Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Вдосконалення механізмів мережевої безпеки на основі сучасних алгоритмів»

Виконав: студент групи К23-1М

Спеціальність 122 Комп'ютерні науки

Конохов І.В.
(прізвище та ініціали)

Керівник к.т.н., доц. Мала Ю. А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Дніпровський державний технічний університет
(місце роботи)

доцент кафедри математичного моделювання та системного аналізу
(посада)

к.т.н., доц. Волосова Н.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Конохов І.В. Вдосконалення механізмів мережевої безпеки на основі сучасних алгоритмів.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки» – Університет митної справи та фінансів, Дніпро, 2025.

Магістерська робота присвячена дослідженню сучасних алгоритмів безпеки, які забезпечують захист інформації в комп'ютерних мережах. У роботі виконано аналіз основних загроз і типів атак, що виникають у процесі обміну даними, а також запропоновано методи протидії їм на основі сучасних криптографічних підходів і протоколів захисту. Особливу увагу приділено криптографічним алгоритмам AES, RSA та ECC, які використовуються для забезпечення конфіденційності, цілісності й автентичності даних, а також для створення стійких систем захисту в умовах сучасних кіберзагроз.

У роботі висвітлено актуальні питання аутентифікації користувачів, включаючи багатофакторну аутентифікацію та використання біометричних даних, які є перспективними технологіями в умовах зростаючих вимог до безпеки. Розглянуто такі протоколи, як Kerberos, OAuth, OpenID, SSL/TLS і IPSec, що забезпечують захист даних на різних рівнях мережевої архітектури. Окремо проаналізовано методи захисту голосових і відеокомунікацій, а також використання хеш-функцій і цифрових підписів для забезпечення цілісності та невідмовності інформації.

Розділ програмної реалізації спрямований на розробку і реалізацію програмного забезпечення для демонстрації ефективності сучасних алгоритмів безпеки.

Ключові слова: безпека комп'ютерних мереж, криптографія, аутентифікація, багатофакторний захист, протоколи безпеки, цифрові підписи, хешування, кіберзагрози.

ABSTRACT

Konokhov I.V. Improving network security mechanisms based on modern algorithms.

Diploma thesis for obtaining a master's degree in specialty 122 «Computer Science» – University of Customs and Finance, Dnipro, 2025.

The master's thesis is devoted to the study of modern security algorithms that provide information security in computer networks. The work analyzes the main threats and types of attacks that arise in the process of data exchange, and proposes methods to counteract them based on modern cryptographic approaches and security protocols. Particular attention is paid to the cryptographic algorithms AES, RSA and ECC, which are used to ensure the confidentiality, integrity and authenticity of data, as well as to create resilient protection systems in the face of modern cyber threats.

The paper highlights the current issues of user authentication, including multifactor authentication and the use of biometric data, which are promising technologies in the face of growing security requirements. Protocols such as Kerberos, OAuth, OpenID, SSL/TLS, and IPSec are considered, which provide data protection at different levels of network architecture. Methods of protecting voice and video communications, as well as the use of hash functions and digital signatures to ensure the integrity and reliability of information are analyzed separately.

The practical section of the work is aimed at developing and implementing software to demonstrate the effectiveness of modern security algorithms.

Keywords: computer network security, cryptography, authentication, multifactor protection, security protocols, digital signatures, hashing, cyber threats.

ЗМІСТ

ВСТУП

Інформаційні технології стали невід'ємною частиною сучасного суспільства, їхнє застосування охоплює майже всі сфери людської діяльності: від економіки і фінансів до медицини, науки та освіти. Зростання обсягу та складності обробки даних, а також зростаюча інтеграція комп'ютерних мереж у повсякденне життя зумовлюють нові виклики у сфері безпеки інформації. Всі ці фактори вимагають розвитку та вдосконалення алгоритмів безпеки, що забезпечують захист даних від різноманітних загроз та атак, зокрема з боку кіберзлочинців, конкурентів і державних структур [1]. Безпека комп'ютерних мереж є критично важливою складовою для збереження цілісності інформаційних систем і забезпечення конфіденційності та доступності даних. Саме тому питання забезпечення безпеки комп'ютерних мереж є надзвичайно актуальним в умовах сучасних технологій, постійно змінюваних загроз і нових методів атак.

Актуальність теми кваліфікаційної роботи полягає в необхідності дослідження та вдосконалення алгоритмів безпеки, які використовуються для захисту комп'ютерних мереж від зовнішніх та внутрішніх загроз. Оскільки кіберзлочинність, а також економічні кібер-атаки набувають все більшої популярності, пошук ефективних алгоритмів для забезпечення конфіденційності, цілісності та доступності інформації набуває особливого значення. Вдосконалення алгоритмів криптографії, аутентифікації, захисту від атак, а також розробка нових підходів до виявлення та запобігання вторгненням є необхідними для підтримки безпеки в глобальних комп'ютерних мережах.

Мета дослідження полягає в аналізі та порівнянні сучасних алгоритмів безпеки в комп'ютерних мережах, а також у визначенні їх ефективності в умовах сучасних загроз. Для досягнення цієї мети необхідно провести

ґрунтовний огляд існуючих технологій, а також розробити рекомендації щодо їх використання та вдосконалення.

Завданнями дослідження є:

– огляд сучасних алгоритмів криптографії, аутентифікації та захисту інформації;

– аналіз переваг та недоліків різних методів захисту даних в комп’ютерних мережах;

– вивчення новітніх підходів до запобігання вторгненням та атак на комп’ютерні системи;

– порівняння ефективності алгоритмів безпеки в реальних умовах з точки зору їх швидкодії, надійності та стійкості до атак;

– розробка рекомендацій щодо оптимального використання алгоритмів для захисту сучасних комп’ютерних мереж.

Об’єктом дослідження є комп’ютерні мережі, які використовуються для передачі і зберігання інформації в різноманітних організаціях та системах, як у приватному, так і в публічному секторах. Важливими аспектами, що охоплюються в роботі, є методи і засоби захисту даних, що застосовуються в таких мережах, а також алгоритми, що використовуються для забезпечення безпеки на різних рівнях.

Предметом дослідження є сучасні алгоритми безпеки, що використовуються для захисту інформації в комп’ютерних мережах. Це включає криптографічні алгоритми, протоколи аутентифікації та авторизації, а також методи виявлення вторгнень та аналізу безпеки мереж.

Методи дослідження. Методи дослідження, що використовуються в роботі, включають теоретичний аналіз існуючих наукових джерел та технологій, порівняльний аналіз ефективності алгоритмів безпеки, математичне моделювання та симуляцію роботи комп’ютерних мереж з використанням різних методів захисту. Крім того, застосовуються методи експериментальної перевірки ефективності алгоритмів у реальних умовах.

Практична значимість роботи полягає в розробці практичних рекомендацій щодо використання найбільш ефективних алгоритмів безпеки для захисту комп'ютерних мереж. Результати дослідження можуть бути застосовані для вдосконалення існуючих систем захисту даних в компаніях, установах, а також у розробці нових протоколів безпеки для мобільних і корпоративних мереж. Висновки, отримані в процесі дослідження, сприятимуть зниженню ризиків атак та підвищенню ефективності захисту інформації в умовах сучасних кіберзагроз.

Наукова новизна роботи полягає в систематизації та порівнянні найновіших алгоритмів безпеки, застосовуваних у різних сферах комп'ютерних мереж. В ході роботи пропонується новий підхід до комбінування криптографічних та антивірусних технологій, що дозволяє підвищити ефективність захисту від складних багатоступінчатих атак. Крім того, у роботі представлено розширений аналіз слабких місць сучасних алгоритмів, що дозволить запропонувати шляхи для їх удосконалення.

Загалом, дослідження в цій сфері має велике значення для подальшого розвитку теорії і практики безпеки інформаційних технологій. Обраний напрямок дослідження відкриває нові можливості для підвищення ефективності систем захисту в сучасному інформаційному середовищі.

Структура кваліфікаційної роботи. Кваліфікаційна робота складається з трьох розділів, обсяг роботи – 93 сторінки, робота містить 17 рисунків. Список використаних джерел складає 16 посилань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Поняття та принципи безпеки комп’ютерних мереж

Безпека комп’ютерних мереж є однією з найбільш актуальних і важливих тем в області інформаційних технологій. Зокрема, в умовах стрімкого розвитку технологій, інтеграції різних пристроїв і платформ у глобальну мережу Інтернет, забезпечення захисту інформації стає критично важливим завданням [1, 2]. Комп’ютерні мережі, будучи складними інформаційними системами, взаємодіють між собою через канали зв’язку, що забезпечує обмін даними. Цей процес має численні уразливості, які можуть бути використані злочинними або зловмисними акторами для несанкціонованого доступу до конфіденційної інформації, порушення нормальної роботи системи або навіть для її повного руйнування. Саме тому безпека комп’ютерних мереж охоплює комплекс заходів, спрямованих на запобігання несанкціонованому доступу, пошкодженню, зміні чи знищенню даних та забезпечення цілісності і конфіденційності інформації в процесі її обміну через мережі.

Основою безпеки комп’ютерних мереж є її принципи, що охоплюють різноманітні аспекти організації захисту інформації на всіх етапах її передачі, зберігання та обробки. Найважливішими принципами безпеки є конфіденційність, цілісність, доступність та автентичність інформації. Кожен з цих принципів має свою особливу роль і разом вони створюють основу для побудови ефективних систем захисту.

Схема комп’ютерної мережі продемонстрована на рисунку 1.1.

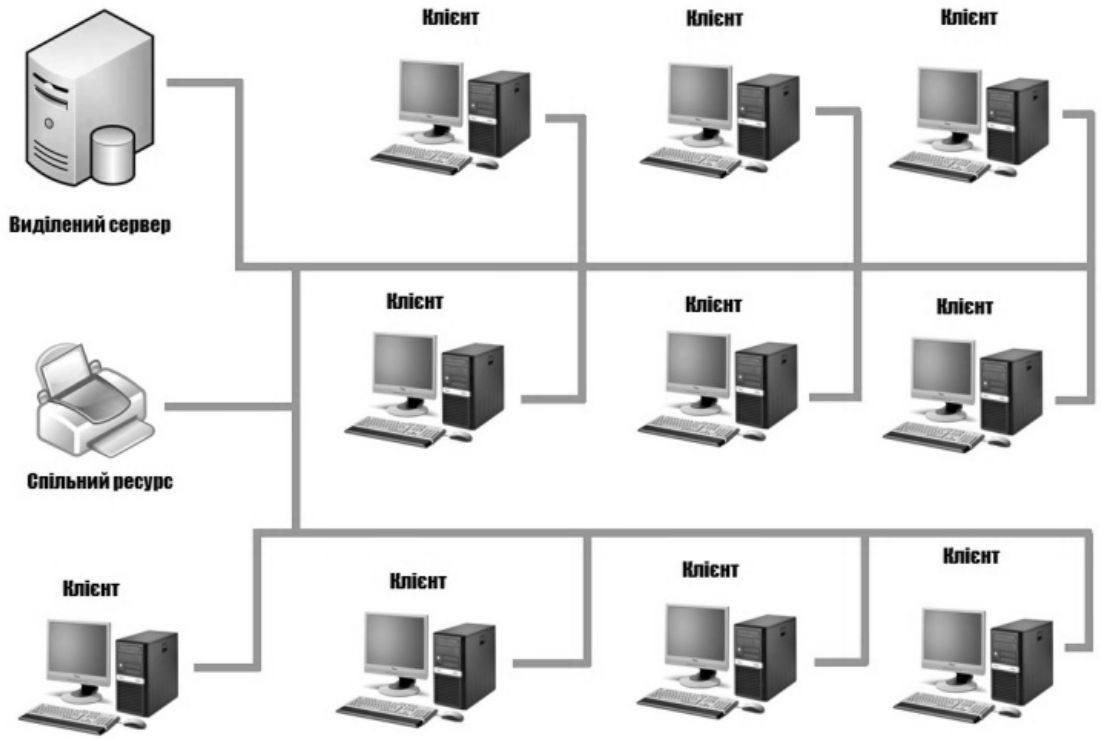

Рисунок 1.1 – Комп’ютерна мережа

Конфіденційність є принципом, який визначає необхідність захисту даних від несанкціонованого доступу. Вона забезпечує, щоб тільки авторизовані користувачі могли отримати доступ до певної інформації. Це досягається шляхом використання різних методів шифрування, а також контролю доступу, що дозволяє встановлювати, хто має право доступу до конкретної інформації в мережі. Одним з основних завдань забезпечення конфіденційності є виявлення і попередження можливих загроз, таких як хакерські атаки, несанкціоноване прослуховування мережевого трафіку або витік інформації через уразливості програмного забезпечення [3].

Цілісність інформації передбачає збереження її вірності та точності протягом усіх етапів її обробки і зберігання. Знищення або несанкціоновані зміни інформації можуть привести до серйозних наслідків, таких як непередбачувані збитки для організацій чи навіть втрати даних, що мають критичне значення. Для забезпечення цілісності застосовуються технології хешування та цифрові підписи, що дозволяють перевіряти цілісність

інформації, а також методи резервного копіювання та відновлення даних, які допомагають відновити інформацію після її пошкодження.

Доступність означає здатність системи забезпечувати безперервний доступ до інформації для авторизованих користувачів, навіть в умовах зовнішніх загроз чи внутрішніх збоїв. Важливим аспектом забезпечення доступності є надійність та відмовостійкість мережевих пристроїв і серверів. Для цього використовуються технології резервування та балансування навантаження, що дозволяють забезпечити безперервну роботу системи навіть у разі виходу з ладу окремих компонентів.

Автентичність є принципом, що визначає необхідність підтвердження особи користувача або джерела інформації. Це важливо для того, щоб забезпечити, що інформація надходить від надійного та перевіреного джерела, а також для уникнення фальсифікацій чи підробки даних. Для забезпечення автентичності використовуються цифрові сертифікати, двофакторна аутентифікація та інші методи ідентифікації користувачів.

Однією з важливих складових безпеки комп’ютерних мереж є управління ризиками. Ризики в контексті безпеки комп’ютерних мереж виникають внаслідок впливу різноманітних загроз, таких як хакерські атаки, природні катастрофи, технічні несправності чи людські помилки. Управління ризиками включає в себе визначення можливих загроз, оцінку їх ймовірності та потенційного впливу на систему, а також розробку і впровадження заходів для мінімізації або усунення цих ризиків. Одним з основних підходів до управління ризиками є застосування стратегії захисту, що поєднує технічні, організаційні та фізичні заходи, що забезпечують всебічний захист від різноманітних загроз.

У зв’язку зі швидким розвитком технологій, зокрема хмарних обчислень, Інтернету речей, штучного інтелекту, питання безпеки комп’ютерних мереж набувають все більшої актуальності. Хмари дають змогу забезпечувати гнучкість і масштабованість, однак водночас відкривають нові

вразливості, такі як загроза доступу до конфіденційних даних через недостатньо захищені канали зв’язку або відсутність контролю над фізичним доступом до серверів [3]. Інтернет речей, що з’єднує мільйони пристроїв, потребує нового рівня безпеки, оскільки навіть невеликі вразливості можуть бути використані для здійснення масштабних атак. Водночас, інтеграція штучного інтелекту в системи безпеки надає нові можливості для автоматичного виявлення загроз і реагування на них в реальному часі.

Зважаючи на складність та динамічний характер сучасних комп’ютерних мереж, безпека вимагає комплексного підходу, який включає в себе як технічні, так і організаційні заходи. Це включає в себе не тільки використання сучасних засобів захисту, таких як шифрування, фаєрволи, системи виявлення вторгнень, але й навчання персоналу, розробку політик безпеки, проведення регулярних аудитів та тестувань на вразливості. Крім того, важливим аспектом є дотримання міжнародних стандартів та нормативних актів, що регулюють питання захисту інформації, таких як стандарт ISO/IEC 27001 або Регламент захисту персональних даних (GDPR).

Таким чином, безпека комп’ютерних мереж є багатогранною проблемою, що вимагає інтеграції різних технологій, стратегій та підходів для досягнення належного рівня захисту. Принципи безпеки, такі як конфіденційність, цілісність, доступність та автентичність, складають основу побудови ефективних механізмів захисту інформації. Однак успішне забезпечення безпеки можливе тільки за умови комплексного підходу, що враховує не тільки технічні, але й організаційні та правові аспекти.

1.2 Рівні захисту інформації в комп’ютерних мережах

Захист інформації в комп’ютерних мережах є невід’ємною частиною будь-якої сучасної інформаційної інфраструктури, оскільки забезпечення безпеки даних і збереження їх цілісності мають критичне значення для

організацій, підприємств і навіть для особистої безпеки користувачів. У глобальному контексті розвитку інформаційних технологій, коли комп'ютерні мережі стають все більш взаємопов'язаними та доступними [4, 5], ризики зловмисних атак та витоків конфіденційної інформації зростають. Для ефективного захисту інформації необхідно забезпечити її безпеку на різних рівнях, починаючи від фізичного доступу до мережі і закінчуючи захистом на рівні додатків та даних. Кожен рівень захисту в комп'ютерних мережах має свої особливості і стратегії, а також використовує різні методи для попередження несанкціонованого доступу, знищення чи зміни даних, а також для забезпечення стабільності та доступності мережевих ресурсів [6].

Основними рівнями захисту інформації в комп'ютерних мережах є фізичний рівень, мережевий рівень, рівень операційних систем, рівень додатків і рівень даних (рис. 1.2). Кожен з цих рівнів є важливим компонентом комплексної системи безпеки, і їх ефективне функціонування в сукупності дозволяє досягти належного рівня захисту даних від різноманітних загроз.

Перший рівень захисту – фізичний рівень. Цей рівень безпеки передбачає фізичний захист інфраструктури комп'ютерних мереж, що включає сервери, мережеві пристрої, кабелі, комутатори, маршрутизатори та інші елементи. Фізичний доступ до обладнання є одним з найважливіших аспектів безпеки, оскільки навіть найкраще захищена система може бути зламаною, якщо зловмисник отримає безпосередній доступ до фізичних ресурсів. Для цього використовуються різні методи, включаючи обмеження доступу до серверних кімнат і дата-центрів, встановлення фізичних бар'єрів, таких як замки, відеоспостереження, біометричні системи і контроль за відвідуванням. Також важливим елементом фізичного захисту є резервне живлення і заходи проти стихійних лих, таких як пожежі чи повені, що можуть пошкодити обладнання. Резервні джерела живлення, системи охолодження і аварійне відновлення після збоїв забезпечують безперервність

функціонування критичних елементів інфраструктури, знижуючи ризик втрати даних чи доступу до них.

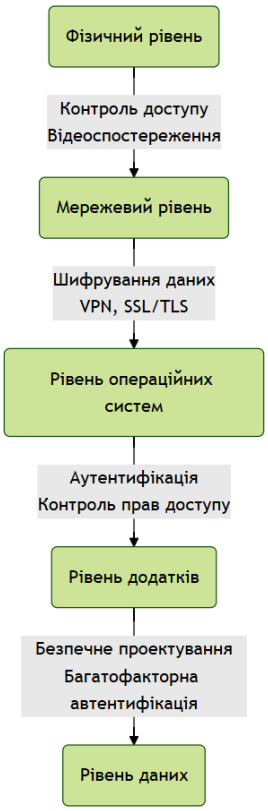

Рисунок 1.2 – Рівні захисту інформації

Другим важливим рівнем є мережевий рівень захисту, який орієнтований на захист інформації під час її передачі по каналах зв’язку. Мережевий рівень охоплює забезпечення безпеки при обміні даними між пристроями в межах локальної або глобальної мережі. Одним з основних аспектів мережевого захисту є контроль доступу до мережі. Мережеві пристрої, такі як маршрутизатори і комутатори, повинні бути налаштовані для обмеження доступу і забезпечення безпечної маршрутизації трафіку. Фаєрволи, що працюють на мережевому рівні, контролюють вхідний і вихідний трафік, блокуючи непотрібні чи потенційно небезпечні з’єднання. Також важливим елементом мережевого захисту є шифрування даних, яке дозволяє забезпечити конфіденційність інформації навіть у випадку її

перехоплення під час передачі [5-7]. Використання протоколів захищеного з'єднання, таких як SSL/TLS для веб-трафіку або VPN для віддаленого доступу, дає можливість забезпечити надійне шифрування даних і захистити їх від перехоплення.

Третім рівнем захисту є рівень операційних систем. Операційна система є основною платформою для виконання програм і управління ресурсами комп'ютера, тому її захист є критичним для безпеки всіх мережевих додатків і сервісів. На цьому рівні безпеки важливо забезпечити контроль доступу до системних ресурсів. Для цього використовуються механізми аутентифікації і авторизації, що дозволяють обмежити доступ до операційної системи лише для авторизованих користувачів. Важливими інструментами захисту є контроль прав доступу до файлів, каталогів, процесів і інших системних компонентів. Операційні системи також повинні бути захищені від шкідливих програм, таких як віруси, трояни, руткіти та інше шкідливе програмне забезпечення. Для цього використовуються антивірусні програми, які повинні регулярно оновлюватися, а також системи виявлення вторгнень, які моніторять поведінку операційної системи для виявлення аномальних дій. Системи безпеки на цьому рівні також включають регулярні оновлення і патчі для виправлення вразливостей, що можуть бути використані для атак.

Четвертим рівнем є рівень додатків. Програмне забезпечення, яке працює на комп'ютерах і серверах, є ще однією важливою частиною системи захисту інформації. Зловмисники можуть використовувати уразливості в додатках для виконання атак, таких як SQL-ін'єкції, міжсайтові скрипти (XSS) або атаки на уразливості в протоколах комунікації. Для захисту додатків необхідно застосовувати методи безпеки, які враховують специфіку кожного конкретного додатку [6, 7]. Важливими аспектами є правильне проектування додатків з точки зору безпеки, включаючи використання шифрування для зберігання чутливої інформації, обмеження прав доступу до баз даних та інші методи захисту. Крім того, на рівні додатків важливо реалізовувати механізми

безпечної аутентифікації користувачів, включаючи багатофакторну аутентифікацію, а також виявляти і запобігати атакам за допомогою систем безпеки додатків (WAF – web application firewall).

Останнім рівнем захисту є рівень даних. Цей рівень охоплює заходи безпеки, які застосовуються безпосередньо до інформації, що зберігається в мережі або передається по каналах зв’язку. Захист даних передбачає як фізичну безпеку їх зберігання, так і захист від несанкціонованого доступу, змін або знищення. На рівні даних використовуються методи шифрування для забезпечення конфіденційності, а також механізми цифрових підписів для перевірки автентичності і цілісності даних. Регулярне створення резервних копій і зберігання їх в безпечних місцях є важливою частиною стратегії захисту даних, оскільки це дозволяє відновити інформацію у разі її втрати або пошкодження. Крім того, важливими є процедури обмеження доступу до даних, а також моніторинг і аудит використання даних для виявлення несанкціонованих спроб доступу або змін.

Таким чином, захист інформації в комп’ютерних мережах є складною та багаторівневою системою, яка включає в себе заходи на фізичному, мережевому, операційному, додатковому та рівні даних. Кожен з цих рівнів відіграє важливу роль у забезпеченні загальної безпеки і дозволяє створити надійний захист від різноманітних загроз і атак. Сучасна кібербезпека вимагає інтегрованого підходу, де всі ці рівні працюють у комплексі для того, щоб гарантувати безпеку даних і забезпечити належний рівень захисту на всіх етапах обробки, передачі і зберігання інформації.

1.3 Загальні принципи криптографії

Криптографія є однією з основних складових сучасної інформаційної безпеки, що дозволяє забезпечити конфіденційність, цілісність, автентичність та невідмовність даних, які передаються через відкриті канали зв’язку.

Розвиток криптографії, як науки та практики, бере початок у давнину, однак з розвитком інформаційних технологій та масовим використанням комп’ютерних мереж її значення значно зросло, оскільки сучасні комунікаційні системи потребують надійного захисту перед численними кіберзагрозами [5, 7]. Криптографія розглядається не тільки як набір математичних методів для захисту інформації, але і як широка дисципліна, що охоплює різні аспекти безпеки: від алгоритмів шифрування до механізмів забезпечення автентичності та захисту від підробок.

Загальні принципи криптографії передбачають використання різноманітних методів і технологій для забезпечення конфіденційності, цілісності та автентичності даних у процесі їх обміну. Ключовими принципами криптографії є: забезпечення конфіденційності, цілісності, автентичності, невідмовності та довіри.

Конфіденційність даних означає, що тільки авторизовані особи або системи мають доступ до інформації, що передається чи зберігається в системах. Вона досягається шляхом застосування алгоритмів шифрування, що перетворюють відкритий текст в шифрований вигляд, який неможливо прочитати без відповідного ключа дешифрування. Ключі шифрування є критично важливими для підтримки конфіденційності, оскільки без них неможливо повернути зашифровану інформацію в її первісний вигляд. Важливою особливістю криптографії є необхідність створення такої системи, яка б дозволяла забезпечити конфіденційність навіть за умови того, що зловмисники можуть знати сам алгоритм шифрування. Саме на основі цієї ідеї виникли сучасні методи симетричного і асиметричного шифрування, кожен з яких має свої переваги і недоліки, але обидва гарантують високий рівень захисту даних при правильному використанні.

Другим важливим принципом криптографії є цілісність. Цілісність даних передбачає, що інформація, яка передається або зберігається, не повинна бути змінена, втрачена чи спотворена в процесі її обробки чи

передачі. Для досягнення цього принципу використовуються різноманітні методи перевірки цілісності, зокрема контрольні суми, хеш-функції та цифрові підписи [1, 5-7]. Хеш-функції є алгоритмами, які приймають на вхід дані довільного розміру і генерують короткий за розміром унікальний ідентифікатор – хеш. Якщо дані були змінені, то хеш-функція на новому наборі даних згенерує інший результат, що дозволяє виявити будь-які спроби втручання. Цілісність забезпечується також за допомогою цифрових підписів, які дозволяють не лише перевірити, що дані не були змінені, але й підтвердити їх походження. Цей принцип є важливим для захисту від атак, що націлені на зміну інформації в процесі її передачі, таких як атаки «людина посередині», що дозволяють зловмисникам змінювати дані без відома їхніх учасників.

Автентичність означає, що дані, що передаються через мережу, дійсно належать тим особам або системам, які їх відправляють. Для цього використовуються механізми аутентифікації, що базуються на перевірці особистості користувача або системи за допомогою різних методів. Одним із найпоширеніших методів є використання паролів, токенів, смарт-карток або біометричних даних. Однак криптографія дозволяє також реалізувати більш надійні методи аутентифікації, такі як використання цифрових сертифікатів і протоколів для перевірки автентичності на основі криптографічних підписів. Використання таких методів дозволяє гарантувати, що обмінювана інформація належить саме тому учаснику, якому вона адресована, що є важливим для уникнення атак, спрямованих на підробку особистостей і шкідливі маніпуляції.

Принцип невідмовності є також важливим аспектом криптографії. Невідмовність означає, що жоден з учасників обміну даними не може заперечити свою участь у здійсненій операції або відмовитися від її виконання. Це принцип забезпечується за допомогою цифрових підписів та механізмів журналювання, що фіксують всі дії з даними. Цифровий підпис створюється за допомогою приватного ключа відправника, що дозволяє зберегти доказ його

участі в операції. Таким чином, навіть якщо відправник спробує заперечити свою участь в операції, цифровий підпис дозволить довести, що саме він був автором повідомлення або дії, що гарантує юридичну та технічну непідтверджуваність.

Надійність криптографії також базується на принципі довіри. Важливою складовою будь-якої криптографічної системи є довіра до алгоритмів і ключів, які використовуються для шифрування [3, 4]. Технології, що забезпечують довіру, включають публічні ключі, інфраструктури відкритих ключів (PKI), сертифікаційні центри та інші засоби перевірки криптографічних ключів. Протоколи і сертифікати, що використовуються в цих системах, дозволяють створити середовище, де учасники можуть взаємодіяти, знаючи, що обмінювані дані захищені і перевірені через надійні криптографічні інструменти. Інфраструктура відкритих ключів, що базується на публічних і приватних ключах, дозволяє організувати безпечний обмін інформацією між незнайомими сторонами, що є критично важливим для електронної комерції та інших сфер застосування криптографії.

Загалом криптографія є комплексною дисципліною, яка охоплює широкий спектр теоретичних і практичних аспектів, що дозволяють забезпечити захист даних на різних етапах їх обробки, зберігання та передачі. Ключові принципи криптографії – конфіденційність, цілісність, автентичність, невідмовність і довіра – утворюють основу для розробки і впровадження надійних систем безпеки, що використовуються в різних сферах, від комунікацій до фінансових операцій та електронної торгівлі. З огляду на те, що з розвитком нових технологій і зростанням кіберзагроз криптографія постійно еволюціонує, важливою частиною її розвитку є вдосконалення алгоритмів, методів шифрування та інтеграції криптографії в складні інформаційні системи для забезпечення їх безпеки.

1.4 Актуальні криптографічні алгоритми

Криптографічні алгоритми є основою для забезпечення безпеки в сучасних інформаційних системах. Вони використовуються для шифрування даних, забезпечення їх цілісності, аутентифікації користувачів і гарантування конфіденційності в електронних комунікаціях. Враховуючи стрімкий розвиток технологій і зростаючі загрози з боку кіберзлочинців, криптографія стає невід'ємною частиною будь-якої інформаційної інфраструктури. Одними з найактуальніших криптографічних алгоритмів на сьогодні є AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman) і ECC (Elliptic Curve Cryptography). Ці алгоритми займають провідні позиції в галузі криптографії завдяки своїй ефективності, надійності і здатності забезпечувати високий рівень захисту в різних застосуваннях – від захисту даних до забезпечення безпеки в електронних транзакціях [7-9].

Актуальні криптографічні алгоритми наведено на рисунку 1.3.

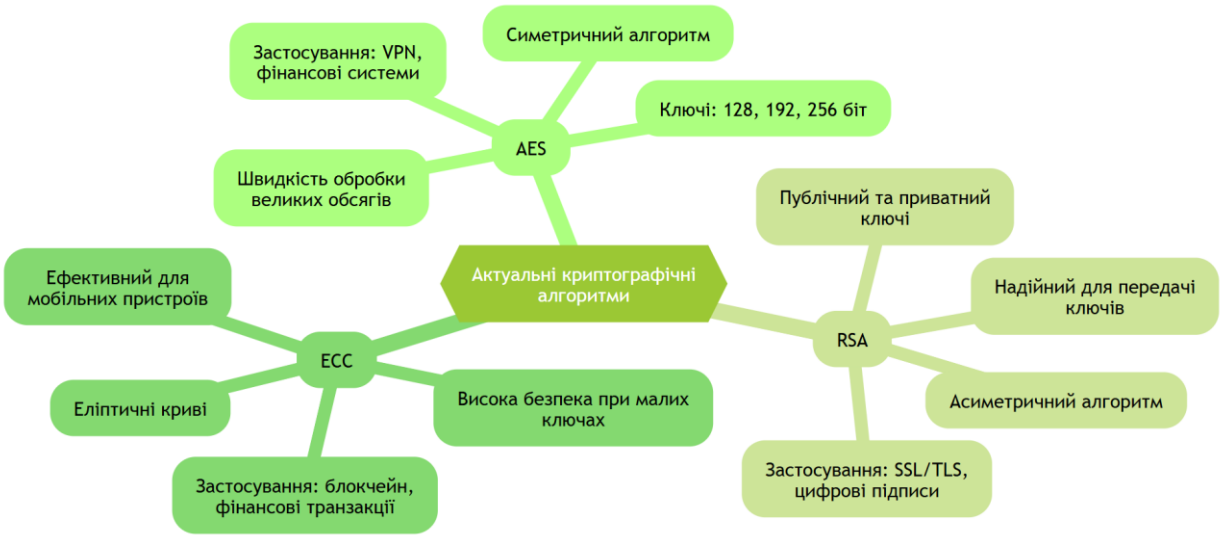

Рисунок 1.3 – Актуальні криптографічні алгоритми

Першим з криптографічних алгоритмів, який заслуговує на увагу, є AES. AES є симетричним алгоритмом шифрування, що був стандартизований урядом США в 2001 році після проведення конкурсу серед криптографічних алгоритмів. Його основною особливістю є використання одного і того ж ключа для шифрування та дешифрування інформації. Симетричні алгоритми мають високі швидкості обробки даних, що робить їх дуже ефективними для шифрування великих обсягів інформації. AES працює на основі блочного шифрування, що означає, що він оперує з блоками даних фіксованого розміру, в даному випадку – 128 бітами. AES використовує ключі різної довжини – 128, 192 або 256 біт, що дозволяє вибирати оптимальний рівень безпеки в залежності від вимог конкретної системи. Вибір довжини ключа має безпосередній вплив на рівень безпеки: чим довший ключ, тим складніше здійснити атаку методом грубої сили. AES широко застосовується в багатьох сферах, включаючи захист даних в банківських і фінансових системах, VPN-з'єднаннях, протоколах безпеки для веб-додатків і інших інфраструктурах, де важлива швидкість і ефективність шифрування. Незважаючи на високу ефективність симетричних алгоритмів, таких як AES, вони мають одну суттєву проблему – необхідність передачі секретного ключа між сторонами, що взаємодіють [9, 10]. Якщо зловмисник перехопить цей ключ під час його передачі, він отримає доступ до зашифрованої інформації. Це призвело до розвитку асиметричних алгоритмів, зокрема RSA, який є одним із найбільш популярних методів для забезпечення безпечної передачі ключів і здійснення аутентифікації.

RSA (Rivest-Shamir-Adleman) є класичним асиметричним алгоритмом шифрування, розробленим в 1977 році. В асиметричних алгоритмах використовуються два різних ключі: публічний, який може бути відомим усім, і приватний, який зберігається в таємниці у власника. Публічний ключ використовується для шифрування інформації, а приватний – для її дешифрування. Це дозволяє безпечно обмінюватися зашифрованими даними

навіть через відкриті канали зв’язку. RSA базується на математичній складності факторизації великих чисел на прості множники, що робить його надійним засобом для забезпечення конфіденційності і автентичності. Ключі в RSA є набагато більшими, ніж в симетричних алгоритмах, що забезпечує більшу стійкість до атак методом грубої сили, але й збільшує витрати на обчислення. Зазвичай RSA використовується для захисту невеликих обсягів даних, таких як криптографічні ключі або цифрові підписи, в той час як для шифрування великих обсягів даних часто використовується в поєднанні з симетричними алгоритмами, такими як AES. RSA є важливим елементом сучасних протоколів безпеки, таких як SSL/TLS, що забезпечують захист веб-комунікацій, а також в електронних підписах для підтвердження автентичності документів. Проте RSA також має свої обмеження, які пов’язані з великими обсягами обчислень для генерації ключів і виконання шифрування. Для підвищення ефективності в багатьох системах використовуються інші методи асиметричного шифрування, такі як алгоритми на основі кривих Еккартів, або ECC [10].

ECC (Elliptic Curve Cryptography) є сучасним і високоефективним підходом до асиметричного шифрування, заснованим на властивостях еліптичних кривих. Вибір еліптичних кривих для криптографії обумовлений їх здатністю забезпечувати високий рівень безпеки при порівняно малих розмірах ключів. Це дає значну перевагу в порівнянні з іншими методами, такими як RSA, де для досягнення високої безпеки потрібні дуже великі ключі, що веде до високих вимог до обчислювальних ресурсів. При використанні ECC для досягнення того ж рівня безпеки, що й RSA з ключем 2048 біт, достатньо буде ключа довжиною всього 256 біт, що забезпечує менші витрати на обчислення, зберігання і передачу ключів. Завдяки цим перевагам ECC набирає популярності в сучасних мобільних та вбудованих системах, а також у протоколах, які потребують високої продуктивності та енергозбереження.

Принцип роботи ECC полягає в застосуванні алгебри еліптичних кривих над скінченними полями для виконання операцій шифрування, підписів і аутентифікації. Однією з ключових особливостей є задача дискретного логарифмування на еліптичних кривих, яка є надзвичайно складною для обчислення навіть за допомогою сучасних комп'ютерів, що забезпечує високий рівень безпеки. Системи на основі ECC застосовуються в різних сферах, зокрема в мобільних пристроях, в хмарних обчисленнях, а також для безпеки у фінансових транзакціях і блокчейн-технологіях. ECC є основою для популярних криптографічних стандартів, таких як ECDSA (Elliptic Curve Digital Signature Algorithm) для цифрових підписів і ECDH (Elliptic Curve Diffie-Hellman) для обміну ключами.

Узагальнюючи, можна сказати, що кожен з криптографічних алгоритмів – AES, RSA і ECC – має свої переваги і області застосування. AES є ефективним алгоритмом для шифрування великих обсягів даних, RSA широко використовується для забезпечення безпеки передачі ключів і цифрових підписів, тоді як ECC забезпечує високу ефективність і надійність при значно менших витратах на обчислення, що робить його ідеальним для сучасних мобільних і вбудованих систем. Всі ці алгоритми активно використовуються в протоколах безпеки, таких як SSL/TLS, VPN, електронні підписи і багатьох інших, що підкреслює їх важливість для забезпечення конфіденційності і безпеки в сучасних інформаційних системах.

1.5 Хешування та цифрові підписи

Хешування та цифрові підписи є ключовими елементами сучасної криптографії, які виконують важливі функції у забезпеченні безпеки та довіри в електронних системах. Вони використовуються для підтвердження цілісності даних, автентичності джерела повідомлення та запобігання підробці інформації. Зрозуміти основи цих криптографічних процесів важливо не лише

для фахівців у галузі безпеки, але й для тих, хто прагне застосовувати сучасні технології захисту інформації в будь-яких видах діяльності, пов’язаних з електронною комунікацією або обробкою даних [5, 8]. Ці методи дозволяють забезпечити високий рівень захисту інформації, а також сприяють розвитку новітніх технологій, таких як блокчейн, електронні платіжні системи, а також системи для захисту особистих даних.

Хешування є одним із основних криптографічних процесів, що дозволяє перетворювати довільно великі обсяги інформації в короткий, фіксований за розміром результат, званий хешем. Хеш-функція – це математичний алгоритм, який приймає на вхід будь-які дані і генерує унікальний хеш, який є своєрідним «відбитком» цих даних. Важливою особливістю хеш-функцій є те, що навіть найменша зміна вхідних даних призводить до зміни хешу. Це забезпечує високу чутливість хеш-функцій до будь-яких змін у даних, що робить їх ідеальними для перевірки цілісності інформації. Ідея використання хешування базується на тому, що на основі хешу неможливо відновити оригінальні дані, тобто хеш-функції є необоротними. Вони також повинні мати властивість детермінованості, що означає, що для одних і тих самих вхідних даних хеш-функція повинна генерувати один і той самий хеш. Іншою важливою властивістю хеш-функцій є їх швидкість: хешування повинно виконуватись дуже швидко, оскільки цей процес є основою для перевірки цілісності даних у реальному часі.

Один з основних застосувань хешування – це перевірка цілісності інформації. Хеш-функція широко використовується для створення контрольних сум файлів або повідомлень. Наприклад, при завантаженні програмного забезпечення з Інтернету користувач може перевірити, чи не було змінено або пошкоджено файл під час передачі [9, 10]. Для цього на сервері створюється хеш-функція оригінального файлу, і користувач може порівняти отриманий хеш після завантаження з наданим на сайті. Якщо ці хеші співпадають, можна з упевненістю стверджувати, що файл не зазнав змін і є

автентичним. Хешування також широко використовується в криптовалютних системах, зокрема у блокчейні, де хеші використовуються для забезпечення безпеки транзакцій і збереження цілісності даних у розподілених реєстрах.

Існують різні алгоритми хешування, серед яких найбільш відомими є MD5, SHA-1 та SHA-256. MD5 був одним з перших широко застосовуваних алгоритмів хешування, але з часом виявилося, що він має вразливості, які дозволяють проводити атаки на колізії, тобто знаходити два різних набори даних, які дають однаковий хеш. З цієї причини MD5 був визнаний недостатньо безпечним для сучасних застосувань, і його використання у системах безпеки було значно обмежене. SHA-1 також зазнав подібних критичних атак і на сьогодні не рекомендується для використання в безпечних системах. Натомість алгоритм SHA-256, який є частиною сімейства SHA-2, став стандартом для багатьох сучасних криптографічних протоколів завдяки своїй високій безпеці і стійкості до атак.

Цифрові підписи є ще одним важливим криптографічним інструментом, який дозволяє підтвердити автентичність джерела інформації та забезпечити її цілісність. Цифровий підпис є результатом застосування приватного ключа до хешу повідомлення, що створює зашифрований відбиток цього повідомлення. Одержувач, використовуючи публічний ключ відправника, може перевірити, чи збігається хеш повідомлення з його підписом, що дозволяє впевнитися в автентичності джерела і в тому, що повідомлення не було змінене під час передачі.

Процес створення цифрового підпису починається з хешування повідомлення за допомогою хеш-функції. Потім отриманий хеш шифрується приватним ключем відправника. В результаті отримуємо цифровий підпис, який додається до повідомлення. Одержувач, отримавши повідомлення разом з підписом, спочатку розшифровує підпис за допомогою публічного ключа відправника, отримує хеш підписаного повідомлення і порівнює його з хешем отриманого повідомлення [8, 9]. Якщо хеші співпадають, це означає, що

повідомлення не було змінене і дійсно походить від того, хто стверджує, що його відправив. Якщо хеші не співпадають, це вказує на те, що повідомлення було змінено або підроблено.

Цифрові підписи забезпечують не тільки автентичність, але й цілісність та невідмовність. Невідмовність означає, що відправник не може заперечити факт підписання повідомлення, оскільки тільки він має доступ до свого приватного ключа, який використовувався для створення підпису. Цей принцип є основою для забезпечення юридичної сили цифрових підписів у багатьох країнах, де вони визнаються еквівалентними власноручним підписам у юридичних документах.

Алгоритми цифрових підписів, такі як RSA, DSA (Digital Signature Algorithm) і ECDSA (Elliptic Curve Digital Signature Algorithm), використовуються для створення і перевірки підписів. RSA, як вже згадувалося, є класичним алгоритмом, який використовує пару публічного та приватного ключів для здійснення криптографічних операцій, зокрема підписання. DSA – це стандартний алгоритм, який був розроблений для створення цифрових підписів і використовується в багатьох криптографічних протоколах. ECDSA є різновидом DSA, який застосовує еліптичні криві для забезпечення більшої безпеки при коротших довжинах ключів, що робить його популярним у сучасних системах, таких як криптовалюти і мобільні платформи.

Використання цифрових підписів є ключовим аспектом забезпечення безпеки в багатьох сучасних інтернет-протоколах, таких як SSL/TLS, які забезпечують захищену передачу даних у веб-браузерах, а також в електронній комерції та фінансових транзакціях. Вони також використовуються в системах електронного уряду, для підписання юридичних документів і контрактів, а також у цифрових сертифікатах для підтвердження ідентичності користувачів і сервісів.

Сучасні технології забезпечення безпеки інформації базуються на складних і надійних криптографічних методах, серед яких хешування і цифрові підписи відіграють важливу роль. Хешування дозволяє гарантувати цілісність даних і їх незмінність під час передачі, а цифрові підписи забезпечують автентичність джерела і підтверджують факт підписання повідомлення, що забезпечує невідмовність і правову силу. Разом ці технології формують основу для безпечної та довіреної комунікації в сучасному цифровому світі.

1.6 Аналіз сучасної літератури

Стаття [1] обговорює актуальні проблеми безпеки комп’ютерних мереж у сучасному світі, зокрема у контексті швидкого розвитку економіки та покращення рівня життя, що супроводжуються широким використанням комп’ютерів і мереж. Завдяки їх проникненню в усі сфери життя та роботи людей, з’являються нові виклики для забезпечення безпеки, такі як вірусні атаки, шкідливий код, масові розсилки небажаної інформації, крадіжки та витоки даних. Це стає серйозною проблемою на глобальному рівні, оскільки мільярди комп’ютерних вірусів завдають значної шкоди, впливаючи на повсякденне життя та роботу комп’ютерних систем. Тому питання захисту комп’ютерних мереж набуває все більшого значення і є важливим завданням для багатьох країн та регіонів. Стаття зосереджується на різних аспектах забезпечення безпеки комп’ютерних мереж, підкреслюючи їх важливість для нормального функціонування таких систем.

Стаття [2] пропонує методи захисту інформаційної безпеки комп’ютерних мереж за допомогою алгоритму кластеризації великих даних та аналізує різні ризики і загрози, пов’язані з сучасними комп’ютерними мережами. Використовуючи цей алгоритм, оцінюється відповідний рівень безпеки інформації комп’ютерних мереж на основі даних з облікових записів

цивільної авіації, зокрема даних про повітряний рух, і проводиться перевірка за допомогою симуляції. Результати дослідження показують, що запропонований метод добре підходить для обробки нелінійних характеристик безпеки комп'ютерних мереж, дозволяючи системно і точно відображати стан їхньої безпеки та надаючи ефективні ідеї для роботи з їх захистом.

Стаття [3] розглядає питання безпеки інформаційних мереж у сучасному суспільстві, де комп'ютерні мережі стають невід'ємною частиною життя людей, впливаючи на безпеку та економічний розвиток різних галузей. Зі зростанням залежності від комп'ютерних мереж, від державних таємниць до особистих розваг, проблема безпеки стає особливо актуальною. Оскільки комп'ютери мають певні вразливості, а передбачення щодо безпеки комп'ютерних систем в майбутньому є важливими, стаття акцентує на необхідності дослідження заходів захисту та алгоритмів оцінки безпеки комп'ютерних мереж. Зокрема, вона використовує методи нечіткого кластерного аналізу для оцінки проблеми захисту інформаційної безпеки мереж та надає стратегії і алгоритми для покращення рівня безпеки комп'ютерних мереж.

Стаття [4] розглядає проблему безпеки комп'ютерних мереж, яка стає все більш актуальною з розвитком комп'ютерних технологій. Традиційні методи криптографії вже не можуть повністю забезпечити захист особистої інформації, майна та конфіденційності. У цьому контексті особливу увагу приділено дослідженню алгоритмів машинного навчання, які можуть зробити процес захисту більш ефективним та доступним у реальному житті. Стаття вводить поняття машинного навчання та криптографії, а також аналізує сучасний стан досліджень у цих областях. Пропонується система виявлення вторгнень на основі випадкової просторової моделі та методів штучних нейронних мереж, а також інші стратегії захисту. Для реалізації багатовимірного розпізнавання ідентичності використовується алгоритм машинного навчання з кластеризацією для зберігання та класифікації різних

типів інформації. Проведене експериментальне дослідження за допомогою MATLAB показало, що система виявлення та оцінки забезпечує високу точність і швидкість виявлення шкідливого ПЗ з низьким рівнем хибних спрацьовувань, що підтверджує її ефективність для реальних користувачів.

Стаття [5] досліджує використання алгоритму нейронних мереж BP (Back Propagation) для оцінки безпеки комп'ютерних мереж. Цей алгоритм є методом навчання з учителем, який дозволяє визначити важливі вузли в нейронній мережі та оптимізувати їх поведінку для виявлення шкідливих дій у комп'ютерних мережах. Алгоритм BP застосовується для навчання і розпізнавання патернів, що дозволяє швидко і ефективно виявляти шкідливе програмне забезпечення з мінімальним впливом на продуктивність. Стаття також аналізує існуючі методи оцінки ризиків інформаційних систем, що через свою складність і взаємозалежність важко оцінити. Запропоновано використання вдосконаленого методу оцінки ризиків на основі алгоритму BP, який покращує традиційний метод аналітичної ієрархії. Такий підхід дозволяє динамічно оцінювати ризики безпеки комп'ютерних мереж, підвищуючи точність оцінки та відповідаючи реальним потребам забезпечення безпеки. Застосування цієї методики до оцінки інформаційних систем комп'ютерних мереж в Тибеті допомагає виявити приховані ризики та покращити індикатори роботи мережі, що сприяє комплексному захисту мереж.

Стаття [6] розглядає проблеми безпеки інформаційних систем комп'ютерних мереж, пов'язані з вразливістю їхньої структури та характеристик, що призводить до виникнення численних загроз для безпеки мережі. Для вирішення обмежень існуючих методів ідентифікації загроз безпеки комп'ютерних мереж, автори пропонують використання алгоритму кластеризації великих даних, детально описуючи налаштування набору даних та експериментальне середовище. Стаття також охоплює проектування робочого процесу та архітектури моделі, а також аналізує застосування розробленої технології ідентифікації загроз. Експериментальні результати

показують високу ефективність цієї технології, з точністю 91,5%, показником відгуку 91,4% і рівнем виявлення 86,4%, що підтверджує високу застосовність технології ідентифікації загроз безпеки комп'ютерних мереж на основі алгоритму кластеризації великих даних.

Стаття [7] досліджує технології безпеки комп'ютерних мереж в епоху великих даних, акцентуючи увагу на важливості зміцнення операційного середовища мережі та запобігання витоку даних. Вона підкреслює основні характеристики мережевих даних, такі як високий рівень безпеки, швидка обчислювальна ефективність і можливість спільного використання ресурсів. Стаття також аналізує різні типи безпекових ризиків комп'ютерних мереж, зокрема ризики вірусних вторгнень, вразливості мережевої безпеки та операційних систем. Зосереджуючись на реалізації технологій безпеки, вона розглядає застосування таких методів, як технології брандмауера, шифрування даних і технології інфраструктури відкритих ключів (PKI) для забезпечення належного захисту мережі.

Стаття [8] розглядає проблему безпеки комп'ютерних мереж в умовах швидкого розвитку соціально-економічної сфери та покращення рівня життя, що призводить до зростання значення інформаційних платформ для ефективної комунікації та взаємодії людей. Вона підкреслює, що з розвитком великих даних питання безпеки комп'ютерних мереж стає все більш критичним, оскільки забезпечення безпеки людей і майна вимагає особливої уваги. Стаття пропонує алгоритм шифрування інформації комп'ютерних мереж на основі нечіткої кластеризації (FCM), який спрямований на підвищення рівня захисту даних. Результати симуляцій показують, що після кількох ітерацій цей метод демонструє значно кращі результати порівняно з іншими алгоритмами, зменшуючи помилки на 35,24% і досягаючи показника відгуку 95,69%, що на 15,33% перевищує ефективність порівняного методу. Це дозволяє знизити ризик крадіжки та втрати даних, забезпечити безпечну експлуатацію даних і вирішити проблеми, пов'язані з перехресними зв'язками,

надаючи теоретичну підтримку для захисту інформаційної безпеки комп’ютерних мереж.

Стаття [9] розглядає застосування нейронних мереж для покращення безпеки комп’ютерних мереж, зокрема в умовах швидкого розвитку комп’ютерних технологій, які активно використовуються в повсякденному житті та виробництві. Проте, незважаючи на їх поширеність, комп’ютерні мережі не позбавлені проблем, таких як хакерські вторгнення, вразливості в системах безпеки та розповсюдження вірусів. Автори стверджують, що використання нейронних мереж може допомогти усунути ці недоліки, забезпечуючи обчислення інформації, контроль та розпізнавання, що дозволяє підвищити ефективність роботи та безпеку комп’ютерних мереж. Стаття зосереджується на огляді нейронних мереж та комп’ютерних мереж, а також аналізує їхнє застосування в оцінці безпеки комп’ютерних мереж, проводячи глибоке дослідження моделі симуляції оцінки безпеки за допомогою нейронних мереж для подальшого вдосконалення цієї сфери.

Стаття [10] аналізує поточний стан захисту інформаційної безпеки комп’ютерних мереж підприємств, які стали об’єктами численних вірусних атак та хакерських вторгнень, що ставить під загрозу як безпеку мережевої інформації, так і безпеку штучного інтелекту підприємств. У відповідь на ці загрози, дослідження пропонує серію заходів захисту, орієнтуючись на управління безпекою та технології захисту. Стаття детально розглядає методи оцінки заходів захисту комп’ютерних мереж на основі складного мережевого середовища, фокусуючись на чотирьох аспектах: ідеї алгоритмів оцінки, словник даних, модель розрахунку ефективності оборони та розрахунки на прикладах. Цей підхід і система конфігурацій безпеки допомагають керівникам підприємств краще захищати мережеву інформацію та безпеку штучного інтелекту, що має важливе практичне значення для забезпечення безпеки інформаційних мереж і контролю штучного інтелекту на підприємствах.

Стаття [11] розглядає актуальні проблеми безпеки комп’ютерних мереж в сучасному інформаційному суспільстві, зокрема загрози, пов’язані з витоками даних та хакерськими атаками. В якості ефективного заходу захисту інформації застосовується технологія шифрування даних, яка здатна забезпечити конфіденційність, цілісність і доступність даних, знижуючи ризики безпеки. У статті через огляд літератури аналізується прогрес досліджень і стан застосування стратегій безпеки комп’ютерних мереж, що базуються на технології шифрування даних, вивчаються її переваги, недоліки та напрямки подальшого розвитку. Дослідження показало, що після впровадження протоколу TLS/SSL кількість порушень цілісності електронної пошти зменшилась з 378 до 73, а порушень конфіденційності – з 318 до 81, що свідчить про ефективність шифрування у забезпеченні безпеки даних. Проте технологія шифрування має й певні обмеження, зокрема складність і швидкість алгоритмів шифрування, які потребують подальшого вдосконалення. Таким чином, майбутні дослідження повинні зосередитися на покращенні цієї технології для підвищення її безпеки.

Стаття [12] розглядає проблему комп’ютерної безпеки як глобальну задачу, яка викликає увагу багатьох країн. Зокрема, досліджується застосування нейронних мереж, як нового напрямку в галузі комп’ютерних технологій, що активно використовуються в умовах Інтернету. Зі збільшенням кількості користувачів комп’ютерних мереж зростають і ризики для безпеки, що ставить перед науковцями завдання знайти ефективні методи вирішення проблем безпеки в складних і мінливих умовах. У статті порівнюються різні технології комп’ютерної безпеки за допомогою експериментальних і порівняльних методів. Результати експериментів показують, що система з двома з двох гарантованих рівнів безпеки має найкращу продуктивність, з помилкою, що не перевищує 1-2 секунди. Використання згорткових нейронних мереж виявляється ефективним і швидким способом для обчислень у сфері комп’ютерної безпеки.

Стаття [13] розглядає роль високих інформаційних технологій, зокрема комп’ютерних технологій, в епоху великих даних, їхнє широке застосування в різних сферах життя, таких як будівництво, фінанси та торгівля. З розвитком цих технологій виникають нові вимоги до комп’ютерних користувачів і з’являються нові виклики для розвитку та застосування комп’ютерних систем. Однією з основних проблем є безпека комп’ютерних мереж, особливо в умовах великих даних, що призводить до нових загроз. Стаття пропонує використання технології шифрування даних як ефективного способу захисту мережевої безпеки, демонструючи, що точність цієї технології досягає 95,3%. Висновки дослідження покликані допомогти працівникам і користувачам у відповідних сферах забезпечити надійний захист комп’ютерних мереж.

Стаття [14] розглядає проблеми безпеки комп’ютерних мереж, які залишаються актуальними, незважаючи на розвиток Інтернет-технологій. Оскільки мережа є важливою частиною повсякденного життя і використовується в різних сферах, таких як економіка, технології, освіта та військові справи, питання її безпеки набувають особливої важливості, оскільки вони безпосередньо впливають на соціальний розвиток і національну безпеку. У статті досліджуються рішення для покращення безпеки мереж на основі інтелектуальних алгоритмів. Описується використання таких алгоритмів для моделювання безпеки комп’ютерних мереж, а також проводиться аналіз експериментальних даних. Виявлено, що при високій швидкості руху вузлів понад 25 м/с, протокол на основі алгоритму колонії мурах (ant_T) значно перевищує AODV за показниками доставки пакетів, що підтверджує ефективність безпечного маршрутизації в умовах високошвидкісного руху в мережах ad hoc.

Стаття [15] розглядає проблему безпеки інформаційних технологій у контексті їх поширення в повсякденному житті людей, оскільки комп’ютерні мережі стали важливим засобом комунікації. Хоча використання цих мереж у інформаційну епоху значно полегшило життя, вони також несуть серйозні

загрози для конфіденційності та безпеки. Організації, які використовують комп’ютерні мережі для обміну даними, повинні вжити заходів для захисту чутливої інформації від крадіжок або несанкціонованих змін. Стаття пропонує метод оцінки інформаційної безпеки в складному мережевому середовищі, який допоможе організаціям ефективно захищати свої дані та забезпечити надійність мережевого захисту.

Стаття [16] досліджує питання безпеки комп’ютерних мереж, зокрема в умовах їхнього поширення в повсякденному житті, що призводить до нових загроз, таких як хакерські атаки, промислове шпигунство та зловмисні вторгнення. Автори пропонують модель управління безпекою комп’ютерних мереж, що включає оцінку ризиків і прогнозування стану безпеки мережі. Для цього використовується модель штучної нейронної мережі, що базується на алгоритмі GABP (генетичний алгоритм зворотного поширення помилки). Експериментальні результати показують високу точність цього алгоритму: 95% для тренувальних даних і 99% для прогнозованих, що свідчить про його ефективність у оцінці і верифікації безпеки комп’ютерних мереж.

1.7 Висновки до першого розділу

У першому розділі було здійснено комплексний аналіз предметної області безпеки комп’ютерних мереж, а також визначено основні проблеми і задачі, що потребують дослідження в даній сфері. Було з’ясовано, що питання забезпечення безпеки в комп’ютерних мережах є надзвичайно актуальним у контексті сучасних технологій та зростаючих кіберзагроз. Основними аспектами безпеки є захист від зовнішніх і внутрішніх загроз, а також забезпечення цілісності, конфіденційності та доступності інформації.

Визначено, що сучасні комп’ютерні мережі стикаються з різноманітними загрозами, серед яких можна виділити атаки, спрямовані на порушення цілісності переданої інформації, крадіжку даних або отримання

несанкціонованого доступу до мережевих ресурсів. Важливим є розуміння різних типів атак, таких як атаки «відмова в обслуговуванні» (DoS), перехоплення даних, маніпулювання ними, а також соціальна інженерія, що вимагає розробки ефективних методів захисту.

У розділі також були висвітлені основні принципи криптографії як основного інструменту забезпечення безпеки в комп'ютерних мережах. Приділено увагу актуальним криптографічним алгоритмам, таким як AES, RSA і ECC, які використовуються для захисту даних на різних рівнях систем безпеки. Окрему увагу було приділено методам хешування та цифровим підписам, які забезпечують цілісність інформації та автентичність її джерела.

В результаті аналізу було встановлено, що сучасні методи захисту інформації, такі як криптографічні алгоритми, хешування та цифрові підписи, є основою для створення ефективних і надійних систем безпеки, що здатні протистояти новітнім загрозам. Проте з огляду на постійне вдосконалення методів атак, дослідження в області безпеки комп'ютерних мереж має бути спрямоване на розробку нових і вдосконалення існуючих механізмів захисту.

Задачі, поставлені в даному розділі, вимагають подальшого дослідження в контексті практичного застосування криптографічних методів для створення більш надійних і масштабованих рішень з безпеки комп'ютерних мереж, зокрема для захисту даних у реальному часі і забезпечення стійкості до новітніх атак.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ СУЧАСНИХ АЛГОРИТМІВ БЕЗПЕКИ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

2.1 Методи аутентифікації користувачів

Методи аутентифікації користувачів є невід'ємною частиною сучасних інформаційних систем (рис. 2.1), які забезпечують безпеку доступу до ресурсів і захист від несанкціонованого вторгнення. Аутентифікація, в своєму найширшому розумінні, означає процес перевірки того, чи є користувач тим, ким він себе видає [11]. Цей процес є важливим етапом у більшості інфраструктур безпеки, адже він передує наданню користувачеві прав доступу до системи чи її окремих ресурсів. Важливість правильного і надійного механізму аутентифікації важко переоцінити, оскільки у разі його порушення можливе серйозне порушення безпеки даних, втручання в роботу інформаційних систем і навіть витік конфіденційної інформації.

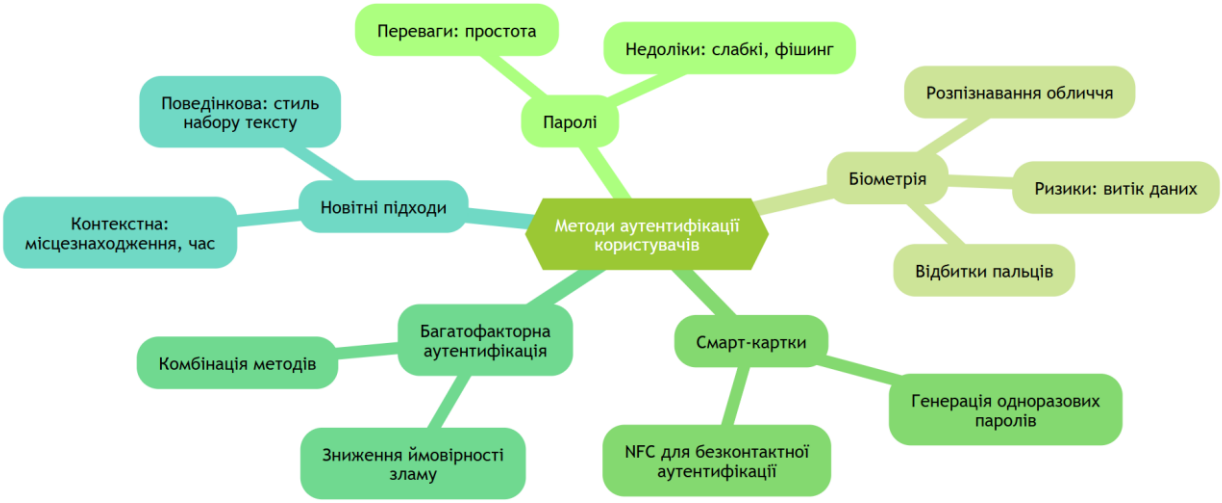

Рисунок 2.1 – Методи аутентифікації користувачів

Серед найбільш розповсюджених методів аутентифікації можна виділити такі, як аутентифікація за допомогою пароля, біометричних даних, смарт-карток, багатофакторна аутентифікація, а також новітні підходи, які використовують поведінкові або контекстуальні фактори для підтвердження особи користувача. Кожен з цих методів має свої переваги та недоліки, і їх вибір залежить від конкретних вимог безпеки, зручності та масштабів застосування. Наприклад, використання паролів є найпоширенішим методом аутентифікації, однак він має низку вразливостей, серед яких – можливість використання слабких паролів, їх несанкціоноване отримання через фішинг, а також уразливість до атак методом підбору. Одним із найбільш поширених методів аутентифікації є використання паролів. Це один із найпростіших і найбільш доступних способів перевірки користувача, однак він є вразливим до низки атак. Для покращення безпеки паролі часто комбінуються з додатковими параметрами, такими як вимоги до довжини, складності (використання великих і малих літер, цифр, спеціальних символів) та обов’язкові зміни пароля через певний проміжок часу. Водночас, оскільки люди часто вибирають паролі, які легко запам’ятовуються, це може призвести до того, що пароль буде простим для зламування або легко вгадуваним. Щоб протистояти цим проблемам, деякі організації застосовують додаткові методи захисту, такі як введення пароля в поєднанні з додатковими елементами автентифікації, які значно підвищують рівень безпеки.

Враховуючи слабкість традиційних паролів, з’являється потреба в більш надійних способах аутентифікації, таких як біометричні методи. Біометрична аутентифікація грунтується на використанні фізичних або поведінкових характеристик користувача, які унікальні для кожної людини. Найбільш поширеними біометричними параметрами є відбитки пальців, сітківка ока, розпізнавання обличчя, а також голосова аутентифікація [11, 12]. Біометричні методи є значно більш надійними, оскільки важко підробити або вкрасти фізичні характеристики людини. Однак, попри їх вищу ефективність, ці

методи також мають свої недоліки. Наприклад, можливі помилки в системах розпізнавання, а також порушення приватності та ризик витоку біометричних даних, що є особливо чутливими. Іншим важливим методом аутентифікації є використання смарт-карток або токенів. Ці фізичні пристрої генерують одноразові паролі або інші коди, які користувач вводить разом із своїм основним паролем для підтвердження своєї особи. Цей метод має перевагу в тому, що навіть якщо пароль користувача буде зламано, атака на смарт-картку або токен зазвичай є набагато складнішою. Крім того, багато сучасних смарт-карток інтегруються з технологією безконтактної передачі даних (NFC), що дозволяє здійснювати аутентифікацію без фізичного підключення картки до пристрою. Однак цей метод також не є абсолютно безпечним, адже сучасні технології зчитування карт можуть бути використані для копіювання даних з картки. Ще одним ефективним способом підвищення рівня аутентифікації є багатофакторна аутентифікація (MFA), яка поєднує два або більше методи аутентифікації для підтвердження особи користувача. Наприклад, це може бути комбінація пароля та біометричного даного або пароля та смарт-картки. Застосування кількох факторів значно знижує ймовірність несанкціонованого доступу, адже для успішного обходу системи зловмисник має обійти декілька рівнів захисту одночасно. Важливим аспектом багатофакторної аутентифікації є зручність для користувачів, адже надмірна кількість факторів може викликати дискомфорт і знизити ефективність використання цієї системи.

Новітніми методами аутентифікації є контекстна та поведінкова аутентифікація. Контекстна аутентифікація ґрунтується на зборі інформації про поведінку користувача або його оточення в процесі використання системи. Це може включати аналіз місцезнаходження користувача, пристрою, з якого він здійснює доступ до системи, а також часу доби або типу мережі [13]. Такі системи можуть виявляти аномалії в поведінці користувача і блокувати доступ, якщо система визначить, що спроба доступу виходить за межі звичайних параметрів. Поведінкова аутентифікація орієнтована на аналіз

індивідуальних патернів поведінки користувача, таких як стиль набору тексту, манера руху миші чи інші характеристики взаємодії з пристроєм. Ці технології є перспективними, оскільки вони дозволяють здійснювати аутентифікацію в режимі реального часу, не вимагаючи від користувача додаткових дій, але при цьому вони потребують великої обчислювальної потужності та можуть мати обмеження щодо точності.

З огляду на розвиток новітніх технологій, для забезпечення безпеки аутентифікації важливо враховувати не лише методи підтвердження особи користувача, але й процеси захисту інформації, що передається під час аутентифікації. Використання технологій шифрування даних є необхідним елементом для забезпечення безпеки при передачі аутентифікаційних даних через інтернет чи інші мережі, що є особливо важливим у сучасному цифровому середовищі, де багато систем підключені до глобальної мережі. Використання SSL/TLS протоколів, а також надійне шифрування паролів та біометричних даних, стає стандартом для збереження конфіденційності інформації.

Методи аутентифікації користувачів, незважаючи на свою ефективність, мають певні обмеження та вразливості. У зв’язку з постійним розвитком технологій безпеки необхідно удосконалювати існуючі методи, а також розробляти нові рішення, які зможуть забезпечити надійний захист від все більш складних загроз. У майбутньому можна очікувати, що нові підходи до аутентифікації, включаючи глибоке навчання та штучний інтелект, можуть значно підвищити точність і надійність аутентифікаційних систем [14]. Однак, важливо, щоб разом з розвитком технологій забезпечувалася і відповідна етика використання таких методів, зокрема, щодо захисту особистої інформації користувачів.

Висновуючи, можна сказати, що методи аутентифікації користувачів є ключовим елементом безпеки в інформаційних системах, і їх розвиток тісно пов’язаний з розвитком нових технологій, які стають необхідними для

забезпечення надійного захисту. Вибір методу аутентифікації залежить від багатьох факторів, таких як рівень безпеки, зручність використання та технологічні можливості, що дає змогу організаціям знайти оптимальне рішення для кожного конкретного випадку.

2.2 Протоколи аутентифікації

Протокол Kerberos (рис. 2.2) є однією з основних технологій для забезпечення безпечного обміну інформацією в мережах, зокрема в корпоративних середовищах. Його розробка почалася в 1980-х роках у Массачусетському технологічному інституті (MIT) в рамках проекту Athena. Назва «Kerberos» походить від імені грецького міфологічного пса, який охороняв вхід до підземного світу, що символізує безпеку доступу до обчислювальних ресурсів. Цей протокол надає механізм, що дозволяє користувачам та сервісам у мережі аутентифікуватися одне перед одним без необхідності передавати паролі у відкритому вигляді. Kerberos використовує метод симетричного шифрування та концепцію «білетів», які видаються авторитетним сервером аутентифікації, званим сервером ключів (Key Distribution Center, KDC) [13]. Процес аутентифікації в Kerberos складається з кількох етапів. Спочатку користувач вводить свій пароль, який використовується для отримання білета, що дозволяє йому отримати доступ до конкретного сервісу або ресурсу в мережі. Усі взаємодії між клієнтом і сервером здійснюються через KDC, що забезпечує безпеку, зменшуючи ризики, пов’язані з перехопленням паролів або іншою несанкціонованою діяльністю. Однією з основних переваг Kerberos є його здатність працювати в середовищах з великою кількістю користувачів і ресурсів, де необхідно забезпечити централізовану аутентифікацію. Протокол ефективно підтримує як одноразову аутентифікацію, так і багаторазову верифікацію користувачів без необхідності повторного введення паролів. Однак протокол також має

деякі недоліки, зокрема він вимагає наявності централізованого сервера KDC, що може стати єдиною точкою відмови в системі. Крім того, Kerberos має певні обмеження в контексті мобільних пристроїв та використання за межами корпоративних мереж.

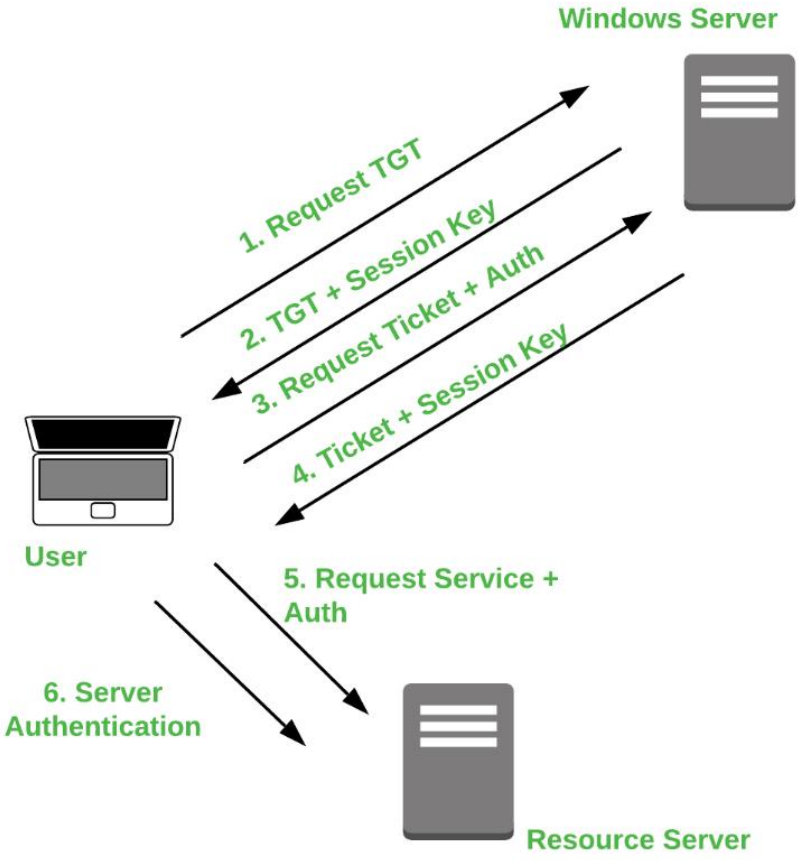

Рисунок 2.2 – Протокол Kerberos

Протокол OAuth (рис. 2.3) є іншим важливим механізмом аутентифікації, який активно використовується для забезпечення доступу до веб-сервісів. OAuth був розроблений для того, щоб спростити і безпечно передавати права доступу до ресурсу між різними сервісами без необхідності розголошення паролів. Цей протокол дозволяє користувачам надавати обмежений доступ до своїх ресурсів на одному сервісі стороннім додаткам без необхідності надавати їм свої облікові дані. OAuth працює за принципом делегування, де один сервер (ресурсний сервер) надає доступ до своїх даних іншим сервісам на основі дозволу, наданого користувачем. Процес

аутентифікації за допомогою OAuth включає кілька етапів, починаючи з того, що користувач надає дозвіл на доступ до свого профілю або ресурсів сторонньому сервісу. Далі цей сервіс отримує токен доступу, який дозволяє йому здійснювати запити до ресурсів користувача без необхідності вводити пароль або інші конфіденційні дані.

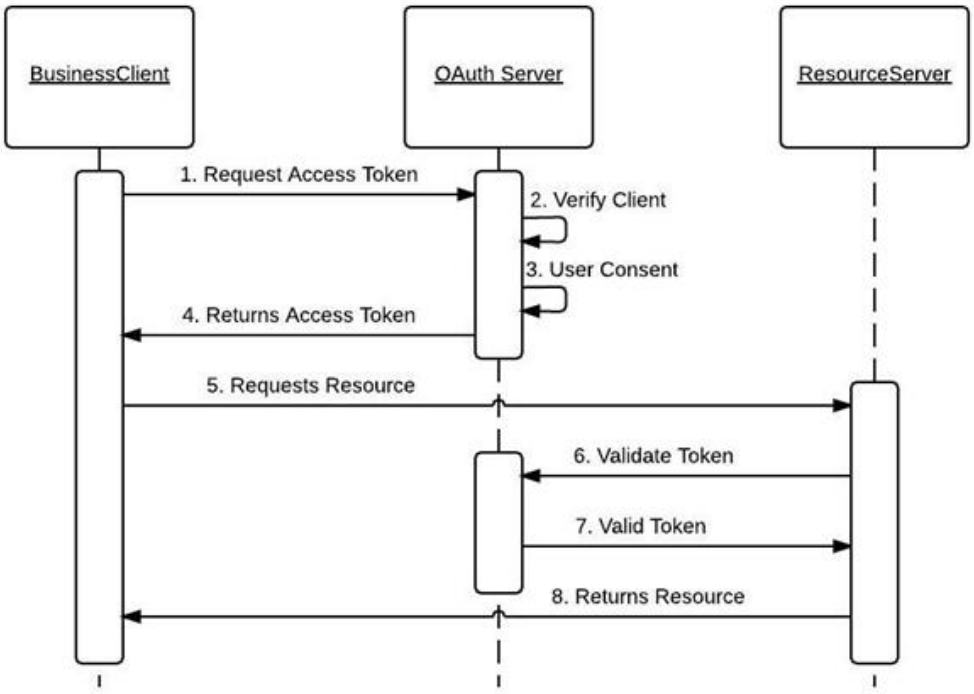

Рисунок 2.3 – Протокол OAuth

Одним з найбільш популярних варіантів використання OAuth є інтеграція з такими великими платформами, як Google, Facebook, Twitter, де користувачі можуть авторизуватися на сторонніх веб-сайтах за допомогою своїх облікових записів на цих платформах [14, 15]. Перевагою OAuth є можливість контролю доступу та безпечної взаємодії між різними сервісами без передачі паролів. Він підтримує механізми токенізації, що дозволяє уникнути витоку паролів при обміні інформацією. Крім того, OAuth забезпечує масштабованість і дозволяє інтегрувати сторонні сервіси без втручання в базову аутентифікаційну систему. Проте OAuth не є протоколом

аутентифікації в чистому вигляді, оскільки він не забезпечує перевірки особи користувача безпосередньо, а лише делегує доступ до ресурсів. Тому для забезпечення повної аутентифікації часто використовується разом з іншими протоколами, такими як OpenID Connect.

OpenID є протоколом аутентифікації, що також використовується для забезпечення безпечного і зручного доступу до різних веб-ресурсів. Основною метою OpenID є надання можливості користувачам використовувати один єдиний обліковий запис для авторизації на численних веб-сайтах і сервісах. Це дозволяє знизити кількість облікових записів, які користувач повинен пам’ятати, і одночасно забезпечує високий рівень безпеки. OpenID є децентралізованим протоколом, що дозволяє користувачам вибирати свого провайдера ідентифікації (наприклад, Google, Facebook або інші), який буде відповідати за аутентифікацію користувача на різних веб-ресурсах. Під час аутентифікації за допомогою OpenID користувач направляється на сторінку авторизації обраного провайдера, де вводить свої облікові дані. Після цього провайдер повертає токен доступу, який підтверджує ідентичність користувача. OpenID може працювати як окремо, так і у поєднанні з OAuth, забезпечуючи як аутентифікацію, так і делегування доступу до ресурсів.

Основною перевагою OpenID є його здатність об’єднувати декілька веб-сервісів під одним обліковим записом, що спрощує процес аутентифікації для користувачів. Крім того, цей протокол дозволяє знизити ризик фішингових атак, оскільки користувач не вводить свої облікові дані на сторонніх сайтах. Однак для організацій, які потребують більш високого рівня контролю за аутентифікацією, OpenID може бути менш прийнятним варіантом, оскільки він залежить від зовнішніх постачальників і може мати обмеження щодо безпеки та конфіденційності.

Таким чином, протоколи аутентифікації, такі як Kerberos, OAuth і OpenID, є основою для забезпечення безпечного доступу до інформаційних систем і ресурсів. Кожен з цих протоколів має свої переваги і недоліки, і їх

застосування залежить від конкретних вимог безпеки та зручності. Kerberos є ідеальним для централізованих корпоративних мереж, OAuth дозволяє безпечно делегувати доступ до сторонніх сервісів, а OpenID спрощує процес аутентифікації на численних веб-сайтах, забезпечуючи при цьому зручність і зниження ризиків. Вибір конкретного протоколу залежить від специфіки середовища та вимог до безпеки, що дозволяє організаціям та індивідуальним користувачам вибирати найбільш відповідні методи для захисту своїх даних та інформаційних ресурсів.

2.3 Алгоритми багатофакторної аутентифікації

Основною ідеєю багатофакторної аутентифікації є використання декількох елементів перевірки, що включають знання (наприклад, пароль), володіння (наприклад, смарт-картка або одноразовий код) і інколи фізичні характеристики користувача, такі як відбитки пальців, сітківка ока чи голос. Алгоритми, що лежать в основі цих систем, мають на меті не тільки підтвердити ідентичність користувача, а й забезпечити високий рівень захисту від різноманітних атак, таких як фішинг, атаки на основі підбору паролів або використання викрадених даних. Одним з основних підходів до організації багатофакторної аутентифікації є використання кількох незалежних факторів, кожен з яких підтверджує, що користувач має доступ до певної інформації або має фізичні характеристики, що відповідають його особистості. Перший фактор зазвичай полягає в чомусь, що користувач знає, наприклад, паролі або пін-коди [14, 16]. Другий фактор зазвичай полягає в чомусь, що користувач має при собі, наприклад, мобільний телефон або смарт-картку. І останнім часом додається третій фактор, пов’язаний з фізичними характеристиками, такими як біометрія.

Алгоритми, що реалізують багатофакторну аутентифікацію, значною мірою залежать від кожного з цих факторів і їх інтеграції. Наприклад, на

початковому етапі користувач вводить свій логін і пароль, які є першою лінією захисту. Цей етап перевіряє перший фактор аутентифікації – знання. Однак, навіть якщо пароль виявляється слабким або скомпрометованим, це не надає доступу до ресурсу, оскільки на наступному етапі може бути виконано перевірку другого фактора.

Другий фактор зазвичай передбачає використання фізичних пристроїв, таких як мобільні телефони, смарт-картки або одноразові паролі, що генеруються апаратними токенами або мобільними додатками. Одним з поширених прикладів є використання коду, що надсилається користувачу через SMS або через спеціалізовані мобільні додатки (наприклад, Google Authenticator або Authy). Ці коди змінюються через певні проміжки часу і є одноразовими, що значно знижує ймовірність їх перехоплення і несанкціонованого використання. Система, перевіряючи введений користувачем код, може порівняти його з тим, що був згенерований в момент запиту, тим самим переконуючись у правильності даних.

Крім того, існують так звані апаратні токени, що генерують одноразові паролі для аутентифікації. Такі пристрої можуть бути у вигляді фізичних карток, ключів або USB-пристроїв. Вони є дуже зручними в корпоративних середовищах, де забезпечення безпеки доступу до конфіденційної інформації є пріоритетним завданням. Токен генерує код на основі алгоритмів, які можуть використовувати як час, так і певні криптографічні ключі для створення унікального коду для кожного запиту. Цей метод дозволяє знизити залежність від Інтернет-з’єднання та додатково підвищує рівень захисту.

Третім фактором багатофакторної аутентифікації є біометричні методи, які використовують фізичні або поведінкові характеристики користувача. Оскільки ці характеристики є унікальними для кожної людини, вони становлять потужний засіб для підтвердження особи. Біометрія може включати відбитки пальців, сітківку ока, голос, або навіть відстеження рухів обличчя або манери писати [12]. Алгоритми, що працюють з біометрією,

зазвичай використовують складні методи обробки та порівняння даних, які дозволяють досягти високої точності та надійності в ідентифікації. Ці технології застосовуються для забезпечення високого рівня безпеки, особливо у випадках, коли інші методи можуть бути менш ефективними або зручними для користувачів.

Особливе місце серед алгоритмів БФА займають криптографічні методи, які забезпечують надійність і конфіденційність переданої інформації, а також запобігають можливим атакам. Багато сучасних систем багатофакторної аутентифікації використовують методи, які ґрунтуються на застосуванні асиметричного шифрування. Наприклад, під час аутентифікації з використанням смарт-карток або токенів може бути використано криптографічний процес, в якому генерується унікальний цифровий підпис або ключ для кожного сеансу доступу. Це дозволяє значно знизити ймовірність крадіжки і використання підроблених даних.

Важливим аспектом алгоритмів багатофакторної аутентифікації є їх здатність до інтеграції з іншими системами і забезпечення зручності для кінцевих користувачів. Застосування таких алгоритмів може бути інтегроване в різноманітні платформи і сервіси, починаючи від корпоративних мереж і закінчуючи мобільними додатками або інтернет-сервісами. Більшість сучасних рішень для багатофакторної аутентифікації використовують відкриті стандарти та протоколи, які дозволяють взаємодіяти з іншими системами без необхідності змінювати існуючі інфраструктури.

Завдяки високій гнучкості алгоритмів БФА, вони можуть застосовуватись в різних сценаріях, від високонавантажених серверних систем до мобільних додатків, де особливо важливо забезпечити високу зручність і швидкість аутентифікації. Водночас, саме багатофакторність дозволяє мінімізувати ймовірність несанкціонованого доступу навіть у разі компрометації одного з елементів аутентифікації, таких як пароль [13].

У майбутньому, з розвитком технологій і підвищенням рівня загроз кібербезпеці, алгоритми багатофакторної аутентифікації будуть ставати все більш досконалими і багатогранними. Вже зараз можна спостерігати активне використання штучного інтелекту та машинного навчання для аналізу поведінки користувачів і виявлення аномалій у їхній активності, що дозволяє створювати адаптивні системи аутентифікації, які автоматично вибирають найбільш відповідний рівень перевірки в залежності від контексту і ризику. Таким чином, алгоритми багатофакторної аутентифікації є важливим елементом забезпечення кібербезпеки і забезпечують високий рівень захисту, що відповідає сучасним вимогам цифрового світу.

2.4 Використання біометричних даних в аутентифікації

Принцип біометричної аутентифікації полягає в тому, що для ідентифікації або аутентифікації користувача використовуються його фізіологічні або поведінкові характеристики. Ці характеристики є унікальними для кожної людини, що забезпечує високий рівень точності в процесі верифікації. Існують різні типи біометричних даних, які можуть бути використані для аутентифікації, серед яких найбільш поширеними є відбитки пальців, розпізнавання обличчя, сітківка ока, голос і геометрія руки. Крім того, з розвитком технологій з’являються нові методи, наприклад, розпізнавання ходи або аналіз відгуків на дотик. Один із основних аспектів використання біометричних даних в аутентифікації полягає в їхній унікальності. Наприклад, відбитки пальців кожної людини є неповторними. Також сітківка ока, що використовується для ідентифікації, є таким же унікальним біометричним маркером, який майже неможливо підробити або змінити. Аналогічно, фізичні характеристики обличчя також є відносно стабільними протягом життя людини, хоча їх точність може знижуватися в разі значних змін у зовнішності, викликаних хворобами або віковими змінами [10]. Одним з найбільш

розповсюджених методів біометричної аутентифікації є використання відбитків пальців. Алгоритми розпізнавання відбитків пальців базуються на принципі порівняння та аналізу дрібних деталей, таких як лінії, вигини, шрами та інші особливості, що є характерними для кожної людини. Технології сканування відбитків пальців використовують різноманітні підходи для зчитування та аналізу цих характеристик, від оптичних датчиків до технологій ультразвукових сканерів. Порівняння збережених відбитків пальців із зразком, отриманим під час аутентифікації, дозволяє точно ідентифікувати користувача.

Розпізнавання обличчя є ще одним поширеним біометричним методом, який набув популярності завдяки здатності працювати безконтактно. Технології розпізнавання обличчя використовують спеціалізовані камери, які знімають зображення обличчя користувача. Ці зображення потім обробляються за допомогою алгоритмів машинного навчання, які дозволяють виділити характерні ознаки обличчя, такі як відстань між очима, форма носа і підборіддя, контури щелепи та інші геометричні особливості. Після аналізу зображення, система порівнює отриману інформацію з базою даних, що містить попередньо збережені біометричні дані, і визначає, чи відповідає новий зразок особі користувача.

Технології розпізнавання сітківки ока та райдужної оболонки також є ефективними методами біометричної аутентифікації. Сітківка ока є високоточним біометричним маркером, оскільки вона має велику кількість унікальних характеристик, які важко змінити навіть в результаті хвороб або травм. Використовувані для цього технології здатні зчитувати та аналізувати малесенькі судини сітківки, що дозволяє забезпечити високий рівень точності під час верифікації. Іншим методом є використання розпізнавання голосу, яке здебільшого застосовується в системах телефону або для дистанційного доступу до певних ресурсів. Звукові хвилі, що виходять від голосових органів людини, можуть бути унікальними для кожної особи і використовуються для

підтвердження ідентичності. Одним із сучасних напрямків в біометрії є використання поведінкових характеристик, таких як манера писати, рухи рук або навіть хода [11]. Ці методи є особливо перспективними, оскільки вони дозволяють проводити аутентифікацію в реальному часі, не потребуючи спеціальних пристроїв. Наприклад, аналіз ходи може бути здійснений за допомогою спеціальних датчиків, встановлених в смартфоні, що дозволяє визначити індивідуальні особливості пересування людини. Це відкриває нові можливості для аутентифікації в середовищах, де використання традиційних біометричних методів може бути обмеженим.

Основними перевагами біометричної аутентифікації є її унікальність і труднощі підробки. Для зловмисників значно важче підробити або викрасти біометричні дані порівняно з паролями або іншими традиційними методами аутентифікації. Крім того, біометрія знижує потребу в запам’ятовуванні численних паролів та інших даних, що значно покращує зручність для користувачів. З біометрією стає можливим надання доступу до системи без необхідності введення коду чи пароля, що спрощує процес аутентифікації.

Однак, незважаючи на численні переваги, використання біометрії також має певні недоліки та виклики. Одним із основних проблем є забезпечення конфіденційності та захисту біометричних даних. Оскільки ці дані є постійними і неможливо їх змінити, їх викрадення або компрометація може мати серйозні наслідки для безпеки користувача. Якщо пароль або пін-код буде викрадений, користувач завжди може змінити їх, а ось біометричні дані неможливо замінити. Тому для збереження безпеки необхідно використовувати додаткові методи захисту, такі як шифрування біометричних даних і застосування технологій зберігання їх у розподіленому вигляді, щоб мінімізувати ризик їх компрометації.

Іншою проблемою є ймовірність помилкових спрацьовувань, таких як помилкове підтвердження особи (false accept) або відмова в доступі через помилкове невизнання (false reject). Це може статися, якщо біометричні дані

погано зчитуються або якщо є фізіологічні зміни у користувача, такі як травми, зміни в зовнішності або вік [13]. Для мінімізації таких ризиків часто використовуються комбіновані системи, де біометрія доповнюється іншими факторами, такими як паролі або смарт-карти.

Отже, біометрична аутентифікація є потужним інструментом у забезпеченні безпеки, що використовує унікальні фізіологічні або поведінкові характеристики людини для перевірки її ідентичності. Використання біометрії стає все більш поширеним у різних сферах, від мобільних пристроїв до високозахищених корпоративних і урядових систем. Проте з розвитком технологій і підвищенням рівня загроз безпеки виникає необхідність у постійному вдосконаленні методів захисту біометричних даних, а також у розробці нових підходів для інтеграції біометрії з іншими методами аутентифікації для досягнення оптимального рівня безпеки.

2.5 Протоколи захисту даних на транспортному рівні

SSL і TLS є протоколами криптографічного захисту, які зазвичай використовуються для забезпечення безпечного з’єднання між клієнтом та сервером, а також для захисту переданих даних у веб-застосунках. Зазначені протоколи виконують функції забезпечення конфіденційності переданої інформації, а також гарантують її цілісність, тобто запобігають модифікації даних під час їх передачі. Їхня важливість для безпеки сучасного інтернет-спілкування важко переоцінити, адже більшість сучасних веб-застосунків, від онлайн-банкінгу до електронної комерції, використовують саме ці протоколи для забезпечення захисту персональних даних користувачів. SSL, розроблений компанією Netscape в середині 1990-х років, став першим протоколом, який надав можливість забезпечити безпечну передачу даних через незахищену мережу інтернет. Пізніше, з урахуванням виявлених вразливостей та необхідності вдосконалення безпеки, на основі SSL був розроблений більш

досконалий протокол TLS [15]. Важливо відзначити, що хоча SSL та TLS мають суттєві відмінності в архітектурі та механізмах роботи, в повсякденному використанні обидва терміни часто вживаються взаємозамінно, хоча технічно вони позначають різні версії протоколу.

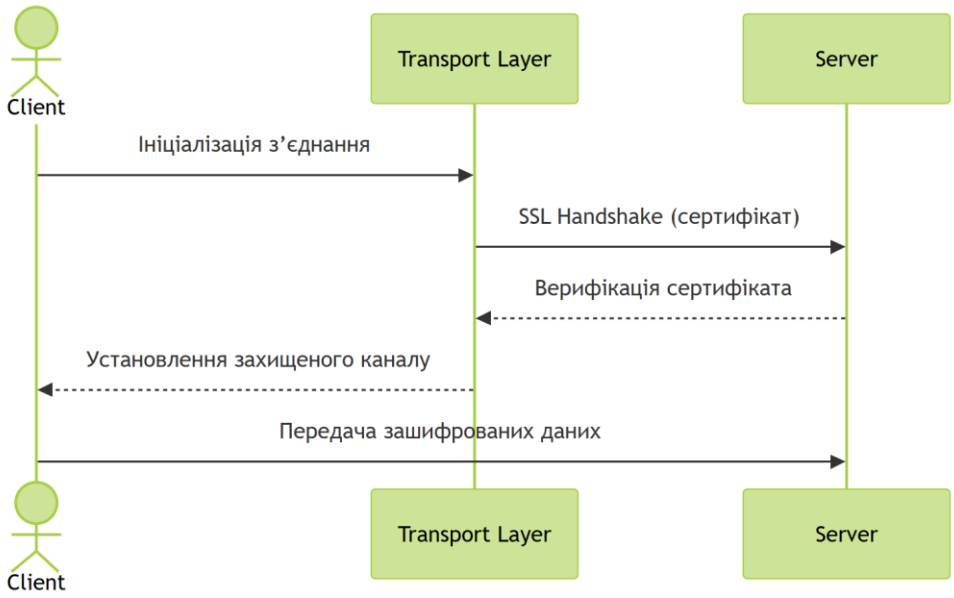

Рисунок 2.4 – Протоколи захисту даних на транспортному рівні

Основною метою як SSL, так і TLS є забезпечення захищеного каналу для передачі даних між двома сторонами, наприклад, між веб-браузером користувача та веб-сервером. Під час встановлення з'єднання через SSL або TLS ініціалізується процедура, яка включає кілька важливих етапів: аутентифікацію серверної сторони, обмін ключами шифрування та встановлення захищеного каналу для передачі даних [7]. На першому етапі клієнт перевіряє автентичність сервера за допомогою цифрових сертифікатів, що видані довіреними сертифікаційними центрами. Цей процес гарантує, що користувач взаємодіє саме з тим сервером, до якого він намагається підключитися, а не з потенційним зловмисником, який може здійснити атаку «людина посередині» (man-in-the-middle).

Однією з ключових особливостей SSL та TLS є використання асиметричного шифрування для обміну ключами шифрування та симетричного шифрування для самих даних. Під час встановлення з'єднання відбувається обмін відкритими ключами за допомогою яких генеруються симетричні ключі, які використовуються для шифрування даних в процесі подальшої взаємодії. Такий підхід дозволяє зберегти баланс між високим рівнем безпеки та ефективністю шифрування. Використання симетричного шифрування, яке є значно швидшим у порівнянні з асиметричним, дозволяє швидко і ефективно передавати великі обсяги даних при високому рівні захисту.

Щодо IPSec, то цей протокол забезпечує захист даних на мережевому рівні, забезпечуючи шифрування та автентифікацію даних, що передаються через IP-мережу. IPSec зазвичай використовується для захисту трафіку між пристроями, такими як маршрутизатори або між комп'ютерами в корпоративних мережах, а також для створення віртуальних приватних мереж (VPN). На відміну від SSL/TLS, які зазвичай використовуються для забезпечення безпеки з'єднань між клієнтами і серверами, IPSec працює на рівні мережевих пакетів, тобто він захищає всю передачу даних між двома кінцевими точками, не залежно від того, який тип трафіку передається.

Один з основних компонентів IPSec – це дві основні моделі роботи, які визначають способи шифрування та автентифікації: транспортний режим та тунельний режим [7, 9]. В транспортному режимі лише корисне навантаження IP-пакету шифрується, а заголовки залишаються незмінними, що підходить для захисту з'єднань між кінцевими пристроями. В тунельному режимі весь пакет, включаючи заголовки, шифрується, що дозволяє створювати захищені «тунелі» для передавання даних між двома мережами, забезпечуючи конфіденційність та цілісність для всього трафіку. Це дає можливість організувати захищений канал для передачі даних через незахищену мережу, наприклад, через інтернет, таким чином створюючи віртуальну приватну

мережу (VPN). Іншою важливою складовою IPSec є використання двох основних протоколів для забезпечення захисту даних: протоколу AH (Authentication Header) і протоколу ESP (Encapsulating Security Payload). Протокол AH забезпечує автентифікацію та цілісність даних, але не здійснює їх шифрування. У свою чергу, ESP забезпечує як автентифікацію, так і шифрування даних, що робить його більш універсальним і широко використовуваним у практиці. У комбінації з різноманітними методами аутентифікації та криптографії, IPSec забезпечує високий рівень безпеки для мережевих з'єднань.

SSL/TLS та IPSec мають кілька спільних рис, але відрізняються за механізмами реалізації та сферами застосування. Ось кілька основних відмінностей, які варто зазначити: протоколи SSL/TLS орієнтовані на захист конкретних з'єднань між кінцевими точками (наприклад, між браузером і веб-сервером), в той час як IPSec захищає все передавання даних між мережами або пристроями в рамках однієї мережі. Протоколи SSL/TLS зазвичай використовуються для веб-базованих додатків та сервісів, де важливо забезпечити безпечну взаємодію користувачів з веб-ресурсами, в той час як IPSec є більш корисним у корпоративних мережах та для створення віртуальних приватних мереж.

Сучасні версії SSL і TLS, такі як TLS 1.2 і TLS 1.3, значно покращили безпеку та ефективність протоколів, включаючи вдосконалені механізми шифрування, а також покращену захищеність від атак типу «людина посередині» та інших форм втручання [8]. Протокол TLS 1.3, який став останнім стандартом в області безпеки інтернет-з'єднань, має значні покращення в порівнянні з попередніми версіями, зокрема, він оптимізує процес встановлення з'єднання, знижує кількість кроків, необхідних для аутентифікації, і забезпечує більш сильне шифрування.

Враховуючи постійну еволюцію технологій, роль протоколів захисту даних на транспортному рівні буде лише зростати, особливо в умовах

зростаючої кіберзагрози та необхідності забезпечення безпеки критичних інфраструктур. SSL/TLS і IPSec є важливими елементами інфраструктури захисту даних, які дозволяють забезпечити високий рівень захисту і надійності передачі інформації в глобальних мережах, таких як Інтернет. Однак постійна увага до оновлень і розвитку цих протоколів, а також інтеграція новітніх криптографічних методів, є необхідною для того, щоб забезпечити максимальний захист від нових типів атак та загроз, які з'являються з кожним роком.

2.6 Протоколи захисту на прикладному рівні

Протоколи захисту на прикладному рівні є важливими елементами для забезпечення безпеки інформації, що передається через мережу, особливо в контексті глобальної мережі Інтернет, де ризики несанкціонованого доступу та зловмисних атак постійно зростають. Від того, наскільки ефективно працюють ці протоколи, залежить безпека конфіденційних даних, таких як паролі, особисті дані користувачів, платіжні реквізити та інші чутливі відомості. У цьому контексті протоколи HTTPS (Hypertext Transfer Protocol Secure), SSH (Secure Shell) і SFTP (Secure File Transfer Protocol) є одними з найпоширеніших інструментів для забезпечення захисту даних на прикладному рівні [11]. Ці протоколи надають механізми для забезпечення конфіденційності, цілісності та автентичності даних, що передаються, і використовуються в широкому спектрі застосувань – від веб-браузерів до віддалених з'єднань між серверами і користувачами.

HTTPS є протоколом, що дозволяє забезпечити безпечну передачу даних між веб-браузером та веб-сервером, захищаючи інформацію, що передається, від можливих атак, таких як підслуховування та зміна даних. HTTPS є зазвичай використовується для забезпечення безпеки в веб-додатках, де важливо, щоб передана інформація, включаючи конфіденційні дані

користувачів, була захищена від зловмисників, які можуть намагатися перехопити або змінити її. Протокол HTTPS є розширенням HTTP (Hypertext Transfer Protocol), однак з додаванням криптографії для забезпечення безпеки. Основним механізмом безпеки в HTTPS є SSL (Secure Sockets Layer) або TLS (Transport Layer Security), які шифрують дані, що передаються, а також забезпечують автентифікацію серверної сторони за допомогою цифрових сертифікатів.

Процес встановлення з'єднання по HTTPS включає кілька ключових етапів. Перш за все, коли користувач намагається підключитися до веб-сайту через HTTPS, веб-браузер встановлює з'єднання з веб-сервером і перевіряє, чи має сервер дійсний цифровий сертифікат, виданий довіреним сертифікаційним центром. Цей сертифікат підтверджує, що сервер, з яким встановлено з'єднання, є тим, за кого він себе видає, і що він дійсно належить відповідному організації чи компанії. У разі позитивної перевірки сертифіката браузер продовжує процес встановлення з'єднання, в ході якого відбувається обмін ключами для симетричного шифрування, а також визначення методів шифрування, які будуть використовуватися для захисту переданих даних. За допомогою цих криптографічних алгоритмів забезпечується конфіденційність і цілісність переданої інформації, а також автентичність комунікацій, що виключає можливість проведення атак типу «людина посередині» (man-in-the-middle) [10].

HTTPS відіграє особливо важливу роль у забезпеченні безпеки онлайн-транзакцій, таких як банківські операції, покупки в інтернет-магазинах, а також при передачі чутливих особистих даних. Завдяки використанню SSL/TLS шифрування HTTPS захищає від підслуховування, перехоплення чи модифікації даних під час їх передачі, що робить його незамінним інструментом для сучасних веб-сервісів, де безпека користувачів є пріоритетом.

Протокол SSH є ще одним важливим інструментом захисту даних, особливо коли йдеться про віддалений доступ до серверів та систем. SSH (Secure Shell) є протоколом для безпечного керування віддаленими комп’ютерами та серверами через незахищені мережі, такі як Інтернет. Це засіб, який дозволяє адмініструвати сервери, виконувати команди, передавати файли, здійснювати налаштування систем без необхідності фізичної присутності біля пристрою, при цьому гарантуючи високий рівень безпеки для всієї взаємодії. Використання SSH дозволяє захистити віддалений доступ, забезпечуючи аутентифікацію та шифрування даних, що передаються, що захищає від атак типу підслуховування, перехоплення даних та несанкціонованого доступу.

Протокол SSH спочатку забезпечує автентифікацію користувача, зазвичай через пару ключів: приватний та публічний. Публічний ключ зберігається на сервері, а приватний – у користувача [9]. Для кожного сеансу SSH генерується унікальний сеансовий ключ, що використовується для шифрування всієї інформації, яка передається між користувачем та сервером. Таким чином, навіть якщо зловмисник перехопить зашифровані дані, він не зможе отримати доступ до змісту переданої інформації без наявності приватного ключа. Крім того, за допомогою аутентифікації на основі ключів, SSH усуває необхідність у введенні пароля при кожному з’єднанні, що значно підвищує рівень безпеки. Важливо, що підключення через SSH є більш безпечним за використання старих, нешифрованих протоколів, таких як Telnet чи Rlogin, які передають паролі і дані у відкритому вигляді, що робить їх уразливими до атак.

Протокол SFTP (Secure File Transfer Protocol) є подібним до SSH у тому сенсі, що він також працює на основі SSH та використовує шифрування для захисту передаваних файлів. SFTP забезпечує безпечну передачу файлів між комп’ютерами по мережі, що дозволяє уникнути атак, таких як перехоплення даних або маніпуляції з файлами під час їх пересилання. На відміну від

традиційного FTP (File Transfer Protocol), який передає дані у відкритому вигляді, SFTP шифрує всі операції, що виконуються, зокрема, і самі файли, що передаються, а також команди управління [7].

SFTP використовує SSH для встановлення безпечного з’єднання між клієнтом і сервером, після чого забезпечує захист усієї переданої інформації, включаючи файли, дані про місце їх зберігання та доступ до них. Протокол SFTP є набагато більш захищеним порівняно з FTP, оскільки в ньому використовується повне шифрування для забезпечення конфіденційності та цілісності переданої інформації. Цей протокол дозволяє ефективно використовувати захист даних при роботі з великими обсягами інформації, а також при здійсненні важливих бізнес-операцій, таких як передача фінансових звітів або конфіденційних корпоративних документів.

Протоколи HTTPS, SSH та SFTP мають спільну мету – забезпечення безпеки передачі даних. Всі ці протоколи використовують різні механізми криптографії для забезпечення конфіденційності і цілісності інформації, що передається. Однак, кожен з цих протоколів має свої особливості і використовується в різних контекстах. HTTPS зазвичай застосовується для забезпечення безпечної взаємодії з веб-сайтами, SSH – для забезпечення безпечного віддаленого доступу до серверів і систем, а SFTP – для захисту передавання файлів. Всі ці протоколи стали стандартом безпеки для сучасних інтернет-технологій і продовжують розвиватися, щоб відповідати новим вимогам і викликам у сфері безпеки.

2.7 Висновки до другого розділу

У дослідницькій частині було розглянуто сучасні алгоритми та методи забезпечення безпеки в комп’ютерних мережах, що є необхідними для захисту даних, які передаються через мережу, а також для забезпечення безпеки користувачів під час їх взаємодії з різними мережевими сервісами. Аналіз було

проведено за кількома напрямками, що охоплюють як методи аутентифікації користувачів, так і протоколи захисту даних на різних рівнях мережі.

Методи аутентифікації користувачів, такі як паролі, токени, сертифікати, а також новітні технології багатофакторної аутентифікації, які комбінують кілька факторів для перевірки особи користувача, є основою для запобігання несанкціонованому доступу до чутливих ресурсів. Окрім того, велика увага приділяється використанню біометричних даних для аутентифікації, що, на відміну від традиційних методів, забезпечує більш високий рівень захисту, оскільки біометричні характеристики є унікальними для кожної особи і важко підробляються.

Протоколи аутентифікації, такі як Kerberos, OAuth і OpenID, мають ключове значення в сучасних мережах, зокрема в інтегрованих і дистрибуційних середовищах. Вони дозволяють забезпечити надійне підтвердження особи користувача та здійснювати автентифікацію на різних рівнях систем, як для доступу до внутрішніх мереж, так і для інтеграції з зовнішніми додатками. Це особливо важливо для підтримки безпеки в хмарних обчисленнях та сервісах, де верифікація користувачів та їх доступів є критичним елементом.

Протоколи захисту даних, зокрема SSL/TLS та IPSec, забезпечують захист інформації на транспортному рівні, використовуючи шифрування і механізми автентифікації для запобігання перехоплення та зміни даних під час їх передачі через мережу. Ці протоколи є основою для забезпечення конфіденційності і цілісності даних, що передаються в Інтернеті, особливо для таких сервісів, як електронна пошта, онлайн-платежі, доступ до чутливих ресурсів. Протоколи захисту на прикладному рівні, такі як HTTPS, SSH, SFTP, гарантують безпеку даних безпосередньо на рівні додатків, забезпечуючи захист під час здійснення веб-операцій або передавання файлів, що робить їх важливими інструментами для підтримки безпеки в глобальних мережах.

Особливе значення мають протоколи захисту голосових і відео-комунікацій, такі як SRTP і ZRTP, які гарантують конфіденційність та безпеку під час передачі голосових та відео-даних в реальному часі, забезпечуючи захист від атак типу «людина посередині» та від несанкціонованого доступу до комунікацій, що особливо важливо в умовах постійного зростання обсягів відеоконференцій і голосових дзвінків через Інтернет.

Дослідження сучасних алгоритмів безпеки в комп’ютерних мережах показує, що для забезпечення комплексного захисту необхідно застосовувати багаторівневі підходи, що включають методи аутентифікації, протоколи шифрування, захисту даних і забезпечення безпеки на прикладному рівні. Використання цих технологій у сукупності дозволяє створити надійний захист для даних, що передаються через мережу, та забезпечити безпеку користувачів і сервісів у сучасних комп’ютерних мережах.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Постановка задачі

У прикладній частині поставлена задача розробити програмну систему для забезпечення безпеки мережевих підключень з наступними функціональними можливостями:

– реєстрація нових користувачів із збереженням хешованих паролів та сольових значень шифрування;

– валідація імен користувачів відповідно до заданих правил;

– надання користувачам ролей (наприклад, адміністратор, аналітик, користувач);

– перевірка автентичності користувачів з урахуванням кількості невдалих спроб входу;

– блокування облікових записів на визначений час після перевищення максимальної кількості невдалих спроб;

– генерація майстер-ключа для шифрування;

– шифрування та розшифрування повідомлень для забезпечення конфіденційності даних;

– запис подій (включаючи автентифікацію, спроби доступу тощо) в журнал аудиту;

– налаштування зберігання логів у файл та консоль;

– відстеження активності клієнтських підключень із підтримкою чорного списку для IP-адрес;

– перевірка IP-адрес клієнтів на відповідність дозволеним мережам;

– ідентифікація сесій користувачів за унікальними ідентифікаторами;

– ведення запису про активні сесії із зазначенням користувача, IP-адреси та часу створення;

– зберігання конфігурацій системи в json-файлі з можливістю автоматичного створення стандартних налаштувань за замовчуванням.

Система повинна функціонувати на базі Python і використовувати SQLite для збереження даних, бібліотеки socket для роботи з мережею, hashlib та cryptography для шифрування. Код має бути структурованим, із застосуванням принципів модульності та читабельності.

3.2 Архітектура програмного забезпечення

Розроблений програмний код представляє складну багатокомпонентну систему мережевої безпеки, яка забезпечує моніторинг, автентифікацію, управління користувачами та захист мережевих підключень. Архітектура цієї системи інтегрує сучасні технології криптографії, обробки даних і мережевого програмування, а також забезпечує високу рівень конфіденційності, цілісності і доступності даних.

На верхньому рівні архітектури виділяється клас AdvancedNetworkSecuritySystem, який виступає ядром всієї системи. Його ініціалізація включає кілька важливих етапів. По-перше, завантажується конфігурація з файлу security_config.json, що дозволяє забезпечити гнучкість у налаштуванні параметрів безпеки, таких як максимальна кількість спроб входу, тривалість блокування IP-адрес, дозволені діапазони мереж, а також порт і хост для мережевого сервера.

Якщо файл конфігурації відсутній, система автоматично генерує його з типовими параметрами, що гарантує її автономну роботу навіть у початкових умовах. По-друге, проводиться налаштування логування, що передбачає створення окремого каталогу для журналів і використання багаторівневої системи логів з підтримкою виведення як у файл, так і в консоль. По-третє, ініціалізується база даних, в якій зберігається інформація про користувачів і аудит. Цей етап включає створення таблиць для облікових записів

користувачів, зберігання хешованих паролів, а також збереження інформації про події системи, таких як спроби входу або підозрілі дії.

У контексті забезпечення криптографічного захисту реалізовано генерацію майстер-ключа на основі алгоритму PBKDF2 із застосуванням SHA-256 як функції хешування. Цей ключ використовується для шифрування і розшифрування конфіденційних повідомлень через механізм бібліотеки Fernet, який гарантує дотримання стандартів безпеки, включаючи використання унікальних токенів і аутентифікації даних. Ця частина архітектури також передбачає використання унікальних солей і багаторазове хешування для надійного зберігання паролів, що робить практично неможливим їх компрометацію навіть у випадку викрадення бази даних.

Додатково в системі реалізовано механізми перевірки дозволених мережевих діапазонів, які визначають, чи підпадає IP-адреса клієнта під дозволені правила. Це досягається за допомогою модулів Python для роботи з IP-адресами та мережами. Такий підхід забезпечує ефективний фільтр для небажаних підключень і сприяє захисту від потенційних атак із зовнішніх мереж.

Процеси автентифікації користувачів передбачають перевірку облікових даних, відстеження кількості спроб входу та динамічне блокування облікових записів у разі перевищення дозволених спроб. Для цього зберігається інформація про час останньої невдалої спроби і кількість таких спроб у словнику connection_attempts. У разі перевищення заданої кількості невдалих входів користувач блокується на певний період, що визначається конфігурацією системи.

Система має можливість реєстрації нових користувачів з валідацією імені користувача за допомогою регулярних виразів. Процес реєстрації включає генерацію унікальної солі для кожного користувача, хешування пароля і збереження цієї інформації в базі даних разом із метаданими, такими як дата створення облікового запису та роль користувача. Ролі, у свою чергу,

дозволяють розподілити права доступу в системі, забезпечуючи базовий рівень багаторівневої безпеки.

Моніторинг мережевої активності здійснюється за допомогою окремого потоку, який працює паралельно із серверними процесами. Цей потік відповідає за очищення застарілих блокувань, оновлення даних про спроби входу та підтримку актуального стану системи. Це рішення дозволяє оптимізувати використання ресурсів і забезпечує безперервність роботи навіть за умов високого навантаження.

Компонент серверної частини реалізує механізм прийому підключень, перевірки IP-адрес і обробки клієнтських запитів. Сервер працює у режимі багатопотоковості, що дозволяє обробляти кілька підключень одночасно. Кожне підключення аналізується на предмет автентичності, а також можливість доступу до певних ресурсів визначається на основі ролі користувача та мережевої адреси.

Значну увагу приділено аудиту дій у системі. Журнал подій включає записи про автентифікацію, дії адміністратора, створення або видалення облікових записів, а також про мережеві події, такі як блокування IP-адрес чи підозрілі спроби доступу. Ця інформація зберігається в базі даних і доступна для аналізу, що дозволяє визначити потенційні загрози та вдосконалювати захисні механізми системи.

Таким чином, розроблений програмний код реалізує комплексну систему мережевої безпеки, яка поєднує сучасні технології криптографії, управління обліковими записами, мережевого моніторингу та багаторівневого захисту. Архітектура системи орієнтована на забезпечення гнучкості, масштабованості та високої продуктивності, що робить її придатною для використання в умовах реальних загроз інформаційній безпеці.

3.3 Опис використаних технологій

Процес розробки програмного забезпечення є багатогранним і потребує використання різноманітних технологій та інструментів для досягнення максимальної ефективності, гнучкості та надійності системи. У цьому контексті особливу увагу було приділено вибору відповідного стеку технологій, які здатні задовольнити вимоги до продуктивності, безпеки, масштабованості й зручності підтримки.

На етапі розробки основною мовою програмування було обрано Python, що забезпечує оптимальний баланс між простотою написання коду, потужністю бібліотек і можливістю інтеграції з іншими компонентами. Ця мова дозволяє ефективно вирішувати завдання обробки даних, реалізації мережевих протоколів, криптографії та управління потоками завдяки своєму багатому екосистемному середовищу. Python також підтримує кросплатформеність, що робить його ідеальним вибором для розробки програмного забезпечення, яке може працювати на різних операційних системах.

Важливим компонентом системи є бібліотека Fernet з модуля cryptography, яка використовується для реалізації криптографічного шифрування даних. Вона забезпечує шифрування, що відповідає стандартам AES із 128-бітним ключем у режимі CBC (Cipher Block Chaining), а також використовує HMAC для забезпечення аутентифікації даних. Використання цього інструмента дозволяє створити надійний механізм захисту конфіденційної інформації, яка передається між клієнтом і сервером, а також зберігається у внутрішніх базах даних. Додатково алгоритм PBKDF2 із функцією хешування SHA-256 використовується для генерації майстер-ключів із заданою кількістю ітерацій, що забезпечує стійкість до атак типу brute-force.

Для роботи з базами даних обрано SQLite як локальне рішення для збереження інформації про користувачів, спроби входу, події системи та мережеві журнали. SQLite має кілька вагомих переваг, серед яких компактність, відсутність необхідності у встановленні серверного програмного забезпечення й повна сумісність із мовою SQL. База даних була інтегрована з Python через бібліотеку sqlite3, яка надає високий рівень абстракції для роботи з SQL-запитами, дозволяючи виконувати операції створення, читання, оновлення та видалення даних. Цей підхід забезпечує зручність управління інформацією й можливість швидкого розширення функціоналу.

Логування є невід'ємною частиною системи, і для цього використано стандартний модуль logging, що входить до складу Python. Цей модуль дозволяє створювати багаторівневі журнали подій із підтримкою різних форматів і рівнів важливості. У рамках реалізації розроблена індивідуальна конфігурація, яка дозволяє зберігати журнали в окремому каталозі, а також забезпечувати виведення в консоль для оперативного моніторингу стану системи. Логування включає обробку виключень, запис інформації про підключення клієнтів, успішні й невдалі спроби автентифікації, а також події, пов'язані з адмініструванням облікових записів.

Для підтримки мережевої взаємодії система використовує модулі socket і threading. Socket забезпечує низькорівневу обробку мережевих з'єднань через протоколи TCP/IP, що дозволяє реалізувати серверну частину для прийому й обробки запитів клієнтів. Threading, у свою чергу, відповідає за багатопотокову обробку, дозволяючи обслуговувати кілька клієнтських з'єднань одночасно. Це забезпечує масштабованість системи та її високу продуктивність навіть у разі значного навантаження. Додатково було реалізовано механізм перевірки дозволених IP-адрес із використанням бібліотеки ipaddress, яка дозволяє виконувати точну перевірку адрес клієнтів відповідно до заданих діапазонів.

Валідація даних під час реєстрації користувачів реалізована за допомогою модуля re для роботи з регулярними виразами. Це дозволяє встановлювати чіткі правила для форматування імен користувачів і паролів, що сприяє підвищенню безпеки системи, уникаючи використання занадто простих чи потенційно небезпечних облікових даних. Усі паролі проходять хешування з використанням алгоритму bcrypt, який забезпечує високу стійкість до атак типу rainbow table, завдяки використанню адаптивної функції хешування.

Додатково інтегровано систему автоматичного очищення застарілих блокувань і спроб входу. Цей компонент використовує окремий потік для періодичного сканування словників, які містять інформацію про тимчасово заблоковані IP-адреси та облікові записи. Це дозволяє зберігати актуальність даних і уникати надмірного використання ресурсів сервера. Весь процес організований таким чином, щоб мінімізувати вплив на основні потоки програми й забезпечити безперервність її роботи.

Для організації взаємодії з клієнтами використовується текстовий протокол передачі даних, що дозволяє обмінюватися повідомленнями між клієнтом і сервером у зручному для аналізу форматі. Це спрощує налагодження, тестування і діагностику проблем. У майбутньому така архітектура дає можливість легко перейти на більш складніші протоколи, наприклад, JSON чи XML, без значних змін у кодовій базі.

Резюмуючи, у процесі розробки було використано широкий спектр інструментів і технологій, які забезпечують високу продуктивність, гнучкість і безпеку системи. Об’єднання цих компонентів дозволило створити багаторівневу архітектуру, здатну ефективно виконувати завдання автентифікації, моніторингу та управління користувачами, одночасно забезпечуючи захист даних від сучасних загроз інформаційній безпеці.

3.4 Опис роботи програмного забезпечення

Розроблене програмне забезпечення представляє собою складну інтегровану систему, яка забезпечує виконання широкого спектру завдань, пов’язаних із автентифікацією користувачів, управлінням доступом, збереженням даних і забезпеченням безпеки інформації. Його робота базується на використанні клієнт-серверної архітектури, що дозволяє забезпечити гнучкість, масштабованість і зручність у використанні. Уся система складається з декількох ключових компонентів, які працюють у тісній взаємодії для забезпечення цілісності та безперервності робочого процесу.

На початковому етапі роботи система ініціалізує серверну частину, яка виконує функцію центрального вузла для обробки запитів клієнтів. Після запуску сервер створює слухач мережевих з’єднань на визначеному порту, використовуючи модуль socket. Цей процес включає відкриття TCP-сокета та прив’язку його до заданої IP-адреси та порту, що забезпечує можливість прийому вхідних запитів. Після ініціалізації сервер переходить у режим очікування, у якому він готовий до обслуговування клієнтських підключень. Одночасно з цим система створює окремий потік для управління журналами подій і логуванням. Це дозволяє забезпечити безперервне відстеження стану програми, записуючи інформацію про всі події, включаючи помилки, з’єднання та запити.

Під час підключення клієнта сервер автоматично виділяє окремий потік для обробки цього з’єднання. Такий підхід забезпечує одночасну обробку множинних запитів без впливу на продуктивність системи. Після встановлення з’єднання сервер перевіряє IP-адресу клієнта, використовуючи заздалегідь визначені правила. Ця перевірка виконується для виявлення можливих загроз, таких як спроби несанкціонованого доступу з невідомих або заблокованих IP-адрес. У разі виявлення невідповідності IP-адреса клієнта вноситься до списку заблокованих, і з’єднання припиняється.

Після успішного підключення клієнт повинен пройти автентифікацію. Цей процес починається з введення користувачем облікових даних, які включають ім’я користувача та пароль. Клієнт передає ці дані у зашифрованому вигляді, використовуючи криптографічні алгоритми, такі як AES у режимі CBC, із ключем, який був згенерований за допомогою алгоритму PBKDF2. Зашифровані дані передаються серверу, де вони розшифровуються за допомогою аналогічного ключа. Після розшифрування сервер порівнює отримані облікові дані з даними, що зберігаються у внутрішній базі даних SQLite. Якщо перевірка пройшла успішно, клієнту надається доступ до системи, і сервер відправляє підтвердження про успішну автентифікацію. У разі невдалої спроби вхідних даних сервер реєструє подію у журналі та відправляє клієнту повідомлення про помилку.

Усі паролі зберігаються у вигляді хешів, які генеруються за допомогою алгоритму bcrypt. Цей підхід забезпечує високу стійкість до атак типу brute-force, оскільки bcrypt є адаптивним алгоритмом, який збільшує складність хешування з кожною новою ітерацією. Крім того, кожен хеш доповнюється унікальним сольовим значенням, яке запобігає використанню попередньо обчислених таблиць хешів (rainbow table). Такий підхід гарантує, що навіть у разі компрометації бази даних паролі користувачів залишаться захищеними.

Крім того, у системі передбачено механізм захисту від надмірної кількості невдалих спроб входу. Якщо клієнт кілька разів поспіль вводить неправильні облікові дані, його IP-адреса тимчасово блокується. Ця інформація зберігається у внутрішніх структурах даних, які періодично перевіряються системою на предмет застарілих записів. Після завершення встановленого періоду блокування IP-адреса автоматично видаляється зі списку блокувань, дозволяючи користувачеві повторно спробувати підключитися до системи.

Після успішної автентифікації користувач отримує доступ до основного функціоналу системи, який може включати перегляд даних, виконання

адміністративних завдань, оновлення інформації або зміну налаштувань облікового запису. Усі дії користувача реєструються в системних журналах, що забезпечує можливість проведення аудиту й моніторингу. Сервер перевіряє кожен запит на відповідність правам доступу користувача. Цей механізм дозволяє захистити конфіденційну інформацію від несанкціонованого доступу навіть у випадках, коли користувач має доступ до системи. Ще одним важливим аспектом роботи програмного забезпечення є регулярне обслуговування й оптимізація бази даних. Система автоматично виконує періодичне очищення даних, що більше не використовуються, а також дефрагментацію таблиць, щоб забезпечити максимальну продуктивність і мінімізувати використання ресурсів. Крім того, реалізовано механізм резервного копіювання, який дозволяє зберігати копії важливих даних у захищеному місці. Це забезпечує високу стійкість до втрати інформації в разі технічних збоїв або інших надзвичайних ситуацій.

Під час завершення сесії клієнт надсилає серверу запит на відключення. Сервер коректно закриває з’єднання, звільняючи всі ресурси, які використовувалися для обслуговування даного клієнта. Усі дані про сесію зберігаються в журналах, що дозволяє проводити аналіз використання системи. Ця функція також включає в себе очищення тимчасових даних, які могли створюватися під час сесії, забезпечуючи безпеку та конфіденційність.

Усі аспекти роботи розробленого програмного забезпечення побудовані таким чином, щоб забезпечити максимально ефективну й безпечну взаємодію користувача із системою. Такий підхід дозволяє досягти високого рівня продуктивності, мінімізувати ризики, пов’язані із захистом даних, і гарантувати стабільність роботи навіть за умови значного навантаження на сервер.

3.5 Кроки оптимізації програмного забезпечення

Оптимізація програмного забезпечення є важливим процесом, спрямованим на поліпшення його продуктивності, зниження споживання ресурсів і підвищення загальної ефективності роботи. В умовах сучасних технологічних реалій, коли складність програмних продуктів зростає разом із потребою в обробці великих обсягів даних, питання оптимізації набувають критичного значення. Кожен етап цього процесу передбачає ретельний аналіз, дослідження й внесення змін до існуючого коду та архітектури системи для досягнення поставлених цілей.

Першим важливим аспектом оптимізації є аналіз продуктивності існуючої системи. Для цього застосовуються спеціалізовані інструменти профілювання, які дозволяють виявити вузькі місця в роботі програмного забезпечення. Профілювання надає змогу зібрати дані про використання пам’яті, завантаження процесора, кількість операцій введення-виведення, а також час виконання ключових функцій. На основі отриманих результатів розробники можуть визначити, які компоненти системи вимагають найбільшої уваги. Наприклад, якщо виявляється, що певні функції споживають надмірну кількість ресурсів або виконуються надто повільно, це вказує на потенційні зони для оптимізації.

Важливим напрямом оптимізації є перегляд алгоритмів і структур даних, які використовуються в програмному забезпеченні. Алгоритмічна оптимізація передбачає заміну неефективних алгоритмів на більш продуктивні. Наприклад, замість використання алгоритму сортування з квадратичною складністю можна впровадити більш швидкісний алгоритм, такий як сортування злиттям або швидке сортування. Крім того, важливо аналізувати вибір структур даних, оскільки вони впливають на час доступу до елементів і операції вставки або видалення. Наприклад, для частих операцій пошуку

більш доцільно використовувати хеш-таблиці або дерева, тоді як для обробки послідовностей даних можуть бути ефективні зв’язкові списки чи масиви.

Не менш важливим етапом оптимізації є зниження витрат на введення-виведення, які часто є одним із найбільших джерел уповільнення роботи системи. Для цього можна використовувати буферизацію, що дозволяє мінімізувати кількість звернень до фізичних носіїв інформації. Наприклад, замість того, щоб кожного разу записувати дані у файл, система може зберігати їх у тимчасовій пам’яті, а потім записувати великими блоками. Це знижує навантаження на дискову систему та покращує загальну швидкість роботи. Аналогічно, кешування дозволяє зменшити кількість звернень до баз даних або віддалених серверів, що особливо важливо для систем із високою частотою запитів.

Крім того, можна звернути увагу на оптимізацію запитів до бази даних, якщо такі використовуються у програмному забезпеченні. Розробники мають можливість мінімізувати складність SQL-запитів, зменшити кількість підзапитів, впроваджувати індексацію для прискорення пошуку й сортування даних, а також використовувати попередньо підготовлені запити для зниження навантаження на сервер. У випадках, коли база даних обробляє велику кількість однотипних запитів, можна реалізувати пул з’єднань для уникнення постійного відкриття і закриття з’єднань, що також покращує продуктивність. Ще одним важливим напрямом є оптимізація управління пам’яттю. Погане управління пам’яттю може призвести до витоків пам’яті, збоїв системи й загального зниження її продуктивності. Для цього необхідно ретельно відслідковувати всі створені об’єкти та їхнє видалення після завершення використання. У мовах програмування з ручним управлінням пам’яттю, таких як C++, це може включати явне звільнення пам’яті за допомогою оператора delete. У мовах із автоматичним управлінням пам’яттю, таких як Python, слід уникати створення циклічних посилань між об’єктами, які можуть ускладнити роботу збирача сміття.

Важливу роль у процесі оптимізації відіграє масштабованість системи. Умови експлуатації програмного забезпечення можуть змінюватися з часом, і система має бути готовою до роботи з більшим обсягом даних або збільшеною кількістю користувачів. Для цього доцільно використовувати такі методи, як горизонтальне та вертикальне масштабування серверів, розподіл навантаження за допомогою балансувальників, а також впровадження механізмів обробки даних у реальному часі. Наприклад, якщо сервер обробляє занадто багато запитів, можна додати нові сервери, які працюватимуть паралельно, розподіляючи навантаження між собою. Не менш важливим напрямом оптимізації є покращення користувацького інтерфейсу. Це включає прискорення завантаження сторінок, зменшення затримок у відгуках інтерфейсу та мінімізацію використання ресурсів на стороні клієнта. Наприклад, можна використовувати асинхронні запити для уникнення блокування інтерфейсу під час виконання складних операцій. Оптимізація графічних компонентів, таких як зображення, відео чи інші мультимедійні елементи, також може значно покращити загальне враження від використання програмного забезпечення.

Окрему увагу слід приділити тестуванню програмного забезпечення після внесення змін, оскільки оптимізація може вплинути на стабільність системи. Для цього використовуються автоматизовані та ручні методи тестування, які дозволяють перевірити, чи всі функції програми працюють відповідно до очікувань. Крім того, необхідно проводити стрес-тестування, щоб переконатися у здатності системи витримувати високі навантаження без втрати продуктивності.

Оптимізація програмного забезпечення є безперервним процесом, який потребує постійного аналізу, експериментів і вдосконалення. Використання сучасних інструментів, методів і підходів дозволяє створювати ефективні, швидкі й надійні системи, які відповідають високим вимогам користувачів і забезпечують стійку роботу навіть у складних умовах експлуатації.

3.6 Демонстрація розроблених аспектів безпеки в комп'ютерних мережах

На рисунку 3.1 наведено логіку управління користувачами. Загалом було зареєстровано чотири користувачів з різними іменами та ролями.

```
=== УПРАВЛІННЯ КОРИСТУВАЧАМИ ===

Реєстрація користувача: admin (роль: admin)
2024-12-13 01:17:40,231 - INFO: Зареєстровано нового користувача: admin
Реєстрація успішна

Реєстрація користувача: user1 (роль: user)
2024-12-13 01:17:40,285 - INFO: Зареєстровано нового користувача: user1
Реєстрація успішна

Реєстрація користувача: analyst (роль: analyst)
2024-12-13 01:17:40,342 - INFO: Зареєстровано нового користувача: analyst
Реєстрація успішна

Реєстрація користувача: testuser (роль: user)
2024-12-13 01:17:40,398 - INFO: Зареєстровано нового користувача: testuser
Реєстрація успішна
```

Рисунок 3.1 – Управління користувачами

На рисунку 3.2 наведено тестування аутентифікації. Тобто було здійснено спроби авторизуватися для раніше зареєстрованих користувачів. З рисунка видно, що користувач з іменем admin авторизувався успішно, в той час як user1 та nonexistent – невдало, оскільки до сервера було передано невірні дані: логін та/або пароль.

```
=== ТЕСТУВАННЯ АВТЕНТИФІКАЦІЇ ===

Спроба входу: admin
Автентифікація успішна

Спроба входу: user1
Автентифікація невдала

Спроба входу: nonexistent
Автентифікація невдала
```

Рисунок 3.2 – Тестування аутентифікації

На рисунку 3.3 відображено систему логування та аудиту. З цього логування видно ідентифікатори записів, події, що відбулися, який користувач намагався взаємодіяти, яка IP-адреса, час та деталі: чи була авторизація успішною, чи ні. Якщо ні, то що саме вийшло невдало: невірний пароль чи вказаного користувача не існувало.

```
=== СИСТЕМА ЛОГУВАННЯ ТА АУДИТУ ===
Останні записи аудиту:
ID: 1, Подія: AUTH, Користувач: admin, IP: 192.168.0.101, Час: 2024-12-13 01:17:40.440389, Деталі: Успіх:
ID: 2, Подія: AUTH, Користувач: user1, IP: 192.168.0.101, Час: 2024-12-13 01:17:40.515089, Деталі: Невдача: Невірний пароль
ID: 3, Подія: AUTH, Користувач: nonexistent, IP: 192.168.0.101, Час: 2024-12-13 01:17:40.537023, Деталі: Невдача: Користувача не знайдено
```

Рисунок 3.3 – Система логування та аудиту

На рисунку 3.4 наведено демонстрацію криптографічних операцій. Тут представлено оригінальні повідомлення, що підлягали шифрування, їх зашифровані версії та розшифровані.

```
=== КРИПТОГРАФІЧНІ ОПЕРАЦІЇ ===

Оригінальне повідомлення: Секретне повідомлення 1
Зашифроване повідомлення: b'gAAAAABnW28UfkYbr9bXoSTgQqSHYa1jNL_tSyCGnpr_CCJC3Uox2PHAafd2pn1v-gxwmepamvabziSZopZIHNwHEqahko-FErCqv7jhy5kmfCRZHux3SEpIwqFrrKnhDQod4bQvKMMz'
Розшифроване повідомлення: Секретне повідомлення 1

Оригінальне повідомлення: Важлива конфіденційна інформація
Зашифроване повідомлення: b'gAAAAABnW28UMVFbX7OKugXe3-EKOO0n9Q-R7Uuvt5AvysxzuR-OKId-JKeX4mXG48y4OWcB8Yt47OBc7bdyCVKVsIMEOlO-93qmHrlSQCvW_xfZmphf1QETVSZCYakDo1Cbcjr0W5ER91v35pp9gW4cnS92JJ6d4A=='
Розшифроване повідомлення: Важлива конфіденційна інформація

Оригінальне повідомлення: Системні налаштування мережі
Зашифроване повідомлення: b'gAAAAABnW28ULLInDghoaru-4X1qjmDPR9Y0YBrhbz54_U4FFbz1REVpY-YI1_MZdGf983fKTViGD3W9SB9iK7s8vNLJOb4zrQyNkOlQ0aR_3lmWHHnBGmLktn9VFCqireYIDBxvzDvDbhKqU_cR5CSFJwv90sRp7g=='
Розшифроване повідомлення: Системні налаштування мережі
```

Рисунок 3.4 – Криптографічні операції

На рисунку 3.5 наведено мережеву конфігурацію: з яких IP-адрес доступ можливий, з яких – заборонений. Також наводиться поточна конфігурація системи.

На рисунку 3.6 представлено управління сесіями: які сесії було відкрито, які з них активні, які вже ні. Також вказуються користувачі, які пов’язані з конкретною сесією, та IP-адреси.

На рисунку 3.7 наведено аналіз зареєстрованих користувачів: вказуються їхні імена та ролі.

```
=== МЕРЕЖЕВА КОНФІГУРАЦІЯ ===
IP 192.168.1.100: Заборонено
IP 10.0.0.50: Дозволено
IP 8.8.8.8: Заборонено

Поточна конфігурація системи:
max_login_attempts: 5
block_duration_minutes: 30
allowed_networks: ['192.168.0.0/24', '10.0.0.0/16']
port: 65432
host: 0.0.0.0
```

Рисунок 3.5 – Мережева конфігурація

```
=== УПРАВЛІННЯ СЕСІЯМИ ===
Активні сесії:
Сесія: deacf76a-1cca-42ed-87ca-21f5c8aba28c
  Користувач: admin
  IP: 192.168.1.100
  Створена: 2024-12-13 01:17:40.559462
Сесія: 1c4b00cc-14f9-487f-8ee9-eba5fb4b9d71
  Користувач: user1
  IP: 10.0.0.50
  Створена: 2024-12-13 01:17:40.559462
```

Рисунок 3.6 – Управління сесіями

```
=== АНАЛІЗ РОЛЕЙ КОРИСТУВАЧІВ ===
Зареєстровані користувачі:
Користувач: admin, Роль: admin
Користувач: user1, Роль: user
Користувач: analyst, Роль: analyst
Користувач: testuser, Роль: user
```

Рисунок 3.7 – Аналіз ролей користувачів

На рисунку 3.8 представлено базу даних, яка зберігає дані про користувачів: логіни, паролі, сіль, яка використовується для шифрування та дешифрування, коли було створено акаунт користувача, коли останній раз він був авторизований та яка в нього роль у системі.

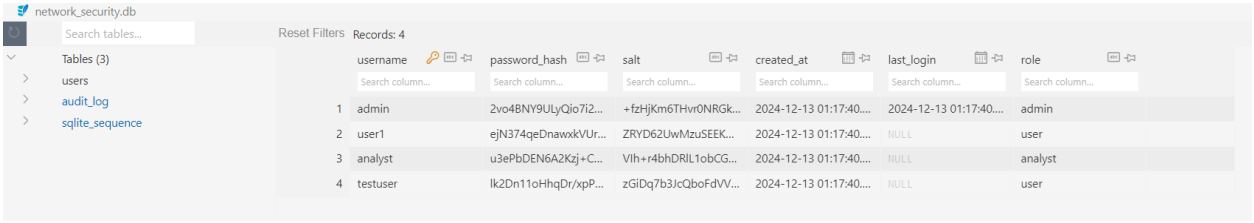

Рисунок 3.8 – База даних для користувачів

security_config.json > ...

```
{
    "max_login_attempts": 5,
    "block_duration_minutes": 30,
    "allowed_networks": [
        "192.168.0.0/24",
        "10.0.0.0/16"
    ],
    "port": 65432,
    "host": "0.0.0.0"
}
```

Рисунок 3.9 – JSON-файл для зберігання налаштувань конфігурації

На рисунку 3.10 наведено логування, яке зберігається у окремому файлі.

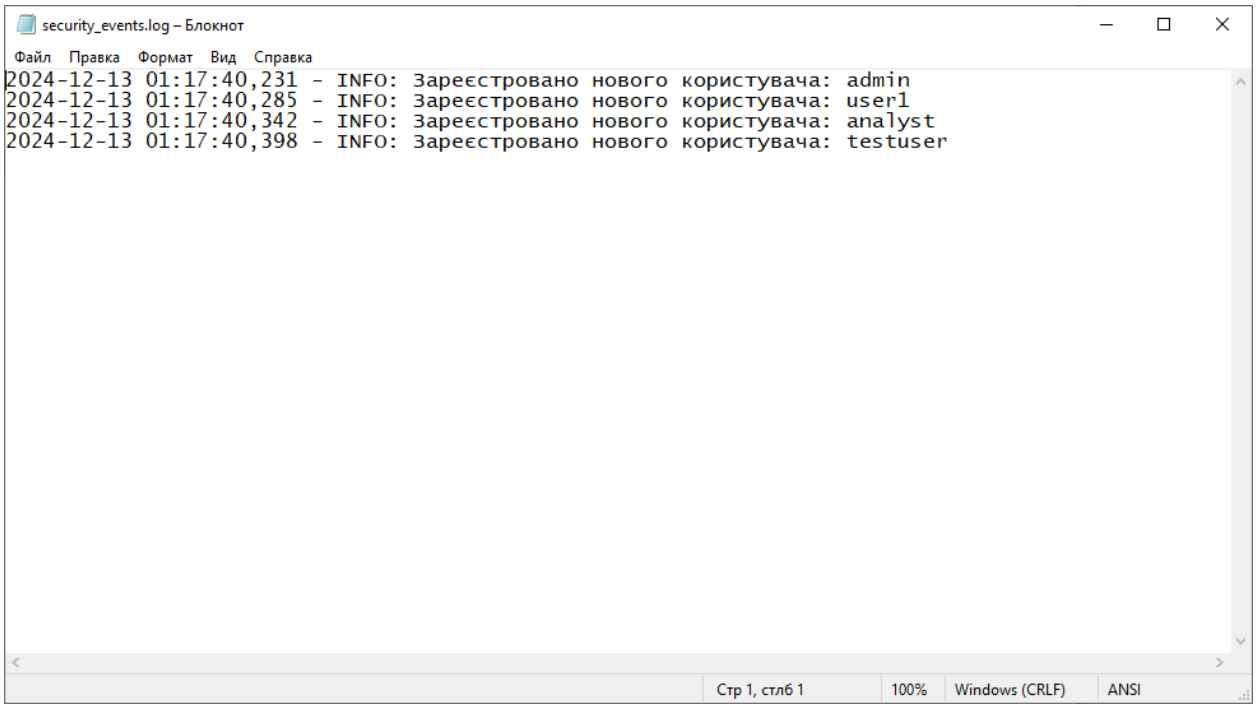

Рисунок 3.10 – Збереження логування в окремому файлі

Усі перелічені компоненти серверної системи (рис. 3.1-3.10) представляють собою комплексний підхід до забезпечення алгоритмів безпеки в комп'ютерних мережах. Це включає шифрування, розшифрування, правильне збереження у базах даних, логування, перевірки для надання чи ненадання доступів тощо.

3.7 Висновки до третього розділу

У третьому розділі детально розглянуто програмну реалізацію розробленого програмного забезпечення, включаючи постановку задачі, архітектурні рішення, вибір технологій, опис функціонування системи та можливі напрями її оптимізації. В результаті проведеного аналізу сформовано комплексне уявлення про структуру, функції та способи покращення роботи програмного продукту.

Постановка задачі дала змогу чітко окреслити мету та основні вимоги до розробки, що стали фундаментом для подальшого проєктування і реалізації. Було визначено ключові функції, які має виконувати програмне забезпечення, і критерії, що забезпечують його ефективність, зручність та масштабованість. Це створило міцну основу для побудови надійної архітектури.

Архітектура програмного забезпечення була розроблена з урахуванням принципів модульності, розширюваності та оптимального використання ресурсів. Вибраний підхід забезпечує гнучкість системи, полегшує її тестування та підтримку, а також сприяє інтеграції нових функціональних можливостей у майбутньому. Особливу увагу було приділено розмежуванню клієнтської і серверної частини, що сприяє ефективному розподілу обчислювальних навантажень. Опис використаних технологій підтвердив обґрунтованість їх вибору відповідно до поставлених задач. Кожна технологія або інструмент забезпечує виконання конкретних функцій системи, зокрема, взаємодію між компонентами, зберігання й обробку даних, а також забезпечення стабільної роботи користувацького інтерфейсу. Використання сучасних технологічних рішень дозволило досягти високого рівня продуктивності та надійності. У процесі опису роботи програмного забезпечення було висвітлено ключові етапи його функціонування, включаючи обробку запитів користувачів, обмін даними між клієнтом і сервером, а також внутрішню обробку даних. Це дозволяє краще зрозуміти механізми роботи системи і забезпечує базу для її подальшого вдосконалення.

Нарешті, розглянуто можливі кроки для оптимізації програмного забезпечення, які спрямовані на підвищення його продуктивності, зниження витрат ресурсів та покращення масштабованості. Запропоновані рішення стосуються оптимізації алгоритмів, структур даних, управління пам’яттю, запитів до бази даних і загальної архітектури системи. Особливу увагу було приділено технологіям кешування, буферизації та балансування навантажень, які дозволяють зменшити час відгуку та покращити досвід користувачів.

ВИСНОВКИ

У кваліфікаційній роботі було проведено ґрунтовне дослідження сучасних алгоритмів безпеки, які застосовуються для захисту інформації в комп’ютерних мережах. Дослідження охопило аналіз існуючих загроз, методів захисту та практичних аспектів застосування криптографії, протоколів аутентифікації, багатофакторних систем доступу і технологій захисту даних на різних рівнях мережевої архітектури. Це дозволило комплексно оцінити поточний стан і проблеми забезпечення інформаційної безпеки в умовах швидкого розвитку інформаційних технологій та зростаючих кіберзагроз.

Робота підкреслює важливість інтегрованого підходу до забезпечення безпеки, який поєднує технічні, організаційні та нормативно-правові заходи. Встановлено, що сучасні виклики у сфері безпеки, зокрема зростання масштабів кіберзлочинності, розвиток нових типів атак і поява складних багатоступеневих загроз, вимагають використання комплексних рішень, які включають в себе криптографічні алгоритми, технології шифрування, хешування, цифрові підписи, а також протоколи аутентифікації та авторизації.

У процесі роботи проведено аналіз ефективності таких криптографічних алгоритмів, як AES, RSA і ECC. Встановлено, що алгоритм AES демонструє високу швидкодію і надійність у завданнях симетричного шифрування, тоді як RSA і ECC є основними інструментами для забезпечення конфіденційності та автентичності при передачі даних. Алгоритми ECC, які базуються на використанні еліптичних кривих, показали значну перевагу завдяки своїй ефективності при менших розмірах ключів, що робить їх особливо перспективними для мобільних і вбудованих систем.

Досліджено роль багатофакторної аутентифікації, яка значно підвищує рівень захищеності системи, комбінуючи кілька різних факторів ідентифікації користувача, таких як паролі, біометричні дані чи фізичні токени. Окрему увагу приділено перспективним методам поведінкової та контекстуальної

аутентифікації, які дозволяють динамічно виявляти загрози та автоматично адаптувати рівень безпеки системи.

Практична цінність роботи полягає у розроблених рекомендаціях щодо вибору алгоритмів і технологій захисту залежно від конкретних умов застосування, зокрема для захисту даних у хмарних системах, корпоративних мережах, мобільних платформах і фінансових транзакціях. Ці рекомендації спрямовані на підвищення стійкості інформаційних систем до сучасних кіберзагроз, таких як атаки «людина посередині», DDoS-атаки, підробка даних та інші.

Наукова новизна дослідження полягає в систематизації сучасних алгоритмів безпеки, аналізі їх переваг і недоліків, а також у розробці підходів до їх вдосконалення. Запропоновано методи комбінування криптографічних рішень із системами виявлення загроз, що дозволяє знизити ризик складних атак і зберегти конфіденційність, цілісність та доступність даних.

Отримані результати роблять значний внесок у сферу інформаційної безпеки, дозволяючи удосконалити існуючі системи захисту, підвищити їх ефективність і масштабованість. Висновки та рекомендації можуть бути використані як основа для розробки нових методів забезпечення безпеки в умовах швидкого розвитку технологій і нових викликів у галузі кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H. Wang, “Application of Collaborative Filtering Algorithm in Computer Network Security Technology Evaluation”, in *2023 Int. Conf. Data Sci. Netw. Secur. (ICDSNS)*, Tiptur, India, Jul. 28–29, 2023. IEEE, 2023. https://doi.org/10.1109/icdsns58469.2023.10245824

2. Y. Han, “Information Security Protection Method for Computer Networks Considering Big Data Clustering Algorithm”, in *2023 Int. Conf. Comput. Simul. Model., Inf. Secur. (CSMIS)*, Buenos Aires, Argentina, Nov. 15–17, 2023. IEEE, 2023. https://doi.org/10.1109/csmis60634.2023.00092

3. D. Jingbo, “Research on Computer Network Information Security Protection Strategy and Evaluation Algorithm Based on Fuzzy Clustering”, in *2021 IEEE 4th Int. Conf. Inf. Syst. Comput. Aided Educ. (ICISCAE)*, Dalian, China, Sep. 24–26, 2021. IEEE, 2021. https://doi.org/10.1109/iciscae52414.2021.9590686

4. H. Bai, Y. Weng, Q. Zhao, and J. Guan, “Multidimensional Detection and Evaluation System of Computer Network Security Based on Machine Learning Algorithm”, in *2022 IEEE 2nd Int. Conf. Mobile Netw. Wireless Commun. (ICMNWC)*, Tumkur, Karnataka, India, Dec. 2–3, 2022. IEEE, 2022. https://doi.org/10.1109/icmnwc56175.2022.10031286

5. L. Xia, Y. Zheng, and J. Gu, “BP Neural Network Algorithm for Computer Network Security Evaluation”, in *2023 2nd Int. Conf. Innov. Technol. (INOCON)*, Bangalore, India, Mar. 3–5, 2023. IEEE, 2023. https://doi.org/10.1109/inocon57975.2023.10101368

6. H. Yang, “Computer Network Information Security Threat Identification Technology Based on Big Data Clustering Algorithm”, in *2022 IEEE 2nd Int. Conf. Mobile Netw. Wireless Commun. (ICMNWC)*, Tumkur, Karnataka, India, Dec. 2–3, 2022. IEEE, 2022. https://doi.org/10.1109/icmnwc56175.2022.10032035

7. B. Guo, W. Zhao, P. Chen, C. Zhang, Z. Sun, and W. Feng, “Research on the Application Risk of Computer Network Security Technology”, in *2023 Int. Conf. Netw., Inform. Comput. (ICNETIC)*, Palermo, Italy, May 29–31, 2023. IEEE, 2023. https://doi.org/10.1109/icnetic59568.2023.00013

8. J. Xu, “Computer Network Information Security Protection Based on Big Data Clustering Algorithm”, in *2023 Int. Conf. Netw., Inform. Comput. (ICNETIC)*, Palermo, Italy, May 29–31, 2023. IEEE, 2023. https://doi.org/10.1109/icnetic59568.2023.00096

9. X. Luo, “Application of neural network in computer network security evaluation”, in *2023 Int. Conf. Netw., Inform. Comput. (ICNETIC)*, Palermo, Italy, May 29–31, 2023. IEEE, 2023. https://doi.org/10.1109/icnetic59568.2023.00036

10. Y. Zhang and Y. Li, “The Security and Protection of Computer Network Information in the Era of Big Data”, in *2023 Int. Conf. Mobile Internet, Cloud Comput. Inf. Secur. (MICCIS)*, Nanjing, China, Apr. 7–9, 2023. IEEE, 2023. https://doi.org/10.1109/miccis58901.2023.00032

11. L. Zhang, Y. Fang, J. Xu, and Y. Shen, “Computer Network Security Strategy Based on Data Encryption Technology”, in *2023 IEEE 15th Int. Conf. Comput. Intell. Communication Netw. (CICN)*, Bangkok, Thailand, Dec. 22–23, 2023. IEEE, 2023. https://doi.org/10.1109/cicn59264.2023.10402235

12. J. Yin, B. He, and J. Zhang, “Computer Security Algorithm Based on Convolutional Neural Network”, in *2023 2nd Int. Conf. Big Data, Inf. Comput. Netw. (BDICN)*, Xishuangbanna, China, Jan. 6–8, 2023. IEEE, 2023. https://doi.org/10.1109/bdicn58493.2023.00062

13. K. Mei, “Computer network security countermeasures based on big data”, in *2021 IEEE Int. Conf. Power Electron., Comput. Appl. (ICPECA)*, Shenyang, China, Jan. 22–24, 2021. IEEE, 2021. https://doi.org/10.1109/icpeca51329.2021.9362621

14. S. Wang, “Intelligent Algorithm in Computer Network Security”, in *2022 Int. Conf. Knowl. Eng. Communication Syst. (ICKECS)*, Chickballapur,

India, Dec. 28–29, 2022. IEEE, 2022. https://doi.org/10.1109/ickecs56523.2022.10059831

15. Y. Guo, J. Xu, H. Yuan, Y. Zhuang, G. Zhu, and Y. Zhang, “Research on Enterprise Computer Network Security Protection Technology Based on Information Technology”, in *2020 IEEE 3rd Int. Conf. Automat., Electron. Elect. Eng. (AUTEEE)*, Shenyang, China, Nov. 20–22, 2020. IEEE, 2020. https://doi.org/10.1109/auteee50969.2020.9315704

16. J. Liu, “Design of Computer Network Security Management System based on Neural Network Technology”, in *2024 Int. Conf. Integr. Circuits Communication Syst. (ICICACS)*, Raichur, India, Feb. 23–24, 2024. IEEE, 2024. https://doi.org/10.1109/icicacs60521.2024.10498613

ДОДАТКИ

Додаток А

Лістинг програмного коду

```
import socket
import threading
import hashlib
import logging
import json
import time
import os
import uuid
from datetime import datetime, timedelta
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
import base64
import sqlite3
import ipaddress
import re

class AdvancedNetworkSecuritySystem:
    def __init__(self, config_path='security_config.json'):
        # Конфігурація системи
        self.load_configuration(config_path)

        # Налаштування логування
        self.setup_logging()

        # Ініціалізація бази даних
        self.init_database()

        # Система моніторингу та захисту
        self.connection_attempts = {}
        self.blocked_ips = set()
        self.active_sessions = {}

        # Криптографічні механізми
        self.generate_master_key()
```

```
def load_configuration(self, config_path):
    """Завантаження конфігурації системи"""
    try:
        with open(config_path, 'r') as f:
            self.config = json.load(f)
    except FileNotFoundError:
        self.config = {
            'max_login_attempts': 5,
            'block_duration_minutes': 30,
            'allowed_networks': ['192.168.0.0/24', '10.0.0.0/16'],
            'port': 65432,
            'host': '0.0.0.0'
        }
        with open(config_path, 'w') as f:
            json.dump(self.config, f, indent=4)

def setup_logging(self):
    """Налаштування системи логування"""
    log_dir = 'security_logs'
    os.makedirs(log_dir, exist_ok=True)

    logging.basicConfig(
        level=logging.INFO,
        format='%(asctime)s - %(levelname)s: %(message)s',
        handlers=[
            logging.FileHandler(f'{log_dir}/security_events.log'),
            logging.StreamHandler()
        ]
    )
    self.logger = logging.getLogger(__name__)

def init_database(self):
    """Ініціалізація бази даних користувачів та аудиту"""
    self.conn = sqlite3.connect('network_security.db', check_same_thread=False)
    self.cursor = self.conn.cursor()

    # Таблиця користувачів
    self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            username TEXT PRIMARY KEY,
            password_hash TEXT,
            salt TEXT,
            created_at DATETIME,
```

```
            last_login DATETIME,
            role TEXT DEFAULT 'user'
        )
    ''')

    # Таблиця аудиту
    self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS audit_log (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            event_type TEXT,
            username TEXT,
            ip_address TEXT,
            timestamp DATETIME,
            details TEXT
        )
    ''')

    self.conn.commit()

def generate_master_key(self):
    """Генерація майстер-ключа для криптографічних операцій"""
    password = os.urandom(32)
    salt = os.urandom(16)
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000
    )
    self.master_key = base64.urlsafe_b64encode(kdf.derive(password))
    self.cipher_suite = Fernet(self.master_key)

def is_ip_allowed(self, ip):
    """Перевірка дозволених мережевих діапазонів"""
    for network in self.config['allowed_networks']:
        if ipaddress.ip_address(ip) in ipaddress.ip_network(network, strict=False):
            return True
    return False

def register_user(self, username, password, role='user'):
    """Реєстрація нового користувача"""
    if not self.validate_username(username):
        self.logger.warning(f"Невалідне ім'я користувача: {username}")
```

```
            return False

        salt = os.urandom(16)
        password_hash = hashlib.pbkdf2_hmac('sha256', password.encode(), salt,
100000)

        try:
            self.cursor.execute('''
                INSERT INTO users
                (username, password_hash, salt, created_at, role)
                VALUES (?, ?, ?, ?, ?)
            ''', (
                username,
                base64.b64encode(password_hash).decode(),
                base64.b64encode(salt).decode(),
                datetime.now(),
                role
            ))
            self.conn.commit()
            self.logger.info(f"Зареєстровано нового користувача: {username}")
            return True
        except sqlite3.IntegrityError:
            self.logger.warning(f"Користувач {username} вже існує")
            return False

    def validate_username(self, username):
        """Валідація імені користувача"""
        return re.match(r'^[a-zA-Z0-9_]{3,20}$', username) is not None

    def authenticate_user(self, username, password):
        """Розширена автентифікація користувача"""
        # Перевірка кількості спроб входу
        if username in self.connection_attempts and \
           self.connection_attempts[username]['attempts'] >=
self.config['max_login_attempts']:
            block_time = self.connection_attempts[username].get('block_until',
datetime.now())
            if datetime.now() < block_time:
                self.logger.warning(f"Обліковий запис {username} заблоковано")
                return False

        # Перевірка облікових даних
```

```
        self.cursor.execute('SELECT password_hash, salt FROM users WHERE
username = ?', (username,))
        result = self.cursor.fetchone()

        if not result:
            self.log_authentication_attempt(username, False, "Користувача не
знайдено")
            return False

        stored_password_hash, stored_salt = result
        stored_password_hash = base64.b64decode(stored_password_hash)
        stored_salt = base64.b64decode(stored_salt)

        # Перевірка хешу
        password_hash = hashlib.pbkdf2_hmac('sha256', password.encode(),
stored_salt, 100000)

        if password_hash == stored_password_hash:
            # Успішна автентифікація
            self.log_authentication_attempt(username, True)
            self.update_last_login(username)
            return True
        else:
            # Невдала спроба входу
            self.log_authentication_attempt(username, False, "Невірний пароль")
            self.track_login_attempts(username)
            return False

    def log_authentication_attempt(self, username, success, details=""):
        """Логування спроб автентифікації"""
        self.cursor.execute('''
            INSERT INTO audit_log
            (event_type, username, ip_address, timestamp, details)
            VALUES (?, ?, ?, ?, ?)
        ''', (
            'AUTH',
            username,
            socket.gethostbyname(socket.gethostname()),
            datetime.now(),
            f"{'Успіх' if success else 'Невдача'}: {details}"
        ))
        self.conn.commit()
```

```
    def track_login_attempts(self, username):
        """Відстеження спроб входу"""
        if username not in self.connection_attempts:
            self.connection_attempts[username] = {'attempts': 1}
        else:
            self.connection_attempts[username]['attempts'] += 1

        if self.connection_attempts[username]['attempts'] >=
self.config['max_login_attempts']:
            block_duration = timedelta(minutes=self.config['block_duration_minutes'])
            self.connection_attempts[username]['block_until'] = datetime.now() +
block_duration
            self.logger.warning(f"Обліковий запис {username} заблоковано на
{block_duration}")

    def update_last_login(self, username):
        """Оновлення часу останнього входу"""
        self.cursor.execute(
            'UPDATE users SET last_login = ? WHERE username = ?',
            (datetime.now(), username)
        )
        self.conn.commit()

    def encrypt_message(self, message):
        """Шифрування повідомлення"""
        return self.cipher_suite.encrypt(message.encode())

    def decrypt_message(self, encrypted_message):
        """Розшифрування повідомлення"""
        return self.cipher_suite.decrypt(encrypted_message).decode()

    def network_monitor(self):
        """Постійний моніторинг мережевої активності"""
        while True:
            # Очищення застарілих блокувань та спроб входу
            current_time = datetime.now()
            self.connection_attempts = {
                user: data for user, data in self.connection_attempts.items()
                if data.get('block_until', current_time) > current_time
            }

            # Додаткова логіка моніторингу
            time.sleep(300)  # Перевірка кожні 5 хвилин
```

```
    def start_server(self):
        """Запуск захищеного сервера"""
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((self.config['host'], self.config['port']))
        server_socket.listen(5)

        self.logger.info(f"[*] Сервер запущено на
{self.config['host']}:{self.config['port']}")

        # Запуск моніторингу
        monitor_thread = threading.Thread(target=self.network_monitor)
        monitor_thread.daemon = True
        monitor_thread.start()

        while True:
            client_socket, addr = server_socket.accept()
            client_thread = threading.Thread(
                target=self.handle_client,
                args=(client_socket, addr)
            )
            client_thread.start()

    def handle_client(self, client_socket, addr):
        """Обробка клієнтських підключень"""
        ip = addr[0]

        # Перевірка IP
        if not self.is_ip_allowed(ip):
            self.logger.warning(f"Заборонене підключення з IP: {ip}")
            client_socket.close()
            return

        try:
            # Автентифікація
            client_socket.send("Username: ".encode())
            username = client_socket.recv(1024).decode().strip()
            client_socket.send("Password: ".encode())
            password = client_socket.recv(1024).decode().strip()

            if self.authenticate_user(username, password):
                # Створення унікальної сесії
                session_id = str(uuid.uuid4())
```

```
            self.active_sessions[session_id] = {
                'username': username,
                'ip': ip,
                'created_at': datetime.now()
            }

            # Захищена відповідь
            encrypted_message = self.encrypt_message(
                f"Успішна автентифікація! Ваш ідентифікатор сесії: {session_id}"
            )
            client_socket.send(encrypted_message)
        else:
            client_socket.send("Невірна автентифікація!".encode())

        client_socket.close()

    except Exception as e:
        self.logger.error(f"Помилка обробки клієнта: {e}")
        client_socket.close()

def main():
    # Створення системи безпеки
    security_system = AdvancedNetworkSecuritySystem()

    def demonstrate_user_management():
        """Демонстрація управління користувачами"""
        print("\n=== УПРАВЛІННЯ КОРИСТУВАЧАМИ ===")

        # Реєстрація нових користувачів
        test_users = [
            ('admin', 'superSecurePassword123!', 'admin'),
            ('user1', 'userPassword456@', 'user'),
            ('analyst', 'securityAnalyst789#', 'analyst'),
            ('testuser', 'weakpassword', 'user')  # Демонстрація слабкого паролю
        ]

        for username, password, role in test_users:
            print(f"\nРеєстрація користувача: {username} (роль: {role})")
            success = security_system.register_user(username, password, role)
            print(f"Реєстрація {'успішна' if success else 'невдала'}")

        # Тестування автентифікації
        print("\n=== ТЕСТУВАННЯ АВТЕНТИФІКАЦІЇ ===")
```

```
    auth_attempts = [
        ('admin', 'superSecurePassword123!'),  # Вірний пароль
        ('user1', 'wrongpassword'),  # Невірний пароль
        ('nonexistent', 'somepassword')  # Неіснуючий користувач
    ]

    for username, password in auth_attempts:
        print(f"\nСпроба входу: {username}")
        result = security_system.authenticate_user(username, password)
        print(f"Автентифікація {'успішна' if result else 'невдала'}")

def demonstrate_logging_and_auditing():
    """Демонстрація системи логування та аудиту"""
    print("\n=== СИСТЕМА ЛОГУВАННЯ ТА АУДИТУ ===")

    # Перевірка логів автентифікації
    security_system.cursor.execute('SELECT * FROM audit_log')
    audit_logs = security_system.cursor.fetchall()

    print("Останні записи аудиту:")
    for log in audit_logs[-5:]:# Показати останні 5 записів
        print(f"ID: {log[0]}, Подія: {log[1]}, Користувач: {log[2]}, "
              f"IP: {log[3]}, Час: {log[4]}, Деталі: {log[5]}")

def demonstrate_encryption():
    """Демонстрація криптографічних можливостей"""
    print("\n=== КРИПТОГРАФІЧНІ ОПЕРАЦІЇ ===")

    # Тестування шифрування/дешифрування
    test_messages = [
        "Секретне повідомлення 1",
        "Важлива конфіденційна інформація",
        "Системні налаштування мережі"
    ]

    for message in test_messages:
        print(f"\nОригінальне повідомлення: {message}")
        encrypted = security_system.encrypt_message(message)
        print(f"Зашифроване повідомлення: {encrypted}")
        decrypted = security_system.decrypt_message(encrypted)
        print(f"Розшифроване повідомлення: {decrypted}")

def demonstrate_network_configuration():
```

```
    """Демонстрація мережевої конфігурації"""
    print("\n=== МЕРЕЖЕВА КОНФІГУРАЦІЯ ===")

    # Перевірка дозволених мереж
    test_ips = [
        '192.168.1.100',   # Дозволена мережа
        '10.0.0.50',       # Дозволена мережа
        '8.8.8.8'          # Заборонена мережа
    ]

    for ip in test_ips:
        allowed = security_system.is_ip_allowed(ip)
        print(f"IP {ip}: {'Дозволено' if allowed else 'Заборонено'}")

    # Виведення поточної конфігурації
    print("\nПоточна конфігурація системи:")
    for key, value in security_system.config.items():
        print(f"{key}: {value}")

def demonstrate_session_management():
    """Демонстрація управління сесіями"""
    print("\n=== УПРАВЛІННЯ СЕСІЯМИ ===")

    # Створення тестових сесій
    test_sessions = [
        ('admin', '192.168.1.100'),
        ('user1', '10.0.0.50')
    ]

    for username, ip in test_sessions:
        session_id = str(uuid.uuid4())
        security_system.active_sessions[session_id] = {
            'username': username,
            'ip': ip,
            'created_at': datetime.now()
        }

    # Виведення активних сесій
    print("Активні сесії:")
    for session_id, session_data in security_system.active_sessions.items():
        print(f"Сесія: {session_id}")
        print(f"  Користувач: {session_data['username']}")
        print(f"  IP: {session_data['ip']}")
```

```
            print(f"  Створена: {session_data['created_at']}")

    def demonstrate_user_roles():
        """Демонстрація ролей користувачів"""
        print("\n=== АНАЛІЗ РОЛЕЙ КОРИСТУВАЧІВ ===")

        security_system.cursor.execute('SELECT username, role FROM users')
        users = security_system.cursor.fetchall()

        print("Зареєстровані користувачі:")
        for username, role in users:
            print(f"Користувач: {username}, Роль: {role}")

    # Послідовний запуск демонстраційних функцій
    demonstrate_user_management()
    demonstrate_logging_and_auditing()
    demonstrate_encryption()
    demonstrate_network_configuration()
    demonstrate_session_management()
    demonstrate_user_roles()

    print("\n=== ДЕМОНСТРАЦІЮ ЗАВЕРШЕНО ===")

    # Запуск серверу в окремому потоці
    # server_thread = threading.Thread(target=security_system.start_server)
    # server_thread.start()

if __name__ == "__main__":
    main()
```