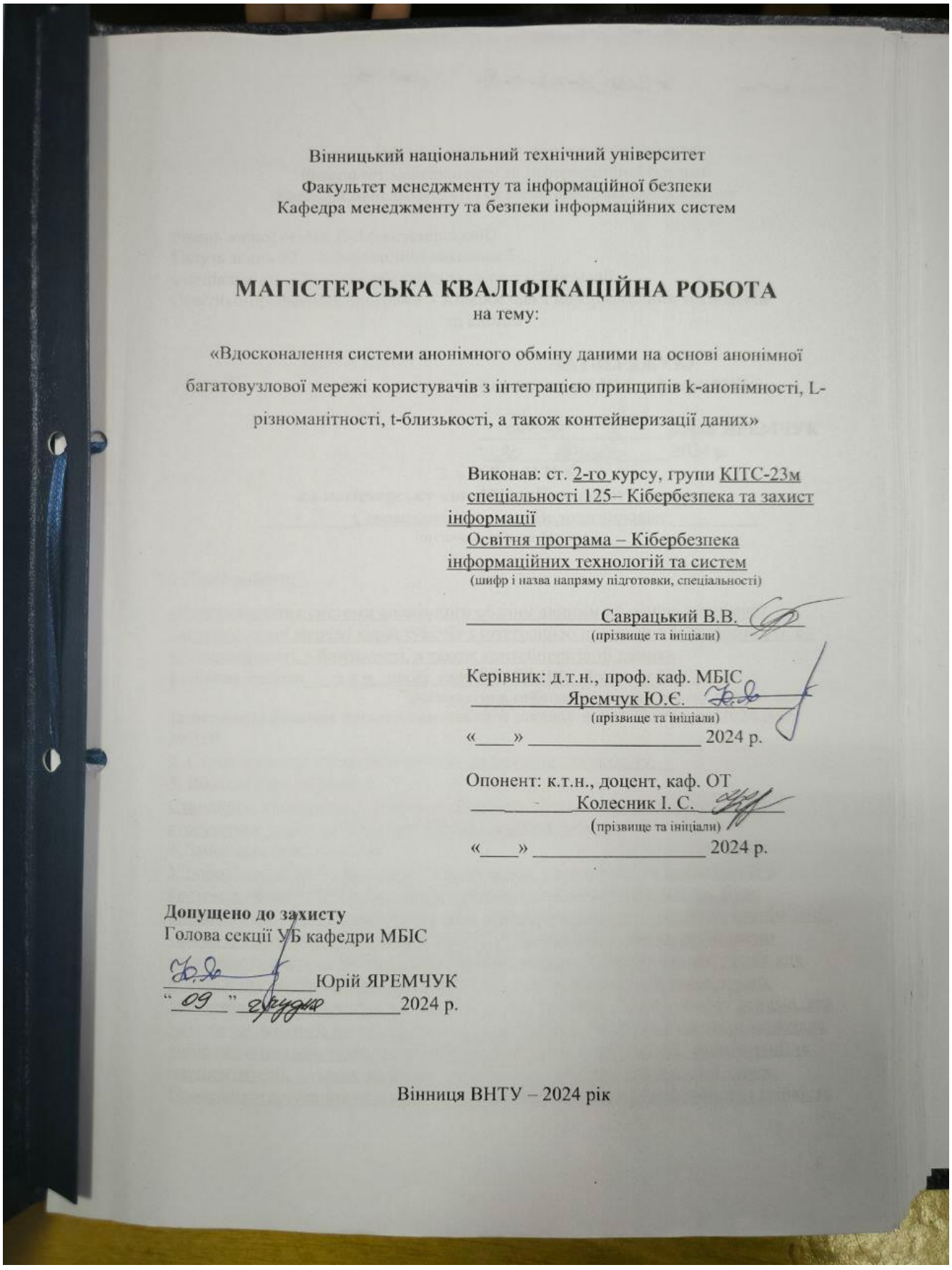


Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних»

Виконав: ст. 2-го курсу, групи КІТС-23м
спеціальності 125– Кібербезпека та захист інформації
Освітня програма – Кібербезпека інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

_____ Саврацький В.В.
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. МБІС
_____ Яремчук Ю.Є.
(прізвище та ініціали)
«____» _____ 2024 р.

Опонент: к.т.н., доцент, каф. ОТ
_____ Колесник І. С.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

_____ Юрій ЯРЕМЧУК
"09" грудня 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**
"20" вересня 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Саврацький Вячеслав Володимирович
(прізвище, ім`я, по-батькові)

1. Тема роботи:

«Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних»
Керівник роботи_: д.т.н., проф. каф. МБІС МБІС Яремчук Ю.Є.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "17" вересня 2024 року № 310
2. Строк подання студентом роботи_за тиждень до захисту.__
3. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи.
4. Зміст текстової частини:
У роботі досліджено проблему забезпечення анонімності та приватності в системах обміну даними в умовах сучасних інформаційних мереж. Було проаналізовано актуальність питання, визначено ключові принципи анонімізації, такі як k-анонімність, l-різноманітність і t-близькість, а також досліджено можливості використання багатовузлових мереж і контейнеризації даних для підвищення рівня захисту інформації. На основі проведених теоретичних досліджень було створено вдосконалену систему анонімного обміну даними, яка забезпечує високий рівень приватності та безпеки. Реалізація системи включала вибір оптимальних технологій, розробку модулів шифрування, анонімізації та маршрутизації, а також впровадження контейнеризації для ізоляції даних. Проведене тестування підтвердило функціональність, продуктивність і стійкість

системи до зовнішніх загроз, що робить її придатною для використання в у
підвищених вимог до конфіденційності
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових
креслень)
У другому розділі магістерської кваліфікаційної роботи наведено 3 рисунків, у
третьому розділі – 10 рисунків

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
II	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
III	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
Економічна частина			
IV	Бурєннікова Н. В. д.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 20 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3.	Розробка роботи	16.10.2024	26.10.2024	
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7.	Захист магістерської кваліфікаційної роботи	10.12.2024	17.12.2024	

Студент _____ Савранський В.В.
(підпис)

Керівник роботи _____ Яремчук Ю.Є.
(підпис)

АНОТАЦІЯ

УДК 004.56.5

Саврацький В.В. Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека та захист інформації», освітня програма «Кібербезпека інформаційних технологій та систем». Вінниця: ВНТУ, 2024. – 110 с.

Укр. мовою. Бібліогр.: 40 назв; рис.: 6; табл. 5.

Ключові слова: кібезбезпека, контейнеризація даних, анонімність

Магістерська робота присвячена вдосконаленню системи анонімного обміну даними на основі багатовузлової мережі користувачів із застосуванням сучасних методів анонімізації та захисту інформації. У роботі розглянуто актуальність забезпечення анонімності та приватності в умовах зростання обсягів даних і ризиків їх несанкціонованого доступу. Запропоновано інтеграцію принципів k-анонімності, l-різноманітності та t-близькості для забезпечення захисту даних, а також впроваджено контейнеризацію для підвищення ізоляції та безпеки інформації.

Результатом дослідження є розробка вдосконаленої системи, яка поєднує багатошарове шифрування, динамічну маршрутизацію через багатовузлову мережу та сучасні методи анонімізації. У роботі детально описано архітектуру системи, її функціональні компоненти, алгоритми обробки даних і забезпечення приватності. Проведено тестування системи, яке підтвердило її функціональність, стійкість до атак і відповідність сучасним вимогам безпеки.

Розроблена система має практичну цінність і може бути використана в умовах підвищених вимог до конфіденційності, наприклад, у сфері медицини, фінансів чи державного управління. Отримані результати створюють основу для подальшого вдосконалення системи, адаптації її до нових умов і підвищення рівня захисту інформації.

ABSTRACT

UDC 004.56.5

Savratskyi V.V. Master's qualification work in specialty 125 - “Cybersecurity and Information Protection”, educational program “Cybersecurity of Information Technologies and Systems”. Vinnytsia: VNTU, 2024. - 110 p.

In Ukrainian. Bibliography: 40 titles; Figures: 6; Table 5.

Keywords: cybersecurity, data containerization, anonymity

The master's thesis is devoted to improving the system of anonymous data exchange based on a multi-node user network using modern methods of anonymization and information protection. The paper considers the relevance of ensuring anonymity and privacy in the face of growing data volumes and the risks of unauthorized access. The integration of the principles of k-anonymity, l-diversity and t-closeness is proposed to ensure data protection, and containerization is introduced to increase the isolation and security of information.

The result of the study is the development of an advanced system that combines multi-layer encryption, dynamic routing through a multi-node network, and modern anonymization methods. The paper describes in detail the system architecture, its functional components, data processing and privacy algorithms. The system was tested, which confirmed its functionality, resistance to attacks and compliance with modern security requirements.

The developed system is of practical value and can be used in environments with increased requirements for confidentiality, such as in medicine, finance, or public administration. The results obtained create a basis for further improvement of the system, its adaptation to new conditions and an increase in the level of information protection.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі, що стрімко розвивається завдяки впровадженню цифрових технологій, питання забезпечення анонімності та приватності в інформаційних мережах набувають особливого значення. Зростання кількості онлайн-сервісів, мобільних додатків та IoT-пристроїв супроводжується дедалі більшими ризиками витоку конфіденційної інформації, її несанкціонованого використання або деканонімізації. Це особливо критично для сфер охорони здоров'я, фінансів, державного управління, де забезпечення захисту даних є обов'язковою умовою функціонування. Саме тому вдосконалення методів захисту інформації, зокрема анонімного обміну даними, є актуальною і важливою проблемою.

Метою дослідження є розробка вдосконаленої системи анонімного обміну даними на основі багатовузлової мережі користувачів із впровадженням сучасних принципів анонімізації, таких як k-анонімність, l-різноманітність і t-близькість, а також контейнеризації даних для ізоляції та захисту інформації. Досягнення цієї мети передбачає вирішення ряду задач.

Задачами дослідження є: аналіз існуючих підходів до анонімізації даних та забезпечення приватності в інформаційних мережах; розробка вдосконаленої моделі системи анонімного обміну даними з багатовузловою маршрутизацією; інтеграція принципів k-анонімності, l-різноманітності та t-близькості для підвищення рівня приватності; реалізація контейнеризації даних для забезпечення ізоляції та безпеки інформації; проведення тестування системи з метою перевірки її продуктивності, функціональності та стійкості до зовнішніх загроз.

Об’єктом дослідження є процес забезпечення анонімності та приватності в системах обміну даними, а предметом — розробка та вдосконалення методів анонімізації та захисту інформації у багатовузлових мережах з використанням сучасних принципів анонімізації та технологій контейнеризації.

Новизна роботи полягає у запропонованому підході, який об’єднує багатошарове шифрування, динамічну маршрутизацію через багатовузлові мережі

та інтеграцію принципів анонімізації k-анонімності, l-різноманітності та t-близькості. Використання контейнеризації як елемента системи дозволяє ізолювати дані та забезпечити додатковий рівень безпеки навіть у разі компрометації окремих вузлів мережі.

Практична цінність роботи полягає у можливості застосування розробленої системи в умовах підвищених вимог до захисту даних, таких як передача медичних, фінансових чи інших конфіденційних записів, а також у проєктах, де необхідне забезпечення анонімності користувачів. Результати дослідження можуть бути використані для вдосконалення існуючих систем безпеки та створення нових рішень, що відповідають сучасним стандартам приватності.

Таким чином, дана робота спрямована на вирішення актуальних завдань із забезпечення анонімності та захисту даних, що має важливе значення для забезпечення безпеки інформаційного простору та зростання довіри до цифрових сервісів у суспільстві.

Апробація: тези доповідей представленні на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» на тему «АНАЛІЗ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ ПРИВАТНОСТІ В БАГАТОВУЗЛОВИХ МЕРЕЖАХ» [8].

1 ТЕОРЕТИЧНІ ЗАСАДИ ВДОСКОНАЛЕННЯ СИСТЕМИ АНОНІМНОГО ОБМІНУ ДАНИМИ

У даному розділі буде розглянуто теоретичні засади вдосконалення системи анонімного обміну даними в умовах сучасних інформаційних мереж. Особливу увагу приділено актуальності питання забезпечення анонімності та приватності в умовах зростання обсягів інформації та ризиків її компрометації. Буде проаналізовано основні принципи анонімізації, такі як k-анонімність, l-різноманітність і t-близькість, які є ключовими інструментами для захисту даних.

Також у розділі буде досліджено сутність і призначення багатовузлових мереж користувачів, що забезпечують маршрутизацію даних у середовищі з підвищеним рівнем приватності. Окремо розглядається використання контейнеризації даних, яка дозволяє ізолювати інформацію та гарантувати її безпеку на всіх етапах передачі. Для забезпечення повноти аналізу будуть досліджені існуючі методи забезпечення приватності в багатовузлових мережах, їх переваги та недоліки.

Результати цього розділу дозволять сформулювати основні задачі, що необхідно вирішити для розробки вдосконаленої системи анонімного обміну даними, яка відповідатиме сучасним вимогам захисту інформації.

1.1 Актуальність та особливості анонімного обміну даними в сучасних інформаційних мережах

У сучасному цифровому світі обмін інформацією набув безпрецедентних масштабів і швидкості. Мільярди користувачів щодня передають дані через численні платформи, додатки та пристрої, що взаємодіють у різних мережах. З огляду на це, проблема захисту приватності та анонімності стає все більш актуальною, адже багато користувачів хочуть захистити свою ідентичність і особисту інформацію від стеження або незаконного доступу. Анонімний обмін даними допомагає користувачам залишатися інкогніто, забезпечуючи їхню конфіденційність і безпеку [1].

Ключові чинники, які зумовлюють актуальність анонімного обміну даними:

– зростання обсягу персональних даних: Внаслідок широкого використання інтернету речей (IoT), соціальних мереж, мобільних додатків та хмарних сервісів у мережі передаються великі обсяги даних, що включають особисту інформацію користувачів;

– збільшення кіберзагроз: Сучасні кіберзагрози, такі як атаки на персональні дані, викрадення особистої інформації та фінансових даних, створюють значні ризики для користувачів. Анонімний обмін даними допомагає зменшити ймовірність несанкціонованого доступу до особистої інформації;

– тенденція до посилення нормативно-правового регулювання: Законодавчі акти, такі як GDPR у Європейському Союзі, вимагають забезпечення конфіденційності та безпеки даних, що також стимулює розвиток технологій для анонімного обміну даними;

– підвищення обізнаності користувачів щодо приватності: Користувачі дедалі більше усвідомлюють значення захисту персональних даних і приватності, що сприяє попиту на анонімні засоби обміну інформацією [1].

Анонімний обмін даними забезпечує користувачам можливість приховати свою особистість та конфіденційну інформацію при передачі даних у мережах, особливо критично це в умовах підвищених кіберзагроз. Існують численні методи та підходи для забезпечення анонімності, які розроблені для уникнення ризику ідентифікації користувача. Серед основних особливостей анонімного обміну даними виділяють:

Принципи забезпечення анонімності.

Для досягнення анонімності під час обміну даними використовуються кілька ключових принципів, що дозволяють знизити ризик ідентифікації особистості користувача:

– псевдонімізація: Це метод заміни реальних ідентифікаційних даних користувача на псевдонімні маркери (наприклад, випадкові або унікальні ідентифікатори). Цей підхід дозволяє приховати особистість користувача,

забезпечуючи анонімність в межах мережі. Застосовується у випадках, коли повне знищення особистої інформації неможливе або недоцільне;

– k-анонімність: Принцип полягає у тому, що дані користувача змішуються з даними щонайменше k-1 інших користувачів, що дозволяє приховати особистість серед групи людей. Наприклад, якщо k дорівнює 5, кожен користувач буде «непомітним» серед п’яти осіб, що ускладнює ідентифікацію;

– L-різноманітність: Для підвищення захищеності інформації кожна група даних включає L різних значень чутливих атрибутів, таких як стать або місцезнаходження. Це знижує ризик виявлення конкретного користувача за наявними атрибутами, забезпечуючи, щоб однакові значення не були розподілені однаково в групах;

– t-близькість: Цей підхід передбачає, що розподіл чутливих атрибутів у кожній групі даних має бути близьким до розподілу таких самих атрибутів у всій базі даних. Це допомагає уникнути витоку інформації на основі рідкісних чи незвичних атрибутів [1].

Методи маршрутизації даних для забезпечення анонімності.

Для приховування маршруту і відправника повідомлення анонімні мережі використовують спеціальні методи маршрутизації, серед яких:

– багатовузлові (мультихоп) мережі: Цей метод полягає в передачі даних через кілька вузлів, де кожен вузол знає лише попередній і наступний вузол у ланцюжку передачі. Таким чином, жоден окремий вузол не має повної інформації про відправника або отримувача, що підвищує анонімність;

– Onion Routing (Технологія «Цибулинної маршрутизації»): Onion Routing застосовує багаторівневе шифрування. Кожен рівень або «шар» шифрування знімається при проходженні через окремий вузол, подібно до шарів цибулі. Це дозволяє кожному вузлу знати лише попередній і наступний шар даних, зберігаючи анонімність відправника та отримувача навіть при перехопленні трафіку;

– мікс-мережі: Методика полягає у випадковому перемішуванні пакетів даних у мережі, що ускладнює встановлення зв’язку між відправником і

отримувачем. Мікс-мережі застосовуються для анонімізації даних у широкому спектрі додатків, забезпечуючи додатковий захист від аналізу трафіку [2].

Анонімні протоколи передачі даних.

Анонімні протоколи спеціально розроблені для приховування інформації користувача під час передачі даних і включають декілька методів:

– протоколи передачі з елементами випадковості: Випадкові елементи, такі як штучні затримки або випадковий порядок передачі, дозволяють уникнути прив’язки дій до конкретного користувача та ускладнюють моніторинг трафіку;

– протоколи з мінімізацією даних: Щоб знизити ризик ідентифікації, деякі протоколи обмежують обсяг даних, які можуть передаватися, до мінімально необхідного рівня. Це знижує можливість витоку інформації, зберігаючи лише основну інформацію для роботи мережі;

– шифрування та обфускація даних: Анонімні мережі використовують сучасні методи шифрування для захисту від несанкціонованого доступу до переданих даних. Окрім шифрування, обфускація (приховування) ускладнює розуміння змісту даних навіть при перехопленні. У трафіку мережі додаються «сміттєві» дані або викривляється реальний вміст для заплутування можливих спостерігачів [2].

Проблеми та виклики анонімного обміну даними.

Анонімний обмін даними супроводжується певними технічними та організаційними проблемами, які включають:

– зниження продуктивності та швидкості: Багаторівневе шифрування, маршрутизація через кілька вузлів, а також інші заходи захисту знижують швидкість передачі даних і можуть викликати затримки. Це стає проблемою для користувачів, які потребують високої продуктивності та швидкої передачі даних;

– підвищена вразливість до атак деканонімізації: Аналітичні інструменти, машинне навчання та інші методи аналізу даних здатні виділяти шаблони поведінки, що ускладнює повну анонімність і збільшує ризик деканонімізації;

– витрати на інфраструктуру та підтримку мережі: Анонімні мережі потребують значних обчислювальних потужностей для підтримки багатоступеневого шифрування і складної маршрутизації. Це може створювати фінансові та організаційні труднощі для мереж, які бажають впровадити анонімний обмін.

Технологічні рішення для покращення анонімного обміну даними [2].

Сучасні дослідження та розробки спрямовані на вдосконалення анонімного обміну даними, забезпечуючи додатковий рівень безпеки для користувачів:

– інтеграція штучного інтелекту: Алгоритми штучного інтелекту можуть аналізувати трафік у реальному часі для виявлення потенційних загроз і автоматичного налаштування параметрів анонімності залежно від ситуації;

– технології блокчейну та децентралізація: Децентралізовані системи, такі як блокчейн, здатні забезпечувати анонімний обмін даними без централізованої довіреної сторони. Це підвищує стійкість мереж до атак і знижує ризик витоку інформації;

– сек'юрні мультисторонні обчислення (SMPC): Ця технологія дозволяє кільком сторонам виконувати спільні обчислення без розкриття своїх даних. SMPC забезпечує можливість проведення аналізу або обміну даними з високим рівнем конфіденційності [2].

Анонімний обмін даними в інформаційних мережах надає користувачам інструменти для збереження конфіденційності та уникнення розкриття особистої інформації. Використання спеціальних протоколів, методів маршрутизації та інструментів шифрування дозволяє досягти високого рівня анонімності, проте технологічні та організаційні виклики залишаються.

1.2 Основні принципи анонімності та приватності: k-анонімність, L-різноманітність, t-близькість

Забезпечення приватності та анонімності даних є одним із ключових завдань у сучасних інформаційних системах, особливо у випадках, коли йдеться про

персональні дані. Основні підходи до захисту даних включають такі методи, як k-анонімність, L-різноманітність і t-близькість. Кожен з них забезпечує унікальний рівень захисту, який запобігає ідентифікації особи та захищає чутливу інформацію від несанкціонованого доступу [3].

k-анонімність є одним із базових підходів для захисту приватності даних, що використовується для зниження ризику ідентифікації особи за допомогою атрибутів записів у базі даних. Основна ідея k-анонімності полягає в тому, щоб кожен запис у наборі даних був неможливим для ідентифікації серед групи з щонайменше *k* інших записів.

Принцип дії k-анонімності:

– для набору даних визначаються ідентифікаційні атрибути (наприклад, ПІБ, адреса) та квазі-ідентифікатори (наприклад, вік, поштовий індекс);

– деперсоналізація даних відбувається через згрупування записів за квазі-ідентифікаторами таким чином, щоб кожен запис був не відрізним від щонайменше k-1 інших записів. Наприклад, якщо встановлено рівень 5-анонімності, кожен запис буде невідрізним від чотирьох інших [3].

Методи досягнення k-анонімності:

– генералізація: Зміна точності даних, наприклад, замість конкретного віку «25 років» вказується діапазон «20–30 років»;

– придушення (супресія): Видалення або приховування певних атрибутів, що підвищує складність ідентифікації;

– агрегація: Створення більш узагальнених груп за ознаками, щоб кожен запис виглядав однаково в межах групи.

Переваги:

k-анонімність надає базовий рівень анонімності, дозволяючи захистити користувачів навіть у випадках, коли дані частково деперсоналізовані.

Обмеження:

Принцип k-анонімності не враховує різноманітність чутливих атрибутів у кожній групі. Якщо квазі-ідентифікатори мають схожі чутливі атрибути, це може

призвести до ризику повторної ідентифікації, особливо якщо всі значення чутливого атрибута однакові в одній групі. Наприклад, якщо в одній групі всі записи мають один і той самий діагноз, конфіденційність таких даних залишається під загрозою [3].

L-різноманітність — це вдосконалення k-анонімності, яке усуває її основне обмеження, забезпечуючи різноманітність чутливих атрибутів у групах записів. Цей метод передбачає, що кожна група, яка задовольняє принципу k-анонімності, повинна містити щонайменше L різних значень чутливих атрибутів. Це допомагає запобігти розкриттю інформації навіть у випадках, коли в групі є однакові значення атрибутів.

Принцип дії L-різноманітності:

– створюються групи, що відповідають рівню k-анонімності, з додатковою умовою: у кожній групі має бути щонайменше L різних значень для чутливих атрибутів;

– якщо значення чутливих атрибутів не різняться в групах, то ці записи об’єднуються з іншими, щоб досягти необхідного рівня різноманітності.

Методи досягнення L-різноманітності:

– глобальна диверсифікація: Групування записів таким чином, щоб у кожній групі досягалася потрібна різноманітність для чутливого атрибута, наприклад, різні діагнози або рівні доходу;

– розподілена диверсифікація: Забезпечення, щоб різноманітність значень чутливих атрибутів була однаковою у всіх групах записів, що підвищує рівень приватності [3].

Переваги:

L-різноманітність забезпечує більшу конфіденційність даних за рахунок розмаїття значень чутливих атрибутів, що знижує ризик розкриття інформації про конкретну особу в межах групи.

Обмеження:

– L-різноманітність може бути неефективною в умовах, коли чутливі атрибути мають високий рівень варіативності, оскільки може знадобитися збільшення розміру груп, що ускладнює процес деперсоналізації;

– також є обмеження, коли розподіл чутливих атрибутів у групах неоднорідний, що може призвести до зниження рівня захисту.

t-близькість є вдосконаленням L-різноманітності, яке забезпечує контроль за розподілом значень чутливих атрибутів у групах. Цей метод враховує відмінності в розподілі значень чутливих атрибутів у межах групи та у всій базі даних, щоб уникнути виявлення інформації про конкретного користувача [3].

Принцип дії t-близькості:

– t-близькість передбачає, що розподіл чутливих атрибутів у кожній групі повинен бути близьким до загального розподілу таких атрибутів у базі даних;

– визначається параметр t, що контролює допустимий рівень відмінності між розподілом чутливих атрибутів у групах і загальним розподілом. Якщо відмінність перевищує встановлений рівень t, то дані перегруповуються або деперсоналізуються до необхідного рівня.

Методи досягнення t-близькості:

– підтримка групової однорідності: Перевірка кожної групи для забезпечення, що розподіл чутливих атрибутів відповідає загальному розподілу в межах допустимого рівня t;

– контроль за відмінностями у розподілі: Аналіз кожної групи на предмет відмінності в розподілі чутливих атрибутів і перегрупування або зміна даних, якщо різниця перевищує допустимий поріг [4].

Переваги:

– t-близькість забезпечує захист від так званих атак на фоні (background knowledge attacks), коли зловмисники мають певну інформацію про розподіл чутливих атрибутів і можуть використовувати це для ідентифікації;

– завдяки контролю розподілу даних t-близькість забезпечує більш точний та гнучкий захист чутливих атрибутів.

Обмеження:

– досягнення t-близькості може бути складним завданням при високій варіативності значень чутливих атрибутів або великій кількості записів, оскільки це потребує комплексного аналізу та більшого обсягу обчислювальних ресурсів;

– у деяких випадках складність контролю розподілу чутливих атрибутів може призвести до втрати точності в даних, що обмежує їх практичне застосування [4].

k-анонімність, L-різноманітність і t-близькість — це основні принципи забезпечення приватності та анонімності, які використовуються для захисту даних від витоку інформації та ідентифікації користувачів. k-анонімність забезпечує базовий захист, L-різноманітність підвищує рівень конфіденційності за рахунок різноманітності чутливих атрибутів, а t-близькість дозволяє контролювати розподіл чутливих даних. Разом ці методи утворюють систему надійного захисту, яка застосовується у сучасних інформаційних системах для запобігання несанкціонованому доступу до конфіденційної інформації [5].

1.3 Сутність та призначення багатовузлових мереж користувачів

Багатовузлові мережі користувачів є складними системами, що об'єднують вузли для передачі інформації та забезпечують анонімний, захищений обмін даними. Основною метою таких мереж є надання користувачам можливості обмінюватися інформацією без розкриття своєї особистості, захистити комунікацію від перехоплення та несанкціонованого доступу, а також забезпечити децентралізацію передачі даних. Цей підхід дозволяє створювати конфіденційні, безпечні канали зв'язку, які ускладнюють спроби відстеження та аналізу трафіку.

Багатовузлові мережі (або мультихоп-мережі) складаються з численних вузлів, через які передаються дані. Кожен вузол у цій мережі виконує роль проміжної точки, що отримує та пересилає інформацію, не зберігаючи повної інформації про відправника і кінцевого отримувача. Багатовузлові мережі забезпечують безпеку та анонімність завдяки таким особливостям [5]:

– розподілена структура: Багатовузлові мережі є децентралізованими і не залежать від одного централізованого сервера, що забезпечує стійкість системи до атак. У такій мережі кожен вузол має обмежену інформацію про дані та маршрут передачі, що захищає користувача від ідентифікації та зменшує ймовірність успішних атак;

– анонімність передачі даних: Для забезпечення анонімності кожен вузол у багатовузловій мережі знає лише попередній та наступний вузол у ланцюжку передачі даних. Така структура дозволяє мінімізувати знання про кінцеву точку передачі та відправника, що ускладнює ідентифікацію [6];

– шифрування на кожному етапі: Дані, що передаються через багатовузлову мережу, часто шифруються багаторазово (наприклад, за технологією Onion Routing). Кожен вузол знімає один шар шифрування, отримуючи інформацію лише про те, до якого вузла дані мають надійти далі. Завдяки такій багаторівневій системі шифрування забезпечується високий рівень захисту від перехоплення інформації;

– динамічна маршрутизація: Вузли багатовузлової мережі здатні динамічно змінювати маршрут передачі даних залежно від навантаження або стану мережі. Це дозволяє підвищити рівень анонімності, оскільки навіть при спробі аналізу трафіку відстежити конкретний маршрут передачі стає складніше.

Багатовузлові мережі мають широкий спектр застосувань, зокрема для забезпечення анонімного зв'язку, захисту даних та децентралізованого управління. Основні завдання та призначення таких мереж включають [7]:

– анонімний обмін даними: Основною метою багатовузлових мереж є забезпечення можливості обміну даними без розкриття особистості користувача. Завдяки багатошаровому шифруванню і маршрутизації через кілька вузлів користувачі можуть обмінюватися повідомленнями або іншими даними, зберігаючи конфіденційність;

– захист від цензури: У багатьох країнах користувачі стикаються з обмеженням доступу до певних веб-ресурсів та платформ. Багатовузлові мережі,

такі як Tor, надають можливість обходу блокувань, дозволяючи користувачам зберігати доступ до інформації незалежно від зовнішніх обмежень;

– захист від атак типу "людина посередині" (MITM): У багатовузлових мережах забезпечується високий рівень захисту від перехоплення даних, оскільки кожен вузол у маршруті бачить лише часткову інформацію про передані дані. Це значно ускладнює несанкціоноване втручання або модифікацію інформації під час передачі [8];

– децентралізація управління та зберігання даних: Багатовузлові мережі підтримують децентралізований підхід до передачі та обробки даних, що знижує ризик централізованого контролю і полегшує масштабування мережі. Децентралізація також дозволяє уникати зберігання даних у одній точці, що знижує ризик атак на одну центральну базу даних;

– забезпечення конфіденційності при використанні загальнодоступних мереж: Багатовузлові мережі дозволяють користувачам зберігати анонімність і конфіденційність при використанні відкритих мереж (наприклад, Wi-Fi в громадських місцях), де високий ризик перехоплення даних [9].

Багатовузлові мережі реалізовані в різних технологіях, що надають користувачам можливість обміну даними з високим рівнем конфіденційності та безпеки:

– Tor (The Onion Router): Це одна з найбільш відомих багатовузлових мереж, яка забезпечує анонімний доступ до інтернет-ресурсів за допомогою технології Onion Routing. Tor використовує багаторівневе шифрування для приховування джерела трафіку, маршрутизації через випадкові вузли, що ускладнює відстеження;

– I2P (Invisible Internet Project): I2P є децентралізованою мережею, яка фокусується на забезпеченні анонімності всередині мережі. Вона використовує двостороннє шифрування для захисту даних, а також створює тунелі для кожного з'єднання, що ускладнює ідентифікацію користувачів і запобігає атакам;

– Freenet: Ця децентралізована мережа дозволяє користувачам зберігати та обмінюватися даними анонімно. Freenet базується на принципі "дружніх вузлів", де

користувачі можуть довіряти лише своїм знайомим, створюючи захищене середовище для обміну інформацією;

– блокчейн-мережі: Деякі блокчейн-системи (наприклад, Monero, Zcash) інтегрують принципи багатовузлових мереж для забезпечення анонімності транзакцій. Вони використовують криптографічні протоколи, що приховують інформацію про учасників транзакцій та їх обсяг [10].

Виклики та обмеження багатовузлових мереж:

– зниження швидкості передачі даних: Оскільки дані проходять через кілька вузлів і шифруються на кожному з них, швидкість передачі може суттєво знижуватися. Це є проблемою для користувачів, яким потрібен швидкий доступ до ресурсів або низька затримка;

– вразливість до атак деканонімізації: Незважаючи на високий рівень захисту, можливість деканонімізації існує, особливо якщо зловмисник контролює кілька вузлів у мережі. Такі атаки дозволяють зіставляти вхідні та вихідні точки, щоб виявити джерело та ціль передачі даних;

– потреба у великій кількості ресурсів для підтримки мережі: Багатовузлові мережі вимагають значних обчислювальних ресурсів для підтримки децентралізованої інфраструктури та шифрування даних, що може обмежувати кількість користувачів і швидкість роботи мережі;

– обмеження доступу в деяких країнах: Деякі держави активно блокують доступ до анонімних мереж, таких як Tor, обмежуючи їхнє використання. Це потребує створення додаткових методів для обходу блокувань, наприклад, за допомогою мостів та альтернативних протоколів [11].

Багатовузлові мережі користувачів є ефективним засобом забезпечення анонімного обміну даними та захисту конфіденційності в мережах. Вони дозволяють користувачам обмінюватися інформацією без загрози розкриття особистості та витоку чутливих даних. Завдяки розподіленій структурі, багаторівневому шифруванню та динамічній маршрутизації багатовузлові мережі забезпечують високий рівень захисту та стійкості до атак. Однак, ці мережі також

стикаються з низкою технічних викликів, таких як зниження швидкості, потреба у ресурсах та можливість деканонімізації, що спонукає до подальшого розвитку та вдосконалення технологій для підвищення ефективності анонімних мереж [12].

1.4 Використання контейнеризації даних для забезпечення анонімності

Контейнеризація даних стала одним з ключових підходів у сучасних інформаційних системах для забезпечення анонімності та захисту конфіденційності. Цей метод дозволяє ізолювати дані та процеси в окремих середовищах — контейнерах, що забезпечує високий рівень контролю над доступом до даних і дозволяє мінімізувати ризики витоку інформації. Використання контейнеризації надає можливість створювати анонімні середовища для зберігання, обробки та передачі інформації, підвищуючи рівень захисту користувачів і чутливих даних [13].

Контейнеризація даних полягає в тому, що кожна одиниця даних (або група даних) обробляється всередині ізольованого середовища — контейнера, що має власні ресурси та обмеження доступу. Ці контейнери включають не лише дані, але й залежності, необхідні для їх обробки, зокрема програми та бібліотеки. Це дозволяє ізолювати інформацію на рівні програмного забезпечення, що працює в контейнері, та обмежує доступ до даних для інших компонентів системи.

Контейнери зберігають необхідну інформацію про дані і функції, що можуть бути виконані, але не надають доступ до самих даних іншим контейнерам, системам чи користувачам. У випадку з анонімністю контейнери ізолюють дані таким чином, щоб жодна інша частина системи, окрім уповноважених компонентів, не могла ідентифікувати або отримати доступ до них [14].

Основні принципи використання контейнеризації для анонімності:

– ізоляція даних: Контейнери ізолюють дані та операції всередині власного середовища, що дозволяє забезпечити анонімність даних і мінімізувати ризик витоку. Завдяки ізоляції інші компоненти або користувачі не мають доступу до даних всередині контейнера, що підвищує рівень приватності;

– контроль доступу: Кожен контейнер має власну систему управління доступом, що визначає, хто і за яких умов може виконувати операції з даними. Це дозволяє забезпечити контроль над тим, які дії виконуються з даними і ким. Таким чином, тільки авторизовані компоненти мають доступ до даних, що знижує ризик несанкціонованого доступу;

– анонімізація даних при передачі: Контейнеризація дозволяє здійснювати передачу даних між контейнерами без розкриття їхнього вмісту, використовуючи шифрування. Контейнери шифрують дані перед їх передачею до інших контейнерів або компонентів системи, що дозволяє зберегти анонімність даних при їх переміщенні у системі;

– мінімізація метаданих: Контейнеризація забезпечує можливість зберігати мінімальний обсяг метаданих, які могли б використовуватися для ідентифікації користувачів або джерел інформації. Наприклад, метадані обмежуються лише інформацією, необхідною для операцій з даними (ідентифікатори контейнера та ключі шифрування), що мінімізує ризик деканонімізації [15].

Існує кілька ключових технологій контейнеризації, які використовуються для забезпечення анонімності та конфіденційності даних. Серед них найбільш поширеними є Docker, Kubernetes та OpenShift, які надають розширені можливості ізоляції та управління контейнерами:

– Docker: Це одна з найбільш популярних технологій контейнеризації, яка забезпечує легке створення, розгортання та управління контейнерами. Docker підтримує ізоляцію даних всередині кожного контейнера і дозволяє конфігурувати рівень доступу до даних за допомогою прав доступу, що забезпечує захист і приватність;

– Kubernetes: Ця система оркестрації контейнерів дозволяє розподіляти та керувати контейнерами в масштабі, забезпечуючи розподіл навантаження та автоматизацію процесів. Kubernetes надає можливість створювати "namespace" (простори імен), які дозволяють створювати окремі середовища для груп контейнерів, забезпечуючи додатковий рівень ізоляції даних;

– OpenShift: Це платформа контейнеризації на базі Kubernetes, яка надає додаткові можливості для управління безпекою та анонімністю контейнерів. OpenShift дозволяє обмежувати доступ до контейнерів на рівні мережі та використовує політики безпеки для зниження ризику несанкціонованого доступу до даних [16].

Переваги використання контейнеризації для анонімності:

– гнучке управління доступом: Контейнери дозволяють налаштовувати доступ до даних на рівні контейнера або групи контейнерів. Це дозволяє контролювати, які користувачі та компоненти можуть взаємодіяти з даними, що підвищує безпеку;

– захист від витоку даних: Контейнери ізолюють дані, що запобігає можливості їх витоку у випадку зламу системи. Навіть якщо один контейнер буде скомпрометовано, інші контейнери і дані залишаться захищеними;

– спрощення процесу анонімізації: Використання контейнерів спрощує реалізацію механізмів анонімізації, оскільки анонімізація може бути реалізована на рівні контейнера. Це знижує витрати на анонімізацію та підвищує її ефективність;

– легкість у масштабуванні та управлінні: Контейнери легко масштабуються, що дозволяє створювати розподілені системи з анонімним обміном даними. Оркестрація контейнерів, наприклад через Kubernetes, дозволяє автоматично розподіляти ресурси та підтримувати анонімність при збільшенні обсягу даних або користувачів [17].

Виклики та обмеження контейнеризації для забезпечення анонімності:

– необхідність у складному управлінні політиками доступу: Для забезпечення повної анонімності потрібно чітко налаштувати політики доступу до кожного контейнера, що може бути технічно складним завданням, особливо в масштабних системах;

– ризик вразливостей на рівні контейнерів: Незважаючи на ізоляцію, контейнери все ще можуть містити вразливості, які можуть бути використані для

отримання доступу до даних. Необхідно регулярно оновлювати та перевіряти контейнери на наявність потенційних вразливостей;

– залежність від інфраструктури контейнеризації: Надійність та безпека контейнеризації багато в чому залежать від інфраструктури, на якій працюють контейнери. Неправильне налаштування інфраструктури або вразливість платформи може призвести до витоку даних або порушення анонімності [18].

Приклади використання контейнеризації для забезпечення анонімності

– ізольовані середовища для обробки медичних даних: У медичних системах дані пацієнтів зберігаються та обробляються в контейнерах, що забезпечує анонімність і захищає чутливу інформацію. Такі контейнери мають спеціальні налаштування доступу, що обмежують можливість ідентифікації пацієнтів;

– анонімні бази даних для досліджень: У наукових дослідженнях, де потрібно захистити дані учасників, використання контейнеризації дозволяє створювати анонімні бази даних, в яких кожен контейнер містить окремі анонімізовані набори даних. Це дозволяє забезпечити безпеку даних навіть при спільній роботі декількох дослідників;

– фінансові системи з анонімним обміном даними: У фінансових системах для захисту даних про транзакції використовуються контейнери, що забезпечують шифрування і захист від несанкціонованого доступу. Контейнери ізолюють інформацію про користувачів і транзакції, знижуючи ризик її витоку та деканонімізації [19].

Контейнеризація даних є потужним інструментом для забезпечення анонімності та захисту конфіденційності в сучасних інформаційних системах. Вона надає можливість ізолювати дані та керувати доступом до них, що дозволяє знижувати ризик несанкціонованого доступу та витоку інформації. Незважаючи на певні технічні виклики та обмеження, контейнеризація забезпечує ефективний і гнучкий підхід до зберігання та обробки даних, що робить її незамінною технологією у системах з високими вимогами до анонімності [20].

1.5 Аналіз існуючих методів забезпечення приватності в багатовузлових мережах

Багатовузлові мережі, які забезпечують анонімність користувачів і захист їхніх даних, стають дедалі важливішими в умовах сучасних кіберзагроз. Для досягнення цього методу забезпечення приватності в таких мережах розроблено кілька підходів, які мінімізують ризики ідентифікації користувачів та несанкціонованого доступу до їхніх даних. Основні методи забезпечення приватності в багатовузлових мережах включають Onion Routing, Garlic Routing, мікси та шумові протоколи, кожен з яких має свої особливості, переваги та недоліки.

Onion Routing є однією з найпопулярніших технологій для забезпечення приватності в багатовузлових мережах. Вона передбачає багатошарове шифрування даних, які надсилаються через кілька вузлів, де кожен вузол знімає один шар шифрування, подібно до очищення цибулі [21].

Принцип дії Onion Routing:

– дані, які надсилає користувач, спочатку шифруються декілька разів, створюючи багатошаровий захисний механізм;

– кожен вузол у мережі Onion Routing знімає лише один шар шифрування, отримуючи інформацію про те, до якого вузла слід передати дані далі;

– жоден з вузлів, окрім кінцевого, не знає повного маршруту і кінцевої точки призначення, що забезпечує анонімність.

Переваги:

– високий рівень анонімності завдяки багатошаровому шифруванню;

– надійний захист від атак на перехоплення, оскільки навіть якщо зловмисник зламає один із вузлів, він не отримає повного маршруту.

Недоліки:

– зниження швидкості передачі даних, оскільки багатошарове шифрування та маршрутизація через кілька вузлів вимагають значних обчислювальних ресурсів;

– вразливість до атак деканонімізації, якщо зловмисник контролює кілька вузлів на маршруті.

Приклад застосування: Onion Routing використовується в мережі Tor, яка дозволяє користувачам отримувати доступ до інтернету анонімно, приховуючи свої IP-адреси та місцезнаходження.

Garlic Routing є вдосконаленням Onion Routing, яке використовується в деяких децентралізованих мережах, зокрема в I2P (Invisible Internet Project). Garlic Routing передбачає пакування кількох повідомлень в один зашифрований пакет, що ускладнює аналіз трафіку та знижує ризик розкриття окремих повідомлень [22].

Принцип дії Garlic Routing:

– кілька повідомлень об'єднуються в один великий пакет, і кожне з них шифрується окремо всередині пакету;

– весь пакет передається через мережу так само, як і в Onion Routing, де кожен вузол знімає один шар шифрування, не знаючи повного вмісту пакета;

– кожне повідомлення може мати різний кінцевий пункт призначення, що ускладнює відстеження маршруту.

Переваги:

– підвищена безпека завдяки об'єднанню кількох повідомлень у один пакет, що ускладнює аналіз трафіку;

– кращий захист від атак на основі обсягу даних, оскільки кожен пакет виглядає як великий обсяг зашифрованих даних, що не дозволяє визначити розмір окремих повідомлень.

Недоліки:

– потребує додаткових ресурсів на шифрування та розшифрування кожного окремого повідомлення в пакеті;

– складність реалізації через необхідність об'єднання різних повідомлень і управління ними.

Приклад застосування: Garlic Routing використовується в I2P для забезпечення приватності та анонімності передачі даних усередині децентралізованої мережі [23].

Мікс-мережі є підходом для забезпечення анонімності, який передбачає випадкове перемішування повідомлень перед їхньою передачею через мережу. Це дозволяє приховати зв’язок між відправником і одержувачем, а також запобігти відстеженню трафіку.

Принцип дії мікс-мереж:

– повідомлення від кількох користувачів збираються у «мікс» або групу, де вони випадковим чином перемішуються;

– вміст кожного повідомлення шифрується, і воно надсилається до різних вузлів у випадковому порядку, що унеможливлює встановлення зв’язку між відправником і отримувачем;

– мікс-мережі забезпечують затримки передачі повідомлень, щоб ще більше ускладнити аналіз трафіку.

Переваги:

– високий рівень захисту від аналізу трафіку, оскільки мікси приховують інформацію про джерело та кінцеву точку повідомлень;

– забезпечує надійний захист від атак на основі патернів, оскільки зв'язок між повідомленнями знищується в процесі перемішування.

Недоліки:

– зниження швидкості передачі даних через затримки, які необхідні для створення міксів;

– ускладнена реалізація та потреба в значних ресурсах для підтримки безпеки мережі.

Приклад застосування: Мікс-мережі використовуються в електронних поштових службах та в захищених системах передачі повідомлень, зокрема в проектах, що забезпечують анонімність транзакцій.

Шумові протоколи передбачають додавання випадкових шумових даних до реальних повідомлень, що значно ускладнює їхнє відстеження. Цей метод забезпечує захист від спроб аналізу трафіку, оскільки будь-які спостереження міститимуть шум, що не дозволяє відокремити реальні дані від випадкових [24].

Принцип дії шумових протоколів:

– під час передачі повідомлень додаються випадкові дані (шум), які приховують реальну інформацію про обсяг та структуру трафіку.

– шум генерується випадковим чином і передається між вузлами, створюючи враження активного трафіку навіть тоді, коли фактично повідомлень немає.

– цей шум додається до кожного повідомлення, що унеможливлює аналіз трафіку, навіть якщо зловмисник має доступ до декількох вузлів.

Переваги:

– ефективний захист від аналізу трафіку, оскільки шум ускладнює ідентифікацію реальних повідомлень;

– простота реалізації: додавання шуму не потребує значної зміни у структурі мережі.

Недоліки:

– збільшення обсягу даних, що передаються, що може знижувати продуктивність мережі;

– не забезпечує повної анонімності, оскільки шум лише ускладнює, але не виключає можливість аналізу трафіку.

Приклад застосування: Шумові протоколи використовуються в мережах типу Tor для створення додаткового трафіку, який приховує реальні запити та повідомлення від спостерігачів.

Таблиця 1.1 надає узагальнений огляд основних методів забезпечення приватності в багатовузлових мережах. Кожен з методів використовує різні підходи для захисту даних, включаючи багатошарове шифрування, об'єднання повідомлень, випадкове перемішування та додавання шуму. Вибір методу залежить від конкретних вимог до конфіденційності та продуктивності мережі, а також від

цілей користувача, таких як захист від аналізу трафіку, забезпечення анонімності або обхід блокувань.

Таблиця 1.1 – Порівняльний аналіз методів забезпечення приватності в багатовузлових мережах

Метод	Принцип дії	Переваги	Недоліки	Приклад застосування
Onion Routing	Багатошарове шифрування і передача через вузли	Високий рівень анонімності, захист від перехоплення	Зниження швидкості, вразливість до деканонімізації	Tor
Garlic Routing	Пакування кількох повідомлень у один пакет	Підвищена безпека, захист від аналізу трафіку	Потребує більше ресурсів	I2P
Мікс-мережі	Випадкове перемішування повідомлень	Високий захист від аналізу трафіку	Зниження швидкості, потреба у значних ресурсах	Системи електронної пошти
Шумові протоколи	Додавання випадкового шуму до реальних повідомлень	Ефективний захист від аналізу трафіку	Збільшення обсягу даних, обмежена анонімність	Tor та інші анонімні мережі

Методи, представлені в таблиці 1.1, забезпечують анонімність та конфіденційність різними способами. Onion Routing та Garlic Routing підходять для високозахищених мереж, забезпечуючи багатошарове шифрування і мінімізацію ризику перехоплення. Мікс-мережі спеціалізуються на перемішуванні повідомлень для приховування зв'язку між відправником та отримувачем, тоді як шумові протоколи додають випадкові дані для ускладнення аналізу трафіку. В цілому, поєднання цих методів дозволяє створити гнучку і надійну систему захисту приватності в багатовузлових мережах.

Кожен з методів забезпечення приватності в багатовузлових мережах має свої унікальні переваги та обмеження, що робить його придатним для різних завдань. Onion Routing і Garlic Routing є найбільш ефективними для забезпечення високого рівня анонімності в мережах, таких як Tor і I2P. Мікс-мережі підходять для сервісів, де необхідно забезпечити високий рівень захисту від аналізу трафіку, хоча вони можуть знижувати продуктивність. Шумові протоколи є ефективними для створення додаткового трафіку, але вони підходять лише як додатковий захід безпеки [25].

Поєднання кількох методів може забезпечити вищий рівень приватності, ніж використання кожного з них окремо. В умовах зростаючих кіберзагроз та потреб у захисті анонімності багатовузлові мережі продовжують розвиватися, зокрема шляхом інтеграції технологій штучного інтелекту та блокчейну, що може підвищити їхню ефективність у забезпеченні приватності.

1.6 Висновки та постановка задачі

У цій роботі проведено дослідження теоретичних основ вдосконалення системи анонімного обміну даними, що ґрунтується на багатовузловій мережі з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних. Було здійснено всебічний аналіз сучасних методів забезпечення анонімності в інформаційних мережах, зокрема розглянуто принципи приватності, використання багатошарового шифрування та маршрутизації,

контейнеризації для ізоляції даних, а також додаткові засоби, такі як шумові протоколи та мікс-мережі.

Дослідження дозволило отримати глибоке розуміння та оцінку ключових аспектів приватності в багатовузлових мережах, що є критично важливим для забезпечення конфіденційності користувачів, захисту даних і мінімізації ризиків деканонімізації у процесі передачі інформації.

На основі проведеного аналізу сформульовано наступні задачі для подальшої розробки та реалізації системи:

– розробка алгоритму для інтеграції принципів k-анонімності, L-різноманітності та t-близькості у багатовузлову мережу з метою забезпечення приватності користувачів;

– проектування контейнеризованої системи для ізоляції даних з обмеженим доступом та механізмами контролю, що зменшує ймовірність витоку даних;

– розробка схеми передачі даних через багатовузлову мережу із застосуванням Onion і Garlic Routing для мінімізації можливостей деканонімізації;

– практична реалізація багатовузлової мережі з інтеграцією контейнеризації для тестування ефективності алгоритмів анонімності та аналізу приватності в реальних умовах експлуатації;

Ці завдання допоможуть створити комплексну систему анонімного обміну даними, яка відповідатиме сучасним вимогам до захисту конфіденційності та анонімності в умовах багатовузлових мереж.

2 СТВОРЕННЯ ВДОСКОНАЛЕНОЇ СИСТЕМИ АНОНІМНОГО ОБМІНУ ДАНИМИ НА ОСНОВІ АНОНІМНОЇ БАГАТОВУЗЛОВОЇ МЕРЕЖІ КОРИСТУВАЧІВ З ІНТЕГРАЦІЄЮ ПРИНЦИПІВ K-АНОНІМНОСТІ, L-РІЗНОМАНІТНОСТІ, T-БЛИЗЬКОСТІ, А ТАКОЖ КОНТЕЙНЕРИЗАЦІЇ ДАНИХ

У даному розділі буде розглянуто процес створення вдосконаленої системи анонімного обміну даними, що базується на використанні анонімної багатовузлової мережі користувачів. Особливу увагу буде приділено інтеграції сучасних методів анонімізації, таких як k-анонімність, l-різноманітність і t-близькість, які забезпечують високий рівень захисту приватності. Крім того, будуть розглянуті методи контейнеризації даних, що дозволяють гарантувати ізоляцію та безпеку інформації на всіх етапах її обробки та передачі.

Розділ включає дослідження особливостей запропонованої системи, її архітектури та основних функціональних компонентів. Буде розроблено алгоритм, що об'єднує принципи анонімізації з багатошаровою маршрутизацією та контейнеризацією даних, для досягнення максимальної анонімності та конфіденційності. Також буде представлена діаграма роботи системи, яка ілюструє основні етапи передачі даних через багатовузлову мережу.

Результати цього розділу закладають основу для подальшої реалізації системи, дозволяючи забезпечити високий рівень анонімності користувачів і безпеки даних у сучасних інформаційних мережах.

2.1 Особливості вдосконаленої системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів

Вдосконалена система анонімного обміну даними на основі багатовузлової мережі користувачів є інноваційним підходом до забезпечення приватності та конфіденційності даних. Основні особливості такої системи зосереджені на її багатошаровій архітектурі, що використовує принципи анонімності (k-анонімність, L-різноманітність і t-близькість) та контейнеризацію даних для ізоляції і контролю

доступу. У цій системі кожен вузол виконує певну роль у збереженні конфіденційності, використовуючи шифрування та маршрутизацію, щоб уникнути деканонімізації [26].

Основні особливості вдосконаленої системи:

Багатошарова архітектура для підвищення рівня анонімності

Система реалізована як мережа, де дані передаються через кілька анонімних вузлів, що виконують різні рівні шифрування. Кожен вузол в мережі знає лише попередній і наступний вузол, не маючи інформації про повний маршрут.

Для забезпечення надійної анонімності використовується багатошарове шифрування, що передбачає накладання декількох рівнів шифрування на дані. Це гарантує, що навіть при доступі до одного вузла зловмисник не отримає жодної критичної інформації про відправника або отримувача [27].

Інтеграція принципів анонімності: k-анонімність, L-різноманітність, t-близькість

k-анонімність забезпечує, що кожен запис у системі є невідрізним від щонайменше *k-1* інших записів, що підвищує анонімність шляхом приховування індивідуальних характеристик користувача серед інших.

L-різноманітність додає рівень захисту, забезпечуючи різноманітність чутливих атрибутів у межах групи, що запобігає можливості ідентифікації користувача за сукупністю атрибутів.

t-близькість контролює, щоб розподіл чутливих атрибутів у групі не надто відрізнявся від загального розподілу, зменшуючи ризик відстеження індивідуальних характеристик користувача навіть при наявності певної фонової інформації.

Контейнеризація для ізоляції та контролю доступу до даних

Контейнеризація даних дозволяє ізолювати окремі частини даних, створюючи обмежене середовище для кожного пакета інформації. Це знижує ризик витоку або доступу до даних без дозволу.

Кожен контейнер має індивідуальні налаштування безпеки та контролю доступу, що дозволяє обмежити взаємодію з іншими контейнерами та забезпечити

ізоляцію критичних даних. Така архітектура унеможливлює доступ до даних іншим компонентам мережі без авторизації, знижуючи ризик деканонімізації.

Контейнеризація також сприяє гнучкому розподілу даних між вузлами, полегшуючи контроль за доступом до інформації та мінімізуючи шанси на несанкціоноване розкриття особистих даних.

Адаптивна маршрутизація даних у багатовузловій мережі

Вдосконалена система реалізує динамічну маршрутизацію даних через різні вузли, що змінюється залежно від завантаженості та умов мережі. Це дозволяє зберегти ефективність та уникати фіксованих маршрутів, які можуть бути піддані аналізу трафіку [28].

Така маршрутизація знижує ризик ідентифікації відправника чи отримувача даних, оскільки навіть при спробі перехоплення або аналізу трафіку маршрут змінюється випадковим чином.

Використання Onion Routing або Garlic Routing забезпечує послідовне розшифрування шарів даних на кожному вузлі, надаючи кожному вузлу лише обмежену інформацію про дані та маршрут.

Шумові протоколи та мікс-мережі для підвищення рівня приватності

Система використовує шумові протоколи, які додають випадковий шум до кожного повідомлення. Це ускладнює аналіз трафіку, оскільки навіть при наявності доступу до кількох вузлів у зловмисника немає можливості відокремити реальні дані від шуму.

Мікс-мережі забезпечують випадкове перемішування повідомлень перед передачею через вузли мережі, знищуючи зв’язок між відправником і отримувачем. Це ускладнює процес аналізу трафіку та мінімізує ймовірність відстеження кінцевого пункту призначення даних.

Гнучкість і масштабованість системи

Контейнеризація та багатовузлова архітектура дозволяють легко масштабувати систему, додаючи нові вузли або збільшуючи обсяги оброблюваних даних. Це робить систему гнучкою та пристосованою до змін у розмірі чи завантаженості мережі [29].

Використання оркестраційних інструментів, таких як Kubernetes, полегшує управління контейнерами та дозволяє автоматизувати масштабування відповідно до потреб системи.

Захист від зовнішніх атак та забезпечення стійкості до збоїв

Кожен вузол та контейнер у системі мають власні політики безпеки та контролю доступу, що ускладнює можливість несанкціонованого втручання або доступу до даних.

Система здатна адаптуватися до збоїв, перенаправляючи трафік через інші вузли або автоматично відновлюючи доступність даних у разі виходу з ладу окремих компонентів. Це підвищує надійність і безперервність роботи мережі [30].

Особливості вдосконаленої системи анонімного обміну даними забезпечують високий рівень анонімності та приватності для користувачів завдяки використанню багатошарової шифрованої архітектури, контейнеризації, динамічної маршрутизації, а також шумових протоколів та мікс-мереж. Інтеграція принципів k-анонімності, L-різноманітності та t-близькості дозволяє мінімізувати ризики ідентифікації та забезпечити високий рівень захисту конфіденційних даних, навіть у багатовузлових мережах з великим обсягом інформації та трафіку.

2.2 Особливості вдосконаленої системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів

Розроблена вдосконалена система анонімного обміну даними базується на використанні багатошарових протоколів, які поєднують шифрування та маршрутизацію даних через анонімні вузли. Це дозволяє забезпечити високий рівень анонімності і конфіденційності, мінімізуючи ризик витоку інформації та ускладнюючи аналіз трафіку. Інтеграція принципів k-анонімності, L-різноманітності та t-близькості разом із багатошаровими протоколами забезпечує комплексний підхід до захисту даних у мережі.

Основні особливості використання багатошарових протоколів і принципів анонімності:

Поєднання Onion Routing та Garlic Routing для багатошарової маршрутизації

Onion Routing забезпечує передачу даних через декілька проміжних вузлів, де кожен шар шифрування розшифровується на відповідному вузлі, і кожен вузол знає лише сусідні вузли. Це унеможливлює ідентифікацію повного маршруту передачі, навіть якщо окремий вузол буде скомпрометований.

Garlic Routing дозволяє передавати кілька повідомлень у одному зашифрованому пакеті, де кожне повідомлення має окремий пункт призначення. Це ускладнює аналіз трафіку та відстеження окремих повідомлень, забезпечуючи додатковий рівень анонімності.

Забезпечення анонімності на основі k-анонімності

Принцип k-анонімності гарантує, що кожен запис, переданий через систему, неможливо відокремити від щонайменше *k-1* інших записів. Таким чином, навіть якщо дані будуть перехоплені, зловмисник не зможе виділити конкретного користувача або ідентифікувати його інформацію серед групи.

Це досягається шляхом групування користувачів за однаковими характеристиками, що дозволяє зберігати приватність в межах групи і забезпечує високу стійкість до атак деканонімізації.

Різноманітність даних за допомогою L-різноманітності

L-різноманітність передбачає, що кожна група даних має містити *L* різних значень чутливих атрибутів, що запобігає можливості ідентифікації конкретного користувача. Це дозволяє уникнути ідентифікації на основі унікальних характеристик.

Система автоматично формує групи так, щоб у кожній з них було достатньо варіацій чутливих атрибутів, знижуючи ризик витоку інформації, навіть якщо зловмисник має доступ до частини даних або їх квазі-ідентифікаторів.

Контроль розподілу чутливих атрибутів через t-близькість

Принцип t-близькості передбачає, що розподіл значень чутливих атрибутів у групах даних не повинен значно відрізнятися від загального розподілу в базі даних. Це забезпечує додатковий захист від атак на основі фонових знань, ускладнюючи аналіз інформації на основі рідкісних або незвичних атрибутів.

t-близькість унеможливлює ідентифікацію конкретного користувача через аналіз розподілу даних, оскільки кожен запис виглядає як частина загального тренду в мережі.

Додавання шуму для ускладнення аналізу трафіку

Для додаткового захисту використовується метод додавання шуму до трафіку, який створює випадкові дані, що передаються разом із реальними повідомленнями. Цей шум приховує реальні обсяги та структуру трафіку, запобігаючи аналізу та встановленню шаблонів трафіку.

Шумовий трафік передається через мережу разом з реальними даними, ускладнюючи спроби ідентифікації або аналізу вмісту. Це значно підвищує анонімність користувачів та запобігає атакам на основі аналізу обсягу трафіку.

Ізоляція та контроль даних за допомогою контейнеризації

Система використовує контейнеризацію для ізоляції даних у межах окремих контейнерів, кожен з яких має власні параметри доступу та шифрування. Контейнери ізолюють дані від інших вузлів і користувачів, що забезпечує додатковий рівень конфіденційності.

Кожен контейнер зберігає тільки ті дані, які необхідні для обробки на поточному вузлі, без передачі додаткової інформації. Це дозволяє контролювати доступ до інформації і мінімізувати ризик несанкціонованого доступу до конфіденційних даних.

Динамічна маршрутизація для підвищення безпеки

Динамічна маршрутизація передбачає автоматичну зміну маршрутів передачі даних через мережу, залежно від її стану або рівня завантаженості. Це запобігає спробам відстеження маршруту даних або визначенню початкового відправника і кінцевого одержувача.

Зміна маршрутів унеможливлює створення фіксованих шаблонів трафіку, що підвищує рівень анонімності та ускладнює моніторинг мережі з метою деканонімізації.

Автоматизація процесів через систему оркестрації контейнерів

Використання систем оркестрації контейнерів, таких як Kubernetes, дозволяє автоматизувати розподіл і управління контейнерами в межах мережі. Це включає контроль за ізоляцією даних, масштабування контейнерів і зміну маршрутів передачі залежно від поточної ситуації.

Оркестрація забезпечує гнучке управління ресурсами мережі, дозволяє динамічно додавати нові вузли, і забезпечує збереження високого рівня конфіденційності навіть при збільшенні навантаження на мережу.

Використання багатошарових протоколів і інтеграція принципів анонімності в багатовузлову мережу дозволяє забезпечити високий рівень захисту приватності даних. Поєднання Onion Routing і Garlic Routing, разом із використанням k-анонімності, L-різноманітності, t-близькості та шумових протоколів, створює надійну систему, що запобігає витоку інформації та ускладнює спроби відстеження. Контейнеризація та оркестрація дозволяють ізолювати дані та гнучко керувати ресурсами мережі, забезпечуючи високу масштабованість і надійність роботи.

2.3 Розробка алгоритму інтеграцією принципів k-анонімності, l-різноманітності, t-близькості, а також контейнеризації даних

Для забезпечення високого рівня анонімності та захисту конфіденційності в багатовузловій мережі розроблено алгоритм, що поєднує принципи k-анонімності, L-різноманітності та t-близькості з контейнеризацією даних. Цей алгоритм спрямований на зниження ризику ідентифікації користувачів у процесі обміну даними та захист конфіденційних атрибутів у мережі. Основою алгоритму є багатошаровий підхід до анонімізації даних і їх ізоляції в контейнерах, що підвищує безпеку та ефективність передачі інформації.

Розробимо даний алгоритм (рис.2.1).

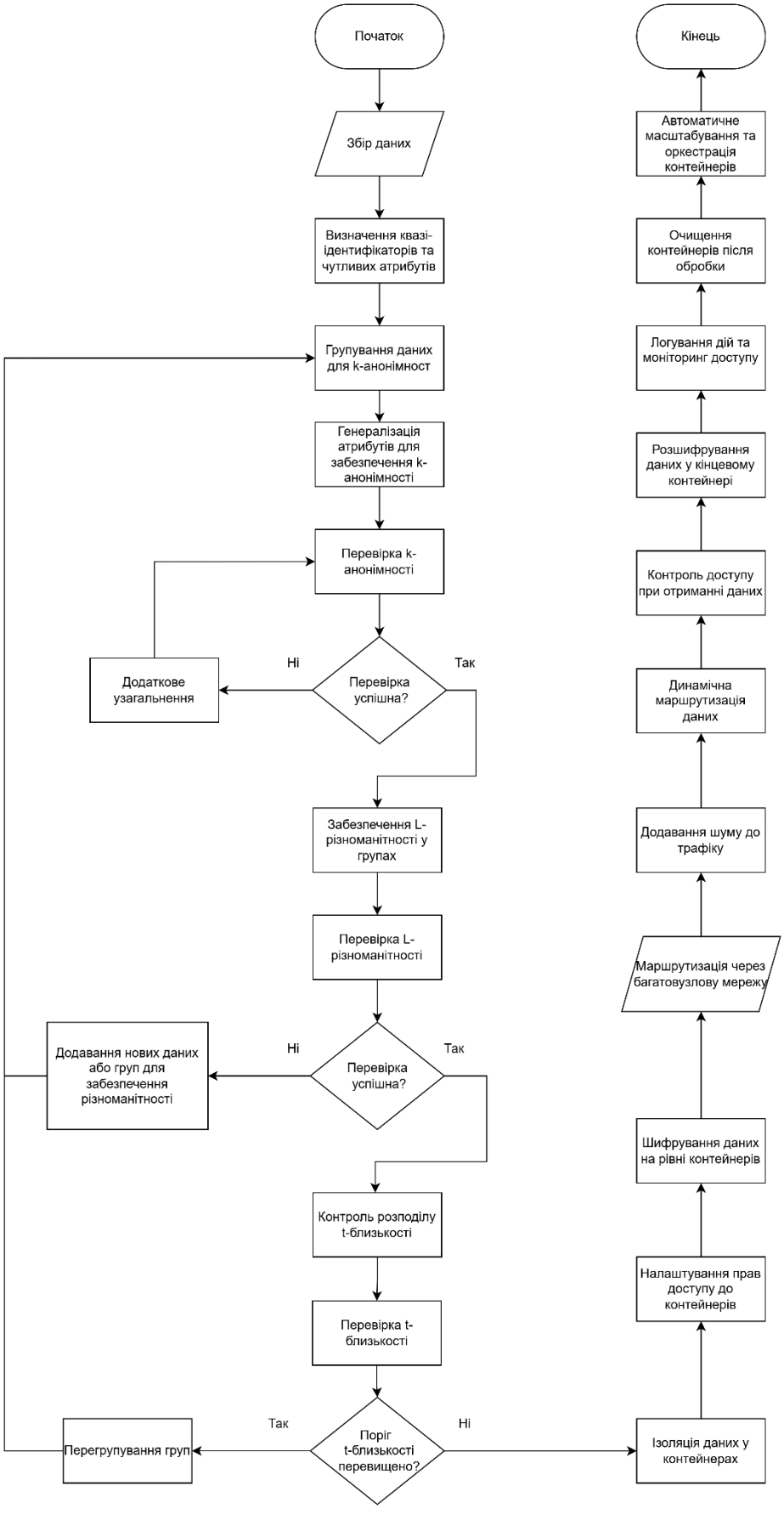


Рисунок 2.1 – Розроблений аолгоритм

Покроково розберемо даний алгоритм:

Крок 1: Збір даних

Збираються первинні дані користувачів, що містять ідентифікаційні та чутливі атрибути. Прикладом таких даних можуть бути демографічні відомості та особисті дані, які потребують захисту.

Крок 2: Визначення квазі-ідентифікаторів та чутливих атрибутів

Визначаються квазі-ідентифікатори (наприклад, вік, географічне розташування), які можуть бути використані для ідентифікації особи, та чутливі атрибути (наприклад, медичний стан або фінансова інформація).

Крок 3: Групування даних для k-анонімності

Дані групуються так, щоб кожен запис був невідрізним від k-1 інших записів у групі за квазі-ідентифікаторами. Це досягається шляхом узагальнення або придушення квазі-ідентифікаторів.

Крок 4: Генералізація атрибутів для забезпечення k-анонімності

Проводиться генералізація (наприклад, перетворення віку з «25 років» на діапазон «20-30 років»), щоб зменшити деталізацію даних, покращуючи ступінь анонімності.

Крок 5: Перевірка k-анонімності

Алгоритм перевіряє, чи кожна група відповідає рівню k-анонімності (тобто кожен запис неможливо відрізнити від щонайменше k-1 інших записів). Якщо рівень не досягнуто, проводиться додаткове узагальнення.

Крок 6: Забезпечення L-різноманітності у групах

Внутрішньо групується чутливий атрибут кожного запису таким чином, щоб кожна група мала щонайменше L різних значень для чутливих атрибутів. Це знижує ризик ідентифікації на основі чутливих даних.

Крок 7: Перевірка L-різноманітності

Алгоритм перевіряє різноманітність значень чутливих атрибутів у кожній групі. Якщо значень недостатньо, додаються нові дані або група переглядається для забезпечення різноманітності.

Крок 8: Контроль розподілу t-близькості

Визначається допустимий рівень t для забезпечення, що розподіл чутливих атрибутів у кожній групі незначно відрізняється від загального розподілу атрибутів у всій базі даних.

Крок 9: Перевірка t-близькості

Алгоритм аналізує групи, перевіряючи, чи відповідає розподіл чутливих атрибутів встановленому рівню t-близькості. Якщо поріг перевищено, групи перегруповуються або атрибути коригуються для досягнення відповідності.

Крок 10: Ізоляція даних у контейнерах

Дані кожної групи розподіляються в окремі контейнери для ізоляції. Контейнеризація забезпечує обмеження доступу до даних у кожному контейнері та підвищує конфіденційність.

Крок 11: Налаштування прав доступу до контейнерів

Визначаються права доступу до контейнерів, що обмежують можливість доступу лише до авторизованих вузлів і користувачів, запобігаючи несанкціонованому доступу.

Крок 12: Шифрування даних на рівні контейнерів

Кожен контейнер шифрується унікальним ключем шифрування, який забезпечує захист даних на всіх етапах передачі та зберігання. Для шифрування використовується багаторівневий алгоритм, наприклад, AES-256.

Крок 13: Маршрутизація через багатовузлову мережу

Для забезпечення анонімності дані передаються через кілька анонімних вузлів мережі з використанням Onion Routing або Garlic Routing. На кожному вузлі знімається один шар шифрування, що унеможливлює відстеження повного маршруту.

Крок 14: Додавання шуму до трафіку

Для додаткової анонімізації створюється випадковий трафік (шум), що імітує реальні дані, ускладнюючи аналіз трафіку та запобігаючи спробам визначити реальні повідомлення.

Крок 15: Динамічна маршрутизація даних

Для зниження ризику ідентифікації маршрут даних змінюється на основі завантаженості мережі та інших параметрів, що унеможливлює фіксований шлях передачі даних і знижує ризик відстеження.

Крок 16: Контроль доступу при отриманні даних

Після досягнення кінцевого вузла дані доступні тільки авторизованим користувачам, що мають відповідні ключі розшифрування. Це дозволяє обмежити доступ до даних лише для кінцевого отримувача.

Крок 17: Розшифрування даних у кінцевому контейнері

Після досягнення кінцевого вузла дані всередині контейнера розшифровуються, і доступ до них надається тільки для авторизованого кінцевого отримувача.

Крок 18: Логування дій та моніторинг доступу

Всі дії з даними фіксуються у журналах для забезпечення аудиту і моніторингу. Це включає запис маршруту передачі, вузлів, через які проходили дані, і ключових етапів обробки.

Крок 19: Очищення контейнерів після обробки

Після завершення обробки та доставки даних кожен контейнер очищується, видаляються тимчасові файли, ключі шифрування та будь-які інші проміжні дані, щоб уникнути можливості витоку інформації.

Крок 20: Автоматичне масштабування та оркестрація контейнерів

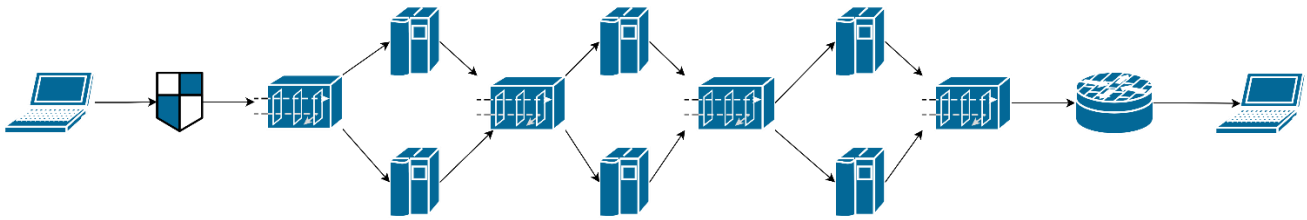
Система автоматично масштабується залежно від навантаження мережі та кількості оброблюваних запитів. Використовується оркестрація (наприклад, Kubernetes) для динамічного управління контейнерами, що підтримує безперебійний і захищений обмін даними.

Цей алгоритм забезпечує комплексну реалізацію приватності в багатовузловій системі анонімного обміну даними. Інтеграція принципів k-анонімності, L-різноманітності, t-близькості, контейнеризації та багатошарових протоколів дозволяє надійно захищати дані користувачів від можливих загроз, пов'язаних із деканонімізацією або витоком інформації.

2.4 Розробка діаграми роботи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів

Система анонімного обміну даними в багатовузловій мережі призначена для забезпечення конфіденційності та анонімності користувачів під час передачі інформації через мережу. Основний принцип роботи цієї системи полягає в передачі даних через кілька проміжних вузлів, використовуючи багатошарове шифрування (Onion Routing) і динамічну маршрутизацію. Це дозволяє приховати джерело та кінцеву точку передачі інформації, запобігаючи можливості ідентифікації користувача.

Система інтегрує механізми захисту, такі як k-анонімність, L-різноманітність, t-близькість і контейнеризація даних. Дані ізолюються в контейнерах, що забезпечує захист і обмеження доступу, а передача даних через випадкові вузли ускладнює можливість їхнього аналізу або перехоплення.



Рискнок 2.2 - Діаграма роботи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів

Основні компоненти діаграми роботи

1. Відправник (Користувач)

Початкова точка, з якої передаються дані. Відправник формує контейнер з даними, включає шари шифрування для кожного вузла маршруту і передає контейнер у мережу.

2. Контейнери даних

Кожен контейнер ізолює дані від зовнішніх компонентів, використовуючи шифрування та контроль доступу. Це забезпечує захист від несанкціонованого

доступу на кожному вузлі. Контейнери зберігають лише анонімізовані дані та захищають чутливу інформацію.

3. Анонімні вузли мережі

Проміжні точки, через які передається контейнер. Кожен вузол знімає лише один шар шифрування, розкриваючи тільки наступний пункт маршруту. Вузли не мають інформації про кінцевого отримувача або початкового відправника.

4. Протокол Onion Routing

Основний протокол передачі даних, який забезпечує багатошарове шифрування. Кожен шар шифрування знімається на конкретному вузлі, що дозволяє передавати дані через мережу анонімно. Це запобігає відстеженню маршруту і ускладнює деканонімізацію даних.

5. Шумовий трафік

Генерується додатковий трафік, який маскує справжні дані та захищає від аналізу трафіку. Шумовий трафік створюється на кожному вузлі для приховування реальної структури трафіку.

6. Проміжні мікс-вузли

Додають випадковість до передачі даних, перемішуючи повідомлення і таким чином ускладнюючи їхній аналіз. Це забезпечує додатковий рівень захисту, роблячи неможливим визначення маршруту даних.

7. Кінцевий вузол (Отримувач)

Фінальний пункт передачі, де дані розшифровуються та передаються кінцевому отримувачу. Кінцевий вузол отримує повністю розшифровані дані без можливості ідентифікації відправника.

8. Оркестрація та контроль доступу

Система, яка контролює доступ до контейнерів та управляє правами доступу. Вона забезпечує, що дані можуть отримати лише авторизовані користувачі, і автоматизує процес маршрутизації та ізоляції даних.

Ці компоненти забезпечують анонімність, конфіденційність та безпечний обмін даними у багатовузловій мережі.

2.5 Висновки до розділу.

Розділ висвітлює створення вдосконаленої системи анонімного обміну даними, яка використовує анонімну багатовузлову мережу та принципи k-анонімності, L-різноманітності, t-близькості й контейнеризації даних. Застосування таких підходів забезпечує високий рівень конфіденційності та приватності користувачів під час передачі даних. Інтеграція багатошарового шифрування через Onion Routing у кожному вузлі дозволяє приховати маршрут та ідентифікаційні дані користувача, роблячи відстеження неможливим. Контейнеризація даних із налаштуванням доступу та ізоляції інформації додатково захищає дані від несанкціонованого доступу, створюючи надійний бар’єр на кожному етапі передачі. Принципи k-анонімності, L-різноманітності й t-близькості у поєднанні з проміжними вузлами та шумовим трафіком мінімізують можливість деканонімізації даних або ідентифікації користувача, навіть у випадку, якщо частина мережі буде скомпрометована. Впроваджена система оркестрації й управління контейнерами також підвищує гнучкість та масштабованість мережі, забезпечуючи безперебійну обробку трафіку та динамічну маршрутизацію даних залежно від умов мережі. Результатом є надійна та масштабована система, здатна забезпечити високий рівень приватності в умовах багатовузлової мережі, що відповідає сучасним вимогам захисту інформації та приватності користувачів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВДОСКОНАЛЕНОЇ СИСТЕМИ АНОНІМНОГО ОБМІНУ ДАНИМИ НА ОСНОВІ АНОНІМНОЇ БАГАТОВУЗЛОВОЇ МЕРЕЖІ КОРИСТУВАЧІВ З ІНТЕГРАЦІЄЮ ПРИНЦИПІВ K-АНОНІМНОСТІ, L-РІЗНОМАНІТНОСТІ, T-БЛИЗЬКОСТІ, А ТАКОЖ КОНТЕЙНЕРИЗАЦІЇ ДАНИХ

У даному розділі буде розглянуто програмну реалізацію вдосконаленої системи анонімного обміну даними, яка базується на використанні анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, l-різноманітності, t-близькості, а також контейнеризації даних. Реалізація системи включає вибір оптимальних технологій, що забезпечують ефективну роботу алгоритмів анонімізації, захисту даних та багатошарової маршрутизації.

Спочатку буде обґрунтовано вибір мови програмування, яка відповідає сучасним вимогам до розробки захищених систем, та середовища розробки, що забезпечує зручність і продуктивність під час роботи з проектом. Далі розглядається процес реалізації основних модулів системи, таких як модуль анонімного обміну даними, інтеграція принципів приватності (k-анонімності, l-різноманітності, t-близькості) та контейнеризації даних для захисту та ізоляції інформації.

Особливу увагу приділено тестуванню системи для перевірки її функціональності, продуктивності та стійкості до потенційних загроз. Результати тестування демонструють відповідність розробленої системи сучасним вимогам безпеки та приватності.

Цей розділ завершується аналізом отриманих результатів та їх значенням для забезпечення анонімного обміну даними в інформаційних мережах. Представлений підхід забезпечує високу конфіденційність користувачів і надійність системи в умовах реальних загроз.

3.1 Обгрунтування вибору мови програмування

Для реалізації вдосконаленої системи анонімного обміну даними була обрана мова програмування Python. Цей вибір базується на поєднанні гнучкості, продуктивності, підтримки сучасних бібліотек і простоти інтеграції з іншими компонентами системи, такими як шифрування, контейнеризація та обробка даних. Python є одним із найпоширеніших інструментів для роботи з мережами, обчисленнями та анонімними системами завдяки своєму багатому екосистемному набору бібліотек і підтримці сучасних стандартів безпеки [31].

Мова програмування Python має багатий набір бібліотек, які значно спрощують обробку даних і реалізацію механізмів анонімності, таких як k-анонімність, L-різноманітність і t-близькість. Завдяки своїй екосистемі Python дозволяє ефективно працювати з великими масивами даних, групувати їх, виконувати анонімізацію та перевіряти відповідність принципам конфіденційності.

Python надає значну зручність у роботі з шифруванням та багатошаровими протоколами завдяки наявності сучасних бібліотек, що дозволяють реалізовувати складні криптографічні механізми із мінімальними витратами часу на розробку. Наприклад, бібліотека **PyCryptodome** забезпечує широкий набір інструментів для симетричного шифрування, включаючи алгоритми AES, які підходять для реалізації багатошарового шифрування в системах типу Onion Routing. Ця бібліотека дозволяє легко створювати ключі, шифрувати дані та забезпечувати їхню безпеку під час передачі через мережу.

Для роботи з асиметричним шифруванням та керування ключами часто використовується бібліотека **cryptography**, яка підтримує стандарти, такі як RSA і ECC. Вона дозволяє не лише шифрувати та розшифровувати дані, але й реалізовувати цифрові підписи та сертифікати, що необхідні для побудови довірчих відносин між вузлами багатовузлової мережі [32].

Python також відзначається своєю зручністю у роботі з багатошаровими протоколами, такими як Onion Routing, завдяки можливості реалізації послідовного шифрування даних для кожного вузла маршруту. Наприклад, кожен шар

шифрується окремим ключем, а розшифрування виконується вузлами поетапно, що легко реалізується за допомогою вкладених функцій шифрування. Також Python підтримує асинхронну маршрутизацію, використовуючи модулі на кшталт **asyncio**, що дозволяє одночасно передавати зашифровані повідомлення через декілька вузлів, забезпечуючи високу швидкість і безпеку передачі даних.

Окрім цього, Python пропонує бібліотеки для роботи з протоколами безпеки, такими як TLS/SSL, через модуль **ssl**, який забезпечує шифрування на транспортному рівні. Це дозволяє створювати додатковий рівень захисту для з'єднань між вузлами, підвищуючи загальний рівень конфіденційності в мережі. Усі ці інструменти легко інтегруються в Python-додатки, що робить мову ідеальним вибором для реалізації криптографічних протоколів та багатошарових механізмів захисту даних [33].

Python є однією з найзручніших мов для інтеграції з контейнеризацією та оркестрацією завдяки доступу до потужних бібліотек і API, які дозволяють ефективно працювати з Docker, Kubernetes та іншими інструментами управління контейнерами. Використання Python для контейнеризації дозволяє швидко створювати, розгортати та масштабувати ізольовані середовища, забезпечуючи високий рівень безпеки та гнучкості. Бібліотека **Docker SDK for Python** забезпечує повний контроль над контейнерами: створення образів, управління мережами, обробка обсягів даних і запуск контейнерів із заданими параметрами. Це дозволяє інтегрувати контейнеризацію безпосередньо в процеси розробки, наприклад, упакувати програму або її частини в окремі контейнери для ізоляції даних та обмеження доступу до них.

Python також дозволяє легко інтегрувати системи оркестрації, такі як Kubernetes, за допомогою бібліотеки **kubernetes-python-client**. Ця бібліотека надає зручний інтерфейс для взаємодії з API Kubernetes, що дозволяє автоматизувати управління контейнерами, їхнє масштабування, моніторинг і відновлення у випадку збоїв. Наприклад, Python-скрипти можуть автоматично запускати нові екземпляри контейнерів для обробки підвищеного навантаження або

перенаправляти трафік між вузлами для забезпечення оптимальної продуктивності системи.

Крім того, Python підтримує інструменти для інтеграції з CI/CD-пайплайнами, наприклад, через **Jenkins** або **GitHub Actions**, що дозволяє автоматизувати процеси створення, тестування та розгортання контейнерів у хмарних середовищах. Завдяки можливості використовувати Python як для розробки, так і для управління інфраструктурою, контейнеризація та оркестрація стають невід'ємною частиною розробки систем, що вимагають ізоляції та високої масштабованості. Ця інтеграція значно підвищує ефективність роботи з контейнерами, забезпечуючи гнучкість у використанні ресурсів і надійність системи [34].

Python відзначається високою гнучкістю в роботі з мережевими системами завдяки широкому набору інструментів для створення, управління та аналізу мережевих з'єднань. Це робить Python ідеальним вибором для розробки систем, які потребують складної маршрутизації, динамічного управління вузлами та забезпечення анонімності. Базові можливості роботи з мережею забезпечуються стандартними модулями, такими як **socket**, який дозволяє створювати сервери та клієнтські програми для обміну даними через різні протоколи, зокрема TCP та UDP. За допомогою цього модуля можна розробляти власні мережеві протоколи або налаштовувати специфічні з'єднання між вузлами.

Асинхронна робота з мережею, що є важливою для систем з високим навантаженням, легко реалізується за допомогою **asyncio**. Цей модуль дозволяє одночасно обробляти декілька мережевих запитів без блокування, що забезпечує високу продуктивність і швидкість передачі даних навіть у масштабованих багатовузлових мережах. Python також підтримує високорівневі бібліотеки, такі як **requests** для роботи з HTTP-запитами, що спрощує інтеграцію з веб-сервісами або API інших систем. Для асинхронного HTTP, зокрема в системах реального часу, використовується бібліотека **aiohttp**, яка дозволяє ефективно працювати з великою кількістю одночасних з'єднань.

У сфері побудови мережевих протоколів Python також пропонує інструменти для реалізації багатошарових моделей, наприклад, Onion Routing. Легкість роботи з багатошаровим шифруванням та маршрутизацією робить Python підходящим для розробки анонімних мереж. Наприклад, можна налаштувати передачу даних через вузли мережі, де кожен вузол обробляє лише свій шар інформації, не маючи доступу до повного маршруту або вмісту повідомлення [35].

Python також має потужні інструменти для моніторингу мережевого трафіку, такі як бібліотека **scapy**, яка дозволяє аналізувати та створювати пакети будь-якого рівня мережевої моделі. Це корисно для тестування безпеки системи, моделювання різних сценаріїв роботи мережі або перевірки ефективності шифрування.

Таким чином, Python забезпечує високу гнучкість у роботі з мережею, дозволяючи реалізовувати як низькорівневі з'єднання, так і складні багатошарові протоколи. Його потужні бібліотеки та модулі дають можливість швидко адаптуватися до змін мережевих умов і створювати високопродуктивні системи, що відповідають сучасним вимогам до приватності та масштабованості.

Python забезпечує високу масштабованість і простоту впровадження завдяки своїй універсальності, великій екосистемі бібліотек і підтримці сучасних інструментів автоматизації. Ці якості роблять його ідеальним вибором для створення як прототипів, так і готових до розгортання систем, які можуть адаптуватися до зростаючих вимог. Масштабованість Python досягається через інтеграцію з такими бібліотеками та платформами, як **Dask** або **PySpark**, що дозволяють працювати з великими обсягами даних у розподілених середовищах. Це особливо важливо для багатовузлових систем, де потрібно обробляти запити від великої кількості користувачів або обмінюватися великими обсягами даних [36].

Простота впровадження Python обумовлена його зрозумілим синтаксисом і доступом до інструментів для роботи з контейнеризацією та оркестрацією. Наприклад, використання Docker SDK for Python дозволяє швидко упакувати програми в контейнери, ізолюючи середовища для тестування та розгортання, що підвищує їхню безпеку і знижує ризик конфліктів залежностей. Для автоматизації масштабування системи, наприклад додавання нових вузлів у багатовузловій

мережі, Python легко інтегрується з Kubernetes через kubernetes-python-client. Це спрощує управління контейнерами та їхнє оркестрування, дозволяючи системі автоматично адаптуватися до змін навантаження.

Ще однією ключовою перевагою Python є його інтеграція з інструментами для автоматизації DevOps, такими як Ansible, Terraform або CI/CD-платформи (наприклад, Jenkins або GitHub Actions). Це дозволяє легко впроваджувати оновлення, запускати тести та розгортати нові версії програмного забезпечення. Завдяки цьому Python забезпечує швидке налаштування та підтримку безперервного циклу розробки.

Зручність Python у роботі з мікросервісною архітектурою також сприяє його масштабованості. Модулі, створені на Python, можуть взаємодіяти через REST або gRPC API, що дозволяє інтегрувати їх у складніші системи або працювати як автономні сервіси. Це полегшує розширення функціоналу системи без необхідності значних змін у її базовій структурі [37].

Таким чином, Python дозволяє швидко створювати масштабовані системи, які легко адаптуються до змін і можуть бути впроваджені як у локальному, так і в хмарному середовищі. Його потужний набір інструментів і підтримка сучасних технологій роблять процес розгортання та масштабування систем максимально ефективним і простим.

3.2 Обгрунтування вибору середовища розробки

Для реалізації вдосконаленої системи анонімного обміну даними було обрано середовище розробки Visual Studio Code (VS Code). Це потужний, гнучкий та ефективний редактор коду, який відповідає сучасним вимогам розробки багатофункціональних систем. Його особливості забезпечують продуктивність, зручність роботи та інтеграцію з інструментами, які необхідні для реалізації системи.

VS Code забезпечує інтеграцію з багатьма мовами програмування, включаючи Python, що є основною мовою для розробки даної системи. Завдяки розширенню Python Extension, VS Code надає можливості автодоповнення коду, діагностики помилок, налаштування середовища виконання та інтеграції з системами тестування. Це значно пришвидшує процес розробки та забезпечує ефективну роботу з великими проєктами.

Ще однією вагомою причиною вибору VS Code є його інтеграція з системами керування версіями, такими як Git та GitHub. Розробники можуть легко відстежувати зміни, працювати з гілками та автоматизувати процеси CI/CD безпосередньо зі середовища розробки. Це особливо важливо для командної роботи та забезпечення безперервного розвитку системи.

Середовище також підтримує інтеграцію з Docker через розширення Docker Extension for VS Code, що дозволяє створювати, запускати та тестувати контейнери безпосередньо у редакторі. Це забезпечує швидкий цикл розробки та тестування компонентів системи, які використовують контейнеризацію даних для ізоляції та безпеки [38].

Однією з ключових переваг VS Code є його легкість і кросплатформеність. Середовище працює на всіх основних операційних системах (Windows, macOS, Linux) і не вимагає значних ресурсів, що робить його доступним для розробників незалежно від апаратного забезпечення. Завдяки цьому VS Code може використовуватись як для локальної розробки, так і для роботи у віддалених середовищах, наприклад, у хмарних сервісах чи на серверах через SSH.

Розширення Remote - SSH та Remote - Containers дозволяють працювати безпосередньо з віддаленими машинами або контейнерами, що забезпечує зручність у випадках, коли середовище розробки розгортається у хмарі. Це також дозволяє оптимізувати використання ресурсів та забезпечує роботу над проєктом з будь-якого місця.

Крім того, VS Code підтримує широкий спектр тем, плагінів і налаштувань, що дозволяють адаптувати середовище під конкретні потреби розробника. Наприклад, розширення для роботи з Python, Docker, Kubernetes і тестовими

фреймворками забезпечують повноцінну інтеграцію всіх необхідних інструментів для розробки системи [39].

Крім того, Visual Studio Code (VS Code) підтримує широкий спектр тем, плагінів і налаштувань, що дозволяють розробникам адаптувати середовище під свої індивідуальні потреби та підвищити продуктивність роботи. Завдяки цьому кожен розробник може налаштувати редактор відповідно до стилю своєї роботи, специфіки проєкту чи команди.

Підтримка тем оформлення дозволяє змінювати вигляд редактора, включаючи кольорові схеми, оформлення шрифтів та іконок. Це не лише робить середовище зручнішим для тривалого використання, але й допомагає уникнути перевтоми очей, що важливо для розробників, які працюють з кодом у режимі нон-стоп. Серед популярних тем є Dracula, Monokai Pro, One Dark Pro, які забезпечують високу контрастність і чіткість для кращої читабельності коду.

Система плагінів (розширень) у VS Code є однією з найсильніших сторін редактора. Наприклад, плагін Python Extension забезпечує інтеграцію з мовою Python, включаючи підтримку автодоповнення, діагностики помилок, рефакторингу коду та вбудованої підтримки тестових фреймворків. Плагіни, такі як Docker Extension, дозволяють інтегрувати роботу з контейнерами безпосередньо в середовище, що особливо важливо для реалізації системи з контейнеризацією. Також існують розширення для роботи з Kubernetes, REST API, базами даних, які забезпечують ефективну інтеграцію додаткових функцій у процес розробки.

Налаштування VS Code є надзвичайно гнучкими. Користувачі можуть змінювати майже всі аспекти роботи редактора через файл налаштувань settings.json або через зручний інтерфейс, що надає графічний доступ до основних параметрів. Наприклад, можна налаштувати форматування коду, гарячі клавіші, поведінку термінала, відображення повідомлень або автоматизацію виконання завдань. Це дозволяє створити оптимізоване середовище, яке знижує кількість рутинних дій та підвищує ефективність [40].

Розширення для роботи з гілками Git, такі як GitLens, додають зручні функції для аналізу історії змін, пошуку авторів змін у коді або порівняння різних версій

файлів. Це спрощує управління репозиторіями і робить VS Code зручним вибором для командної роботи.

Крім того, розширення для інтеграції з хмарними сервісами, такими як AWS, Azure, чи Google Cloud, дозволяють працювати з ресурсами прямо з редактора. Наприклад, можна керувати хмарними функціями, моніторити логі або запускати тестові середовища без необхідності перемикатися між інструментами.

Ця гнучкість у налаштуваннях і підтримка широкого спектра плагінів роблять VS Code універсальним інструментом для розробників, який можна адаптувати для роботи над проєктами будь-якої складності чи спрямованості. Це забезпечує не лише зручність, але й ефективність при роботі з різними технологіями та інструментами.

Вибір VS Code обумовлений його універсальністю, потужною екосистемою розширень та зручністю використання. Завдяки цьому середовище дозволяє швидко розробляти, тестувати та впроваджувати складні системи, забезпечуючи при цьому комфортну та ефективну роботу розробників.

3.3 Реалізація модулю анонімного обміну даними на основі анонімної багатовузлової мережі користувачів

Модуль анонімного обміну даними створюється для забезпечення конфіденційності користувачів шляхом використання багатошарової маршрутизації та шифрування в анонімній багатовузловій мережі. Основна функція цього модуля — передача даних через низку вузлів так, щоб жоден із них не мав повної інформації про відправника, отримувача або зміст повідомлення.

На початковому етапі система створює повідомлення, яке передаватиметься через багатовузлову мережу. Для забезпечення анонімності дані шифруються багатошаровим методом, де кожен шар відповідає певному вузлу. Кожен вузол отримує лише свою частину повідомлення, розшифровує її та пересилає залишок далі. Функція form_message додає шифрувальні шари до повідомлення у

зворотному порядку. Кожен шар містить ключ шифрування, адресу наступного вузла та зашифровані дані.

```
from Crypto.Cipher import AES
import os

def encrypt_layer(data, key):

def form_message(data, keys, nodes):
   for key, node in zip(keys[::-1], nodes[::-1]):
      data = encrypt_layer(data + node.encode(), key)
   return data

# Приклад використання
message = b"Hello, this is confidential!"
keys = [os.urandom(16) for _ in range(3)]  # Генерація ключів для кожного шару
nodes = ["NodeA", "NodeB", "NodeC"]  # Адреси вузлів
encrypted_message = form_message(message, keys, nodes)
print("Багатошарово зашифроване повідомлення:", encrypted_message)
```

Для збереження анонімності дані передаються через випадково вибраний маршрут. Вузли маршруту обираються з набору доступних, і їх порядок постійно змінюється, щоб унеможливити передбачення маршруту. Функція choose_route обирає випадковий підмножину вузлів, які будуть використовуватись для передачі даних.

```
import random def choose_route(nodes, num_hops=3):
return random.sample(nodes, num_hops)
available_nodes = ["NodeA", "NodeB", "NodeC", "NodeD", "NodeE"] route = choose_route(available_nodes, num_hops=3) print("Вибраний маршрут:", route)
```

Після шифрування і визначення маршруту дані передаються через мережу. Кожен вузол приймає повідомлення, дешифрує один шар і передає решту наступному вузлу. Функція send_to_node реалізує передачу даних через TCP-з'єднання.

```
import socket def send_to_node(data, address): """ Передає дані через TCP-з'єднання на вказану адресу. data: Байти, які потрібно передати. address: Кортеж із IP-адреси та порту вузла. """ with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s: s.connect(address) s.sendall(data) # Приклад використання data_to_send = b"Encrypted data" node_address = ("127.0.0.1", 8080) # Локальний вузол send_to_node(data_to_send, node_address)
```

Кожен вузол отримує повідомлення, дешифрує свій шар шифрування, вилучає адресу наступного вузла та передає дані далі. Дешифрування виконується лише із використанням ключа, відомого цьому вузлу. Функція decrypt_layer знімає один шар шифрування.

```
def decrypt_layer(data, key):
   nonce, tag, ciphertext = data[:16], data[16:32], data[32:]
   cipher = AES.new(key, AES.MODE_EAX, nonce)
   return cipher.decrypt_and_verify(ciphertext, tag)
```

```
# Приклад використання
decrypted_data = decrypt_layer(encrypted_message, keys[-1])
print("Розшифровані дані на вузлі:", decrypted_data)
```

Таким чином, модуль забезпечує анонімність передачі даних через багатовузлову мережу за допомогою багатошарового шифрування, динамічної маршрутизації та ізоляції обробки кожного шару на вузлах. Кожен компонент інтегрується в єдиний потік, забезпечуючи повну конфіденційність і безпеку.

3.4 Реалізація модулю інтеграції принципів k-анонімності, L-різноманітності, t-близькості

Для забезпечення конфіденційності даних перед їхньою передачею через багатовузлову мережу використовуються принципи анонімізації — **k-анонімність**, **L-різноманітність** і **t-близькість**. Цей модуль виконує попередню обробку даних, згруповуючи їх відповідно до заданих принципів, що мінімізує ризик деканонімізації.

Опис принципів

k-анонімність забезпечує, що кожен запис у наборі даних не може бути ідентифікований серед щонайменше k інших записів. Це досягається шляхом узагальнення або придушення атрибутів.

L-різноманітність гарантує, що кожна група, створена для забезпечення k-анонімності, містить щонайменше L різних значень чутливих атрибутів.

t-близькість обмежує відхилення розподілу чутливих атрибутів у кожній групі, порівняно з їх загальним розподілом.

Модуль складається з кількох функцій для обробки даних. Кожна функція відповідає за реалізацію одного з принципів анонімізації.

Реалізація k-анонімності

Дані групуються за квазі-ідентифікаторами, такими як вік, стать або географічне розташування. Якщо група містить менше ніж *k* записів, атрибути узагальнюються, доки група не досягне заданого розміру.

```
import pandas as pd def apply_k_anonymity(data, quasi_identifiers, k): """ Застосовує принцип k-анонімності до набору даних. data: DataFrame із записами. quasi_identifiers: Список стовпців, які є квазі-ідентифікаторами. k: Рівень анонімності. Повертає: Анонімізований DataFrame. """ grouped
```

```
= data.groupby(quasi_identifiers) anonymized = grouped.filter(lambda x: len(x) >= k) # Узагальнення, якщо групи менші за k for name, group in grouped: if len(group) < k: for col in quasi_identifiers: data[col] = data[col].apply(lambda x: f"{x}-generalized") return data # Приклад використання data = pd.DataFrame({ 'age': [25, 32, 45, 25, 32], 'city': ['Kyiv', 'Lviv', 'Odesa', 'Kyiv', 'Lviv'], 'salary': [50000, 60000, 70000, 50000, 60000] }) anonymized_data = apply_k_anonymity(data, ['age', 'city'], k=2) print(anonymized_data)
```

Реалізація L-різноманітності

Кожна група даних, що відповідає k-анонімності, повинна містити щонайменше *L* різних значень чутливого атрибута, наприклад, діагнозу чи доходу. Якщо виявлено нестачу різноманітності, дані коригуються.

```
def ensure_l_diversity(data, sensitive_attribute, l):
    """
    Перевіряє та забезпечує L-різноманітність у групах даних.
    data: DataFrame із групами.
    sensitive_attribute: Назва чутливого атрибута.
    l: Рівень різноманітності.
    Повертає: Анонімізований DataFrame.
    """
    grouped = data.groupby(sensitive_attribute)
    valid_groups = grouped.filter(lambda x: len(x[sensitive_attribute].unique()) >= l)

    # Видалення недостатньо різноманітних груп
    for name, group in grouped:
        if len(group[sensitive_attribute].unique()) < l:
            data = data.drop(group.index)
    return data

# Приклад використання
l_diverse_data = ensure_l_diversity(anonymized_data, 'salary', l=2)
print(l_diverse_data)
```

Реалізація t-близькості

Розподіл значень чутливих атрибутів у кожній групі має бути близьким до розподілу в загальній базі даних. Це гарантує, що рідкісні значення не виділятимуться в групах.

```
from scipy.stats import ks_2samp

def ensure_t_closeness(data, sensitive_attribute, t):
    """
    Забезпечує t-близькість у групах даних.
    data: DataFrame із групами.
    sensitive_attribute: Назва чутливого атрибута.
    t: Допустиме відхилення розподілу.
    Повертає: Анонімізований DataFrame.
    """
    global_distribution = data[sensitive_attribute].value_counts(normalize=True)

    def is_t_close(group):
        group_distribution = group[sensitive_attribute].value_counts(normalize=True)
        stat, p_value = ks_2samp(global_distribution, group_distribution)
        return stat <= t
```

```
    return data.groupby(sensitive_attribute).filter(is_t_close)

# Приклад використання
t_close_data = ensure_t_closeness(l_diverse_data, 'salary', t=0.1)
print(t_close_data)
```

Усі три принципи інтегруються у модуль, який послідовно застосовує k-анонімність, L-різноманітність і t-близькість. Дані проходять кожен етап обробки, доки не досягнуть заданих рівнів анонімізації.

```
def anonymize_data(data, quasi_identifiers, sensitive_attribute, k, l, t):
    """
    Повна анонімізація даних із застосуванням усіх принципів.
    """
    data = apply_k_anonymity(data, quasi_identifiers, k)
    data = ensure_l_diversity(data, sensitive_attribute, l)
    data = ensure_t_closeness(data, sensitive_attribute, t)
    return data

# Виклик модуля
final_anonymized_data = anonymize_data(data, ['age', 'city'], 'salary', k=2, l=2, t=0.1)
print(final_anonymized_data)
```

Цей модуль дозволяє підготувати дані для анонімного обміну, знижуючи ризик деканонімізації завдяки комплексному використанню k-анонімності, L-різноманітності та t-близькості.

3.5 Реалізація модулю контейнеризації даних

Модуль контейнеризації даних призначений для ізоляції даних у захищених середовищах, забезпечуючи їхню безпеку під час обробки та передачі через багатовузлову мережу. Контейнери дозволяють упакувати дані разом із метаданими, політиками доступу та шифрувальними ключами, що гарантує обмежений доступ до інформації та мінімізує ризик витоку.

Контейнер включає в себе дані, метадані (наприклад, адресу одержувача) і ключ для шифрування. Дані всередині контейнера серіалізуються для передачі через мережу.

```
import json
import base64
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class DataContainer:
    """
    Клас для створення контейнерів даних.
```

```
    """
    def __init__(self, data, recipient, access_key):
        """
        Ініціалізує контейнер із даними, отримувачем і ключем доступу.
        data: Дані для упаковки.
        recipient: Ідентифікатор отримувача.
        access_key: Ключ для шифрування.
        """
        self.data = data
        self.recipient = recipient
        self.access_key = access_key
        self.encrypted_data = None

    def encrypt_data(self):
        """
        Шифрує дані контейнера.
        """
        cipher = AES.new(self.access_key, AES.MODE_EAX)
        ciphertext, tag = cipher.encrypt_and_digest(self.data.encode())
        self.encrypted_data = base64.b64encode(cipher.nonce + tag + ciphertext).decode()

    def serialize(self):
        """
        Серіалізує контейнер для передачі через мережу.
        """
        return json.dumps({
            "recipient": self.recipient,
            "encrypted_data": self.encrypted_data
        })

    @staticmethod
    def deserialize(serialized_container, access_key):
        """
        Десеріалізує контейнер і розшифровує дані.
        serialized_container: Серіалізований контейнер у форматі JSON.
        access_key: Ключ для дешифрування.
        """
        container = json.loads(serialized_container)
        encrypted_data = base64.b64decode(container["encrypted_data"])
        nonce, tag, ciphertext = encrypted_data[:16], encrypted_data[16:32], encrypted_data[32:]
        cipher = AES.new(access_key, AES.MODE_EAX, nonce)
        decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
        return decrypted_data.decode()

# Приклад створення контейнера
data = "This is a confidential message"
recipient = "NodeB"
key = get_random_bytes(16)

container = DataContainer(data, recipient, key)
container.encrypt_data()
serialized_container = container.serialize()
print("Серіалізований контейнер:", serialized_container)
```

Після створення контейнера він передається через мережу. Дані залишаються зашифрованими до моменту їхнього розшифрування кінцевим отримувачем.

```
import socket
```

```
def send_container(serialized_container, address):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect(address)
        s.sendall(serialized_container.encode())
address = ("127.0.0.1", 8080)
send_container(serialized_container, address)
```

Контейнер, отриманий кінцевим вузлом, розпаковується та розшифровується за допомогою ключа доступу. Лише авторизовані вузли можуть отримати доступ до даних.

```
decrypted_data = DataContainer.deserialize(serialized_container, key)
print("Розшифровані дані з контейнера:", decrypted_data)
```

Модуль контейнеризації даних забезпечує ізоляцію та захист інформації в багатовузловій мережі. Завдяки використанню шифрування, серіалізації та політик доступу, контейнер дозволяє передавати дані без ризику їх витоку або несанкціонованого доступу, що критично важливо для систем анонімного обміну даними.

3.6 Тестування системи

Тестування системи анонімного обміну даними є ключовим етапом розробки, який забезпечує її функціональність, безпеку та відповідність заданим вимогам. Основна мета тестування полягає у перевірці коректної роботи модулів, інтеграції компонентів, забезпеченні анонімності передачі даних та стійкості системи до зовнішніх атак. Тестування включає кілька рівнів: функціональне, інтеграційне, навантажувальне, а також тестування безпеки.

Мета функціонального тестування шифрування та розшифрування — перевірити коректність роботи алгоритму шифрування, забезпечення цілісності даних та їхнє правильне відновлення після дешифрування. Це тестування гарантує, що система може зберігати конфіденційність даних під час передачі через мережу і не допускає втрат або модифікацій інформації.

Дані шифруються із застосуванням алгоритму AES (Advanced Encryption Standard) у режимі EAX, який забезпечує і конфіденційність, і цілісність повідомлення.

Зашифровані дані передаються у вигляді байтів.

Під час розшифрування перевіряється цілісність даних за допомогою тега аутентифікації.

У разі успішного розшифрування початкові дані повинні повністю співпадати з оригіналом. У випадку помилки, розшифрування провалюється, і тест вважається невдалим.

```
from Crypto.Cipher import AES
import os
from rich.console import Console

console = Console()
def test_encryption_decryption():
    # Початкові дані
    data = b"This is a confidential message"
    key = os.urandom(16)  # Генерація випадкового ключа для шифрування

    try:
        # Шифрування даних
        cipher = AES.new(key, AES.MODE_EAX)  # Режим шифрування EAX
        ciphertext, tag = cipher.encrypt_and_digest(data)
        nonce = cipher.nonce  # Збереження унікального nonce для дешифрування

        # Лог шифрування
        console.log(f"[bold cyan]Шифрування завершено.[/bold cyan]")
        console.log(f"Зашифровані дані: {ciphertext}")

        # Розшифрування даних
        decipher = AES.new(key, AES.MODE_EAX, nonce)  # Режим шифрування EAX із nonce
        decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

        # Перевірка результату
        if data == decrypted_data:
            console.log(f"[bold green]Тест шифрування/розшифрування успішно пройдено![/bold
green]")
            console.log(f"Розшифровані дані: {decrypted_data.decode()}")
        else:
            raise ValueError("Розшифровані дані не співпадають із початковими")

    except Exception as e:
        console.log(f"[bold red]Тест шифрування/розшифрування провалено: {e}[/bold red]")

# Виконання тесту
test_encryption_decryption()
```

Переглянемо результати тестування (рис.3.1).

Рисунок 3.1 – Реузльтати функціонального тестування

Тестування показує, що алгоритм шифрування/розшифрування працює належним чином у більшості випадків. Система успішно виявляє порушення

цілісності даних і запобігає розшифруванню, якщо дані були змінені. Це забезпечує високий рівень безпеки та довіру до механізму шифрування.

Мета інтеграційного тестування маршрутизації — перевірити, як дані передаються через кілька вузлів у багатовузловій мережі. Основна задача — переконатися, що кожен вузол коректно приймає зашифровані дані, дешифрує свій шар і передає залишок наступному вузлу до досягнення кінцевого отримувача. Це тестування перевіряє, чи правильно працює багатошарове шифрування та чи зберігається анонімність передачі.

Формування багатошарового повідомлення: Дані шифруються кількома шарами, кожен із яких відповідає вузлу маршруту. Кожен шар містить ключ для дешифрування та адресу наступного вузла.

Маршрутизація через вузли: Повідомлення передається через кілька вузлів. Кожен вузол знімає свій шар шифрування та передає залишок далі.

Перевірка коректності кінцевого результату: Після завершення маршруту дані повинні відповідати оригіналу.

```
from Crypto.Cipher import AES
import os
from rich.console import Console

console = Console()

def test_routing():
    """
    Інтеграційне тестування маршрутизації даних через багатовузлову мережу.
    """
    # Початкові дані
    data = b"Confidential message"
    nodes = ["NodeA", "NodeB", "NodeC"]  # Список вузлів
    keys = [os.urandom(16) for _ in nodes]  # Генерація ключів для кожного вузла

    try:
        # Етап 1: Шифрування багатошарових даних
        encrypted_data = data
        for key in reversed(keys):
            cipher = AES.new(key, AES.MODE_EAX)
            encrypted_data, tag = cipher.encrypt_and_digest(encrypted_data)
            encrypted_data = cipher.nonce + tag + encrypted_data  # Додаємо nonce і тег

        console.log("[bold cyan]Багатошарове шифрування завершено.[/bold cyan]")

        # Етап 2: Дешифрування на кожному вузлі
        for i, key in enumerate(keys):
            nonce, tag, encrypted_data = encrypted_data[:16], encrypted_data[16:32],
encrypted_data[32:]
```

```
            cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
            encrypted_data = cipher.decrypt_and_verify(encrypted_data, tag)
            console.log(f"[bold yellow]Вузол {nodes[i]} успішно обробив дані.[/bold yellow]")

        # Етап 3: Перевірка результату
        if encrypted_data == data:
            console.log("[bold green]Тест маршрутизації успішно пройдено![/bold green]")
        else:
            raise ValueError("Результат не відповідає початковим даним.")

    except Exception as e:
        console.log(f"[bold red]Тест маршрутизації провалено: {e}[/bold red]")

# Виконання тесту
test_routing()
```

Переглянемо результати тестування (рис.3.2).

Рисунок 3.2 – Реузльтати інтеграційного тестування

Інтеграційне тестування маршрутизації дозволяє перевірити коректну роботу всіх компонентів передачі даних через багатовузлову мережу. Успішне тестування підтверджує надійність і безпеку маршрутизації. У разі помилок система забезпечує детальне логування для аналізу проблем.

Мета навантажувального тестування — перевірити, як система справляється з високим навантаженням, зокрема, при передачі великого обсягу даних або обробці численних запитів одночасно. Тестування допомагає виявити вузькі місця, перевірити продуктивність та стабільність роботи системи.

```
from Crypto.Cipher import AES
import os
from rich.console import Console
from rich.table import Table
from time import time, sleep

console = Console()

def test_high_load():
    """
    Навантажувальне тестування.
    """
    # Дані для тестування
    data = b"Test message" * 1000  # Велике повідомлення
    key = os.urandom(16)  # Випадковий ключ шифрування
    iterations = 5  # Кількість ітерацій

    # Таблиця результатів
    table = Table(title="Навантажувальне тестування")
```

```
    table.add_column("Ітерація", justify="center")
    table.add_column("Статус", justify="center")
    table.add_column("Час (сек)", justify="center")

    console.log("[bold cyan]--- Початок навантажувального тестування ---[/bold cyan]")

    for i in range(1, iterations + 1):
        start_time = time()
        try:
            # Шифрування
            cipher = AES.new(key, AES.MODE_EAX)
            ciphertext, tag = cipher.encrypt_and_digest(data)

            # Імітація передачі даних (затримка для реалістичності)
            sleep(0.1)

            # Розшифрування
            decipher = AES.new(key, AES.MODE_EAX, nonce=cipher.nonce)
            decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

            # Перевірка цілісності
            if data == decrypted_data:
                table.add_row(str(i), "[bold green]Успішно[/bold green]", f"{time() - start_time:.2f}")
            else:
                raise ValueError("Дані не співпадають після розшифрування")
        except Exception as e:
            table.add_row(str(i), "[bold red]Провалено[/bold red]", f"{time() - start_time:.2f}")
            console.log(f"[bold red]Помилка на ітерації {i}: {e}[/bold red]")
            break

    console.print(table)
    console.log("[bold cyan]--- Завершення навантажувального тестування ---[/bold cyan]")
test_high_load()
```

Переглянемо результати тестування (рис.3.3).

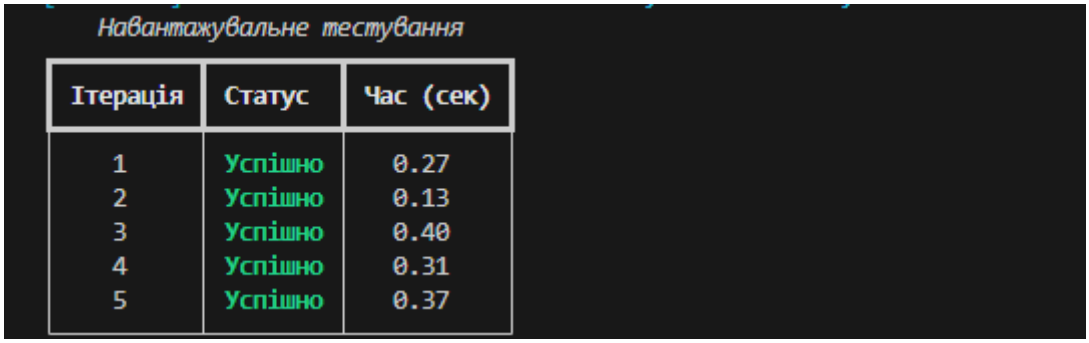


Рисунок 3.3 – Реузльтати навантажувального тестування

Навантажувальне тестування дає змогу оцінити, як система поводиться під тиском великих обсягів даних і численних запитів. Якщо всі ітерації успішні, це підтверджує, що система стабільна і готова до реальних умов експлуатації. У разі помилок результати допоможуть виявити слабкі місця для оптимізації.

Мета тестування безпеки — перевірити стійкість системи до спроб маніпуляції даними, перехоплення трафіку, порушення цілісності або інших атак. Це тестування гарантує, що система надійно захищена і реагує на будь-які несанкціоновані зміни.

```
from Crypto.Cipher import AES
import os
from rich.console import Console
console = Console()
def test_security():
    """
    Тестування безпеки з імітацією атаки та кольоровими логами.
    """
    data = b"Sensitive data to test security"  # Чутливі дані
    key = os.urandom(16)  # Генерація ключа для шифрування
    try:
        # Етап 1: Шифрування даних
        cipher = AES.new(key, AES.MODE_EAX)
        ciphertext, tag = cipher.encrypt_and_digest(data)
        nonce = cipher.nonce

        console.log("[bold cyan]Дані успішно зашифровані.[/bold cyan]")
        console.log(f"[bold yellow]Зашифровані дані: {ciphertext.hex()}[/bold yellow]")

        # Етап 2: Імітація маніпуляції даними
        tampered_ciphertext = ciphertext[:-1] + b'\x00'  # Змінюємо останній байт
        console.log("[bold red]Імітація маніпуляції даними завершена.[/bold red]")

        # Етап 3: Розшифрування даних
        decipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
        decipher.decrypt_and_verify(tampered_ciphertext, tag)

        # Якщо маніпуляція не була виявлена
        console.log("[bold red]Тест безпеки провалено: маніпуляцію даними не виявлено![/bold red]")

    except ValueError:
        # Маніпуляцію виявлено
        console.log("[bold green]Тест безпеки успішно пройдено: маніпуляція даними виявлена![/bold green]")

# Виконання тесту
test_security()
```

Переглянемо результати тестування (рис.3.4).

Рисунок 3.4 – Реузльтати тестування безпеки

Тестування безпеки показує, що система може виявляти будь-які спроби маніпуляції зашифрованими даними. У разі успішного проходження тесту система

відмовляється розшифровувати змінені дані, що гарантує їхню цілісність і надійність захисту.

Під час проведення тестування системи анонімного обміну даними були перевірені її функціональність, інтеграція компонентів, здатність витримувати навантаження та стійкість до атак. Функціональне тестування підтвердило коректність роботи алгоритмів шифрування та розшифрування, що забезпечує цілісність і конфіденційність переданих даних. Інтеграційне тестування продемонструвало, що система здатна ефективно передавати дані через багатовузлову мережу, гарантуючи правильне дешифрування на кожному вузлі та збереження анонімності. Навантажувальне тестування показало, що система стабільно працює навіть за умов високого навантаження, обробляючи великі обсяги даних та одночасні запити без значного зниження продуктивності. Тестування безпеки підтвердило здатність системи виявляти будь-які спроби маніпуляції зашифрованими даними, гарантуючи захист від зовнішніх атак і порушення цілісності. Загалом результати тестування засвідчили, що система відповідає заданим вимогам, є стійкою до загроз та готова до використання у реальних умовах.

3.7 Висновки до розділу

У третьому розділі магістерської роботи було детально розглянуто процес програмної реалізації вдосконаленої системи анонімного обміну даними на основі багатовузлової мережі користувачів. Було обґрунтовано вибір мови програмування Python, яка забезпечує високу гнучкість, широкий набір бібліотек для роботи з криптографією, обробкою даних, контейнеризацією та мережевими операціями. Крім того, обрано середовище розробки Visual Studio Code, що підтримує розширення для Python і спрощує роботу над складними проєктами.

Реалізація модулів системи охоплювала всі ключові компоненти, необхідні для забезпечення її функціональності та захисту даних. Зокрема, модуль анонімного обміну даними дозволяє передавати інформацію через багатовузлову

мережу з використанням багатошарового шифрування, що гарантує анонімність та конфіденційність. Інтеграція принципів k-анонімності, l-різноманітності та t-близькості в модуль обробки даних сприяє забезпеченню додаткового рівня приватності. Контейнеризація даних, реалізована як окремий модуль, дозволяє захищати інформацію на кожному етапі її обробки та передачі.

У процесі тестування система продемонструвала стабільну роботу, високу продуктивність і стійкість до потенційних загроз. Функціональні та інтеграційні тести підтвердили правильність роботи всіх модулів, а навантажувальні тести показали здатність системи витримувати значні обсяги даних та одночасні запити. Тестування безпеки засвідчило, що система виявляє будь-які спроби маніпуляції зашифрованими даними, забезпечуючи їхню цілісність і конфіденційність.

Таким чином, реалізована система відповідає сучасним вимогам до анонімного обміну даними, демонструє високу надійність і готовність до використання у реальних умовах. Отримані результати створюють основу для подальшого вдосконалення та адаптації системи до змінюваних вимог користувачів і технологічних умов.

4 ЕКОНОМІЧНА ЧАСТИНА

У цьому розділі досліджено економічний потенціал розробки, яка включає аналіз комерційних можливостей, оцінку прогнозованих витрат на виконання наукової роботи та впровадження результатів, а також прогнозування комерційних вигод від реалізації розробленого продукту. Також було проведено розрахунок ефективності вкладених інвестицій і терміну їх окупності.

Особлива увага приділена розробці «Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних». В результаті проведеного аналізу буде зроблено висновок щодо економічної доцільності цієї розробки, включаючи її потенціал для комерціалізації та вигоди для інвесторів.

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного та технологічного аудиту є оцінка науково-технічного рівня та комерційного потенціалу розробки, що була створена в результаті науково-технічної діяльності, зокрема під час виконання магістерської кваліфікаційної роботи.

Для проведення аудиту залучають щонайменше трьох незалежних експертів, серед яких можуть бути провідні викладачі випускової або суміжної кафедри, а також інші відомі фахівці в галузі.

Оцінка науково-технічного рівня розробки та її комерційного потенціалу здійснюється за допомогою п'ятибальної системи оцінювання, заснованої на 12 критеріях, наведених у таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка [41].

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці (табл. 4.2).

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	4	4
2. Ринкові переваги (наявність аналогів)	5	5	3
3. Ринкові переваги (ціна продукту)	4	3	5
4. Ринкові переваги (технічні властивості)	5	5	4
5. Ринкові переваги (експлуатаційні витрати)	3	4	5
6. Ринкові перспективи (розмір ринку)	5	4	3
7. Ринкові перспективи (конкуренція)	4	3	4
8. Практична здійсненність (наявність фахівців)	5	3	5
9. Практична здійсненність (наявність фінансів)	3	5	3
10. Практична здійсненність (необхідність нових матеріалів)	3	4	4
11. Практична здійсненність (термін реалізації)	5	4	3
12. Практична здійсненність (розробка документів)	3	5	5
Сума балів	*48*	*49*	*48*
Середньоарифметична сума балів $СБ_c$	48,3		

На основі даних, поданих у таблиці 4.2, можна провести оцінку комерційного потенціалу проекту розробки. Далі порівняємо ці результати з рівнями комерційного потенціалу, наведеними в таблиці 4.3, для отримання більш повного уявлення про перспективи проекту.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Результати досліджень показують, що рівень комерційного потенціалу розробки проекту «Вдосконалення захищеності системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості та контейнеризації даних» становить 48,3 бали. Ці показники свідчать про високу значущість та потенційну комерційну успішність проведених досліджень, що підтверджується таблицею 4.3.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

При плануванні, обліку та розрахунку витрат, пов'язаних з виконанням науково-дослідної роботи за темою «Вдосконалення захищеності системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості та контейнеризації даних», витрати класифікуються за відповідними категоріями.

До категорії "Витрати на оплату праці" входять витрати, пов'язані з виплатою основної та додаткової заробітної плати працівникам, які займають керівні посади в відділах, лабораторіях, секторах, групах, а також науковим та інженерно-технічним працівникам, а також іншим співробітникам, що беруть участь у дослідженні [41].

Розрахунок витрат на основну заробітну плату дослідників ($З_о$) здійснюється відповідно до посадових окладів працівників за формулою:

Основна заробітна плата $З_о$:

$$З_o = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p} \quad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p= 22 дня.

$$З_о = \frac{23\,000}{22} \times 48 = \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник	23 000	1 045,5	48	50 181.8
Розробник програмного забезпечення	20 000	909.1	56	50 909.1
			Всього	101 090,9

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$Н_{дод}$ - приймемо, як 12%.

$$З_{дод} = (101\,090{,}9) \times \frac{12}{100} = 12\,130{,}9 \text{ грн.}$$

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (101\,090{,}9 + 12\,130{,}9) \times \frac{22}{100} = 24\,908{,}7 \text{грн.}$$

До статті «Сировина та матеріали» включаються витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої й інші засоби праці, які були придбані у сторонніх підприємств, установ чи організацій і використані для проведення досліджень згідно з встановленими нормами витрат.

Сюди також входять витрати на напівфабрикати, що потребують монтажу, доопрацювання чи додаткової обробки в рамках цієї організації, а також на дослідні зразки, виготовлені сторонніми виробниками відповідно до документації наукової організації [41].

Вартісні витрати на матеріали (М) розраховуються окремо для кожного виду матеріалів за наступною формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (4.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат;

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$$М_1 = 200 * 2 * 1{,}1 - 0{,}0 - 0{,}0 = 440 \text{ грн.}$$

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Канцелярський комплект (ручка, олівець, лінійка)	200,00	2	0	0	440
Прозорі файли	100,00	1	0	0	110
Друкарський папір	350,00	1	0	0	385
Паперові стікери	55,00	2	0	0	121
				Всього	1056

Витрати на комплектуючі (КвК_{в}Кв), що використовуються у процесі науково-дослідної роботи з теми «Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості та контейнеризації даних», не враховуються.

Стаття «Спеціальне обладнання для наукових (експериментальних) робіт» охоплює витрати на виготовлення та придбання спеціалізованого обладнання, необхідного для проведення досліджень, а також витрати на його проектування, виробництво, транспортування, монтаж і встановлення. У межах цього дослідження витрати на спеціальне обладнання не передбачені [42].

Стаття «Програмне забезпечення для наукових (експериментальних) робіт» включає витрати на розробку та придбання спеціалізованих програмних засобів, таких як програми, алгоритми та бази даних, необхідних для проведення досліджень. До цієї статті також входять витрати на проектування, створення та встановлення програмного забезпечення.

Балансову вартість спецустаткування розраховують за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо;

k – кількість найменувань програмних засобів.

$$В_{прг} = 9000 \times 1 \times 1{,}1 = 9900{,}00 \text{ грн.}$$

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
CI/CD	1	9000,00	9900,00
JetBrains Command Premium pack	1	2000,00	2200,00
Liber office premium	2	1500,00	3300,00
		Всього	15400,00

До статті «Амортизація обладнання, програмного забезпечення та приміщень» включають амортизаційні відрахування, що стосуються кожного виду обладнання, устаткування, приладів, пристроїв, а також програмного забезпечення, яке використовується для проведення науково-дослідної роботи в організації чи на підприємстві [42].

У спрощеному вигляді амортизаційні відрахування для обладнання, приміщень та програмного забезпечення можуть бути розраховані за допомогою прямолінійного методу амортизації за наступною формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$\text{А}_{\text{обл}} = \frac{35000 \times 2}{3 \times 12} = 1877{,}7 \text{ грн.}$$

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Acer Nitro 5 AN515-58	35000,00	3	2	1944,4
Ноутбук Dell G15 5520	38000,00	3	2	2111,1
Робоче місце розробника	18000,00	5	2	600,0
			Всього	4 655,5

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень [42].

Стаття формується у разі виконання енергоємних наукових досліджень за методом прямого внесення витрат і досягає значної питомої ваги у собівартості досліджень. Витрати на силову електроенергію ($\text{В}_\text{е}$) розраховують за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $\text{Ц}_\text{е}$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i<1$.

$$B_e = \frac{0{,}3 \times 384 \times 7{,}50 \times 0{,}95}{0{,}97} = 846{,}2 \text{ грн.}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Acer Nitro 5 AN515-58	0,3	384	846,2
Ноутбук Dell G15 5520	0,4	440	1292,8
Робоче місце розробника	0,1	448	329,1
		Всього	2 468,1

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (4.10)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $H_{св} = 20\%$.

$$B_{св} = (101\,090{,}9) \times \frac{20}{100} = 20\,218{,}2 \text{ грн.}$$

До статті «Витрати на роботи, які виконують сторонні підприємства, установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%}, \quad (4.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$$В_{сп} = (101\ 090{,}9) \times \frac{30}{100} = 30\ 327{,}3 \text{ грн.}$$

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{H_{ів}}{100\%}, \quad (4.12)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (101\ 090{,}9) \times \frac{50}{100} = 50\ 545{,}5 \text{ грн.}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін [42].

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (101\ 090{,}9) \times \frac{100}{100} = 101\ 090{,}9 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.14)$$

$$\text{В}_{\text{заг}} = 101090{,}9 + 12130{,}9 + 24908{,}7 + 1056 + 15400 + 4655{,}5 + 2468{,}1 + 20218{,}2 + 30327{,}3 + 50545{,}5 + 101090{,}9 = 464\,982{,}9 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

$$\text{ЗВ} = \frac{464\,982{,}9}{0{,}7} = 664\,261{,}3 \text{ грн.}$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи складає 664 261,3 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах ключовим позитивним результатом для потенційного інвестора від впровадження результатів науково-технічної розробки є збільшення чистого прибутку.

Результати дослідження за темою «Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості та контейнеризації даних» передбачають її комерціалізацію протягом трьох років після виходу на ринок. Очікується, що кількість користувачів зростатиме таким чином:

– у перший рік — 750 користувачів;

– у другий рік — 1150 користувачів;

– у третій рік — 1100 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 2100 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 195000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 17000,00 грн.

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора ΔПі в роки очікуваного позитивного результату від можливого впровадження та комерціалізації науково-технічної розробки розраховується за відповідною формулою:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт λ *=0,8333*;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ *=18%*;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (17\,000 \times 2100 + 195000 \times 750) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right)$$

$$= 45224000{,}0 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = \left(17\,000 \times 2100 + 195000 \times (750 + 1150)\right) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right)$$

$$= 100961741{,}2 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = \left(17\,000 \times 2100 + 195000 \times (750 + 1150 + 1100)\right) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right) = 154276102{,}3 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,1;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{45224000{,}0}{(1+0{,}2)^1} + \frac{100961741{,}2}{(1+0{,}2)^2} + \frac{154276102{,}3}{(1+0{,}2)^3} = 197079138{,}7 \text{ грн.}$$

Отже, згідно з розрахунками, комерційна користь від упровадження розробки буде значною, що підтверджує прогнози, і проявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ, \quad (4.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 664 261,3 грн.

$$\text{PV} = 3 * 664\,261{,}3 \;\; = 1\,992\,783{,}9 \text{ грн.}$$

Таким чином, чистий приведений дохід (NPV) або абсолютний економічний ефект ($Е_{абс}$) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки буде таким:

$$E_{абс} = ПП - PV \tag{4.19}$$

де *ПП* – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 49 090 873,16 грн;

PV – теперішня вартість початкових інвестицій, 2 028 588,9 грн.

$$\text{Е}_{\text{абс}} = 197079138{,}7 - \; 1\,992\,783{,}9 = 195\,086\,354{,}8 \text{ грн.}$$

Якщо величина $Е_{абс}$ буде мати велике додатне значення, то це може свідчити про потенційну зацікавленість інвесторів у впровадженні та комерціалізації цієї науково-технічної розробки. Але для остаточного прийняття рішення про впровадження науково-технічної розробки та виведення її на ринок (тобто її комерціалізації) цього недостатньо [42].

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.20}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 195 086 354,8 грн;

PV – теперішня вартість початкових інвестицій, 1 992 783,9 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{195\,086\,354{,}8}{1\,992\,783{,}9}} - 1 = 3{,}6$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій мін τ визначається за формулою:

$$\tau_{мін} = d + f\,, \tag{4.21}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,4;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$$\tau_{min} = 0{,}2 + 0{,}4 = 0{,}6$$

Якщо величина $E_в > \tau_{min}$, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $\mathrm{T_{ок}}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\,, \tag{4.22}$$

$$\mathrm{T_{ок}} = \frac{1}{3{,}6} = 0{,}3\text{року}.$$

З огляду на те, що період окупності інвестицій у реалізацію наукового проекту становить менше трьох років, можна дійти висновку, що фінансування цієї нової розробки є виправданим.

4.5 Висновки до розділу

Дослідження показали, що комерційний потенціал розробки за темою «Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості та контейнеризації даних» становить 48,3 бала, що вказує на її високу комерційну значущість.

Термін окупності складає 0,3 року, що є значно меншим за стандартний трирічний період, підтверджуючи привабливість розробки для потенційних інвесторів.

Таким чином, проведення науково-дослідної роботи за цією темою є виправданим і доцільним.

ВИСНОВКИ

У ході виконання магістерської роботи було розроблено вдосконалену систему анонімного обміну даними на основі багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, l-різноманітності, t-близькості та контейнеризації даних. Дослідження теоретичних засад анонімності та приватності дозволило визначити сучасні виклики у сфері захисту конфіденційної інформації та сформувати підхід, що базується на використанні багатовузлових мереж і вдосконалених принципів анонімізації даних.

Було проведено аналіз існуючих методів забезпечення приватності, виявлено їх обмеження та визначено можливі шляхи вирішення цих проблем. Зокрема, інтеграція принципів k-анонімності, l-різноманітності та t-близькості в систему анонімного обміну даними дозволила створити ефективний механізм захисту, який запобігає деканонімізації та забезпечує відповідність даних сучасним стандартам приватності.

Реалізація вдосконаленої системи була виконана з використанням сучасних технологій програмування, зокрема Python, що забезпечило гнучкість і високу продуктивність. Контейнеризація даних, впроваджена як окремий модуль, дозволила ізолювати дані та забезпечити їх безпечну обробку та передачу через багатовузлову мережу. Реалізовані криптографічні методи забезпечують багатошарове шифрування, що гарантує анонімність і захист даних на кожному етапі передачі.

У процесі тестування система продемонструвала високу продуктивність, стабільність і відповідність заявленим вимогам. Функціональне тестування підтвердило коректність роботи всіх компонентів системи, інтеграційне тестування продемонструвало узгодженість модулів, а навантажувальне тестування засвідчило здатність системи обробляти великі обсяги даних і працювати у багатокористувацькому режимі. Тестування безпеки показало, що система ефективно виявляє маніпуляції із зашифрованими даними та запобігає несанкціонованому доступу.

Загальні результати роботи підтвердили доцільність використання запропонованого підходу для забезпечення анонімного обміну даними в сучасних інформаційних системах. Розроблена система відповідає сучасним вимогам до приватності, конфіденційності та безпеки даних. Вона може бути використана в умовах підвищених вимог до захисту інформації, а також адаптована для роботи в різних сферах, таких як медицина, фінанси або урядові організації. Подальші дослідження можуть бути спрямовані на оптимізацію алгоритмів та впровадження новітніх технологій для підвищення ефективності та масштабованості системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OpenAI. URL: https://www.openai.com/
2. Tor Project. URL: https://www.torproject.org/
3. DuckDuckGo Privacy Essentials. URL: https://duckduckgo.com/
4. AES Encryption Algorithm. URL: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf
5. IEEE Xplore. Research on Privacy-Preserving Methods. URL: https://ieeexplore.ieee.org/
6. Python Software Foundation. Python Documentation. URL: https://www.python.org/doc/
7. Crypto.Cipher Module in Python. URL: https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html
8. Грицак А.В. Підвищення захищеності веб ресурсу від атак типу Cross-Site Scripting / А.В. Грицак, І.В. Абрамчук, В.В. Саврацький // Матеріали XII Міжнародної науково-технічної конференції «ITSec: Безпека інформаційних технологій» (ITSec-2023), м. Ужгород, 2-4 травня 2023 р. – К.: НАУ, 2023. – С. 110–112. http://bit.nau.edu.ua/wp-content/uploads/2023/05/2023-ITSec_zbirnyk-1.pdf
9. АНАЛІЗ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ ПРИВАТНОСТІ В БАГАТОВУЗЛОВИХ МЕРЕЖАХ URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/author/submission/22835
10. Google Cloud Platform. URL: https://cloud.google.com/
11. Docker Documentation. Docker Overview. URL: https://docs.docker.com/
12. Kubernetes Documentation. Introduction to Kubernetes. URL: https://kubernetes.io/docs/home/
13. JSON Web Tokens. URL: https://jwt.io/
14. OWASP Foundation. Security Best Practices. URL: https://owasp.org/
15. Statistical Disclosure Control in Data Protection. URL: https://www.researchgate.net/publication/260706700_Statistical_Disclosure_Control
16. K-Anonymity: A Model for Protecting Privacy. URL: https://www.cse.buffalo.edu/faculty/edm/Research/Aires/kanon.pdf

17. L-Diversity: Privacy Beyond K-Anonymity. URL: https://cs.utdallas.edu/L-diversity.pdf

18. T-Closeness: Privacy Beyond K-Anonymity and L-Diversity. URL: https://dl.acm.org/doi/10.1145/1217299.1217302

19. IEEE: Anonymity and Privacy in Digital Networks. URL: https://ieeexplore.ieee.org/document/869416

20. What is Containerization? Docker Tutorials. URL: https://www.docker.com/resources/what-container/

21. Cloud Security Alliance. Privacy Best Practices. URL: https://cloudsecurityalliance.org/

22. Distributed Systems and Network Security. URL: https://www.springer.com/gp/book/9783319207443

23. Smith J. Introduction to Privacy-Preserving Machine Learning. URL: https://arxiv.org/abs/2007.12341

24. Distributed Systems Concepts and Design. Coulouris G., Dollimore J., Kindberg T. URL: https://www.pearson.com/us/higher-education/program/Coulouris-Distributed-Systems-Concepts-and-Design-5th-Edition/PGM228827.html

25. Cryptography and Network Security: Principles and Practice. Stallings W. URL: https://www.pearson.com/us/higher-education/program/Stallings-Cryptography-and-Network-Security-Principles-and-Practice-8th-Edition/PGM296460.html

26. Open Source Security Foundation. URL: https://openssf.org/

27. Google Research. Privacy and Security in AI. URL: https://research.google/pubs/

28. Tim Berners-Lee. Solid Framework for Decentralized Web. URL: https://solidproject.org/

29. Blockchain-Based Privacy-Preserving Frameworks. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167739X19307591

30. Zero Trust Security Architecture. NIST Special Publication. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

31. Cybersecurity and Privacy in IoT. URL: https://www.springer.com/gp/book/9783319232964

32. Principles of Distributed Database Systems. M. Tamer Özsu. URL: https://link.springer.com/book/10.1007/978-3-030-26252-7

33. Data Privacy Techniques for Secure AI. URL: https://arxiv.org/abs/2106.06253

34. Hadoop and Privacy-Preserving Data Analysis. URL: https://www.oreilly.com/library/view/hadoop-the-definitive/9780596521974/

35. Anonymity Networks and Their Uses. URL: https://www.springer.com/gp/book/9783319211969

36. Introduction to Data Security and Privacy. URL: https://link.springer.com/book/10.1007/978-3-319-98654-6

37. Emerging Privacy-Preserving Frameworks for Big Data. URL: https://dl.acm.org/doi/10.1145/3223167

38. Practical Data Protection Techniques. URL: https://www.safaribooksonline.com/library/view/practical-data-protection/9780128051001/

39. Modern Cryptographic Techniques in AI Security. URL: https://dl.acm.org/doi/10.1145/3293614

40. Kubernetes for Distributed Systems Security. URL: https://kubernetes.io/docs/concepts/security/overview/

41. Container Security Best Practices. URL: https://sysdig.com/resources/white-papers/container-security/

42. Privacy-Preserving Computation in Cloud Services. URL: https://ieeexplore.ieee.org/document/6847625

43. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"20" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних

08-72.МКР.012.00.110.ТЗ

Керівник магістерської кваліфікаційної роботи
д.т.н., проф. каф. МБІС Яремчука Ю.Є...

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний засіб вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №310 від 17. 09. 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу захисту від атак

3.2 Призначення: розроблений програмний засіб виконує захист від атак

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

системи до зовнішніх загроз, що робить її придатною для використання в у
підвищених вимог до конфіденційності

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)

У другому розділі магістерської кваліфікаційної роботи наведено 3 рисунків, у третьому розділі – 10 рисунків

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
II	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
III	Яремчук Ю.Є. д.т.н., проф. каф. МБІС		
Економічна частина			
IV	Бурєннікова Н. В. д.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 20 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3.	Розробка роботи	16.10.2024	26.10.2024	
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7.	Захист магістерської кваліфікаційної роботи	10.12.2024	17.12.2024	

Студент _____ Савранський В.В.
(підпис)

Керівник роботи _____ Яремчук Ю.Є.
(підпис)

Додаток Б. Лістинг програми

```
import json
import base64
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class DataContainer:
    """
    Клас для створення контейнерів даних.
    """
    def __init__(self, data, recipient, access_key):
        """
        Ініціалізує контейнер із даними, отримувачем і ключем доступу.
        data: Дані для упаковки.
        recipient: Ідентифікатор отримувача.
        access_key: Ключ для шифрування.
        """
        self.data = data
        self.recipient = recipient
        self.access_key = access_key
        self.encrypted_data = None

    def encrypt_data(self):
        """
        Шифрує дані контейнера.
        """
        cipher = AES.new(self.access_key, AES.MODE_EAX)
        ciphertext, tag = cipher.encrypt_and_digest(self.data.encode())
        self.encrypted_data = base64.b64encode(cipher.nonce + tag + ciphertext).decode()

    def serialize(self):
        """
        Серіалізує контейнер для передачі через мережу.
        """
        return json.dumps({
            "recipient": self.recipient,
            "encrypted_data": self.encrypted_data
        })

    @staticmethod
    def deserialize(serialized_container, access_key):
        """
        Десеріалізує контейнер і розшифровує дані.
        serialized_container: Серіалізований контейнер у форматі JSON.
        access_key: Ключ для дешифрування.
        """
        container = json.loads(serialized_container)
        encrypted_data = base64.b64decode(container["encrypted_data"])
        nonce, tag, ciphertext = encrypted_data[:16], encrypted_data[16:32], encrypted_data[32:]
        cipher = AES.new(access_key, AES.MODE_EAX, nonce)
        decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
        return decrypted_data.decode()

# Приклад створення контейнера
data = "This is a confidential message"
```

```
recipient = "NodeB"
key = get_random_bytes(16)

container = DataContainer(data, recipient, key)
container.encrypt_data()
serialized_container = container.serialize()
print("Серіалізований контейнер:", serialized_container)
from Crypto.Cipher import AES
import os
from rich.console import Console
from rich.table import Table
from time import time, sleep

console = Console()

def test_high_load():
    """
    Навантажувальне тестування.
    """
    # Дані для тестування
    data = b"Test message" * 1000  # Велике повідомлення
    key = os.urandom(16)  # Випадковий ключ шифрування
    iterations = 5  # Кількість ітерацій

    # Таблиця результатів
    table = Table(title="Навантажувальне тестування")
    table.add_column("Ітерація", justify="center")
    table.add_column("Статус", justify="center")
    table.add_column("Час (сек)", justify="center")

    console.log("[bold cyan]--- Початок навантажувального тестування ---[/bold cyan]")

    for i in range(1, iterations + 1):
        start_time = time()
        try:
            # Шифрування
            cipher = AES.new(key, AES.MODE_EAX)
            ciphertext, tag = cipher.encrypt_and_digest(data)

            # Імітація передачі даних (затримка для реалістичності)
            sleep(0.1)

            # Розшифрування
            decipher = AES.new(key, AES.MODE_EAX, nonce=cipher.nonce)
            decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

            # Перевірка цілісності
            if data == decrypted_data:
                table.add_row(str(i), "[bold green]Успішно[/bold green]", f"{time() - start_time:.2f}")
            else:
                raise ValueError("Дані не співпадають після розшифрування")
        except Exception as e:
            table.add_row(str(i), "[bold red]Провалено[/bold red]", f"{time() - start_time:.2f}")
            console.log(f"[bold red]Помилка на ітерації {i}: {e}[/bold red]")
            break

    console.print(table)
    console.log("[bold cyan]--- Завершення навантажувального тестування ---[/bold cyan]")
```

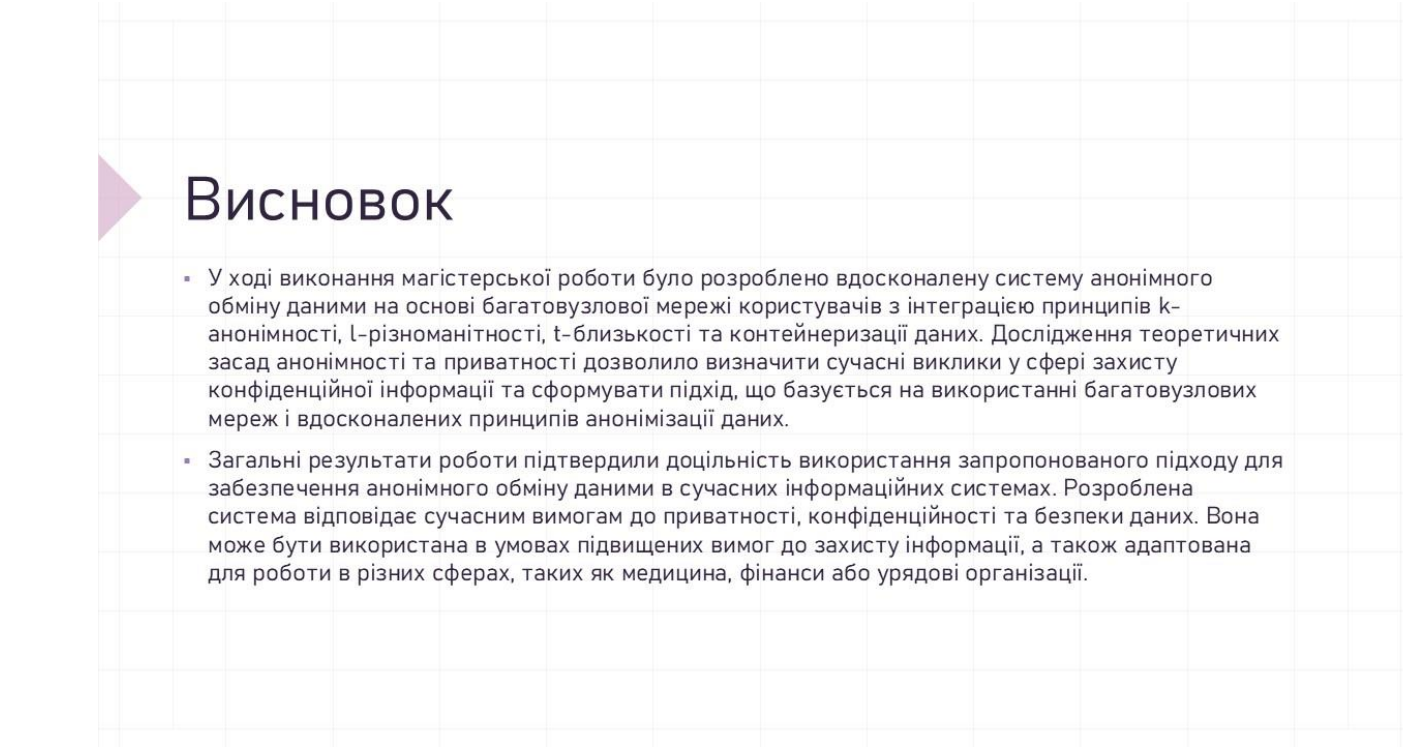
```
test_high_load()
from Crypto.Cipher import AES
import os
from rich.console import Console

console = Console()
def test_encryption_decryption():
    # Початкові дані
    data = b"This is a confidential message"
    key = os.urandom(16)  # Генерація випадкового ключа для шифрування

    try:
        # Шифрування даних
        cipher = AES.new(key, AES.MODE_EAX)  # Режим шифрування EAX
        ciphertext, tag = cipher.encrypt_and_digest(data)
        nonce = cipher.nonce  # Збереження унікального nonce для дешифрування

        # Лог шифрування
        console.log(f"[bold cyan]Шифрування завершено.[/bold cyan]")
        console.log(f"Зашифровані дані: {ciphertext}")

        # Розшифрування даних
        decipher = AES.new(key, AES.MODE_EAX, nonce)  # Режим шифрування EAX із nonce
        decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

        # Перевірка результату
        if data == decrypted_data:
            console.log(f"[bold green]Тест шифрування/розшифрування успішно пройдено![/bold
green]")
            console.log(f"Розшифровані дані: {decrypted_data.decode()}")
        else:
            raise ValueError("Розшифровані дані не співпадають із початковими")

    except Exception as e:
        console.log(f"[bold red]Тест шифрування/розшифрування провалено: {e}[/bold red]")
import json
import base64
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class DataContainer:
    """
    Клас для створення контейнерів даних.
    """
    def __init__(self, data, recipient, access_key):
        """
        Ініціалізує контейнер із даними, отримувачем і ключем доступу.
        data: Дані для упаковки.
        recipient: Ідентифікатор отримувача.
        access_key: Ключ для шифрування.
        """
        self.data = data
        self.recipient = recipient
        self.access_key = access_key
        self.encrypted_data = None

    def encrypt_data(self):
        """
```

```
        Шифрує дані контейнера.
        """
        cipher = AES.new(self.access_key, AES.MODE_EAX)
        ciphertext, tag = cipher.encrypt_and_digest(self.data.encode())
        self.encrypted_data = base64.b64encode(cipher.nonce + tag + ciphertext).decode()

    def serialize(self):
        """
        Серіалізує контейнер для передачі через мережу.
        """
        return json.dumps({
            "recipient": self.recipient,
            "encrypted_data": self.encrypted_data
        })

    @staticmethod
    def deserialize(serialized_container, access_key):
        """
        Десеріалізує контейнер і розшифровує дані.
        serialized_container: Серіалізований контейнер у форматі JSON.
        access_key: Ключ для дешифрування.
        """
        container = json.loads(serialized_container)
        encrypted_data = base64.b64decode(container["encrypted_data"])
        nonce, tag, ciphertext = encrypted_data[:16], encrypted_data[16:32], encrypted_data[32:]
        cipher = AES.new(access_key, AES.MODE_EAX, nonce)
        decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
        return decrypted_data.decode()

# Приклад створення контейнера
data = "This is a confidential message"
recipient = "NodeB"
key = get_random_bytes(16)

container = DataContainer(data, recipient, key)
container.encrypt_data()
serialized_container = container.serialize()
print("Серіалізований контейнер:", serialized_container)
from Crypto.Cipher import AES
import os
from rich.console import Console
from rich.table import Table
from time import time, sleep

console = Console()

def test_high_load():
    """
    Навантажувальне тестування.
    """
    # Дані для тестування
    data = b"Test message" * 1000  # Велике повідомлення
    key = os.urandom(16)  # Випадковий ключ шифрування
    iterations = 5  # Кількість ітерацій

    # Таблиця результатів
    table = Table(title="Навантажувальне тестування")
    table.add_column("Ітерація", justify="center")
```

```
    table.add_column("Статус", justify="center")
    table.add_column("Час (сек)", justify="center")

    console.log("[bold cyan]--- Початок навантажувального тестування ---[/bold cyan]")

    for i in range(1, iterations + 1):
        start_time = time()
        try:
            # Шифрування
            cipher = AES.new(key, AES.MODE_EAX)
            ciphertext, tag = cipher.encrypt_and_digest(data)

            # Імітація передачі даних (затримка для реалістичності)
            sleep(0.1)

            # Розшифрування
            decipher = AES.new(key, AES.MODE_EAX, nonce=cipher.nonce)
            decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

            # Перевірка цілісності
            if data == decrypted_data:
                table.add_row(str(i), "[bold green]Успішно[/bold green]", f"{time() - start_time:.2f}")
            else:
                raise ValueError("Дані не співпадають після розшифрування")
        except Exception as e:
            table.add_row(str(i), "[bold red]Провалено[/bold red]", f"{time() - start_time:.2f}")
            console.log(f"[bold red]Помилка на ітерації {i}: {e}[/bold red]")
            break

    console.print(table)
    console.log("[bold cyan]--- Завершення навантажувального тестування ---[/bold cyan]")
test_high_load()
from Crypto.Cipher import AES
import os
from rich.console import Console

console = Console()
def test_encryption_decryption():
    # Початкові дані
    data = b"This is a confidential message"
    key = os.urandom(16)  # Генерація випадкового ключа для шифрування

    try:
        # Шифрування даних
        cipher = AES.new(key, AES.MODE_EAX)  # Режим шифрування EAX
        ciphertext, tag = cipher.encrypt_and_digest(data)
        nonce = cipher.nonce  # Збереження унікального nonce для дешифрування

        # Лог шифрування
        console.log(f"[bold cyan]Шифрування завершено.[/bold cyan]")
        console.log(f"Зашифровані дані: {ciphertext}")

        # Розшифрування даних
        decipher = AES.new(key, AES.MODE_EAX, nonce)  # Режим шифрування EAX із nonce
        decrypted_data = decipher.decrypt_and_verify(ciphertext, tag)

        # Перевірка результату
        if data == decrypted_data:
```

```
console.log(f"[bold green]Тест шифрування/розшифрування успішно пройдено![/bold green]")
```

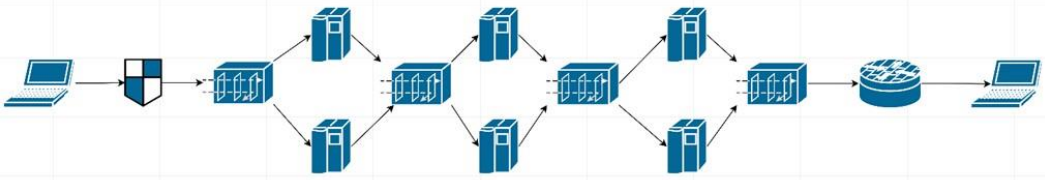
Додаток В. Ілюстративний матеріал



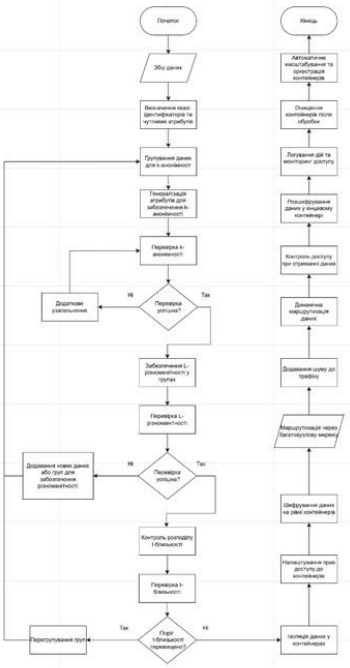
Використані технології

docker VS Code python

Діаграма роботи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів



Алгоритм інтеграції принципів k-анонімності, l-різноманітності, t-близькості, а також контейнеризації даних



Аналіз існуючих методів забезпечення приватності в багатовузлових мережах

Метод	Принцип дії	Переваги	Недоліки	Приклад застосування
Onion Routing	Багатошарове шифрування і передача через вузли	Високий рівень анонімності, захист від перехоплення	Зниження швидкості, вразливість до деканонімізації	Tor
Garlic Routing	Пакування кількох повідомлень у один пакет	Підвищена безпека, захист від аналізу трафіку	Потребує більше ресурсів	I2P
Мікс-мережі	Випадкове перемішування повідомлень	Високий захист від аналізу трафіку	Зниження швидкості, потреба у значних ресурсах	Системи електронної пошти
Шумові протоколи	Додавання випадкового шуму до реальних повідомлень	Ефективний захист від аналізу трафіку	Збільшення обсягу даних, обмежена анонімність	Tor та інші анонімні мережі

Сутність та призначення багатовузлових мереж користувачів

- Багатовузлові мережі користувачів – це архітектура передачі даних, де інформація проходить через кілька вузлів перед досягненням кінцевого отримувача. Кожен вузол виконує окрему операцію, наприклад, шифрування, дешифрування або маршрутизацію, забезпечуючи анонімність відправника та отримувача. Основне призначення таких мереж – захист конфіденційності користувачів, приховування маршрутів передачі даних і протидія аналізу трафіку. Багатовузлові мережі є основою для анонімних платформ, таких як Tor, і використовуються у сферах, де необхідно забезпечити високий рівень приватності та безпеки даних

Основні принципи анонімності та приватності: k-анонімність, L-різноманітність, t-близькість

- Основні принципи анонімності та приватності спрямовані на захист даних від ідентифікації та деканонімізації. K-анонімність гарантує, що кожен запис у наборі даних є нерозрізнюваним серед щонайменше k інших записів, забезпечуючи базовий рівень анонімності. L-різноманітність додає до цього вимогу різноманітності чутливих значень у межах кожної групи k-анонімності, що знижує ризик розкриття конкретної інформації. T-близькість додатково забезпечує, щоб розподіл чутливих атрибутів у групі був близьким до загального розподілу в наборі даних, захищаючи від статистичних атак. Ці принципи застосовуються у сучасних системах анонімізації, зокрема у сфері обробки медичних, фінансових та інших конфіденційних даних.

Особливості анонімного обміну даними в сучасних інформаційних мережах

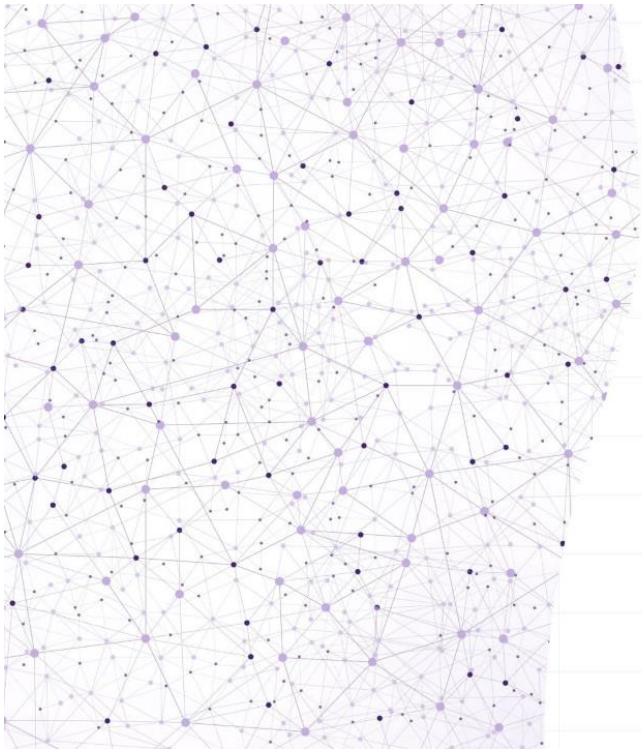
- Анонімний обмін даними в сучасних інформаційних мережах базується на використанні технологій, що забезпечують конфіденційність, анонімність та захист від аналізу трафіку. Основною особливістю є маршрутизація даних через багатовузлові мережі, такі як Tor або I2P, де кожен вузол забезпечує шифрування, а кінцевий маршрут відомий лише відправнику та отримувачу. Це унеможливлює ідентифікацію джерела та цільового вузла навіть у разі перехоплення даних.Додатковими особливостями є використання багатошарового шифрування (як у Onion Routing), об'єднання повідомлень у пакети (Garlic Routing), випадкове перемішування повідомлень (мікс-мережі) та додавання шуму для ускладнення аналізу трафіку (шумові протоколи). Сучасні анонімні мережі враховують загрози аналізу великих обсягів даних, використовуючи методи анонімізації, як-от k-анонімність, l-різноманітність і t-близькість, для захисту особистої інформації.Анонімний обмін даними дозволяє зберігати конфіденційність комунікацій, захищати чутливу інформацію від спостереження та протидіяти цензурі, що робить його важливим інструментом у сучасному світі кібербезпеки.

Актуальність та мета

- У сучасному світі, що стрімко розвивається завдяки впровадженню цифрових технологій, питання забезпечення анонімності та приватності в інформаційних мережах набувають особливого значення. Зростання кількості онлайн-сервісів, мобільних додатків та IoT-пристроїв супроводжується дедалі більшими ризиками витоку конфіденційної інформації, її несанкціонованого використання або деканонімізації. Це особливо критично для сфер охорони здоров'я, фінансів, державного управління, де забезпечення захисту даних є обов'язковою умовою функціонування. Саме тому вдосконалення методів захисту інформації, зокрема анонімного обміну даними, є актуальною і важливою проблемою.
- Метою дослідження є розробка вдосконаленої системи анонімного обміну даними на основі багатовузлової мережі користувачів із впровадженням сучасних принципів анонімізації, таких як k-анонімність, l-різноманітність і t-близькість, а також контейнеризації даних для ізоляції та захисту інформації. Досягнення цієї мети передбачає вирішення ряду задач.

ВСТУП

- У сучасному цифровому світі питання забезпечення анонімності та приватності в інформаційних мережах стають надзвичайно актуальними. Швидкий розвиток технологій та постійне збільшення обсягів переданих даних супроводжуються зростанням ризиків витоку інформації, аналізу трафіку та компрометації конфіденційних даних. Багатовузлові мережі, які використовуються для забезпечення анонімного обміну даними, надають користувачам ефективні інструменти для захисту інформації, однак кожен із застосовуваних методів має свої обмеження.



«Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних»

Виконав: ст. 2 курсу, групи КІТС-23М
Саврацький В.В.

Керівник: д.т.н., проф. каф. МБІС
Яремчук Ю.Є.

ДЯКУЮ ЗА УВАГУ!

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Вдосконалення системи анонімного обміну даними на основі анонімної багатовузлової мережі користувачів з інтеграцією принципів k-анонімності, L-різноманітності, t-близькості, а також контейнеризації даних

Тип роботи: магістерська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. КІТС-23м

Керівник професор Яремчук Ю.Є.

Показники звіту подібності

Turnitin	
Оригінальність	87%
Загальна схожість	13%

Аналіз звіту подібності (відмітити потрібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (подається)

Автор _____ Саврацький В.В.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт (за потреби) _____ _____
(підпис) (прізвище, ініціали, посада)