Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Система керування ключами для забезпечення безпеки

(тема)

Виконав:
здобувач 4 року навчання,
групи
Ігор НОВІКОВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. В'ячеслав РАДЧЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

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

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Новікову Ігору Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система керування ключами для забезпечення безпеки

затверджена наказом по університету від "1" травня 2023 р. № 104 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 2023 р.

3. Вхідні дані до роботи 1) моделі та методи шифрування та дешифрування інформації; 2) сучасні вимоги до систем керування ключами; 3) перелік використаних програмних та апаратних засобів: macOS Monterey, Visual Studio Code

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз сучасних систем керування ключами
2) розглянути види шифрування
3) виявити переваги та недоліки сучасних систем керування ключами
4) проведення загального огляду всіх структур шифрування
5) розробити план для створення власної системи керування ключами для забезпечення безпеки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз сучасних систем керування ключами	05.02.23 – 16.02.23	
2	Огляд технології шифрування та	17.02.23 – 01.03.23	
	дешифрування		
3	Реалізація шифрування	02.03.23 – 13.03.23	
4	Збір додаткової інформації	14.03.23 – 07.06.23	
5	Оформлення пояснювальної записки	08.06.23 – 15.06.23	

Дата видачі завдання 1 травня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Радченко В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 58 с., 7 рис., 1 дод., 10 джерел.

СИСТЕМИ КЕРУВАННЯ, ІНТЕРНЕТ, КЛЮЧ, ПРОТОКОЛ, ШИФРУВАННЯ, ДЕШИФРУВАННЯ, ЗАХИСТ.

Метою кваліфікаційної роботи розробка та реалізація прототипу системи керування криптографічними ключами для забезпечення інформаційної безпеки, використовуючи мову програмування Python у середовищі Google Colab.

У ході виконання кваліфікаційної роботи опрацьовано та оцінено сучасні криптографічні бібліотеки Python, серед яких було обрано найдоцільніші для практичного використання в рамках розробленої системи. На основі аналізу сформовано архітектуру, що охоплює модулі генерації ключів, їх безпечного зберігання, шифрування і дешифрування даних, логування операцій та візуалізації процесів. Реалізована система забезпечує основні функції криптографічного захисту, включаючи створення як симетричних, так і асиметричних ключів, здійснення криптографічних операцій над текстовими повідомленнями, а також ведення журналу подій для забезпечення аудиту та контролю доступу.

Окрему увагу було приділено питанням зручності та наочності використання системи. З цією метою до реалізації було інтегровано інтерактивні елементи, що дозволяють виконувати шифрування безпосередньо з інтерфейсу Colab, а також візуалізацію структури системи у вигляді графічної діаграми.

ABSTRACT

Bachelor’s thesis: 58 pages, 7 figures, 1 appendices, 10 sources.

MANAGEMENT SYSTEMS, INTERNET, KEY, PROTOCOL, ENCRYPTION, DECRYPTION, SECURITY.

The major goal of this thesis is the development and implementation of a prototype key management system to ensure information security, using the Python programming language within the Google Colab environment.

In order to various modern Python cryptographic libraries were reviewed and assessed, with the most appropriate ones selected for practical implementation in the developed system. Based on the conducted analysis, a system architecture was designed, incorporating modules for key generation, secure storage, data encryption and decryption, operational logging, and process visualization. The implemented system supports core cryptographic functionalities, including the creation of both symmetric and asymmetric keys, execution of cryptographic operations on textual data, and maintenance of an event log to enable auditing and access control.

Special attention was given to the system’s usability and visual clarity. To enhance user interaction, interactive components were integrated, enabling encryption directly through the Colab interface, as well as a graphical representation of the system's structure for intuitive understanding.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЕЦП – електронний цифровий підпис

ЄСКД – єдина система конструкторської документації

СКК – система керування ключами

СКП – стандарт з криптографічних послуг

3DES – Потрійний стандарт шифрування даних (англ., Triple Data Encryption Standard)

AES – Розширений стандарт шифрування (англ., Advanced Encryption Standard)

CSPRNG – криптографічно стійкий генератор псевдовипадкових чисел (англ., Cryptographically secure pseudorandom number generator)

DES – стандарт шифрування даних (англ., Data Encryption Standard)

ECC –е криптографія (англ., Elliptic Curve Cryptography)

FIPS – федеральні стандарти обробки інформації (англ., Federal Information Processing Standards)

GDPR – General Data Protection Regulation

HSM – апаратний модуль безпеки

IDEA – міжнародний алгоритм шифрування даних

IoT – Інтернет речей (англ., Internet of Things)

IPSec – IP Security

KMIP – протокол спільним керуванням ключами (англ., Key Management Interoperability Protocol)

KaaS – управління ключами як системою (англ., Key Management as a Service)

MIME – багатоцільові розширення інтернет-пошти (англ., Multipurpose Internet Mail Extensions)

NIST – Національний інститут стандартів і технологій (англ., National Institute of Standards and Technology)

OpenPGP – відкрита досить добра конфіденційність (англ., Open Pretty Good Privacy)

PKCS – стандарти криптографії з відкритим ключем (англ., Public–Key Cryptography Standards)

RSA – абревіатура від фамилій Rivest, Shamir і Adleman

SSL – рівень захищених сокетів (англ., Secure Sockets Layer)

TLS – протокол захисту транспортного рівня (англ., Transport Layer Security)

ВСТУП

Практично кожна сфера діяльності, включаючи бізнес, науку, медицину та громадську діяльність, використовує інформаційні технології, які містять важливу конфіденційну інформацію. Оскільки ці дані є важливими та цінними, вони можуть бути предметом крадіжки або зламу без належного захисту. Тому безпека даних є надзвичайно важливою темою в наш час.

Система керування ключами є однією з основних компонентів безпеки даних, що дозволяє зберігати, керувати та захищати ключі, які використовуються для захисту даних. СКК може використовуватися в багатьох галузях, включаючи фінанси, медицину, військову та наукову сфери.

Одним з викликів при створенні системи керування ключами є забезпечення високої безпеки без затримки в роботі системи та складнощів у використанні для користувачів. Для досягнення цього використовуються сучасні методи шифрування та дешифрування даних, а також механізми контролю доступу до ключів і їх захисту.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Система керування ключами

Система керування ключами - це набір процедур та протоколів, які забезпечують безпеку інформаційних обмінів шляхом створення, збереження, передачі та використання ключів шифрування. В сучасному світі, де обмін інформацією є ключовим процесом для бізнесу та приватного життя, СКК стає все більш важливою.

Існує безліч систем керування ключами, які використовуються у різних сферах. Наприклад, системи керування ключами в банківському секторі, системи керування ключами в телекомунікаційній індустрії, системи керування ключами в урядових структурах та багато інших. Кожна з цих систем має свої особливості та вимоги щодо забезпечення безпеки обміну інформацією. Для огляду існуючих систем керування ключами необхідно проаналізувати їхні особливості та переваги, визначити їх вимоги до забезпечення безпеки та дослідити протоколи, які вони використовують для збереження та передачі ключів. Також важливо розглянути атаки на системи керування ключами та заходи, які можуть бути вжиті для захисту від таких атак.

Один з найпоширеніших методів керування ключами є публічний ключ. У цьому методі, кожен користувач має пару ключів: приватний та публічний. Публічний ключ розповсюджується відкрито, тоді як приватний ключ залишається тільки власнику. Для того, щоб забезпечити безпеку системи криптографічного захисту інформації, ключі повинні бути збережені в надійному і захищеному від доступу місці. У той же час, ключі повинні бути доступні тільки користувачам, які мають дозвіл на їх використання.

Для розв'язання цієї проблеми використовуються системи керування ключами. СКК - це інструмент для генерації, збереження, відновлення,

видачі та скасування криптографічних ключів. Їх використовують для забезпечення безпеки великої кількості даних та ресурсів в інформаційних системах, де важливо зберегти конфіденційність та інтегрітет.

СКК включає в себе методи та протоколи для забезпечення безпеки ключів під час їх транспортування, зберігання та використання. Для цього можуть використовуватися такі методи, як шифрування, цифровий підпис, аутентифікація та ідентифікація. Також це може включати перевірку дійсності ключів, контроль їх використання, відслідковування неправомірного використання Окрім того, СКК забезпечує можливість контролю доступу до ключів та моніторинг їх використання.

Одним з найбільш важливих аспектів СКК є зберігання ключів. Для забезпечення надійного зберігання ключів можуть використовуватися такі технології, як захищена обчислювальна апаратура, розділена база даних, резервне копіювання та відновлення ключів.

Першим етапом управління ключами є створення ключа. Цей етап включає в себе вибір відповідної криптографічної системи та ключового протоколу, що залежить від застосування криптографічного захисту.

Наступним етапом є збереження ключа. Забезпечення безпеки ключів - це критичний аспект управління ключами. Для забезпечення безпеки ключів використовуються різні техніки шифрування та захисту ключів від несанкціонованого доступу. Потім ключ розподіляється користувачам або системам, які його потребують.

Нарешті, ключ використовується для шифрування та розшифрування даних. Ключ може бути використаний один раз або застосовуватися для шифрування багатьох повідомленнь.

Управління ключами є надзвичайно важливим аспектом криптографії, оскільки компрометація криптографічного ключа може призвести до розкриття всієї захищеної інформації. Крім того, управління ключами може вплинути на продуктивність та доступність системи, що використовує криптографічний захист

1.2 Історія шифрування

Історія шифрування відома з давніх часів. Люди використовували різні методи шифрування, щоб зберегти свої таємниці від інших. Одним з найвідоміших методів шифрування є шифрування Цезаря, яке було використано Юлієм Цезарем в Давньому Римі. Цей метод полягав у заміні кожної букви відкритого тексту на букву, яка знаходилася за кілька позицій в алфавіті. Наприклад, букву "А" можна було замінити на букву "Д", букву "Б" - на букву "Е" і т. д. Цей метод був досить простим і легко зламувався, тому з часом люди розробили більш складні методи шифрування. Протягом історії було розроблено багато різних методів шифрування, включаючи традиційні методи, такі як симетричне та асиметричне шифрування. Однак, з появою комп'ютерів і Інтернету з'явилася потреба в більш складних і безпечних системах керування ключами.

СКК були введені в 1970-х роках, коли американська корпорація IBM розробила систему керування ключами, яка дозволяла зберігати ключі шифрування в безпечному місці. Пізніше, у 1980-х роках, було створено стандарт з криптографічних послуг (СКП), який включав у себе протоколи шифрування, цифрові підписи та системи керування ключами. У 1990-х роках СКК стали більш поширеними і були використані в багатьох проектах і стандартах, таких як IPSec, SSL / TLS і S / MIME. Проте, з поширенням інтернету та збільшенням кількості користувачів, проблема управління ключами стала ще більш складною, особливо з точки зору масштабування та безпеки. У 2000-х роках було розроблено кілька нових стандартів та протоколів СКК, які враховують нові виклики та можливості, такі як використання мобільних пристроїв, хмарних технологій та інтернету речей. Один з найважливіших стандартів, розроблений у цей період, - це стандарт AES, який забезпечує ефективне та безпечне симетричне шифрування.

Після 2000-х років застосування СКК значно розширилось. Завдяки швидкому розвитку Інтернету, збільшенню обсягу електронних операцій та

поширенню злочинних дій в Інтернеті, системи керування ключами стали важливим інструментом для захисту конфіденційності та цілісності даних в Інтернеті. З появою електронної пошти та електронних комунікаційних каналів, СКК стали важливим інструментом для забезпечення безпеки електронної переписки та передачі даних. У 2001 році був розроблений стандарт PKCS #12, який визначає формат електронного ключа та захищеного контейнера для зберігання приватного ключа. У 2010 році було запропоновано новий стандарт СКК - Key Management Interoperability Protocol (KMIP), який регулює взаємодію між різними системами керування ключами та дозволяє взаємодіяти з різними типами ключів та їх захищеними контейнерами.

Після 2010 року, у зв'язку з поширенням хмарних технологій та мобільних пристроїв, виникла потреба в більш гнучких і простих методах керування ключами. У зв'язку з цим почали розвиватись системи керування ключами як сервіс (KaaS - Key Management as a Service), що надаються в рамках хмарних платформ. Також з'явилися нові технології шифрування, зокрема шифрування на базі ідентичності, що дозволяє шифрувати дані безпосередньо на основі ідентичності користувача, не використовуючи при цьому сторонніх систем керування ключами. Це відкриває нові перспективи для безпеки даних в області IoT та блокчейн-технологій.

Також зростає значення захисту даних від квантових обчислювачів, тому з'являються нові методи шифрування, що забезпечують захист даних від квантового розкладу. Одним з найбільш відомих методів квантового шифрування є метод BB84, який застосовується для безпечної передачі даних на відстані за допомогою квантових каналів зв'язку.

Узагалі, розвиток технологій та зростання кількості кібератак спонукає до постійного вдосконалення систем керування ключами та методів шифрування, що дозволяє забезпечувати надійний захист даних в сучасному світі.

1.3 Основні види шифрування

Шифрування - це процес перетворення звичайного тексту (повідомлення) в криптований вигляд (шифр), який захищає його від читання або розуміння без дозволу. Існують різні види шифрування, основні з них - симетричне шифрування та асиметричне шифрування. Також можна виділити гомоморфне шифрування, квантове шифрування та інші.

1.3.1 Симетричне шифрування

Симетричне шифрування - це метод шифрування, в якому використовується той самий ключ для шифрування та дешифрування даних. Цей ключ повинен бути відомий як відправнику, так і отримувачу даних. Такий метод шифрування називається симетричним, оскільки ключі для шифрування та дешифрування є ідентичними.

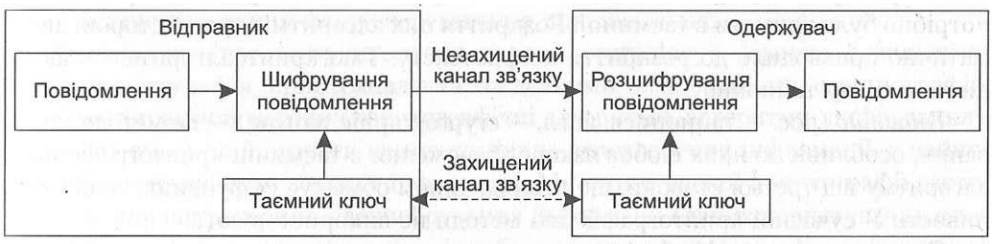

Рисунок 1.1 – Схема симетричного шифрування

Симетричне шифрування має декілька переваг, серед яких - швидкість шифрування та дешифрування, а також висока ефективність при роботі з великими об'ємами даних. Проте, цей метод має одну суттєву недолік - потребує передачі ключа між відправником та отримувачем даних, що може бути небезпечним при передачі через відкриті мережі.

Найпоширеніші алгоритми симетричного шифрування - це AES, DES, 3DES, Blowfish, IDEA та інші. Кожен з цих алгоритмів має свої особливості та переваги, які залежать від конкретного застосування.

Використання симетричного шифрування повинно бути обмежене виключно на закритих системах, де відправник та отримувач даних можуть довіряти один одному та забезпечити безпеку передачі ключа. Для захисту від несанкціонованого доступу до ключа та даних, які були зашифровані, можна використовувати додаткові заходи захисту, такі як хешування ключа та використання аутентифікаційних механізмів.

1.3.2 Асиметричне шифрування

Асиметричне шифрування - це метод шифрування, який використовує два ключі: публічний та приватний. Кожен користувач, що використовує асиметричне шифрування, має пару ключів: приватний ключ, який зберігається в таємниці, і публічний ключ, який розголошується.

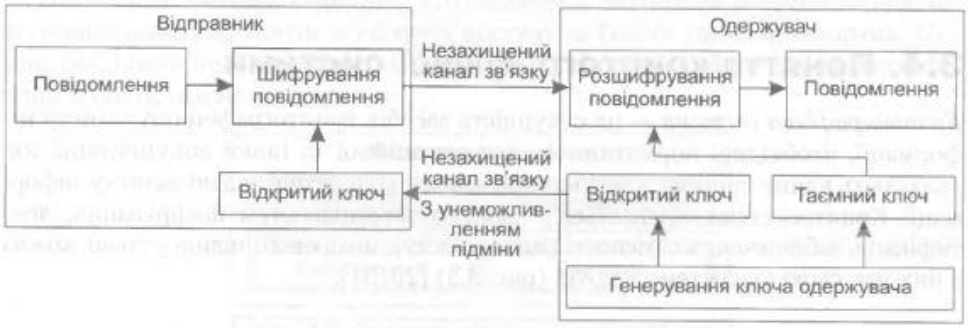

Рисунок 1.2 – Схема асиметричного шифрування

У процесі асиметричного шифрування, перед відправленням повідомлення, відправник шифрує його за допомогою отриманого публічного ключа отримувача. Тільки отримувач може розшифрувати повідомлення, використовуючи свій власний приватний ключ. Таким чином, повідомлення може бути надіслане через небезпечний канал комунікації без ризику доступу до нього третіх осіб.

Асиметричне шифрування використовується не тільки для захисту повідомлень, але й для реалізації системи електронного цифрового підпису

(ЕЦП), яка забезпечує перевірку автентичності повідомлення та ідентифікацію відправника.

Однією з найбільш поширених систем асиметричного шифрування є RSA, яка використовується у багатьох криптографічних протоколах та програмах. У системі RSA, публічний ключ може бути розголошеним, тоді як приватний ключ залишається в таємниці.

1.3.3 Гомоморфне шифрування

Гомоморфне шифрування - це метод шифрування, який дозволяє проводити операції з зашифрованими даними, не розшифровуючи їх. Іншими словами, гомоморфне шифрування дозволяє зашифровані дані залишати в зашифрованому вигляді, проводити з ними різні операції і отримувати результат у зашифрованому вигляді.

Цей метод шифрування має безліч застосувань, зокрема, в області обробки даних та хмарних обчислень, де може бути важко збирати дані в одному місці, але необхідно проводити операції з ними. Також гомоморфне шифрування може бути використано для збереження приватності даних, забезпечуючи можливість проводити обробку даних, не розкриваючи їх зміст.

У гомоморфному шифруванні існують два види: частково гомоморфне шифрування та повне гомоморфне шифрування. Частково гомоморфне шифрування дозволяє проводити обмежені операції з зашифрованими даними, наприклад, додавати два зашифровані числа. Повне гомоморфне шифрування дозволяє проводити будь-які операції з зашифрованими даними, зокрема, множення та ділення.

Одним з прикладів гомоморфного шифрування є RSA-шифрування з використанням модульного множення.

Рисунок 1.3 – Алгоритм гомоморфного шифрування

У цьому методі, якщо два зашифрованих числа зашифровані однією й тією ж парою ключів, то їх сума у зашифрованому вигляді буде еквівалентна зашифрованому добутку відповідних розшифрованих чисел.

1.3.4 Квантове шифрування

Квантове шифрування - це метод захисту інформації від перехоплення та підробки в процесі передачі, який використовує принципи квантової механіки. У квантовому шифруванні використовуються кванти інформації, які можуть бути знаходитися в станах, які відрізняються один від одного, а також засновані на принципах вимірювання в квантовій механіці.

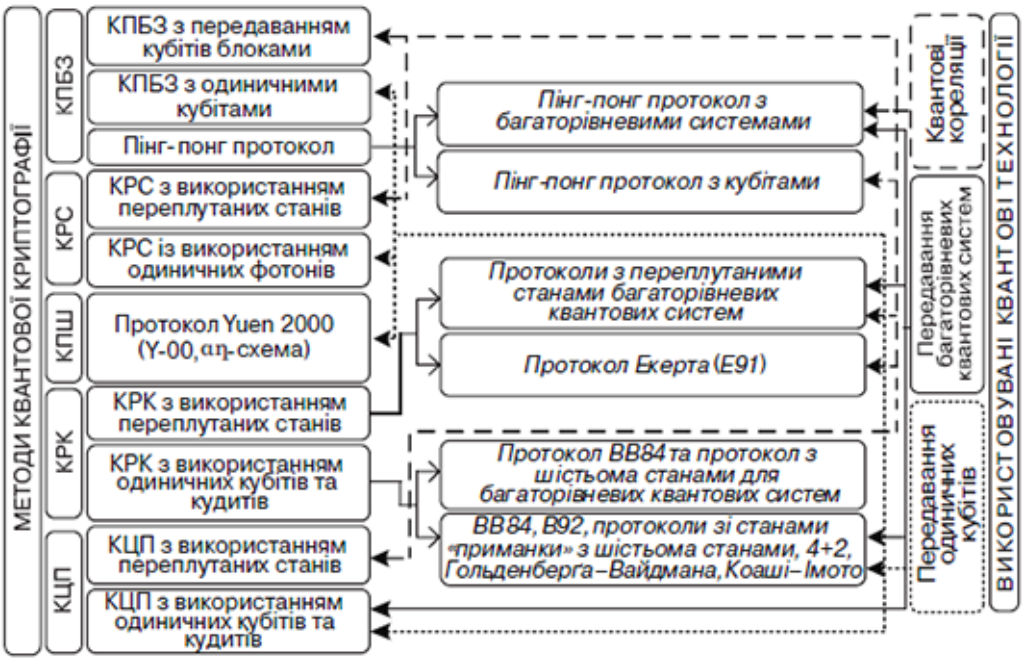

Рисунок 1.4 – Класифікація методів квантової криптографії

Квантове шифрування дозволяє безпечно передавати ключі шифрування з високою ступенем захисту від перехоплення. У звичайному шифруванні для передачі секретного ключа потрібно використовувати захищені канали зв'язку або постійно змінювати ключі, що не завжди може бути зручним або може викликати певні витрати. В квантовому шифруванні використовується принцип квантової нерозривності, який дає змогу забезпечити безпечний обмін секретним ключем між двома сторонами без ризику перехоплення.

Принцип квантової нерозривності полягає в тому, що передача кванту інформації викликає його зміну, тобто вимірювання стану кванту відбувається зміною самого стану кванту. Це означає, що з перехопленням кванту інформації стан кванту зміниться, а в результаті передача секретного ключа буде неможливою.

Найбільш відомі протоколи квантового шифрування - BB84 та E91. Протокол BB84 був запропонований Чарльзом Беннеттом та Девідом Шором у 1984 році. Цей протокол базується на використанні двох неперервних базисів - горизонтальний та вертикальний (X- та Z-базисів) для передачі

бітів, та діагонального та антидіагонального (Y- та Hadamard-базисів) для передачі квантових станів. Протокол використовує взаємодію клієнта та сервера для створення та передачі ключа шифрування. Цей протокол є одним з найбільш розповсюджених та відомих протоколів квантового шифрування.

У цьому протоколі, відправник (Карл) використовує згенеровані ним випадкові послідовності кубітів (qubits), які він передає отримувачу (Боб). Аліса випадково обирає один з двох можливих базисів для кожного qubit - вертикальний/горизонтальний або діагональний/антидіагональний.

Боб також випадково обирає один з двох базисів для кожного qubit. Карл та Боб взаємодіють, щоб визначити, які з qubits Боба були відправлені в тому ж самому базисі, що і qubits Карла. Ці qubits можуть бути використані для створення спільного ключа шифрування, який може бути використаний для захисту конфіденційної інформації.

Якщо присутній хтось, хто може прослуховувати передачу (Джон), то будь-яке спостереження або втручання в передачу qubits призведе до помилкових вимірів і зниження кількості qubits, які можна використати для створення спільного ключа.

Тому, якщо Карл та Боб помічають, що кількість qubits, які можна використати для створення спільного ключа, є недостатньою для забезпечення безпеки передачі, вони переривають зв'язок і повторюють процедуру знову. Це забезпечує високий рівень безпеки, оскільки квантові властивості qubits не можуть бути скопійовані або відтворені без помітного втручання

Протокол E91 був запропонований Артуром Екертом, Андре Леггеро та Антоном Зейлендингом у 1991 році. Цей протокол базується на використанні ентангльованих квантових станів для передачі ключа шифрування. Протокол вимагає створення спільних ентангльованих квантових станів, які потім можуть бути використані для передачі бітів інформації. Цей протокол відрізняється від BB84 тим, що він використовує тільки один тип базису та

не потребує додаткової інформації для розшифрування. Інші відомі протоколи квантового шифрування включають B92, Ekert 91+2 та KMB09.

Одним з головних викликів квантової криптографії є реалізація її у практичних системах. У сучасних квантових системах зазвичай використовуються волоконно-оптичні мережі для передачі квантових бітів. Однак, квантові біти дуже чутливі до шуму та інших втручань, тому їх передача може бути складною і вимагати додаткових заходів безпеки.

Також, наразі квантові комп'ютери все ще знаходяться на ранній стадії розвитку і не мають достатньої потужності для розшифрування складних шифрів. Однак, з розвитком технологій та збільшенням кількості qubits, це може змінитися.

Таким чином, квантова криптографія має великий потенціал для захисту конфіденційної інформації в майбутньому. Вона може бути використана в різних сферах, включаючи фінансову та медичну галузі, де конфіденційність даних є критично важливою. Однак, для того, щоб квантова криптографія стала реалізовуваною у практичних системах, потрібно ще багато роботи та досліджень.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ КЕРУВАННЯ КЛЮЧАМИ

Одним з основних трендів останніх років є зростання використання хмарних СКК, які дозволяють забезпечити безпеку ключів при зберіганні і обміні через хмарні сервіси. Такі системи стали популярними, оскільки вони забезпечують більшу мобільність і зручність у використанні. Водночас, виникає питання про безпеку зберігання ключів в хмарних СКК, а також про можливість злому або підробки ключів через ці сервіси.

Ще одним трендом є розвиток СКК, які забезпечують квантову безпеку. Квантові системи криптографічного захисту дозволяють забезпечити непроникну безпеку шифрування за рахунок використання фізичних принципів квантової механіки. Такі системи, хоча й дуже дорогі в розробці та використанні, вважаються майбутнім криптографічного захисту інформації.

Загалом можна сказати, що сучасні СКК дуже ефективні та надійні, проте вони не є повністю безпечними і не гарантують 100% захисту інформації.

Серед сучасних систем керування ключами варто зазначити відкриті стандарти, такі як PKCS від RSA Security, або OpenPGP, який базується на стандарті PGP і забезпечує шифрування електронної пошти та файлів. Одним з найбільш відомих стандартів в сучасних системах керування ключами є стандарт FIPS, який розробляється Національним інститутом стандартів і технологій США (NIST). Цей стандарт визначає вимоги до захисту інформації в федеральних установах США і включає в себе вимоги до генерації, зберігання та використання криптографічних ключів.

Крім того, існує багато комерційних систем керування ключами, які використовуються в промисловості та фінансових установах. Наприклад, RSA Key Manager від RSA Security, SafeNet KeySecure від Gemalto, та IBM Key Lifecycle Manager.

Незважаючи на різноманіття систем керування ключами, вони всі мають спільну мету - забезпечити безпеку криптографічних ключів та управляти ними в ефективний спосіб. Проте, з розвитком технологій та збільшенням кількості даних, що потребують захисту, системи керування ключами стають все складнішими та потребують більш продуманого та прискореного підходу до керування ключами.

2.1 Переваги

Серед переваг сучасних систем керування ключами можна виділити наступні:

- висока безпека: нові системи керування ключами використовують сучасні алгоритми шифрування, що забезпечують високий рівень безпеки. Крім того, вони можуть використовувати мультифакторну аутентифікацію, що додатково підвищує захист від несанкціонованого доступу;
- легкість у використанні: сучасні СКК мають інтуїтивно зрозумілий і простий інтерфейс, що дозволяє користувачам легко керувати ключами та забезпечувати безпеку своїх даних;
- гнучкість: нові системи керування ключами можуть працювати в різних середовищах та мережах, що забезпечує їх гнучкість і дозволяє легко інтегрувати їх з іншими системами;
- збереження ключів: сучасні системи керування ключами забезпечують зберігання ключів у безпечному місці, що знижує ризик їх втрати або крадіжки;
- автоматизація: нові СКК можуть автоматично генерувати ключі, що знижує ризик помилок користувачів та забезпечує високу швидкість обробки даних;
- інтеграція з іншими системами: сучасні системи керування ключами можуть легко інтегруватися з іншими системами, що забезпечує їх сумісність з існуючими інфраструктурами та додатками;

- керування життєвим циклом ключів: нові СКК дозволяють керувати життєвим циклом ключів, включаючи їх генерацію, розподіл, зміну, знищення та архівування, що забезпечує повну контроль над ключами. Це особливо важливо для організацій, які мають багато ключів, оскільки дозволяє легко відстежувати та контролювати їх стан. Нові СКК також дозволяють автоматично змінювати ключі через певні проміжки часу або після визначених подій, що забезпечує ще більшу безпеку даних;

- інтеграція з іншими системами безпеки: СКК можуть бути легко інтегровані з іншими системами безпеки, такими як системи ідентифікації та аутентифікації, системи керування доступом та системи моніторингу подій. Це дозволяє забезпечити повну безпеку даних та мережі;

- покращена безпека: СКК дозволяють забезпечити більш високий рівень безпеки, ніж традиційні методи керування ключами. Вони дозволяють захистити ключі від викрадення або втрати, а також дозволяють автоматично змінювати ключі через певний час або після певної кількості використань. Більшість нових СКК також дозволяють використовувати різні методи аутентифікації, що додає ще один рівень безпеки;

- легше керування: нові СКК зазвичай мають дуже простий та зрозумілий інтерфейс користувача, що дозволяє легко керувати ключами та виконувати всі необхідні операції. Крім того, вони забезпечують централізований контроль над ключами, що дозволяє забезпечити їх однаковий рівень безпеки в усіх частинах мережі;

- сумісність: більшість нових СКК можуть працювати з різними криптографічними алгоритмами та протоколами, що дозволяє їх використання в різних системах та з різними пристроями. Крім того, вони можуть бути інтегровані з іншими системами безпеки, що забезпечує комплексний підхід до захисту інформації.

Загалом, сучасні системи керування ключами дозволяють забезпечити більш високий рівень безпеки та ефективно керувати ключами в умовах постійного зростання кількості та складності криптографічних застосувань.

Такі системи дозволяють автоматизувати багато процесів, пов'язаних з керуванням ключами, та зменшують ризик витоку інформації або використання її несанкціоновано. Крім того, сучасні СКК забезпечують більш прозорий та документований процес керування ключами, що дозволяє підвищити відповідність до різних регуляторних вимог та стандартів безпеки. Зокрема, застосування сучасних СКК допомагає підприємствам та організаціям виконувати вимоги Підприємницького кодексу, ЄСКД, GDPR та інших стандартів безпеки, що є важливим аспектом для бізнесу в умовах постійно зростаючих кіберзагроз.

2.2 Недоліки

Хоча сучасні системи керування ключами мають багато переваг, вони також мають деякі недоліки:

Висока вартість: деякі сучасні СКК можуть бути дуже дорогими, що може зробити їх недосяжними для менших організацій з обмеженим бюджетом.

Складність впровадження: іноді впровадження нової системи керування ключами може бути складним і вимагати багато зусиль та ресурсів. Це може включати в себе перенесення даних зі старої системи та навчання персоналу використанню нової системи.

Залежність від виробника: сучасні СКК часто базуються на виробниках обладнання або програмного забезпечення, що може створювати залежність від цих виробників та обмежувати можливості організації.

Ризики безпеки: хоча сучасні СКК можуть забезпечити високий рівень безпеки, вони також можуть бути піддаються атакам, зокрема, атакам на програмне забезпечення та криптографічні алгоритми. Якщо хакерам вдасться зламати систему керування ключами, то вони матимуть доступ до всіх захищених даних. Це може призвести до серйозних наслідків для компанії або організації, яка використовує систему керування ключами.

Якщо система керування ключами не оновлюється регулярно, це може стати проблемою, оскільки старі ключі можуть бути вразливими до атак.

Обмежена масштабованість: деякі СКК можуть мати обмежену масштабованість, що може бути проблемою для великих організацій або об'єднаних мереж.

Ризики витоку даних: хоча СКК дозволяють забезпечити високий рівень безпеки, неправильна настройка системи або недбале ставлення до безпеки можуть призвести до витоку даних із системи керування ключами.

Загалом, при впровадженні СКК необхідно враховувати як їх переваги, так і недоліки, та використовувати їх з розумінням

3 ГЕНЕРАЦІЯ КЛЮЧІВ

Генерація ключів - це процес створення криптографічного ключа, який буде використовуватися для зашифрування та розшифрування даних в криптографічній системі. Ключ може бути згенерований за допомогою різних алгоритмів, в залежності від вимог до безпеки та розміру ключа.

Важливою частиною генерації ключів є випадковість. Якщо ключ буде створено з відомих або передбачуваних значень, то це може привести до порушення безпеки системи. Тому, більшість алгоритмів генерації ключів базуються на використанні псевдовипадкових чисел, які генеруються на основі фізичних процесів, таких як шум електронних компонентів, коливання радіочастотного поля тощо.

Важливо також забезпечити, щоб ключ був збережений в безпечному місці і не потрапив у руки несанкціонованої особи. Для цього можуть використовуватися системи керування ключами, які дозволяють керувати життєвим циклом ключів, включаючи їх генерацію, розподіл, зміну, знищення та архівування.

3.1 Приклад симетричного шифрування

Ось загальна структура алгоритму симетричного шифрування:

- ініціалізація ключа: Генерується та ініціалізується секретний ключ для шифрування;
- розбиття повідомлення на блоки: Повідомлення, яке потрібно зашифрувати, розбивається на блоки фіксованої довжини;
- шифрування блоків: Кожен блок повідомлення шифрується з використанням секретного ключа. Це може бути виконано різними симетричними шифрувальними алгоритмами, такими як AES, DES, або іншими;

- збереження або передача зашифрованих блоків: Зашифровані блоки можуть бути збережені в файлі або передані по мережі до отримувача;

- розшифрування блоків: Отримувач використовує той самий секретний ключ для розшифрування блоків зашифрованого повідомлення.

Лістинг 3.1 – Реалізації симетричного шифрування

```
     import random
def generateSecretKey(key_length):

    #Генерація випадкового секретного ключа заданої довжини.

    letters_and_digits =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    secret_key = ''.join(random.choice(letters_and_digits) for _
in range(key_length))
    return secret_key

def encrypt(message, key):

    # Шифрування повідомлення з використанням симетричного
ключа.
    encrypted_message = ""
    for char in message:
        encrypted_char = chr((ord(char) + key) % 256)
        encrypted_message += encrypted_char
    return encrypted_message
def decrypt(encrypted_message, key):

    # Розшифрування зашифрованого повідомлення з використанням
симетричного ключа.

    decrypted_message = ""
    for char in encrypted_message:
        decrypted_char = chr((ord(char) - key) % 256)
        decrypted_message += decrypted_char
    return decrypted_message
# Приклад використання
key_length = 16
secret_key = generateSecretKey(key_length)
print("Секретний ключ:", secret_key)

message = "Hello, World!"
encrypted_message = encrypt(message, len(secret_key))
print("Зашифроване повідомлення:", encrypted_message)
decrypted_message = decrypt(encrypted_message, len(secret_key))
print("Розшифроване повідомлення:", decrypted_message)
```

Результатом цього коду буде:

- секретний ключ: uHrS9h3Na20ETtz0;
- зашифроване повідомлення: Xu||<0g|t1;
- розшифроване повідомлення: Hello, World!.

У цьому прикладі використовується простий алгоритм шифрування, де кожен символ повідомлення зсувається на величину ключа за модулем 256. Функція ‘encrypt’ шифрує повідомлення, додаючи ключ до коду кожного символу. Функція ‘decrypt’ розшифровує зашифроване повідомлення, віднімаючи ключ від коду кожного символу.

3.2 Приклад асиметричного шифрування

Загальна структура алгоритму симетричного шифрування:

- генерація ключів: Спочатку генерується ключова пара, що складається з приватного ключа (private key) і публічного ключа (public key). Приватний ключ залишається в таємниці, а публічний ключ розповсюджується;

- шифрування: Для зашифрування повідомлення використовується публічний ключ одержувача. Повідомлення шифрується з використанням публічного ключа, що перетворює його в незрозумілий для посторонніх символьний рядок або бінарні дані;

- передача шифрованого повідомлення: Зашифроване повідомлення передається одержувачу по відкритому каналу зв'язку. Оскільки повідомлення зашифроване з використанням публічного ключа, його може переглядати будь-хто, але лише особа з відповідним приватним ключем зможе розшифрувати його;

- розшифрування: Одержувач використовує свій приватний ключ для розшифрування отриманого шифрованого повідомлення. Розшифроване повідомлення стає зрозумілим для одержувача.

Лістинг 3.2 – Реалізації асиметричного шифрування

```
from cryptography.hazmat.primitives.asymmetric import rsa,
padding
from cryptography.hazmat.primitives import serialization, hashes

def generate_key_pair():
    # Генерація ключової пари
    private_key =
rsa.generate_private_key(public_exponent=65537, key_size=2048)
    public_key = private_key.public_key()

    # Перетворення ключів у байтовий формат
    private_key_bytes = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    public_key_bytes = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    return private_key_bytes, public_key_bytes

def encrypt_message(public_key, message):
    # Завантаження публічного ключа
    public_key = serialization.load_pem_public_key(public_key)

    # Шифрування повідомлення
    encrypted_message = public_key.encrypt(
        message.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

    return encrypted_message

def decrypt_message(private_key, encrypted_message):
    # Завантаження приватного ключа
    private_key =
serialization.load_pem_private_key(private_key, password=None)

    # Розшифрування повідомлення
    decrypted_message = private_key.decrypt(
```

Лістинг 3.2 – Продовження

```
encrypted_message,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),

    label=None
        )
    )

    return decrypted_message

# Приклад використання

# Генерація ключової пари
private_key, public_key = generate_key_pair()

# Шифрування повідомлення
message = "Hello, World!"
encrypted_message = encrypt_message(public_key, message)

# Розшифрування повідомлення
decrypted_message = decrypt_message(private_key,
encrypted_message)

print("Повідомлення:", message)
print("Зашифроване повідомлення:", encrypted_message)
print("Розшифроване повідомлення:", decrypted_message)
```

Результатом цього коду буде:

- повідомлення: Hello, World!;
- зашифроване повідомлення

b'k\x02\xf9\xd7]\t8x*\x12\xda\x08H\xa8\xed]O
p$\x8dQ\x96\xb4\xe8\xbaK\xe1\x1e\xa1F\xc3St\x9e\xc7\xa2\x93\xec\xcf\xc6\x92
\xde\xb01\x11\x05!\t\x8eg\x89\x95C\xb1\x1d\xc7T\x05\xe4_)y{k\xed4\xbf3\xc3\
x03\x80AQV\xe6X1KY\xf6\xfeG\xecBG\xe49\x97\x8e(Q:\x9f\t\xdb*\x8f\xe7@\x
1bW\x8b\x84K\x1d\x81\xd5r\xf5\xf3\x13<O\xc7\xc2\x8b!419\x82U\x19\xd1l\xbb
\xad\x02\xa0\x9e\x89[H\xbd#vh\xabo\xde\x9d\x92\x93)\x10\x80(\xb5O/-
\xda\x86\xc6I\xde\xebI\xe9\xc8\xcf{\xfb\xebq\r*jc\xb9\x9f\x17\xcf\xb7\x0e\xd2\x
94<\xe0X\xd5\x06n\xf5\x19\xda/Q\xcb\xc2v.\xb4}z\xd6\xc0\xf9\xce\x0e\nmt\x13
g\xd5G~%+\x1e\xc3\xeb\xa6\x8en\xcf\x8e;\xd5\x9f^{\xf0+\x19\x9a\xd5\xc7,Ii\x1

f@\xb4\xe5\x81k\xe7\xecH*\xad\xbb\xbf\x89\xb4E|\xf48\\I\x9e\x99'

- розшифроване повідомлення: b'Hello, World!'.

У цьому прикладі ми використовуємо бібліотеку CRYPTOGRAPHY, яка надає функціональність для роботи з криптографією. Ми генеруємо ключову пару RSA, шифруємо повідомлення з використанням публічного ключа одержувача та розшифровуємо його за допомогою приватного ключа.

3.3 Приклад гомоморфного шифрування

Загальна структура алгоритму гомоморфного шифрування включає декілька етапів.

Лістинг 3.3 – Структура алгоритму гомоморфного шифрування

```
class HomomorphicEncryption:
    def __init__(self, public_key, private_key):
        self.public_key = public_key
        self.private_key = private_key

    def encrypt(self, plaintext):
        # Реалізація шифрування даних
        # ...

    def decrypt(self, ciphertext):
        # Реалізація розшифрування даних
        # ...

    def add(self, ciphertext1, ciphertext2):
        # Реалізація операції додавання
        # ...

    def multiply(self, ciphertext1, ciphertext2):
        # Реалізація операції множення
        # ...

# Приклад використання

# Генерація ключів
public_key, private_key = generate_key_pair()
# Ініціалізація гомоморфного шифрування
encryption = HomomorphicEncryption(public_key, private_key)
# Шифрування даних
plaintext1 = 10
plaintext2 = 5
```

Лістинг 3.3 – Продовження

```
ciphertext1 = encryption.encrypt(plaintext1)
ciphertext2 = encryption.encrypt(plaintext2)

# Виконання операцій
result_add = encryption.add(ciphertext1, ciphertext2)
result_multiply = encryption.multiply(ciphertext1, ciphertext2)

# Розшифрування результатів
decrypted_add = encryption.decrypt(result_add)
decrypted_multiply = encryption.decrypt(result_multiply)

print("Повідомлення 1:", plaintext1)
print("Повідомлення 2:", plaintext2)
print("Результат додавання:", decrypted_add)
print("Результат множення:", decrypted_multiply)
```

Лістинг 3.4 – Реалізація функції шифрування

```
def encrypt(self, plaintext):
    # Перетворення повідомлення в числовий формат
    message = int.from_bytes(plaintext.encode(), 'big')

    # Використання публічного ключа для шифрування повідомлення
    ciphertext = pow(message, self.public_key['e'],
self.public_key['n'])

    return ciphertext
```

Ця функція приймає повідомлення plaintext як вхідний параметр. Спочатку вона перетворює текстове повідомлення на числове значення message, використовуючи int.from_bytes(). Далі вона застосовує публічний ключ для шифрування повідомлення за допомогою операції піднесення до степені з використанням модуля pow(). Результат шифрування ciphertext повертається з функції.

Лістинг 3.5 – Реалізація функції дешифрування

```
def decrypt(self, ciphertext):
    # Використання приватного ключа для розшифрування шифротексту
    decrypted_message = pow(ciphertext, self.private_key['d'],
self.private_key['n'])
    # Повернення розшифрованого повідомлення у вигляді рядка
    plaintext = decrypted_message.to_bytes((decrypted_message.bit_length()
+ 7) // 8, 'big').decode()

    return plaintext
```

Ця функція отримує шифротекст ciphertext як вхідний параметр. Вона застосовує приватний ключ для розшифрування шифротексту за допомогою операції піднесення до степені з використанням модуля pow(). Розшифроване повідомлення decrypted_message перетворюється на рядок plaintext за допомогою методу to_bytes() та decode(). Нарешті, розшифроване повідомлення повертається з функції.

Лістинг 3.6 – Реалізація функції додавання зашифрованих чисел

```
def add(self, ciphertext1, ciphertext2):
    # Використання публічного ключа для виконання операції
додавання на шифротекстах
    result_ciphertext = (ciphertext1 + ciphertext2) %
self.public_key['n']

    return result_ciphertext
```

Ця функція отримує два шифротексти, ciphertext1 та ciphertext2, як вхідні параметри. Вона використовує публічний ключ для виконання операції додавання на шифротекстах за допомогою формули (ciphertext1 + ciphertext2) % self.public_key['n']. Результат операції додавання result_ciphertext повертається з функції.

Лістинг 3.7 - Реалізація функції множення зашифрованих чисел

```
def multiply(self, ciphertext1, ciphertext2):
    # Використання публічного ключа для виконання операції
множення на шифротекстах
    result_ciphertext = (ciphertext1 * ciphertext2) %
self.public_key['n']

    return result_ciphertext
```

Ця функція отримує два шифротексти, ciphertext1 та ciphertext2, як вхідні параметри. Вона використовує публічний ключ для виконання операції множення на шифротекстах за допомогою формули (ciphertext1 * ciphertext2) % self.public_key['n']. Результат операції множення result_ciphertext повертається з функції.

4 БІОМЕТРИЧНИЙ ЗАХИСТ

4.1 Визначення біометричного захисту інформації

Біометрія – сукупність автоматизованих методів і засобів ідентифікації людини, заснованих на її фізіологічній або поведінковій характеристиці.

Основні принципи біометричного захисту це біометричні характеристики, збір біометричних данних та їх обробка.

Біометричні характеристики включають різноманітні фізичні та поведінкові ознаки особи, які можуть бути використані для ідентифікації та автентифікації. Ось декілька видів біометричних характеристик:

- відбитки пальців: є одним з найпоширеніших видів біометричних характеристик. Унікальна рельєфна структура пальцевих відбитків може бути використана для точної ідентифікації особи;
- розпізнавання обличчя: Цей вид біометричної характеристики використовує унікальні риси обличчя, такі як форма обличчя, положення очей, ніздрів та губ, для ідентифікації особи. Він широко застосовується в системах безпеки та розпізнавання;
- розпізнавання структури сітківки: Сітківка ока має унікальну мережу кровоносних судин, яка може бути використана для ідентифікації особи. Для отримання структури сітківки зазвичай використовується спеціальне обладнання, таке як сканери сітківки;
- розпізнавання ірису: Ірис ока має унікальні структури, такі як штрихи і колір, які можуть бути використані для ідентифікації особи. Системи розпізнавання ірису зазвичай вимагають спеціального обладнання для сканування та аналізу ірису;
-
-
-

- розпізнавання голосу: Голос має унікальні акустичні характеристики, такі як тембр, частота та інтонація, які можуть бути використані для ідентифікації особи. Системи розпізнавання голосу аналізують ці характеристики для здійснення автентифікації;

- динаміка письма: Цей вид біометричної характеристики використовується для аналізу особливостей почерку особи. Це може включати швидкість письма, ритм, тиск та стилістику письма.

Існують також інші види біометричних характеристик, такі як динаміка ходьби, структура вен, форма вуха та інші, які також можуть бути використані для ідентифікації особи. Вибір конкретного виду біометричної характеристики залежить від конкретних вимог, контексту застосування та технічних можливостей.

4.2 Збір біометричних даних та їх обробка

Процес збору біометричних даних та їх подальша обробка включає кілька етапів. Збір біометричних даних: Перший етап - збір самої біометричної характеристики від особи. Наприклад, це може бути сканування відбитків пальців, фотографування обличчя, запис голосу або сканування сітківки. Залежно від конкретного виду біометричної характеристики, можуть використовуватися спеціальні датчики або пристрої.

Перетворення в цифровий формат: Після збору біометричних даних, вони зазвичай перетворюються в цифровий формат для подальшої обробки та збереження. Це може включати процеси, такі як дискретизація голосу, витягування ключових особливостей з відбитків пальців або створення шаблонів обличчя.

Витягнення характеристик: Наступним кроком є витягнення ключових характеристик з цифрових даних біометричної характеристики. Це може бути вектор або шаблон, який представляє унікальні особливості біометричного зразка. Наприклад, відбитки пальців можуть бути представлені у вигляді

точок перетину ліній (мінувань), а ірис ока - як шаблон з текстурних ознак.

Збереження шаблонів: Отримані шаблони біометричних даних можуть бути збережені у базі даних або на захищеному сервері. Забезпечення безпеки і конфіденційності біометричних даних - важлива складова цього етапу.

Порівняння і ідентифікація: Під час процесу ідентифікації біометричний зразок порівнюється зі збереженими шаблонами у базі даних. Застосовуються алгоритми порівняння для визначення ступеня відповідності або подібності між зразком і шаблонами.

Підтвердження автентичності: У разі автентифікації біометричний зразок порівнюється з відповідним шаблоном для підтвердження автентичності особи. Це може бути використано для контролю доступу до систем або пристроїв.

Важливо відзначити, що обробка біометричних даних повинна бути виконана з дотриманням відповідних протоколів безпеки та конфіденційності, оскільки ці дані є особистими і чутливими.

4.3 Роль біометричних даних у шифруванні

Однією з головних ролей біометричних даних у шифруванні є їх використання для автентифікації особи. Замість традиційних методів аутентифікації, таких як паролі або PIN-коди, біометричні дані можуть служити унікальним "паролем". Кожна особа має унікальну біометричну характеристику, таку як відбиток пальця, розпізнавання обличчя або голосу, і ці дані можуть бути використані для точної ідентифікації особи.

Біометричні дані також можуть використовуватись для захисту ключів шифрування. Ключі шифрування є важливим елементом у забезпеченні безпеки даних, і використання біометричних даних для створення та захисту цих ключів дозволяє підвищити безпеку. Ключі шифрування можуть бути згенеровані на основі біометричних характеристик особи або збережені у

пристрої, який вимагає біометричну автентифікацію для доступу.

Іншою важливою роллю біометричних даних у шифруванні є захист інформації. За допомогою біометричних даних можна шифрувати файли або документи таким чином, що доступ до них буде мати лише особа, у якої співпадають біометричні характеристики збережених даних. Це надає додатковий рівень безпеки та гарантує, що конфіденційна інформація залишається захищеною від несанкціонованого доступу.

Використання біометричних даних у шифруванні також включає в себе процеси збору та обробки даних. Збір біометричних даних вимагає використання спеціальних датчиків або пристроїв, які реєструють та фіксують біометричні характеристики. Після цього дані перетворюються в цифровий формат, витягуються ключові характеристики та зберігаються у безпечних базах даних.

Зважаючи на важливість безпеки і конфіденційності інформації, використання біометричних даних у шифруванні стає все більш поширеним. Вони надають надійну ідентифікацію особи, захищають ключі шифрування та забезпечують безпеку даних. Проте, важливо враховувати потенційні проблеми з приватністю та захистом даних, пов'язаних з використанням біометричних даних, і розробляти та використовувати відповідні протоколи безпеки та захисту для їх обробки та зберігання.

Передбачається, що майбутні розвиток біометричного захисту буде характеризуватися наступними перспективами:

Розширення використання біометричних характеристик: На сьогоднішній день використовуються такі типи біометричних характеристик, як відбиток пальця, розпізнавання обличчя та голосу. Однак, майбутнє прихильників біометричного захисту полягатиме у використанні більш широкого спектру характеристик, таких як сетчатка ока, венозна структура, походка, електроенцефалограма та інші. Це дозволить створити більш точні та надійні системи ідентифікації.

Застосування штучного інтелекту: Розвиток штучного інтелекту відкриває нові можливості для біометричного захисту. Штучний інтелект може покращити процеси збору, обробки та аналізу біометричних даних, а також підвищити точність ідентифікації. Алгоритми машинного навчання та глибокого навчання можуть використовуватися для розпізнавання та класифікації біометричних характеристик, що покращує ефективність системи.

Забезпечення мультимодальної аутентифікації: Майбутнє біометричного захисту полягатиме у комбінуванні різних типів біометричних характеристик для створення мультимодальних систем аутентифікації. Наприклад, комбінація відбитку пальця з розпізнаванням обличчя або голосу може забезпечити більш надійний рівень ідентифікації, оскільки потрібно співпадіння кількох характеристик для отримання доступу.

Розширення застосування біометричного захисту: Наразі біометричний захист широко використовується в таких галузях, як фізична безпека, фінансові послуги та мобільні пристрої. Однак, в майбутньому його застосування може розширитися на такі сфери, як охорона мережі Інтернет речей, медична сфера, управління документами та багато інших. Біометричний захист може стати необхідним елементом безпеки у всіх аспектах нашого життя.

Усе це вказує на те, що біометричний захист має великий потенціал для подальшого розвитку і впровадження у різні сфери життя. Його використання може забезпечити високий рівень безпеки та надійності, забезпечуючи одночасно зручність для користувачів. Захист даних та ідентифікація особи шляхом використання біометричних характеристик мають потенціал змінити спосіб, яким ми забезпечуємо безпеку і взаємодіємо з технологіями у майбутньому.

Використання біометричного захисту має потенціал змінити спосіб, яким ми забезпечуємо безпеку та взаємодіємо з технологіями. Водночас, необхідно враховувати проблеми приватності та захисту даних, пов'язані з використанням біометричних даних, і розробляти відповідні протоколи безпеки для їх обробки та зберігання.

Загалом, біометричний захист інформації є обіцяючим напрямом у сфері безпеки та конфіденційності даних. Впровадження цієї технології може забезпечити надійну ідентифікацію особи та захист важливої інформації. З урахуванням постійного розвитку технологій і зростання вимог до безпеки, біометричний захист відіграє ключову роль у створенні безпечного та захищеного цифрового середовища.

5 ЕТАПИ СТВОРЕННЯ ВЛАСНОЇ СИСТЕМИ КЕРУВАННЯ КЛЮЧАМИ

Розробка СКК вимагає комплексного підходу та урахування безпеки, доступності та ефективності. Далі наведено загальний опис того, як можна розробити таку систему.

Визначте вимоги безпеки: Перш за все, важливо забезпечити надійне шифрування ключів. Вибір сильних криптографічних алгоритмів, таких як AES, RSA або ECC, є основою для захисту ключів від несанкціонованого доступу та атак на їх конфіденційність. Рекомендується використовувати ключі достатньої довжини для забезпечення стійкості шифрування.

Крім шифрування, система керування ключами повинна мати потужні механізми аутентифікації. Важливо переконатися, що лише правильні користувачі мають доступ до ключів. Застосування двофакторної аутентифікації або біометричних методів допоможе підвищити рівень безпеки і ускладнити процес несанкціонованого доступу до ключів.

Авторизація є ще одним важливим аспектом безпеки системи керування ключами. Встановлення політик авторизації, визначення ролей та прав користувачів, а також застосування механізмів контролю доступу дозволить обмежити доступ до ключів тільки для необхідних осіб. Такий підхід мінімізує ризик недостатньо контрольованого розподілу ключів або використання їх несанкціонованими особами.

Система керування ключами також повинна включати механізми аудиту. Реєстрація подій, пов'язаних зі створенням, розподілом, використанням та відкликанням ключів, дозволяє виявляти ненормальну або підозрілу активність, а також здійснювати перевірку безпеки та аудиту системи. Аудитні журнали можуть бути корисними для виявлення вразливостей та розслідування потенційних інцидентів безпеки.

Захист ключів від втрати та компрометації є ще одним важливим аспектом системи керування ключами. Застосування фізичних та логічних

заходів безпеки, таких як розміщення ключів у безпечних приміщеннях або використання апаратних модулів забезпечення для їх захисту, допомагають убезпечити ключі від небажаного доступу.

Визначте архітектуру: Почнемо з вибору між централізованою та розподіленою архітектурою. Централізована архітектура передбачає наявність центрального сервера ключів, який відповідає за керування всіма ключами в системі. Це може бути зручно для централізованих організацій, де потрібен єдиний пункт керування та контролю над ключами. З іншого боку, розподілена архітектура передбачає розподілення керування ключами між кількома серверами або компонентами системи. Це може бути корисно в розподілених середовищах або великих організаціях з різними відділами або локаціями. Вибір між централізованою та розподіленою архітектурою залежить від розмірів та потреб вашої організації.

Наступним ключовим аспектом є компоненти системи. Система керування ключами може включати різні компоненти, такі як сервери ключів, бази даних для зберігання ключів, апаратне забезпечення для генерації та зберігання ключів, а також інтерфейси для взаємодії з користувачами та іншими системами. Важливо визначити, які компоненти необхідні для вашої системи та як вони будуть взаємодіяти між собою. Наприклад, можливо, вам знадобиться веб-інтерфейс для адміністрування та керування ключами, а також API для інтеграції з іншими системами.

Розробіть процес створення та розподілу ключів: Генерація ключів є початковим етапом управління ключами. При розробці системи керування ключами важливо використовувати надійні методи генерації ключів. Криптографічно стійкі генератори псевдовипадкових чисел (CSPRNG) повинні використовуватись для створення ключів. Ці генератори забезпечують випадковість та непередбачуваність ключів, що є критичним для їх безпеки.

Одним з важливих аспектів управління ключами є збереження та забезпечення конфіденційності ключів. Ключі повинні бути збережені в

безпечному середовищі, що забезпечує захист від несанкціонованого доступу. Використання апаратного забезпечення, такого як апаратні модулі забезпечення (HSM), може бути корисним для фізичного захисту ключів та забезпечення їх конфіденційності.

Крім того, важливим аспектом є управління життєвим циклом ключів. Це включає створення, розподіл, зміну, відкликання та обновлення ключів. Кожен ключ повинен мати визначений термін дії, після якого він повинен бути замінений або оновлений. Механізми для автоматичного керування життєвим циклом ключів, включаючи автоматичну зміну та оновлення ключів, можуть спростити процес управління ключами і забезпечити їх актуальність та безпеку.

Додатковою важливою аспектом є забезпечення аудиту та відстеження використання ключів. Система керування ключами повинна вести реєстрацію подій, пов'язаних з використанням ключів, щоб виявляти ненормальну або підозрілу активність. Це допомагає усунути можливі проблеми безпеки та виявити небажану діяльність, що може вказувати на компрометацію ключів або порушення безпеки.

Реалізуйте шифрування та розшифрування: Ключі повинні бути генеровані з використанням криптографічно стійких алгоритмів та зберігатись в захищеному середовищі, наприклад, за допомогою HSM.

Іншим важливим аспектом є розширення шифрування на рівні даних. У системі керування ключами можна використовувати методи шифрування для захисту конфіденційності даних на рівні файлів, баз даних або навіть на рівні окремих полів у базі даних. Це дозволяє забезпечити додатковий рівень захисту для конфіденційних даних, навіть якщо зовнішній периметр системи порушений.

Забезпечте резервне копіювання: основна мета резервного копіювання - забезпечити можливість відновлення системи та ключів у разі непередбачуваних подій, таких як технічні збої, природні катастрофи або кібератаки.

Стратегія резервного копіювання повинна бути ретельно розроблена та включати кілька важливих етапів. По-перше, важливо ідентифікувати ключові компоненти системи, що потребують резервного копіювання, такі як бази даних ключів, конфігураційні файли, сертифікати та приватні ключі. По-друге, необхідно вибрати відповідні методи та інструменти для резервного копіювання, такі як резервне копіювання на зовнішні носії, в хмарні сховища або використання систем автоматичного резервного копіювання.

Крім того, важливо встановити регулярний розклад резервного копіювання для забезпечення актуальності та цілісності копій даних. Це може включати щоденне, щотижневе або місячне резервне копіювання в залежності від вимог безпеки та критичності інформації. Також слід розглянути можливість автоматизованого резервного копіювання, яке спрощує процес та гарантує регулярність та правильність виконання.

Забезпечте масштабованість: Масштабованість означає здатність системи керування ключами ефективно пристосовуватись до зростаючих обсягів даних, користувачів, операцій та вимог до безпеки. Вимоги до системи керування ключами можуть змінюватись з часом, тому важливо мати гнучкість та масштабованість для вирішення цих змін.

Одним з ключових аспектів масштабованості є горизонтальне масштабування. Це означає здатність системи розширюватись шляхом додавання нових ресурсів, таких як сервери, обчислювальні вузли або бази даних. Горизонтальне масштабування дозволяє розділити навантаження між багатьма ресурсами та забезпечити високу продуктивність та доступність системи.

Для досягнення масштабованості системи керування ключами можна використовувати розподілені архітектури, такі як кластери або розподілені бази даних. Це дозволяє розділити обробку та зберігання даних між різними вузлами системи та забезпечити паралельну обробку операцій.

Окрім того, важливим аспектом масштабованості є використання автоматичного масштабування. Це означає, що система може автоматично

реагувати на зростаюче навантаження та збільшувати ресурси, не потребуючи втручання операторів. Наприклад, можна використовувати хмарні платформи, які автоматично масштабують ресурси в залежності від потреб системи.

Перевірка безпеки: Один з підходів до перевірки безпеки системи керування ключами - це проведення аудиту безпеки. Аудит безпеки включає аналіз архітектури системи, перевірку політик безпеки, оцінку схеми аутентифікації та авторизації, перевірку захисту даних та застосування криптографічних алгоритмів. Аудит допомагає виявити можливі слабкі місця та ризики в системі керування ключами та рекомендує відповідні заходи для їх усунення.

Також важливим аспектом перевірки безпеки є тестування на проникнення. Це процес активного тестування системи з метою виявлення потенційних уразливостей та експлуатації їх для зламу безпеки. Під час тестування використовуються різні техніки та інструменти, що дозволяють проникнути в систему та здійснити атаки для перевірки її стійкості та забезпечення безпеки.

У додаток до аудиту безпеки та тестування на проникнення, існують також інші методи перевірки безпеки, такі як сканування вразливостей, аналіз журналів подій, моніторинг мережі та виявлення аномалій. Ці методи допомагають виявити потенційні проблеми безпеки та події, які можуть вказувати на атаку або порушення безпеки системи керування ключами.

Документація: Документація в системі керування ключами включає всі необхідні записи, інструкції та деталі, які описують різні аспекти системи керування ключами. Це включає технічну документацію, політики безпеки, процедури, стандарти, журнали та інші важливі документи. Документація допомагає забезпечити зрозумілість, стабільність та безпеку в управлінні ключами.

Одним з основних документів в системі керування ключами є політики безпеки. Вони встановлюють правила, процедури та вимоги щодо

використання ключів, зберігання, розподілу, аудиту та видалення. Політики безпеки допомагають забезпечити однорідність та безпеку в управлінні ключами, а також допомагають забезпечити відповідність до внутрішніх та зовнішніх вимог безпеки.

Технічна документація також відіграє важливу роль в системі керування ключами. Це можуть бути специфікації, описи архітектури системи, процедури реєстрації та використання ключів, алгоритми шифрування та інші технічні деталі. Технічна документація допомагає розуміти та ефективно використовувати систему керування ключами, а також спрощує процес впровадження та підтримки системи.

Окрім цього, журнали, звіти та інші документи, які відображають історію дій і подій в системі керування ключами, також є важливою частиною документації. Журнали допомагають відслідковувати активності користувачів, аудиторську інформацію, а також виявляти та вирішувати проблеми безпеки. Звіти та аналізи можуть надавати важливі відомості про ефективність, ризики та рекомендації щодо вдосконалення системи керування ключами.

Для ефективного управління документацією в системі керування ключами рекомендується дотримуватися кількох важливих кроків. По-перше, варто розробити структуровану систему документації, що включає логічну організацію документів та їх зручний доступ. По-друге, документацію слід оновлювати та підтримувати в актуальному стані, забезпечуючи відповідність змінам в системі керування ключами та вимогам безпеки. По-третє, важливо забезпечити відповідну авторизацію та контроль доступу до документації, щоб уникнути несанкціонованого доступу та зловживань.

5.1 Криптографічні бібліотеки

На рисунку 5.1 представлено класифікацію та порівняльну характеристику основних криптографічних бібліотек Python. Візуально

виділено рекомендовані, стандартні, вбудовані та застарілі рішення залежно від рівня безпеки, актуальності та сфери застосування.

У категорію рекомендованих входять бібліотеки з активною підтримкою, високим рівнем надійності та широким функціоналом, зокрема cryptography, PyNaCl, bcrypt і argon2-cffi. Вони охоплюють потреби у симетричному та асиметричному шифруванні, хешуванні паролів, цифрових підписах і захищеному обміні даними.

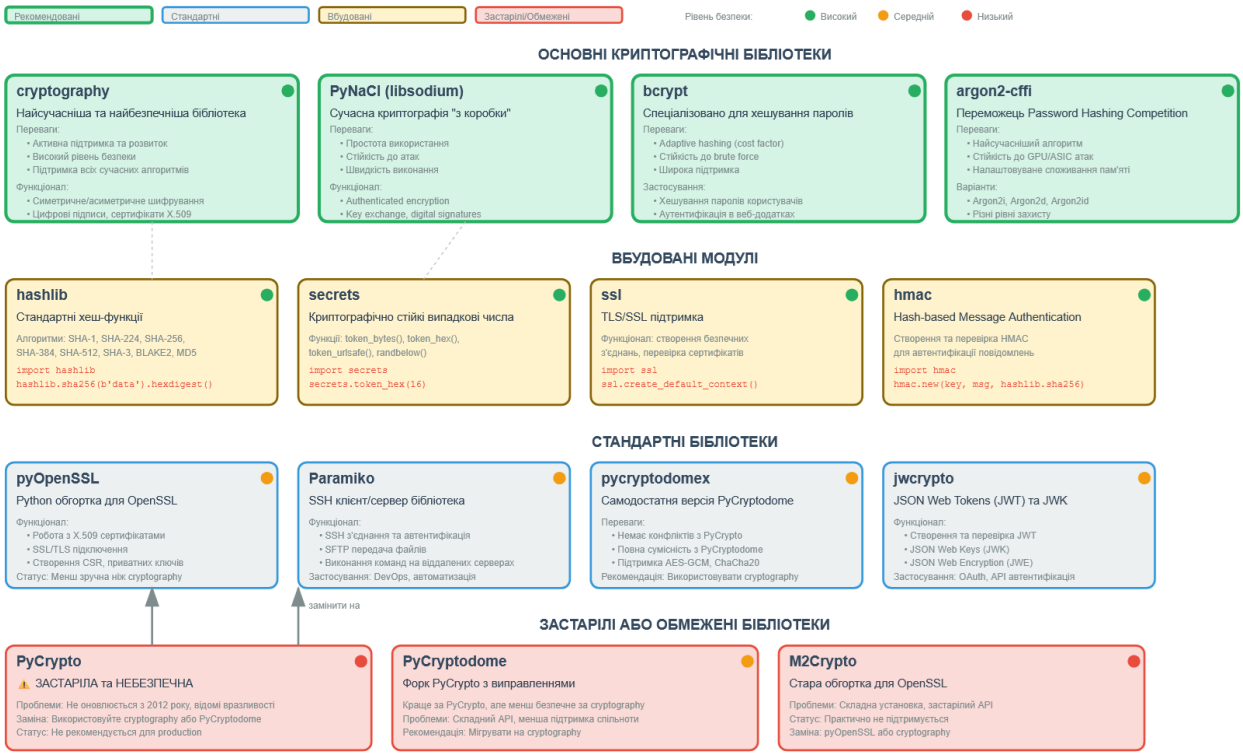

Рисунок 5.1 – Порівняння бібліотек

Вбудовані модулі, такі як hashlib, ssl, secrets та hmac, забезпечують базові криптографічні функції, що вбудовані у Python, і є достатніми для багатьох типових завдань.

Стандартні бібліотеки pyOpenSSL, Paramiko, pycryptodomex і jwcrypto розширюють можливості для роботи з TLS/SSL, SSH, JWT та іншими протоколами, хоча деякі з них мають обмежену безпеку або функціональність порівняно з сучасними альтернативами.

Нарешті, застарілі або небезпечні бібліотеки (PyCrypto, M2Crypto,

частково PyCryptodome) позначено як непридатні для використання у виробничих середовищах через відсутність оновлень, відомі вразливості та складність інтеграції. Їх рекомендується замінити на сучасні та підтримувані рішення.

5.2 Архітектура системи керування ключами

Загальна архітектура системи керування ключами являє собою багаторівневу структуровану модель, яка охоплює основні аспекти життєвого циклу криптографічних ключів – від моменту їх створення до безпечного знищення. Система реалізується у вигляді п’яти взаємопов’язаних модулів, кожен із яких виконує спеціалізовані функції у межах захисту та управління ключовою інформацією.

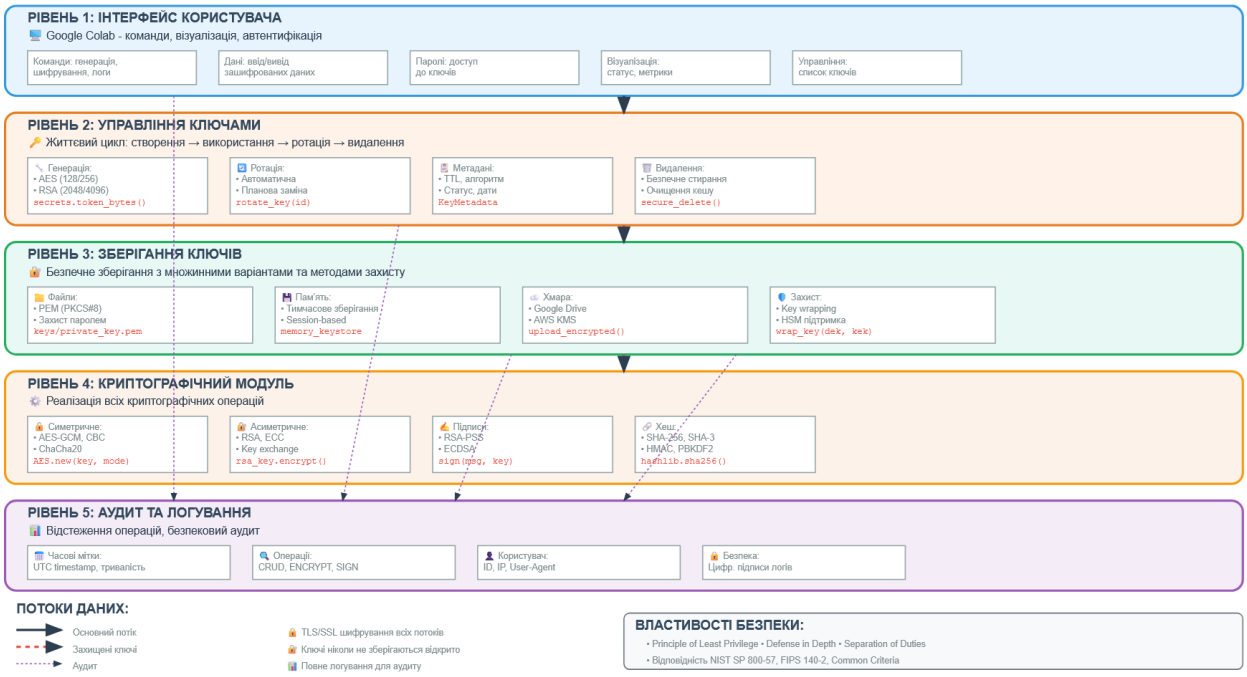

Рисунок 5.2 – Архітектура системи керування ключами

Перший рівень архітектури відповідає за інтерфейс користувача. У контексті реалізації в середовищі Google Colab цей рівень забезпечує взаємодію з системою через введення команд, візуальний контроль

виконання операцій, автентифікацію доступу та обробку зашифрованих або розшифрованих даних. Інтерфейс також дозволяє здійснювати базове адміністрування ключів і перегляд логів.

Другий рівень, що виконує роль модуля управління ключами, реалізує повний життєвий цикл ключів. Його функціональність охоплює генерацію симетричних і асиметричних ключів, підтримку їх ротації, встановлення параметрів часу життя, а також безпечне видалення після завершення використання. Цей модуль є центральним елементом, що координує всі ключові операції.

Третій рівень відповідає за зберігання ключів із застосуванням різних стратегій захисту. Залежно від вимог до безпеки та інфраструктури, ключі можуть зберігатися у зашифрованому вигляді в локальних файлах, оперативній пам’яті або у хмарних сервісах. Для підвищення захищеності використовуються механізми шифрування ключа з використанням іншого ключа (key-wrapping), а також автентифікація за паролем.

На четвертому рівні функціонує криптографічний модуль, що реалізує всі базові та розширені криптографічні операції. Серед них – шифрування й дешифрування даних, формування та перевірка цифрових підписів, генерація хешів, а також підтримка різних режимів блочного шифрування. Цей модуль напряму взаємодіє з ключами, що зберігаються на попередньому рівні.

Завершує архітектуру модуль логування та аудиту. Його призначення полягає в документуванні всіх операцій, що виконуються над ключами, із фіксацією часу дії, типу операції, параметрів сесії користувача та, за потреби, його мережевої ідентифікації. Наявність повноцінного аудиту є критичним чинником для забезпечення прозорості системи та дотримання вимог інформаційної безпеки.

Ця архітектура створює умови для ефективного й безпечного управління криптографічними ключами та може бути масштабована відповідно до потреб конкретної організації або проєкту.

```
# Відображення логів
generate_symmetric_key()
generate_rsa_keys()
encrypt_with_symmetric("тест")
def show_logs():
    with open("logs/actions.log", "r") as f:
        lines = f.readlines()
    df = pd.DataFrame([line.strip().split(" - ") for line in lines], columns=["Час", "Дія"])
    return df

show_logs()
```

```
Симетричний ключ збережено в keys/symmetric.key
RSA ключі збережено в keys/private_key.pem та keys/public_key.pem
```

	Час	Дія
0	2025-06-27 08:51:45.931425	Згенеровано симетричний ключ
1	2025-06-27 08:51:46.024335	Згенеровано RSA ключі
2	2025-06-27 08:51:46.025248	Зашифровано текст симетричним ключем

Рисунок 5.3 – Результати роботи

На рисунку 5.3 представлені результати роботи системи керування ключами.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи розроблено прототип системи керування криптографічними ключами з використанням мови програмування Python у середовищі Google Colab. У ході роботи було проведено теоретичний аналіз фундаментальних принципів криптографії, зокрема конфіденційності, цілісності, автентифікації та незаперечності, що дало змогу визначити основні вимоги до ефективної реалізації систем управління ключами.

Під час роботи було опрацьовано та оцінено сучасні криптографічні бібліотеки Python, серед яких було обрано найдоцільніші для практичного використання в рамках розробленої системи. На основі аналізу сформовано архітектуру, що охоплює модулі генерації ключів, їх безпечного зберігання, шифрування і дешифрування даних, логування операцій та візуалізації процесів. Реалізована система забезпечує основні функції криптографічного захисту, включаючи створення як симетричних, так і асиметричних ключів, здійснення криптографічних операцій над текстовими повідомленнями, а також ведення журналу подій для забезпечення аудиту та контролю доступу.

Окрему увагу було приділено питанням зручності та наочності використання системи. З цією метою до реалізації було інтегровано інтерактивні елементи, що дозволяють виконувати шифрування безпосередньо з інтерфейсу Colab, а також візуалізацію структури системи у вигляді графічної діаграми.

У підсумку можна стверджувати, що створена система є функціональним прототипом, який може бути застосований для навчальних або дослідницьких цілей, а також адаптований до практичного використання з розширенням функціональності. Подальші напрямки вдосконалення можуть включати реалізацію повноцінної багатокористувацької автентифікації, автоматизовану ротацію ключів, інтеграцію з зовнішніми системами та розгортання на хмарних платформах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography. CRC Press, 1996. 780 p.

2. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications. Wiley, 2010. 384 p.

3. Stallings W. Cryptography and Network Security: Principles and Practice. Pearson, 2017. 768 p.

4. Barker E., Roginsky A. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST Special Publication 800-131A, 2019. 112 p.

5. Cooper D., Santesson S., Farrell S., Boeyen S., Housley R., Polk W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, 2008. 151 p.

6. Housley R., Polk W., Ford W., Solo D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Internet Engineering Task Force, RFC 2459, 1999. P. 1-126.

7. Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2. Internet Engineering Task Force, RFC 5246, 2008. P. 1-104.

8. Kent S., Seo K. Security Architecture for the Internet Protocol. Internet Engineering Task Force, RFC 4301, 2005. P. 1-101.

9. Krawczyk H., Bellare M., Canetti R. HMAC: Keyed-Hashing for Message Authentication. Internet Engineering Task Force, RFC 2104, 1997. P. 1-11.

10. Rescorla E., Modadugu N. Datagram Transport Layer Security Version 1.2. Internet Engineering Task Force, RFC 6347, 2012. P. 1-32.