
100
Process of creating public and private keys
try to type two prime numbers that bigger than 40
43 47
Do you want R to be random Y/N
y
your encrypted message is 3896297
Do you want to end this process Y/N
```

Рисунок 3.1 Виконання шифрування над повідомленням

Для дешифрування треба лише вказувати шифротекст та приватний ключ, за допомогою якого раніше цей текст було зашифровано (рис. 3.2).

```
Do you want to encrypt or decrypt (type E or D)
d
DECRYPTION
type your message
3896297
3896297
Process of creating public and private keys
try to type two prime numbers that bigger than 40
43 47
your decrypted message is 100
```

Рисунок 3.2 Виконання дешифрування

Для перевірки відповідності простих чисел до алгоритму та генерації ключів використовуються функції, що знаходять найменше спільне кратне та найбільший спільний дільник, які в коді мають назви LCM та MCD відповідно (рис.3.3).

```
119     private uint MCD(uint number1, uint number2)
120     {
121         while (number2 != 0)
122         {
123             uint temp = number2;
124             number2 = number1 % number2;
125             number1 = temp;
126         }
127         return number1;
128     }
        0 references
129     private long LCM(uint number1, uint number2)
130     {
131         return Math.Abs(number1 * number2) / MCD(number1, number2);
132     }
        1 reference
133     private bool IsCorrectPrimeNumbers(uint number1, uint number2)
134     {
135         // перевірка чи числа прості
136         if (!IsPrime(number1) || !IsPrime(number2))
137             return false;
138         // чи вони менші за 40
139         else if (number1 < 40 || number2 < 40)
140             return false;
141         // перевірка чи виконується умова
142         else if (MCD(number1, number2) != 1)
143             return false;
144         return true;
145     }
        1 reference
146     private void generateKeys(bool state)
147     {
148         // генерація простих чисел
149         setNumbers(state);
150         // надання значень змінним, які будуть використовуються для подальших обчислень
151         n = p * q;
152         nsquare = n * n;
153         phi = (ulong)((p - 1)*(q - 1));
154         g = n + 1;
155         lya = phi * 1;
156         u = ModInverse(L(BigInteger.ModPow(g,lya,nsquare)),n);
157 
158     }
```

Рисунок 3.3 Алгоритм генерації ключів

Для шифрування повідомлення, згідно з підрозділом 2.3, у вигляді коду реалізовано послідовність дій:

```
public void encryption(BigInteger m)
{
    // очищую попередні заначення змінних
    p = 0; q=0;r=0;g=0;lya=0;n = 0;u = 0;
    resultat = 0;

    // вказую прості числа та генерую ключі
    generateKeys(true);

    // перевіряю чи повідомлення менше за ключ n
    checkGenerationOfKeys(m,true);

    // підношу в степінь m число g та по модулю n^2
    step1 = BigInteger.ModPow(g, m, nsquare);

    // підношу в степінь n число r та по модулю n^2
    step2 = BigInteger.ModPow(r, n, nsquare);
    step3 = step1 * step2;
    // отримую остаточний результат шифрування
    resultat = step3 % nsquare;
    Console.WriteLine("your encrypted message is "+ resultat);


}
```

Рисунок 3.4 Алгоритм шифування вхідного повідомлення m

Щоб дешифрувати шифротекст, було здійснено наступний порядок виконання:

```
public void decryption(BigInteger c)
{
    // очищую попередні значення змінних
    p = 0; q=0;r=0;g=0;lya=0;n = 0;u = 0;
    resultat = 0;

    // вказую прості числа та генерую ключі
    generateKeys(false);
    checkGenerationOfKeys(c,false);

    // підношу в степінь lya число c та по модулю n^2
    step1 = BigInteger.ModPow(c, lya, nsquare);

    // обчислюю
    step2 = (step1-1)/n;
    step3 = step2 * u;

    //отримую початкове відкрите повідомлення
    resultat = step3 % n;
    Console.WriteLine("your decrypted message is " + resultat);

}
```

Рисунок 3.5 Алгоритм перетворення шифротексту c у вхідне повідомлення

В результаті виконання всіх обчислень буде отримано значення початкового повідомлення.

3.3 Впровадження другого застосунку в хмарному сервісі Azure

Особливістю другого застосунку є те що він розміщений на стороні хмарного середовища. Він потрібен, щоб виконувати обробку даних, які зашифровані за допомогою гомоморфного алгоритму та отримані зі сторони клієнта. Ці обчислення було реалізовано так, щоб обрахунок здійснювався без необхідності їх розшифрування, що в результаті забезпечить конфіденційність наданої інформації на момент її розміщення в хмарному середовищі, адже навіть при нейтралізації систем безпеки та копроментації хмарної інфраструктури зловмисником, дані клієнта залишаються захищеними, бо в хмарі вони зберігаються лише у вигляді шифротекстів та публічних ключів, а приватні ключі, тобто ті, що потрібні для розшифрування, розміщені лише на стороні клієнта.

В такому способі обробки, де всі операції проводяться із зашифрованими даними, процес обчислень стає складнішим і тривалішим порівняно з обробкою незашифрованих даних і потребує значно більше часу для виконання всіх операцій, запланованих клієнтом, а також вимагає значних ресурсів, особливо, для обробки великих обсягів інформації. Тому впровадження обробки гомоморфного шифрування даних в хмарному середовищі, не завжди є доцільною для компаній. Найбільш корисним застосуванням буде для організацій, що працюють з персональними та конфіденційними даними, прикладами яких можуть бути, як медичні записи пацієнтів, фінансові транзакції, унікальні дані клієнтів або корпоративна інформація, в яких важливість збереження конфіденційності даних перевищує потребу в швидкості обробки результатів.

Крім того, після впровадження цього застосунку в хмарному середовищі, можна інтегрувати його з іншими сервісами та інструментами, що надаються постачальником цієї хмарної інфраструктури, що в результаті допомагає доцільно керувати ресурсами та автоматизувати різноманітні процеси. Так використовуючи

хмарну платформу Azure, застосунок можна поєднати з системами моніторингу та керування подіями, що показує, як часто використовувалась програма та коли і хто її запускав, що допомагає вчасно виявляти і реагувати на всі можливі загрози та аномалії в роботі застосунку.

3.3.1 Підготовка хмарного середовища до реалізацій застосунку

Для впровадження другого додатку, необхідно визначити та налаштувати, певну хмарну інфраструктуру, що буде розміщувати його. З цією метою було вирішено скористатися платформою Azure, яка пропонує вагомий перелік сервісів, серед яких є можливість розміщення власного програмного забезпечення.

Azure, що розроблене компанією Microsoft, є публічною хмарною платформою, що пропонує широкий спектр хмарних послуг, що охоплюють обчислення, аналітику, зберігання великого вмісту файлів, засоби керування мережі й створення, розгортання та управління програмами та сервісами через Інтернет [49].

Azure реалізований, як поєднання чотирьох різних моделей надання послуг в хмарних обчисленнях, тобто об’єднує функції IaaS, SaaS та PaaS.

У цілому, Azure себе представляє, як SaaS, оскільки вона надає широкий спектр готових для використання програмних додатків, доступ до якої надається через веб браузер, й оплата здійснюється лише за ті функції, якими користуєшся. Такими послугами є: Dynamics 365, Outlook і Office 365, керування штучним інтелектом, управлінням та статистикою девайсів, що налаштовані через систему Active Directory та багато інших [50].

З послуг, що притаманні IaaS, вона надає доступ до керування віртуальними обчислювальними ресурсами, можна отримати доступ та налаштувати відповідно до потреб компанії сервера, сховища даних, мереж та іншими компонентами інфраструктури, такими як DNS-сервери, служби Windows Server, наприклад, IIS, мережеві служби, брандмауери або програми сторонніх розробників, тощо.

Серед функцій PaaS, Azure дозволяє впроваджувати та управляти цілим середовищем розробки і розгортання в хмарі. Можна створювати, тестувати, розгортати та керувати програми, не турбуючись про налаштування базовою інфраструктурою.

Щоб розгорнути програмну реалізацію обчислення гомоморфних шифротекстів, було створено план обслуговування додатків у Azure, який надає обчислювальні ресурси, визначає операційну систему, можна під'єднати систему моніторингу подій і регіон, в якому буде надаватися доступ до програми, що буде впроваджена в хмарному середовищі (рис. 3.6).

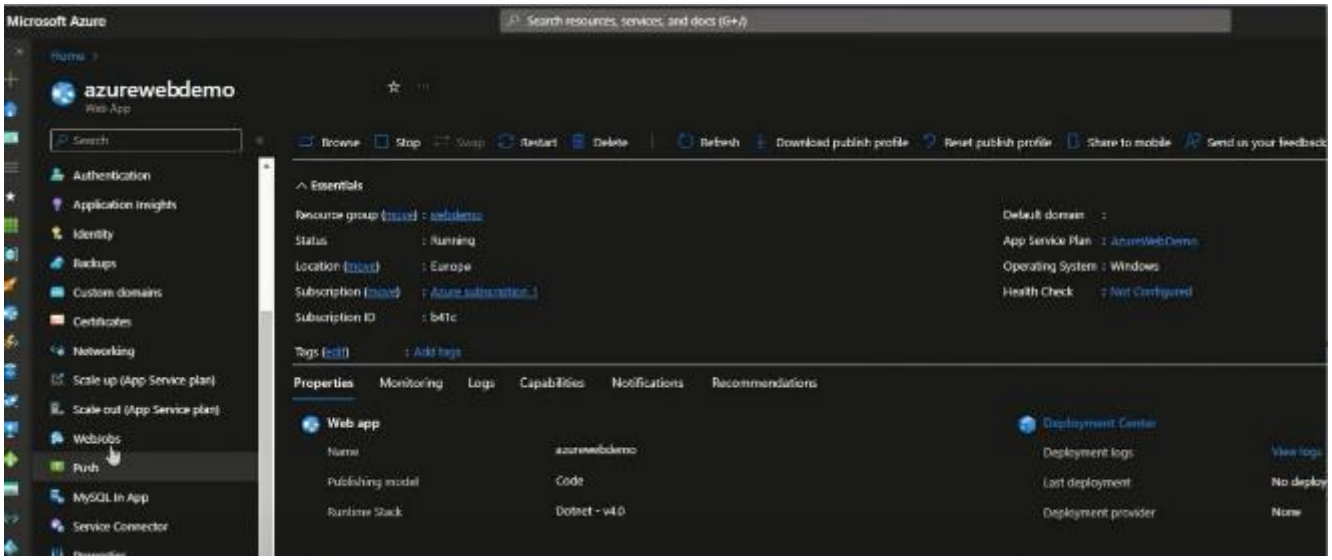


Рисунок 3.6 Створення плану обслуговування додатків

Далі треба налаштувати webjobs, інструмент, який дозволяє виконувати заплановані завдання та ті, що виконуються в фоновому режимі, включаючи консольні додатки, що дозволяє виконувати скрипти або виконувані файли без потреби окремого сервера чи інфраструктури (рис 3.7). Такі завдання можуть включати обробку даних, обслуговування файлів, або інші завдання, які потрібно виконувати періодично або безперервно.

Під час налаштування, треба вказати ім'я програми або процесу, що буде розміщуватись в хмарному середовищі, завантажити файл, в якому знаходиться програмна реалізація додатку, який повинен бути заархівованим в форматі .zip, обрати тип Continuous або Triggered, з яких перший це програми, що після

створення, виконуються у нескінченному циклі, тобто безперервно працюють, а другий це ті, що запускаються тільки вручну або за розкладом. Наступним параметром є Scale, який потрібен тільки для проограм типу Continuous, адже вказує на те, як часто програма буде перезапускатися. Останнім є параметр CRON Expression, що потрібен щоб запускати програму в певний час конкретного дня.

На рисунку 3.7 видно, що для застосунку, який обчислює гомоморфні криптосистеми, було вказана вказана налаштована конфігурація, щоб надалі коритсуватися додатком.

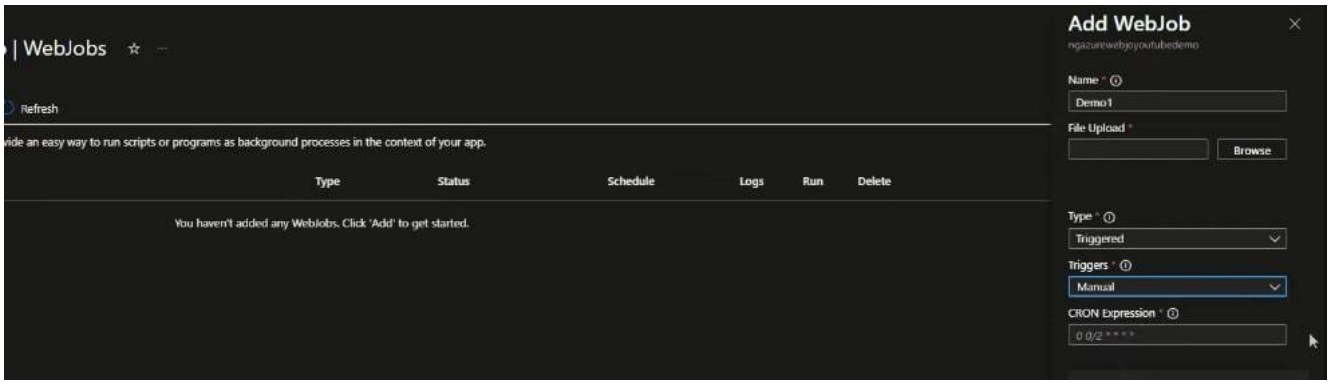


Рисунок 3.7 Конфігурація webjobs

Після налаштування, тепер можливо користуватися програмою в хмарному середовищі Azure (рис.3.8).

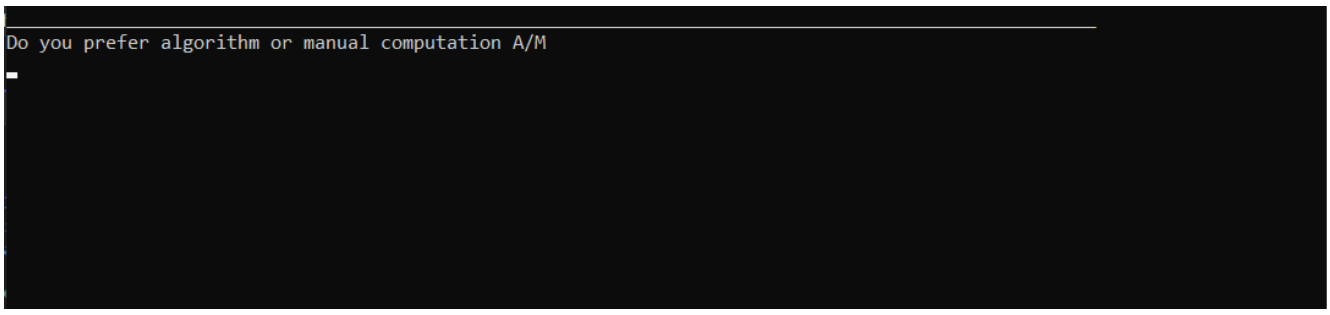


Рисунок 3.8 Запущена програма

3.3.2 Огляд програмної реалізації застосунку для обчислень

Другий застосунок, що обчислює шифротексти, приймає на вхід 2 попередньо зашифрованих повідомлення, над якими будуть потім здійснюватись операції, та публічний ключ, отриманий під час перетворення вхідного тексту. Слід зазначити, що обчислення будуть здійснюватися коректно лише в тому випадку, якщо під час шифрування, ці 2 повідомлення мають однакові значення простих чисел p та q, випадкові числа r та g можуть відрізнятися, бо вони не впливають на обчислення.

Використовуючи для додатку шифрування алгоритм Пайє, дає змогу здійснювати операції додавання та множення над зашифрованими повідомленнями, тому користувач може обрати яку операцію він хоче здійснити.

На рисунку 3.9, було реалізовано рівняння 2.1 та 2.2, що відповідають операціям додавання та множення гомоморфного шифротексту.

```
7   BigInteger AddEncryptedData(BigInteger cipher1,BigInteger cipher2, BigInteger n)
8   {
9       //виконує операцію додавання над шифротекстами
10      return (cipher1* cipher2)%(n*n);
11  }
    2 references
12  BigInteger MultEncryptedData(BigInteger cipher2, BigInteger plaintextOf1, BigInteger n)
13  {
14      // виконує операцію множення над шифротекстом 2 з повідомленням 1
15      return BigInteger.ModPow(cipher2, plaintextOf1, n*n);
16  }
```

Рисунок 3.9 Код для реалізації додавання та множення шифротекстів

Програму реалізовано так, щоб працювала в двух режимах:

1. Manual;
2. Algorithm.

Перший режим, функція якого зображена на рисунку 3.10, відповідає за те, що подальші обрахунки будуть здійснюватись будуть поступовими і користувач під час обчислень може вирішувати, які операції будуть наступними та які шифротексти в подальшому будуть використовуватись (рис. 3.11). В результаті, таке керування послідовністю операцій дає змогу користувачеві зберігати проміжні

результати обчислень для подальшого використання або аналізу та більшу гнучкість у виборі та управлінні процесом обчислень.

```
void manualComputation()
{
    while (true)
        try
        {
            setNumbers();
            break;
        }
        catch (Exception ex)
        {
            Console.WriteLine("Please type only numbers");
        }
    Console.WriteLine("Your two ciphertexts are");
    Console.WriteLine(c1 + " and " + c2);
    try
    {
        Console.WriteLine("Add or multiply A/M");
        question = Console.ReadLine();
        if (question[0].Equals('a') || question[0].Equals('A'))
            Console.WriteLine("result of addition of ciphers are:" + "\n" + AddEncryptedData(c1, c2, n));
        else if (question[0].Equals('m') || question[0].Equals('M'))
        {
            Console.WriteLine("Type your first plaintext");
            BigInteger text=BigInteger.Parse(Console.ReadLine());
            Console.WriteLine("result of multiplication of ciphers are:" + "\n" + MultEncryptedData(c2, text, n));
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Please type A or M");
    }
}
```

Рисунок 3.10 Функція виклику режиму Manual

```
Do you prefer algorithm or manual computation A/M
m
Type your first ciphertext
3896297
Type your second ciphertext
369865
Type public key
2021
Your two ciphertexts are
3896297 and 369865
Add or multiply A/M
a
result of addition of ciphers are:
2825198

Do you prefer algorithm or manual computation A/M
```

Рисунок 3.11 Приклад реалізації режиму Manual

В режимі Algorithm, налаштовано так, щоб користувач на самому початку обчислень вказував у вигляді команди, яку послідовність операцій треба виконувати і при потребі, вказував на шифротекст (рис. 3.12).

```
void algorithmComputation()
{
    byte steps = 0;
    // ввожу алгоритм послідовностей операцій
    Console.WriteLine("write your algorithm."+"\n"+ "Example 'n+ l+ l*'");
    question = Console.ReadLine();
    foreach (char c in question)
        if (c == ' ')
            steps++;
    string[] algorithm = new string[steps];
    algorithm = question.Split(" ");
    Console.WriteLine(algorithm[0] + " " + algorithm[1]+" " + algorithm[2]); ;
```

Рисунок 3.12 Отримання алгоритму послідовності операцій

Команда вводиться у форматі послідовності символів, кожен з яких відповідає за операцію, що потім буде здійнюватися, та над якими даними, наприклад можна вказати (рис.3.13):

$$n+\ \ l+\ \ l*, \tag{3.1}$$

де n – виклик функції введення нових даних, що в подальшому будуть обраховуватися;

l – виклик функції про використання останнього результату в обчисленнях;

+ - функція, що повертає результат адитивної операції над даними;

* - функція, що повертає результат операції множення над даними;

При використанні таких команд, користувач для обчислень, звісно, все одно повинен вводити другий шифротекст між операціями, але в порівнянні з режимом Manual, кількість дій, що повинен він зробити зменшується втричі. В результаті це пришвидшує обрахунки над шифротекстами, що в подальшому допомагає адаптувати цю послідовність операцій до його потреб та інтегрувати з іншими програмними засобами. Також якщо інтегрувати цей додаток з базами даних, що зберігають ці шифротексти, то вплив користувача можна ще зменшити до такої міри, що його участь у керуванні обчисленнями стає взагалі непотрібною.

```
    for (steps=0;steps<algorithm.Length ; steps++)
    {
        // в залежності від того яка послідовність операцій вказана
        // виконую наступні дії
        if (algorithm[steps][0].Equals('n'))
        {
            // втановлюю нові значення для шифротекстів
                setNumbers();
        }
        else if (algorithm[steps][0].Equals('l'))
        {
            // використовую попередній результат для обрахунків
            Console.WriteLine("Type your second text(plain or cipher)");
            c1 = lastresult;
            c2 = BigInteger.Parse(Console.ReadLine());
        }
        if (algorithm[steps][1].Equals('+'))
        {
            // викликаю функцію додавання над шифротекстами
            Console.WriteLine("Addition");
            lastresult = AddEncryptedData(c1, c2, n);
            Console.WriteLine("result of " + steps + " step of algorithm is " + lastresult);
        }
        else if (algorithm[steps][1].Equals('*'))
        {
            // викликаю функцію множення над шифротекстами
            Console.WriteLine("Multiply");
            lastresult = MultEncryptedData(c1, c2, n);
            Console.WriteLine("result of " + steps + " step of algorithm is " + lastresult);
        }
        Console.WriteLine("The last result is " + lastresult);
    }

}
```

Рисунок 3.13 Перевірка введених значень в режимі Algorithm

```
Do you prefer algorithm or manual computation A/M
a
write your algorithm.
Example 'n+ l+ l*'
n+ l+ l*
n+ l+ l*
Type your first ciphertext
3896297
Type your second ciphertext
369865
Type public key
2021
Addition
result of 0 step of algorithm is 2825198
The last result is 2825198
Type your second text(plain or cipher)
369865
Addition
result of 1 step of algorithm is 2979476
The last result is 2979476
Type your second text(plain or cipher)
369865
Multiply
result of 2 step of algorithm is 383824
The last result is 383824
```

Рисунок 3.14 Приклад реалізації режиму Algorithm

3.3.3 Аналіз працездатності другого застосунку та його кооперації з першим

Для перевірки чи працюють обчислення гомоморфних шифротекстів, що перетворені за допомогою алгоритму Пайє, було обрано два повідомлення значення, яких дорівнює 35 та 45, а прості числа для кожного будуть 43 та 47, інші числа згенеровані випадково. Після шифрування було отримано значення 1980271 та 2361935, що потім будуть використані для обчислень (рис.3.15).

```
ENCRYPTION
type your message
35
Process of creating public and private keys
try to type two prime numbers that bigger than 40
43 47
Do you want R to be random Y/N
y
your encrypted message is 1980271
ENCRYPTION
type your message
45
Process of creating public and private keys
try to type two prime numbers that bigger than 40
43 47
Do you want R to be random Y/N
y
your encrypted message is 1723279
```

Рисунок 3.15 Зашифрування повідомлень зі значеннями 35 та 45

Публічним ключем для обох повідомлень буде значення 2021.

Для обчислень вводемо цей публічний ключ та повідомлення, над ними будемо здійснювати як операцію додавання (рис. 3.16) так і множення (рис. 3.17), унікальністю якого в алгоритмі Пайє є те, що потрібно використати одне відкрите повідомлення для обчислень.

```
Do you prefer algorithm or manual computation A/M
m
Type your first ciphertext
1980271
Type your second ciphertext
1723279
Type public key
2021
Your two ciphertexts are
1980271 and 1723279
Add or multiply A/M
a
result of addition of ciphers are:
804227
```

Рисунок 3.16 Отримання результату додавання

```
Do you prefer algorithm or manual computation A/M
m
Type your first ciphertext
1980271
Type your second ciphertext
1723279
Type public key
2021
Your two ciphertexts are
1980271 and 1723279
Add or multiply A/M
m
Type your first plaintext
35
result of multiplication of ciphers are:
2197559
```

Рисунок 3.17 Отримання результату множення

Отримані значеннями додавання є 804227 та множення є 2197559. Тепер залишилось розшифрувати ці результати та аналізувати їх. В результаті обрахунку, що знаходиться на рисунку 3.18, було отримано коректні значення, адже 80 дійсно дорівнює результату додавання значень повідомлень 35 та 45, а в результаті, що отримали в рисунку 3.19 видно, що 1575 співпадає зі значенням , що можна отримати при добутку цих чисел.

```
Do you want to encrypt or decrypt (type E or D)
d
DECRYPTION
type your message
804227
804227
Process of creating public and private keys
try to type two prime numbers that bigger than 40
2021
your input was incorrect
try to type two prime numbers that bigger than 40
43 47
your decrypted message is 80
```

Рисунок 3.18 Дешифрування результатів додавання шифротекстів

```
Do you want to encrypt or decrypt (type E or D)
d
DECRYPTION
type your message
2197559
2197559
Process of creating public and private keys
try to type two prime numbers that bigger than 40
43 47
your decrypted message is 1575
```

Рисунок 3.19 Дешифрування результатів множення шифротекстів

Висновки до розділу 3

Розробка та реалізація обчислень гомоморфних криптосистем покращує системи захисту хмарних інфраструктур, адже допомагає обмежити можливості та ускладнити реалізацію ризиків зловмисникам, шляхом перетворення персональної інформації користувача в певні шифротексти та запровадження методів обчислень на стороні хмари, що приймають тільки значення зашированої інформації.

Було визначено, що для впровадження гомоморфного шифрування в хмарних обчисленнях, потрібно розробити та реалізувати 2 застосунки, перший для клієнта обчислень, другий для хмарного середовища.

Ці застосунки доповнюють один одного та створюють інтегровану систему обробки конфіденційної інформації, адже перший застосунок отримує дані від організації чи користувача й перетворює їх за допомогою алгоритму Пайє на шифротексти, а другий застосунок, реалізований на базі хмарної інфраструктури, отримує результати та обчислює їх згідно з налаштуваннями та потребами і після обробки відправляє їх на дешифрування назад.

Слід зазначити, що впровадження обчислень гомоморфних криптосистем мають недоліки, серед яких є ті, що пов’язані з часом на перетворення великих повідомлень та з використанням значних ресурсів на виконання процесів перетворення. Частину цих незручностей може прибрати використання хмарних платформ, що можуть ефективно адаптовуватись до потреб. Серед таких хмарних платформ є Azure, що забезпечує необхідну масштабованість та обчислювальні потужності для виконанння неохідних операцій гомоморфними шифротекстами, а також з можливістю інтеграції з наявними сервісами та інструментами, що підвищують безпеку та надійність інформаційної системи компанії від небажаних ризиків.

Ці застосунки можна покращити різними заходами, наприклад, інтеграцією з базами даних, що зберігають шифротексти, що дозволяють автоматизувати обчислення та мінімізувати вплив користувача, в результаті підвищуючи їх відмовостійкість та точность.

Організація цього підхіду буде особливо корисною для організацій, що працюють з персональними даними та потребують надійного захисту інформації, розміщених та обчислюваних у хмарі.

ВИСНОВОК

Впродовж виконання роботи було проаналізовано основні елементи впровадження інформаційної безпеки та вимоги до їх реалізації. Досліджено, методи та механізми, що використовуються для реалізації захисту інформаційної системи, результатом реалізації яких, є забезпечення конфіденційності, доступності та цілісності даних, що циркулюють в межах ІС певної компанії. Також, розглянуто проблему зберігання рівноваги між безпекою і витратами на її створенням й утримуванням.

Розлянуто хмарні обчислення як способу віддаленого здійснення обробки, зберігання даних, як методу передачі ризику виникнення інцидентів безпеки до хмарного провайдера. Було здійснено огляд ситуацій, в яких використання хмарних технологій призведе до ефективного функціонування та покращення захисту систем компаній. Розглянуто види хмарних обчислень, зокрема типи моделей надання послуг та моделей розгортання, кожна з яких, була розглянута та описана зважаючи на цілі їх використання, били представлені їх основні переваги та недоліки.

Було проаналізовано змістовний перелік загроз хмарного середовища та вплив недбалості користувачів, особливо при створенні паролів, на успішність реалізації кібератак на систему. Для узагальнення та ретельного аналізу всіх загроз, що можуть бути реалізовані на хмарну інфраструктуру, була розглянута матриця MITRE ATT&CK, як для загальної хмарної системи, так і окремо для IaaS та SaaS.

Як метод забезпечення конфіденційності даних клієнта, при реалізації загроз на хмару, було досліджено шифрування, його ключові аспекти та вплив на забезпечення конфіденційності та цілісності даних. Було викладено різницю між симетричними та асиметричними алгоритмами шифрування, розглянуто процеси, в яких вони використовуються та моменти, де наявні ці два види шифрування.

Визначено та розглянуто, які алгоритми шифрування можна вважати гомоморфними, вказані їх особливості, плюси та мінуси впровадження в ситеми обробки даних. Досліджено різницю між частково та повністю гомоморфним

шифруванням та складність їх реалізації. Описаний процес підготовки зашифрованих даних для відправлення до хмари. Розглянута послідовність перетворення певних повідомлень, згідно з алгоритмом Пайє, та їх обчислень у зашифрованому вигляді.

У практичній частині роботи, було з’ясовано алгоритм здійснення обчислень над шифротекстами в хмарному середовищі. Реалізовані застосунки, які зашифровують повідомлення та обчислюють отримані криптосистеми, згідно з конфігурацією користувача. Впровадження таких застосунків продемонструвало їх ефективність у забезпеченні конфіденційності даних під час обробки у хмарній платформі, навіть при копроментації системи зловмисником.

Загалом, робота відобразила критичну важливість в імплементації засобів захисту інформаційних систем компаній. Показала, важливість використання гомоморфного шифрування під час хмарних обрахунків. Реалізуючи системи обчислення гомоморфних криптосистем, компанії можуть покращити захист даних, особливо, від їх незаконного перегляду. Додатки, що були створенні та впроваджені під час виконання роботи, можуть в подальшому бути покращеними, для зручного використання користувачем, та інтегрованими з іншими інструментами та заходами безпеки, які реалізовані в хмарі, що в результаті, покращить їх кооперацію, зменшуючи занепокоїння компаній щодо захищеності під час обробки в хмарному середовищі своїх персональних та корпоративних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How push to the cloud helped Ukrainian bank keep faith with customers amid war [Електронний ресурс]. – Режим доступу: https://www.nextgov.com/modernization/2023/11/how-push-cloud-helped-ukrainian-bank-keep-faith-customers-amid-war/392375/

2. CrowdStrike Global Threat Report 2024 [Електронний ресурс]. – Режим доступу: https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf

3. What is Homomorphic Encryption? [Електронний ресурс]. – Режим доступу: https://www.keyfactor.com/blog/what-is-homomorphic-encryption/

4. Про основні засади забезпечення кібербезпеки України: Закон України від 5 жовтня 2017 р. № 2163-VIII. Офіц. вид. Відомості Верховної Ради України. 2017. № 45. Ст. 403.

5. Про Положення про технічний захист інформації в Україні: Указ Президента України від 27 вересня 1999 р. № 1229/99. Офіц. вид. Відомості Верховної Ради України. 1999. № 40. Ст. 365.

6. Charles P. Pfleeger, Shari Lawrence Pfleeger. "Security in Computing." Prentice Hall, 2015. – 944 с.

7. Eric Cole. "Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization." Syngress, 2012. – 320 с.

8. Niels Ferguson, Bruce Schneier, Tadayoshi Kohno. "Cryptography Engineering: Design Principles and Practical Applications." Wiley, 2010. – 384 с.

9. Michael Howard, David LeBlanc, John Viega. "24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them." McGraw-Hill, 2009. – 528 с.

10. Dorian Cougias, E. L. Heiberger, Karsten Koop. "The Backup Book: Disaster Recovery from Desktop to Data Center." Schaser-Vartan Books, 2003. – 550 с.

11. Andy Jones, Debi Ashenden. "Risk Management for Computer Security: Protecting Your Network & Information Assets." Butterworth-Heinemann, 2005. – 360 с.

12. Tim Mather, Subra Kumaraswamy, Shahed Latif. "Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance." O'Reilly Media, 2009. – 338 с.

13. Thomas Erl, Ricardo Puttini, Zaigham Mahmood. "Cloud Computing: Concepts, Technology & Architecture." Prentice Hall, 2013. – 528 с.

14. Rountree, Derrick, Castrillo, Ileana. "The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in Theory and Practice." Syngress, 2013. – 172 с.

15. What is Cloud Computing? [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchcloudcomputing/definition/cloud-computing

16. Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi. "Mastering Cloud Computing: Foundations and Applications Programming." Morgan Kaufmann, 2013. – 488 с.

17. Tim Mather, Subra Kumaraswamy, Shahed Latif. "Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance." O'Reilly Media, 2009. – 338 с.

18. Krutz, Ronald L., Vines, Russell Dean. "Cloud Security: A Comprehensive Guide to Secure Cloud Computing." Wiley, 2010. – 384 с.

19. William Stallings. "Effective Cybersecurity: A Guide to Using Best Practices and Standards." Addison-Wesley Professional, 2018. – 800 с.

20. What is Cloud Computing? [Електронний ресурс]. – Режим доступу: https://www.citrix.com/glossary/what-is-cloud-computing.html

21. What is Cloud Computing? Everything You Need to Know About the Cloud [Електронний ресурс]. – Режим доступу: https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/

22. Chris Harding, John Gøtze. "Cloud Computing: The Basics of Multi-Cloud Environments." The Open Group Press, 2019. – 350 с.

23. Cloud Computing Trends: Flexera 2024 State of the Cloud Report [Електронний ресурс]. – Режим доступу: https://www.flexera.com/blog/cloud/cloud-computing-trends-flexera-2024-state-of-the-cloud-report/

24. 7 Types of Cloud Computing Structures [Електронний ресурс]. – Режим доступу: https://www.uniprint.net/en/7-types-cloud-computing-structures/

25. What is IaaS? [Електронний ресурс]. – Режим доступу: https://www.oracle.com/ua/cloud/what-is-iaas/

26. What is PaaS? [Електронний ресурс]. – Режим доступу: https://platform.sh/blog/paas/

27. What is PaaS? [Електронний ресурс]. – Режим доступу: https://www.ibm.com/topics/paas

28. What is SaaS? [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/saas/

29. SaaS vs PaaS vs IaaS: What’s the Difference and How to Choose? [Електронний ресурс]. – Режим доступу: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/

30. Public Cloud Market [Електронний ресурс]. – Режим доступу: https://www.statista.com/outlook/tmo/public-cloud/worldwide

31. Threat Horizons Report H1 2024 [Електронний ресурс]. – Режим доступу: https://services.google.com/fh/files/misc/threat_horizons_report_h12024.pdf

32. MITRE ATT&CK® Cloud Matrix [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/matrices/enterprise/cloud/

33. Cloud Security [Електронний ресурс]. – Режим доступу: https://www.ibm.com/topics/cloud-security

34. Livia Maria BRUMĂ. Data Security Methods in Cloud Computing. The Bucharest University of Economic Studies, Bucharest, Romania.

35. Kire Jakimoski. "Security Techniques for Data Protection in Cloud Computing." International Journal of Grid and Distributed Computing, 9(1):49-56.

36. Jay Heiser, Neil MacDonald. "Security Considerations and Best Practices for Securing Public Cloud Deployments." Gartner, 2020.

37. Ross J. Anderson. "Security Engineering: A Guide to Building Dependable Distributed Systems." Wiley, 2020. – 1232 с.

38. What is Encryption? [Електронний ресурс]. – Режим доступу: https://cloud.google.com/learn/what-is-encryption

39. Bruce Schneier. "Applied Cryptography: Protocols, Algorithms, and Source Code in C." Wiley, 2015. – 937 с.

40. Michael E. Whitman, Herbert J. Mattord. "The Dark Side of Encryption: Ransomware Threats and Mitigation Strategies." Kennesaw State University, 2024. – 350 с.

41. Symmetric Encryption vs Asymmetric Encryption [Електронний ресурс]. – Режим доступу: https://deviceauthority.com/symmetric-encryption-vs-asymmetric-encryption

42. What is a Block Cipher? [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchsecurity/definition/block-cipher#

43. Explain Working of HTTPS [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/explain-working-of-https/

44. Cihan H. Dagli, Monique Ogburna, Claude Turner, Pushkar Dahal. "Homomorphic Encryption."

45. What is Homomorphic Encryption [Електронний ресурс]. – Режим доступу: https://digitalprivacy.ieee.org/publications/topics/what-is-homomorphic-encryption

46. V. Biksham, D. Vasumathi. "Homomorphic Encryption Techniques for Securing Data in Cloud Computing: A Survey."

47. Zhiyong Hong, Zhili Zhang, Pu Duan. "Research on Homomorphic Encryption Techniques."

48. Andrew Troelsen, Philip Japikse. "Pro C# 7: With .NET and .NET Core." Apress, 2017. – 1375 c.

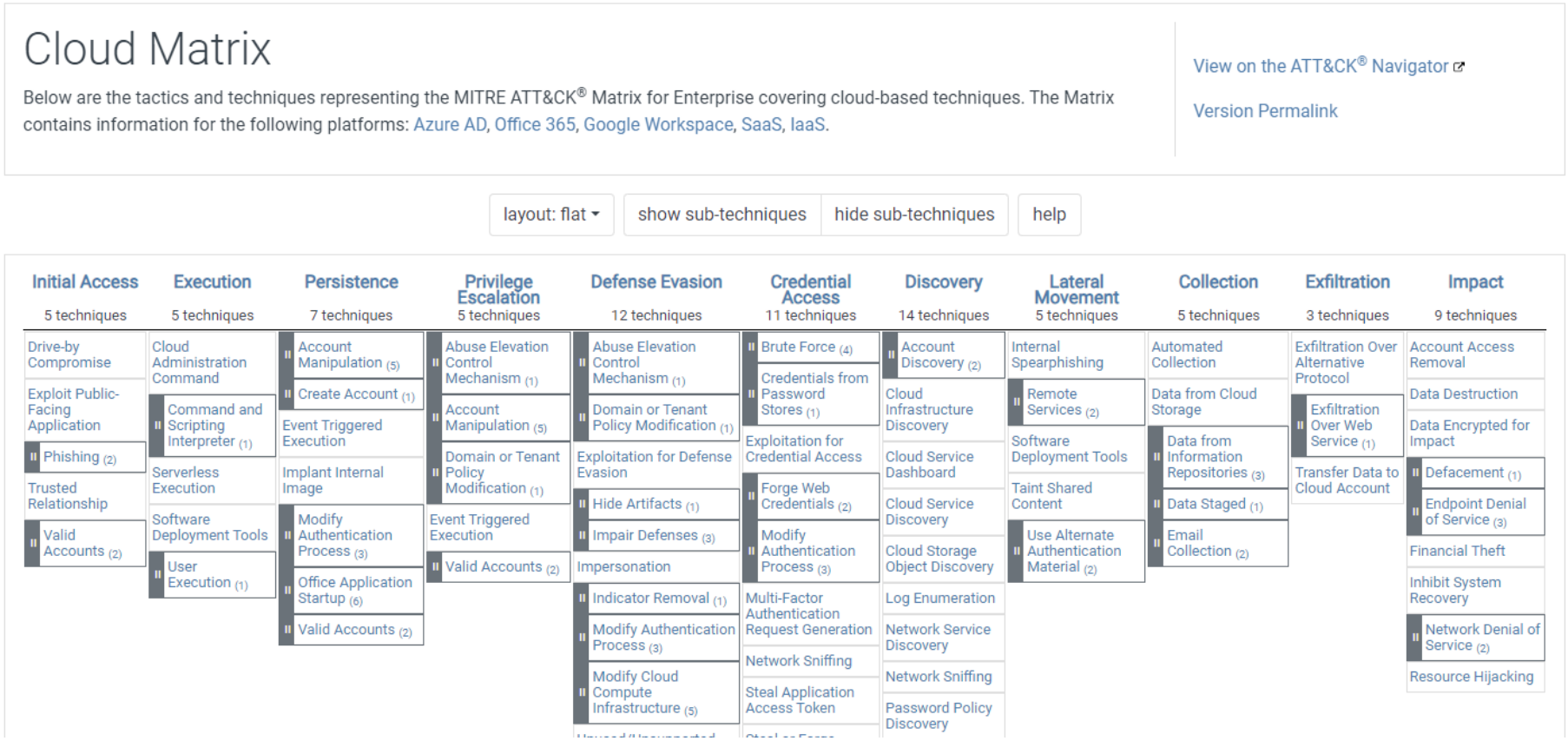
49. What is Azure? [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure

50. Azure Basics: IaaS vs PaaS vs SaaS on Azure [Електронний ресурс]. – Режим доступу: https://sonatafy.com/azure-basics-iaas-vs-paas-vs-saas-on-azure/

ДОДАТКИ

ДОДАТОК А

Матриця атак, що можуть бути імплементовані відносно будь якого хмарного середовища



ДОДАТОК Б

Матриця атак, що можуть бути реалізовані відносно IaaS

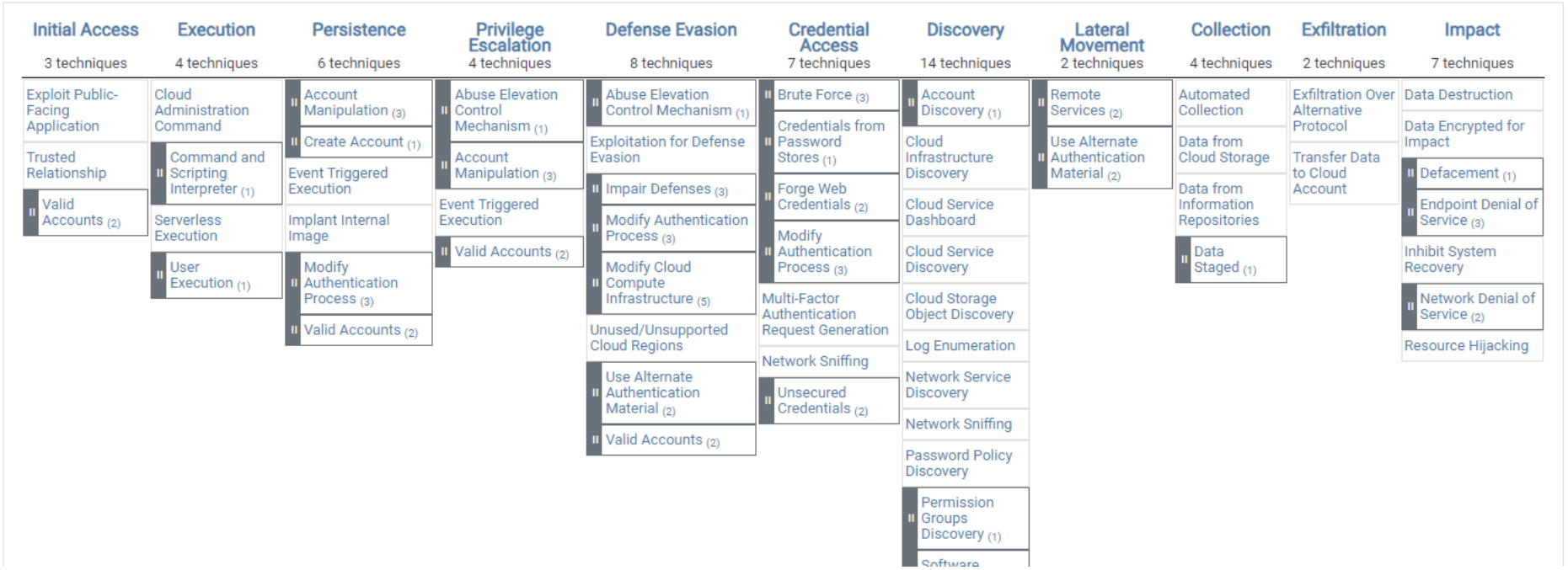
IaaS Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® Matrix for Enterprise covering cloud-based techniques. The Matrix contains information for the IaaS platform.

View on the ATT&CK® Navigator

Version Permalink

layout: flat | show sub-techniques | hide sub-techniques | help



ДОДАТОК В

Матриця атак, що можуть бути реалізовані відносно SaaS

SaaS Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® Matrix for Enterprise covering cloud-based techniques. The Matrix contains information for the SaaS platform.

View on the ATT&CK® Navigator

Version Permalink

layout: flat | show sub-techniques | hide sub-techniques | help

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Impact
4 techniques	2 techniques	5 techniques	4 techniques	6 techniques	7 techniques	4 techniques	4 techniques	3 techniques	3 techniques	4 techniques
Drive-by Compromise	Serverless Execution	Account Manipulation (3)	Account Manipulation (3)	Domain or Tenant Policy Modification (1)	Brute Force (3)	Account Discovery (1)	Internal Spearphishing	Automated Collection	Exfiltration Over Alternative Protocol	Account Access Removal
Phishing (2)	Software Deployment Tools	Create Account (1)	Domain or Tenant Policy Modification (1)	Exploitation for Defense Evasion	Forge Web Credentials (2)	Cloud Service Dashboard	Software Deployment Tools	Data from Cloud Storage	Exfiltration Over Web Service (1)	Endpoint Denial of Service (3)
Trusted Relationship		Event Triggered Execution	Event Triggered Execution	Impersonation	Modify Authentication Process (3)	Cloud Service Discovery	Taint Shared Content	Data from Information Repositories (2)	Transfer Data to Cloud Account	Financial Theft
Valid Accounts (2)		Modify Authentication Process (3)	Valid Accounts (2)	Modify Authentication Process (3)	Multi-Factor Authentication Request Generation	Permission Groups Discovery (1)	Use Alternate Authentication Material (2)			Network Denial of Service (2)
		Valid Accounts (2)		Use Alternate Authentication Material (2)	Steal Application Access Token					
				Valid Accounts (2)	Steal Web Session Cookie					
					Unsecured Credentials (1)					

ДОДАТОК Д

Програмний код додатку, реалізованого в клієнта

```
using System;
using System.Numerics;
public class Pillier
{
    private uint p, q, r;
    private ulong n, lya,g,phi;
    public BigInteger resultat,nsquare,u;
    Random random = new Random();
    private bool IsPrime(uint number)
    {
        for (int i = 2; i < number; i++)
            if (number % i == 0)
                return false;
        return true;
    }
    private uint[] getNumbers(bool state)
    {
        Console.WriteLine("Process of creating public and private keys");
        uint[] mnumb = new uint[4]; byte step = 0;
        string PQ = "", R = "";
        string question;
        while (step < 2)
        {
            string[] numbers = new string[4];
            if (step == 0)
            {
                try
                {
```

```
            Console.WriteLine("try to type two prime numbers that bigger than 40");
            PQ = Console.ReadLine();
            numbers = PQ.Split(" ");
            mnumb[0] = UInt32.Parse(numbers[0]);
            mnumb[1] = UInt32.Parse(numbers[1]);
        }
        catch (Exception ex)
        {
            Console.WriteLine("your input was incorrect");
        }
        // compare whether these numbers are prime and correct

        if (IsCorrectPrimeNumbers(mnumb[0], mnumb[1]))
        {
            step++;
            if (!state) break;
            Console.WriteLine("Do you want R to be random Y/N");
            question = Console.ReadLine();
            if (question[0].Equals('Y') || question[0].Equals('y'))
            {
                mnumb[2] = (uint)random.Next(20, 300);

                break;
            }
        }
    }
    else if (step == 1)
    {
        try
        {
```

```
                Console.WriteLine("type two numbers that bigger than 20");
                R = Console.ReadLine();
                numbers = R.Split(" ");
                mnumb[2] = UInt32.Parse(numbers[0]);
                if (mnumb[2] > 20)
                    step++;
            }
            catch (Exception ex)
            {
                Console.WriteLine("your input was incorrect");
            }
        }
    }
    return mnumb;
}
private void setNumbers(bool state)
{
    uint[] mnub = getNumbers(state);
    p = mnub[0];
    q = mnub[1];
    if(state)
        r = mnub[2];
}
private BigInteger L(BigInteger x)
{
    return (x-1)/n;
}
static BigInteger ModInverse(BigInteger a, BigInteger m)
{
    BigInteger m0 = m;
```

```
    BigInteger y = 0, x = 1;

    if (m == 1)
        return 0;

    while (a > 1)
    {
        // q is quotient
        BigInteger q = a / m;
        BigInteger t = m;

        // m is remainder now, process same as Euclid's algo
        m = a % m;
        a = t;
        t = y;

        // Update x and y
        y = x - q * y;
        x = t;
    }

    // Make x positive
    if (x < 0)
        x += m0;

    return x;
}
private uint MCD(uint number1, uint number2)
{
    while (number2 != 0)
```

```
        {
            uint temp = number2;
            number2 = number1 % number2;
            number1 = temp;
        }
        return number1;
    }
    private long LCM(uint number1, uint number2)
    {
        return Math.Abs(number1 * number2) / MCD(number1, number2);
    }
    private bool IsCorrectPrimeNumbers(uint number1, uint number2)
    {
        // перевірка чи числа прості
        if (!IsPrime(number1) || !IsPrime(number2))
            return false;
        // чи вони менші за 40
        else if (number1 < 40 || number2 < 40)
            return false;
        // перевірка чи виконується умова
        else if (MCD(number1, number2) != 1)
            return false;
        return true;
    }
    private void generateKeys(bool state)
    {
        // генерація простих чисел
        setNumbers(state);
        // надання значень змінним, які будуть використовуються для подальших
обчислень
```

```
        n = p * q;
        nsquare = n * n;
        phi = (ulong)((p - 1)*(q - 1));
        g = n + 1;
        lya = phi * 1;
        u = ModInverse(L(BigInteger.ModPow(g,lya,nsquare)),n);

    }
    private void checkGenerationOfKeys( BigInteger m,bool state)
    {
        while (m > n)
        {

            // set numbers that will be creating public and private keys
            generateKeys(state);
            if (!state)
                break;
            if (m > n)
                Console.WriteLine("yours prime numbers are smaller than message, choose bigger numbers");
            else if (r > n || MCD(r, (uint)n) != 1)
                Console.WriteLine("yours prime numbers are smaller than random one, choose bigger numbers");
        }
    }
    public void encryption(BigInteger m)
    {
        p = 0; q=0;r=0;g=0;lya=0;n = 0;u = 0;
        resultat = 0;
        checkGenerationOfKeys(m,true);
```

```
        resultat = (BigInteger.ModPow(g,m,nsquare)* BigInteger.ModPow(r, n, nsquare))
% nsquare;
        Console.WriteLine("your encrypted message is "+ resultat);



    }
    public void decryption(BigInteger c)
    {
        // очищую попередні значення змінних
        p = 0; q=0;r=0;g=0;lya=0;n = 0;u = 0;
        resultat = 0;

        checkGenerationOfKeys(c,false);
        resultat = (BigInteger)(L(BigInteger.ModPow(c,lya,nsquare))*u) % n;
        Console.WriteLine("your decrypted message is " + resultat);

    }
    public Pillier()
    {
        string state = "";
        while (true)
        {
            Console.WriteLine();

Console.WriteLine("_____
_____");
            Console.WriteLine("Do you want to encrypt or decrypt (type E or D)");
            state=Console.ReadLine();
            try
            {
```

```
        if (state[0].Equals('e') || state[0].Equals('E'))
        {

            Console.WriteLine("ENCRYPTION");
            Console.WriteLine("type your message");
            state = Console.ReadLine();
            encryption(ulong.Parse(state));
        }
        else if (state[0].Equals('d') || state[0].Equals('D'))
        {
            Console.WriteLine("DECRYPTION");
            Console.WriteLine("type your message");
            state = Console.ReadLine();
            Console.WriteLine(state);
            decryption(ulong.Parse(state));
        }
        else
            Console.WriteLine("Incorrect input");
        Console.WriteLine("Do you want to end this process Y/N");
        state = Console.ReadLine();
    }
    catch
    {
        Console.WriteLine("Incorrect input");
    }
    try{
        if (state[0].Equals('y') || state[0].Equals('Y'))
            break;
    }catch(Exception ex) {}
}}}
```

ДОДАТОК Е

Програмний код додатку, реалізованого в хмарному середовищі

```
using System.Numerics;
using System.Runtime.CompilerServices;
BigInteger c1=0, c2=0,n=0;
string question="";
BigInteger lastresult=0;
BigInteger AddEncryptedData(BigInteger cipher1,BigInteger cipher2, BigInteger n)
{
   //виконує операцію додавання над шифротекстами
   return (cipher1* cipher2)%(n*n);
}
BigInteger MultEncryptedData(BigInteger cipher2, BigInteger plaintextOf1, BigInteger
n)
{
   // виконує операцію множення над шифротекстом 2 з повідомленням 1
   return BigInteger.ModPow(cipher2, plaintextOf1, n*n);
}
void setNumbers()
{
   Console.WriteLine("Type your first ciphertext");
   c1 = BigInteger.Parse(Console.ReadLine());
   Console.WriteLine("Type your second ciphertext");
   c2 = BigInteger.Parse(Console.ReadLine());
   Console.WriteLine("Type public key");
   n = BigInteger.Parse(Console.ReadLine());
}
void manualComputation()
{
   while (true)
```

```
        try
        {
            setNumbers();
            break;
        }
        catch (Exception ex)
        {
            Console.WriteLine("Please type only numbers");
        }
    Console.WriteLine("Your two ciphertexts are");
    Console.WriteLine(c1 + " and " + c2);
    try
    {
        Console.WriteLine("Add or multiply A/M");
        question = Console.ReadLine();
        if (question[0].Equals('a') || question[0].Equals('A'))
            Console.WriteLine("result of addition of ciphers are:" + "\n" +
AddEncryptedData(c1, c2, n));
        else if (question[0].Equals('m') || question[0].Equals('M'))
        {
            Console.WriteLine("Type your first plaintext");
            BigInteger text=BigInteger.Parse(Console.ReadLine());
            Console.WriteLine("result of multiplication of ciphers are:" + "\n" +
MultEncryptedData(c2, text, n));
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Please type A or M");
    }
```

```
}
void algorithmComputation()
{
    byte steps = 0;
    // ввожу алгоритм послідовностей операцій
    Console.WriteLine("write your algorithm."+"\n"+ "Example 'n+ l+ l*'");
    question = Console.ReadLine();
    foreach (char c in question)
        if (c == ' ')
            steps++;
    string[] algorithm = new string[steps];
    algorithm = question.Split(" ");
    Console.WriteLine(algorithm[0] + " " + algorithm[1]+" " + algorithm[2]); ;
    for (steps=0;steps<algorithm.Length ; steps++)
    {
        // в залежності від того яка послідовність операцій вказана
        // виконую наступні дії
        if (algorithm[steps][0].Equals('n'))
        {
            // втановлюю нові значення для шифротекстів
                setNumbers();
        }
        else if (algorithm[steps][0].Equals('l'))
        {
            // використовую попередній результат для обрахунків
            Console.WriteLine("Type your second text(plain or cipher)");
            c1 = lastresult;
            c2 = BigInteger.Parse(Console.ReadLine());
        }
        if (algorithm[steps][1].Equals('+'))
```

```
        {
            // викликаю функцію додавання над шифротекстами
            Console.WriteLine("Addition");
            lastresult = AddEncryptedData(c1, c2, n);
            Console.WriteLine("result of " + steps + " step of algorithm is " + lastresult);
        }
        else if (algorithm[steps][1].Equals('*'))
        {
            // викликаю функцію множення над шифротекстами
            Console.WriteLine("Multiply");
            lastresult = MultEncryptedData(c1, c2, n);
            Console.WriteLine("result of " + steps + " step of algorithm is " + lastresult);
        }
        Console.WriteLine("The last result is " + lastresult);
    }
}
while (true) {

Console.WriteLine("_____
_____");

    Console.WriteLine("Do you prefer algorithm or manual computation A/M");
    question = Console.ReadLine();
    if (question[0].Equals('m') || question[0].Equals('M'))
        manualComputation();
    else if (question[0].Equals('a') || question[0].Equals('A'))
        algorithmComputation();
}
```

Короткий звіт подібності

UNICHECK
by Turnitin

Ім'я користувача:
Комп'ютерної математики та інформаційної безпеки...

ID перевірки:
1016359969

Дата перевірки:
14.06.2024 12:06:14 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
14.06.2024 15:00:57 EEST

ID користувача:
100005746

Назва документа: **Диплом_Чикильов**

Кількість сторінок: **59** Кількість слів: **10785** Кількість символів: **83038** Розмір файлу: **1.77 MB** ID файлу: **1016164676**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

0.81%
Схожість

Найбільша схожість: **0.22%** з джерелом з Бібліотеки (ID файлу: **1015719047**)

0.55% Джерела з Інтернету	130	Сторінка 61
0.71% Джерела з Бібліотеки	111	Сторінка 61

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	13
Підозріле форматування	9 сторінок