Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Біловол А.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*“*Інформаційні технології*”*_____
Спеціальність *125 “*Кібербезпека*”*
Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Біловолу Артему Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Біловол А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Біловол А.В. Програмне забезпечення системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування.

Метою розробки є програмне забезпечення системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування.

Результат роботи – програмна реалізація системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, комп'ютерні мережі критичного застосування

ABSTRACT

Bilovol A.V. Software for the cybersecurity system of information transmission in computer networks of critical application. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of information transmission in computer networks of critical application.

The purpose of the development is the software for the cybersecurity system of information transmission in computer networks of critical application.

The result of the work is the software implementation of the cybersecurity system of information transmission in computer networks of critical application.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in Python.

Keywords: cybersecurity, mission-critical computer networks

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ПЗ	–	програмне забезпечення
QoS	–	механізми контролю й керування якістю

ВСТУП

Актуальність теми. В Україні існує велика кількість галузей або виробництв, у яких висуваються критичні вимоги до систем передачі інформації. Це є і атомна енергетика, і аерокосмічна галузь, і транспортна галузь, і безліч інших галузей. У даній бакалаврській роботі за основу взята аерокосмічна галузь, і відповідно проводяться дослідження з розробки програмного забезпечення передачі інформації в комп'ютерних мережах критичного застосування в аерокосмічній галузі. Аналіз нештатних ситуацій для систем критичного застосування в аерокосмічній галузі показує, що більше 50% з них носять технічний характер (пов'язані з різними відмовами і несправностями технічних систем, з порушенням їх функціонування або руйнуванням внаслідок помилкових дій персоналу), причому більше 20% пов'язані з нестачею інформації про повітряну обстановку або несвоєчасним її отриманням. У ряді випадків (складні метеорологічні умови, наявність активних і пасивних перешкод тощо) для безпосереднього управління польотами авіації і забезпечення необхідної повноти інформації виникає необхідність в постачанні суміжним центрам управління повітряним рухом початкової (необробленої) радіолокаційної інформації (РЛІ), яка може бути класифікована як відеоінформація про повітряну обстановку. Інтенсивність інформаційних потоків при передачі цієї інформації для різних радіолокаційних станцій (РЛС) може складати від 1 до 3 Мбіт/с, що в 100 і більше разів перевищує інтенсивність обробленої радіолокаційної інформації, яка передається на даний час. Але рівень пропускної спроможності існуючих каналів зв'язку комп'ютерних мереж систем критичного застосування залишається незмінним. Виникає протиріччя між зростаючими об'ємами інформації, яка повинна передаватися в низькопродуктивних комп'ютерних мережах систем критичного застосування, і обмеженням на час її передачі.

До основних шляхів забезпечення необхідної своєчасності передачі цифрової інформації в комп'ютерних мережах систем критичного застосування при заданому жорстко регламентованому циклі безпосереднього управління польотами відносяться розробка і застосування методів динамічного розподілу мережевого ресурсу.

Аналіз існуючих способів і алгоритмів пошуку найкоротших шляхів передачі інформації і розподілу інформації по них, а також методів обробки даних показав, що вони недостатньо забезпечують передачу відеоінформації про повітряну обстановку в режимі реального часу. Це пов'язано з високою інерційністю процесів збіжності алгоритмів пошуку множини шляхів передачі інформації, низькою ефективністю використання ресурсів комп'ютерної мережі методами, які реалізовані на практиці, і відсутністю механізмів моніторингу якості інформації, що передається. Дане протиріччя можна розв'язати шляхом розробки методу забезпечення своєчасності передачі інформації в комп’ютерних мережах систем критичного застосування, який усуває вищенаведені недоліки.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем передачі інформації в комп’ютерних мережах критичного застосування.

– Дослідження системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

– Програмна реалізація системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі передачі інформації в комп’ютерних мережах критичного застосування.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням системи є реалізація програмного забезпечення передачі інформації в комп’ютерних мережах критичного застосування. Невід’ємною частиною сучасних інформаційно-управляючих систем є комп’ютерні мережі. Це стосується, у першу чергу, розподілених систем управління складними об’єктами та обробки інформації. Одночасно з розширенням меж застосування відкритих мережних технологій і стандартів збільшується і частка відмов інформаційно-управляючих систем, причиною яких є комп'ютерні мережі . Так, за статистикою консалтингової фірми Infonetics, відмови мережного обладнання та збої в роботі комп'ютерної мережі середньостатистичної північноамериканської фірми відбуваються 23,6 рази протягом року, причому 70% з них припадає на кабельну систему, а час, що витрачається на їхнє усунення, становить в середньому близько 5 годин. Матеріальні витрати на ліквідацію пошкоджень, ураховуючи втрачену вигоду та інші збитки, становлять від однієї до п'ятдесяти тисяч доларів за годину. Необхідність забезпечення надійної передачі інформації та задоволення постійно зростаючих потреб користувачів до якості наданих послуг обумовлює актуальність наукових досліджень, присвячених розробці й удосконаленню методів проектування, оптимізації, оцінки й забезпечення надійності мереж зв'язку і передачі даних. Значний внесок у розвиток цих наукових напрямків зробили такі вчені, як Л. Клейнрок, І. Фриш, М. Шварц, В.Г. Лазарєв, Ю.П. Зайченко, Г.П. Захаров, Л.Б. Богуславський. Однак, як свідчить аналіз, існуючі методи й моделі виявляються неефективними для оцінки надійності комп’ютерних мереж, заснованих на відкритих мережних технологіях і стандартах, оскільки не враховують особливості їх організації та побудови.

У той же час такі привабливі характеристики комп'ютерних мереж, заснованих на відкритих мережних технологіях і стандартах, як висока пропускна здатність, можливість масштабування, підтримання різних видів трафіку та багаторічний досвід комерційного використання, відкривають нові перспективи для впровадження цього класу мереж у розподілених інформаційно-управляючих систем критичного застосування, до яких належать авіаційні й ракетно-космічні комплекси, транспортні, енергетичні та інші системи. Особливістю інформаційно-управляючих систем критичного застосування є те, що для цього класу систем існують нормативні документи, що визначають жорсткі вимоги до надійності, безпеки та дотримання найбільш важливих принципів побудови (наприклад, принцип одиничної відмови, що визначає необхідність нейтралізації наслідків відмови будь-якого елемента програмно-апаратних засобів інформаційно-управляючих систем). Крім того, при проектуванні інформаційно-управляючих систем критичного застосування активно реалізують COTS-підхід (Commercial off the Shelf – комерційний продукт "з полки"), що припускає використання комерційного апаратного і програмного забезпечення, якщо воно пройшло відбір на відповідність регулятивним вимогам.

Для інформаційно-управляючих систем критичного застосування, що функціонують у агресивному середовищі, особливого значення набуває оцінка й забезпечення живучості. Причому для таких систем аналіз живучості комп'ютерних мереж може базуватися на виявленні найбільш критичних елементів з урахуванням різної кратності відмов внаслідок впливу екстремальних факторів.

1.2 Область застосування

Областю застосування розроблювальної системи є аерокосмічна галузь. України. країна входить до елітної дев'ятки країн, що мають замкнутий технологічний цикл створення і виробництва авіатехніки. Крім проектування і

виробництва пасажирських і транспортних літаків, в Україні є мережа авіаремонтних підприємств, в тому числі і для відновлення бойових літаків і вертольотів. У березні 2007 р. Кабінет міністрів створив державний авіабудівний концерн «Авіація України» (ДАКАУ), віднесений до управління Міністерства промислової політики. До складу концерну включені 10 держпідприємств: Авіаційний науково-технічний комплекс ім.. Антонова, Київський авіаційний завод «Авіант», Завод №410 цивільної авіації, Харківське державне авіаційне виробниче підприємство, ДП НІІ «Буран», ДП «Харківське агрегатне конструкторське бюро», Харківський машинобудівний завод «ФЕД», Запорізьке машинобудівне КБ «Прогрес» ім. академіка Івченка», підприємство «Новатор», казенне підприємство «Радіовимірювач». Концерн є державним господарським об'єднанням, діє на принципах повної господарської самостійності самоокупності, несе відповідальність за результати своєї господарської діяльності виконання зобов'язань. У статуті сказано, що концерн створено з метою об'єднання розробників і виробників авіаційної техніки в єдиний комплекс з централізованим управлінням. Саме централізація неефективної системи управління для її оптимізації, створення єдиного маркетингового і фінансового центру має допомогти сучасній українській авіаційній промисловості подолати ту складну ситуацію, в якій вона знаходиться.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Утиліта NetSpeedMonitor

NetSpeedMonitor – чудова програма, призначена винятково для моніторингу мережного трафіку, точніше, для визначення швидкості мережних з'єднань і сумарного обліку трафіку. Швидкість вхідних і вихідних потоків, відображається у вигляді іконки, розташованої на панелі завдань. Інтерфейс програми, налаштовується повністю. Існує можливість змінювати шрифт, вид оповіщень, або налаштовувати інші параметри за своїм розсудом.

Можливості програми, передбачають видачу щоденних і щомісячних, деталізованих звітів по трафіку. Установка додаткових драйверів при використанні даної програми не потрібно.

NetWorx 5.2.3.

NetWorx 5.2.3 – програма проводить облік трафіку, що як входить, так і вихідного. Допоможе не витратити весь відпущений ліміт і не залишитися взагалі без мережного підключення. Одна із кращих утиліт, у плані наявності безлічі різноманітних функцій. Здатна відображати три налаштовуються графіка швидкості: основний, на панелі завдань у треї. Так само, має можливість зчитувати трафік на різних мережних адаптерах і на окремих мережних інтерфейсах (Bluetooth, dial-up, провідні, бездротові й т.п.).

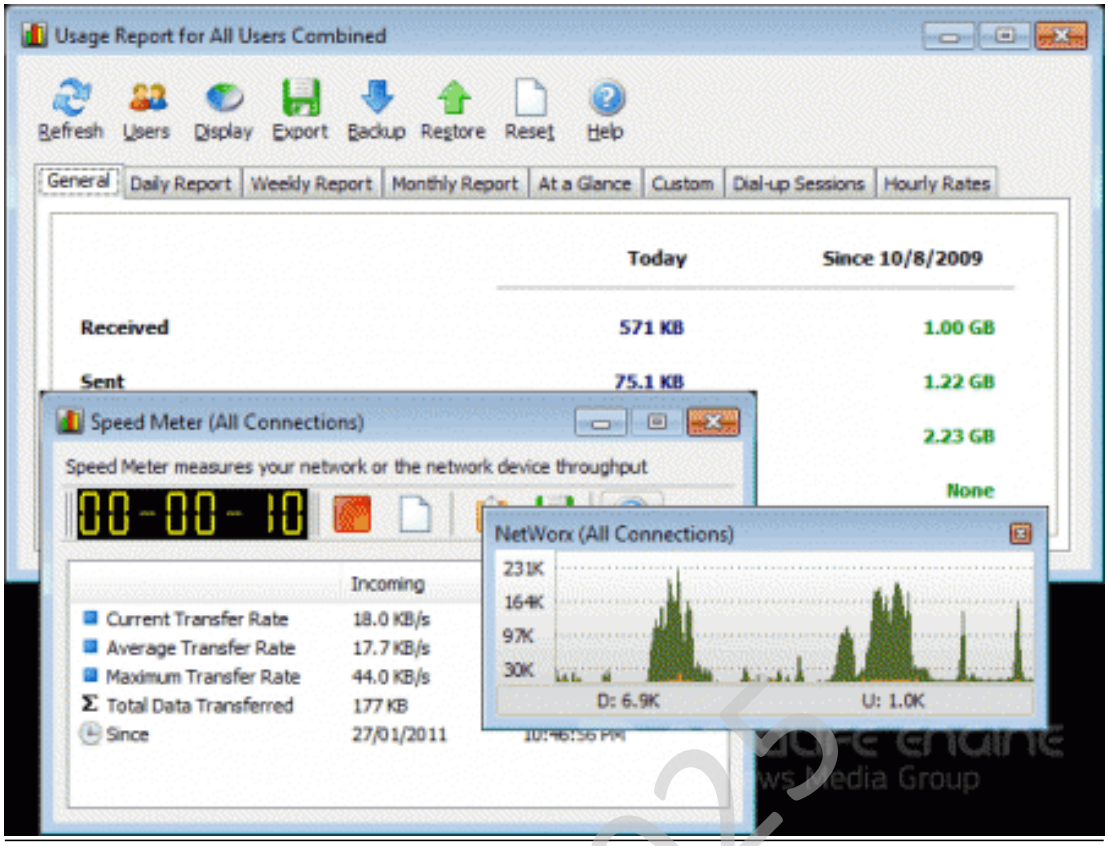


Рисунок 2.1 – Інтерфейс користувача NetWorx 5.2.3

Програма здатна працювати, ігноруючи активність у локальних мережах. Є функція наочного відображення статистики протягом певного періоду часу, або щодо окремо взятого користувача. За допомогою даної утиліти, можливий експорт вхідних даних у різні формати (RTF, XLS, HTML і ін.), а так само збереження їхньої резервної копії.

У підсумку, утиліта NetWorx 5.2.3 – це повнофункціональна програма моніторингу трафіку, що надає користувачеві широкий ряд можливостей, по багатобічній діагностиці мережного підключення, а так само по забезпеченню безпечного використання інтернет-з'єднання.

– Моніторинг трафіку у вигляді графіків, що налаштовуються.

– Вимір поточної швидкості передачі пакетів даних.

– Відображення загальної статистики трафіку, а так само статистики, окремо по кожному користувачі.

– Установка квоти (максимального обсягу переданих і отриманих даних) на трафік, функція корисна у випадку, коли не використовується безлімітний тариф на користування Інтернетом.

– Трасування маршруту передачі даних, дозволяє довідатися перелік проміжних серверів, використовуваних для з'єднання з певним ресурсом.

– Функція «Пінг», визначає швидкість відповіді на посланий запит.

– Інструмент «З'єднання» – дозволяє вести спостереження, за всіма додатками, які вимагають і здійснюють з'єднання з мережею інтернет.

Утиліта BitTally

Програма BitTally, дозволяє робити моніторинг і облік Інтернет трафіку, використовуючи будь-яку кількість мережних інтерфейсів. Містить у собі два модулі: сервер і клієнт, що працюють під керуванням ОС Windows 32-х розрядних версій.

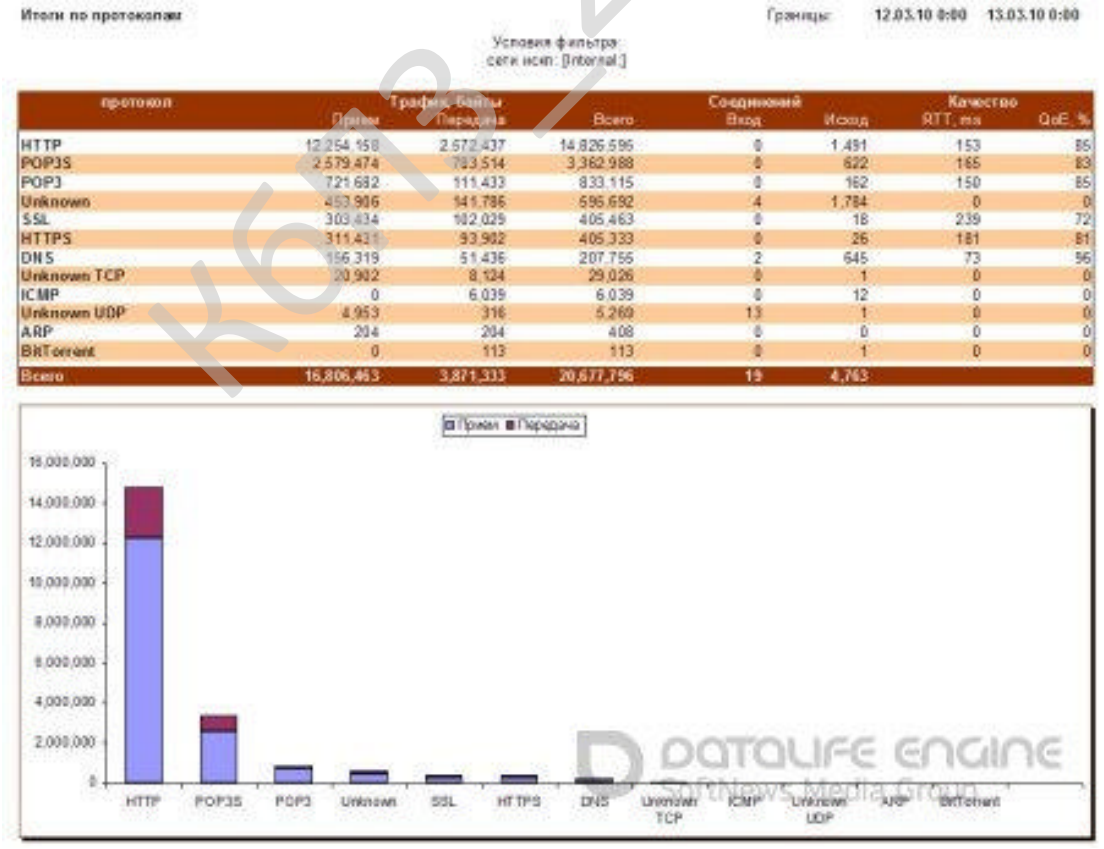


Рисунок 2.2 – Інтерфейс користувача BitTally

Основні можливості даного монітора:

– Створення щохвилинної статистики трафіку, протягом усього періоду роботи.

– Ведення історії всіх звертань до пошукових систем.

– Автоматична відповідна реакція на всі види мережної активності (наприклад, обіг по конкретних мережних протоколах, доменам і категоріям сайтів).

– Контроль трафіку у фоновому режимі, по будь-якій сукупності умов: кількість витраченого трафіку, типи протоколів, користувачі й т.д.

– Активний контроль сумарного трафіку, блокування або перенапрямок з'єднань, що забороняються.

– Установка будь-яких фільтрацій, при перегляді, а також, вивід звітів у файли або печатку.

– Широкий спектр настроювань html-шаблонів для сервера і їхній вивід у будь-якому зручному для користувача виді.

– Функція батьківського контролю, для блокування сайтів з певним умістом.

– Налаштовується функция, що, корпоративного контролю трафіку.

Монітор BitMeter 3.5.9

Не менш гідний представник, серед програм, що здійснюють моніторинг мережного трафіку, розповсюджуваний безкоштовно.

Графік вхідного й вихідного трафіку оформлений з усією повнотою й без надмірностей. Є настроювання для його розміру кольору й інших важливих дріб'язків. Ведеться статистика спожитого трафіку, виведена у вигляді таблиці або графіка. У принципі, функціонал програми обмежений лише веденням контролю над швидкістю з'єднання й над обсягом сумарного трафіку.

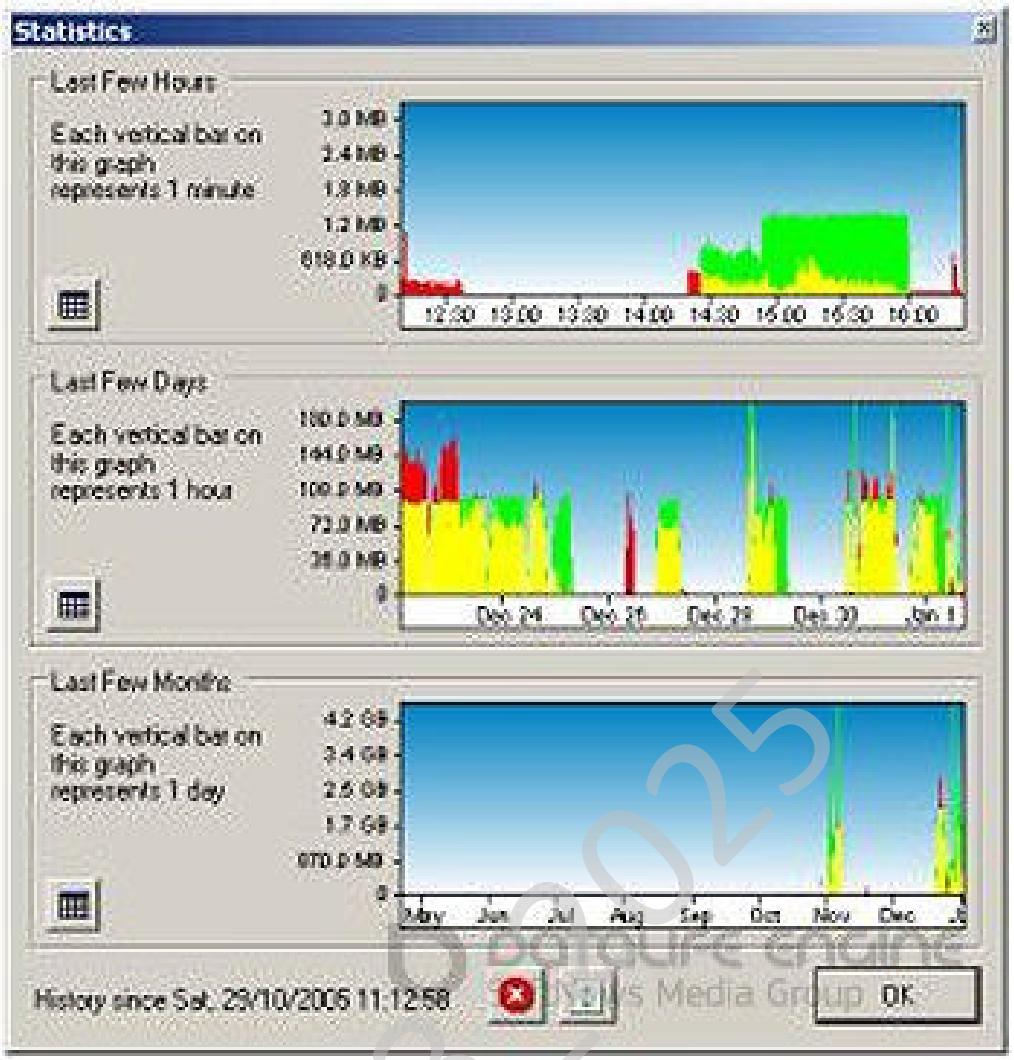


Рисунок 2.3 – Інтерфейс користувача BitMeter 3.5.9

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова

програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітних проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання

чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Під час виконання бакалаврської роботи було проведено аналіз системи управління авіаперевезеннями на загальнодержавному, регіональному та місцевому рівнях, означено місце та роль розподілу мережевого ресурсу в процесі забезпечення своєчасності передачі інформації в комп'ютерних мережах аерокосмічних систем критичного застосування, обґрунтовані вимоги до якості зв'язку при передачі інформації в комп'ютерних мережах систем критичного застосування та проведено аналіз відомих методів розподілу мережевого ресурсу, а також моделей комп'ютерної мережі. З'ясовано, що одним з основних параметрів якості зв'язку при передачі інформації в комп'ютерних мережах систем критичного застосування є своєчасність. Встановлено, що для забезпечення своєчасної передачі інформації про повітряну обстановку допустимий час $T_д$ доставки інформаційного пакету повинен бути не більш 0,5 с, а ймовірність ($Q^{(сч)}$) доставки інформаційного пакету до одержувача за час, який не перевищує допустиме значення, повинна бути не менш ніж 0,9.

Визначено, що найбільш ефективними та такими, які використовуються на практиці, є методи розподілу мережевого ресурсу на основі графових та потокових моделей комп'ютерної мережі.

Удосконалимо аналітичний вираз для розрахунку середнього часу $\bar{t}_{обс}^{(об)}$ обслуговування інформаційних пакетів в комп'ютерних мережах систем критичного застосування та визначено шляхи його зменшення. Розроблена математична модель процесу статистичного мультиплексування в комп'ютерних мережах систем критичного застосування. Підтверджено результат про те, що об'єднана система обслуговування є більш ефективною, ніж система з розділеними ресурсами (за критерієм мінімуму середнього часу доставки інформаційних пакетів). Кількісно оцінено виграш для конкретних розподілів і значень їх параметрів. Проведено

математичне моделювання процесу доставки інформації в комп'ютерних мережах систем критичного застосування за допомогою багатошляхової маршрутизації та визначена цільова функція задачі оптимізації.

В умовах циркуляції в комп'ютерних мережах систем критичного застосування однорідного трафіку задачу управління інформаційними потоками при комутації вирішують за допомогою відомих методів, але забезпечення якості зв'язку при передачі різнорідної інформації від декількох джерел представляється більш складною задачею, яка вимагає використання методу динамічного розподілу мережевого ресурсу. Припустимо, що на вхід мультиплексора надходять два потоки з різними статистичними характеристиками. Розподіли кількості вимог, що поступають в одиницю часу, описуються законами $P_1(k)$ і $P_2(k)$ з інтенсивностями λ_1 і λ_2; розподіли довжин інформаційних пакетів описуються законами $P_1(\ell_p)$ і $P_2(\ell_p)$ з середніми значеннями $\overline{\ell}_p^{(1)}$ і $\overline{\ell}_p^{(2)}$ відповідно. Пропускна спроможність вихідного каналу зв'язку обмежена фіксованою величиною ρ. Потоки можуть обслуговуватися двома мультиплексорами з пропускними спроможностями вихідних каналів ρ_1 і ρ_2, або одним мультиплексором із спільним ресурсом вихідного каналу $\rho_\Sigma = \rho_1 + \rho_2$.

Припустимо, що потоки описуються однаковими законами надходження вимог і законами розподілу довжин та розрізняються лише в перші моменти $(\lambda_1 \neq \lambda_2,\ \overline{\ell}_1 \neq \overline{\ell}_2)$, що є цілком типовим для термінальних ділянок мереж.

Вирішуючи задачу про доцільність об'єднаного або роздільного обслуговування двох статистично різних потоків, використаємо критерій мінімального середнього часу $T_{срд}$ доставки інформаційних пакетів. Припустимо, що на вхід системи обслуговування, яка моделює мультиплексор, надходять пуасоновські потоки з довільним законом розподілу довжин.

Доведено, що для системи, в якій потоки обслуговуються двома роздільними мультиплексорами середній час обслуговування інформаційних пакетів в цілому розраховується як:

$$\bar{t}_{обс}^{(P)} = \frac{\lambda_1}{\lambda_1 + \lambda_2} \cdot \bar{t}_{обс}^{(1)} + \frac{\lambda_2}{\lambda_1 + \lambda_2} \cdot \bar{t}_{обс}^{(2)}, \quad (3.1)$$

а для об'єднаної системи мультиплексування з пропускною спроможністю вихідного каналу ρ_Σ як:

$$\bar{t}_{обс}^{(об)} = t_k^{(об)} + \frac{\frac{\lambda_1}{\lambda^{(об)}} \ell_p^{(1)} + \frac{\lambda_2}{\lambda^{(об)}} \ell_p^{(2)}}{\rho_\Sigma - \left((\lambda_1 + \lambda_2) \cdot k_s^{(i)} \cdot \left(\frac{\lambda_1}{\lambda^{(об)}} \ell_p^{(1)} + \frac{\lambda_2}{\lambda^{(об)}} \ell_p^{(2)} \right) \right)}, \quad (3.2)$$

де $t_к^{(об)}$ – інтервал часу між моментами прийому інформаційного пакету в *i*- му мультиплексорі і його встановлення в чергу на подальшу передачу; λ_i – інтенсивність вхідного потоку інформації в *i*-й мультиплексор.

Визначено, що система мультиплексування із спільним канальним ресурсом при пуасонівському вхідному потоці й експоненціальному обслуговуванні (3.2) завжди краще (за критерієм мінімуму середнього часу доставки інформаційних пакетів), ніж система з роздільним обслуговуванням (3.1). На рисунку 3.1 наведені криві залежності $\Delta\bar{t}_{обс} = \bar{t}_{обс}^{(P)} - \bar{t}_{обс}^{(об)}$ з параметром сімейства $\bar{\ell}_p$ від завантаження системи. При розрахунках були прийняті наступні початкові дані: $\rho_\Sigma = 400$ Кбит / с; $\lambda_1 = \lambda_2$.

З рисунка 3.1 видно, що виграш в часі обслуговування зростає із збільшенням завантаження, і виявляється особливо суттєвим в режимі великих завантажень. Але для фіксованого ρ_Σ виграш в часі обслуговування падає із зменшенням довжини інформаційного пакету. Це пояснюється тим, що абсолютні значення $\bar{t}_{обс}^{(P)}$ і $\bar{t}_{обс}^{(об)}$ також стають незначними.

Таким чином, проведений аналіз показав, що об'єднана система обслуговування є більш ефективною, ніж система з розділеними ресурсами (за критерієм мінімального середнього часу доставки інформаційних пакетів). Представлені результати дозволяють кількісно оцінити отриманий виграш для конкретних розподілів і значень їх параметрів. Вдосконалення аналітичного виразу

дозволило до 4 разів підвищити точність оцінки середнього часу обслуговування інформаційних пакетів у вузлах зв'язку комп'ютерних мереж систем критичного застосування.

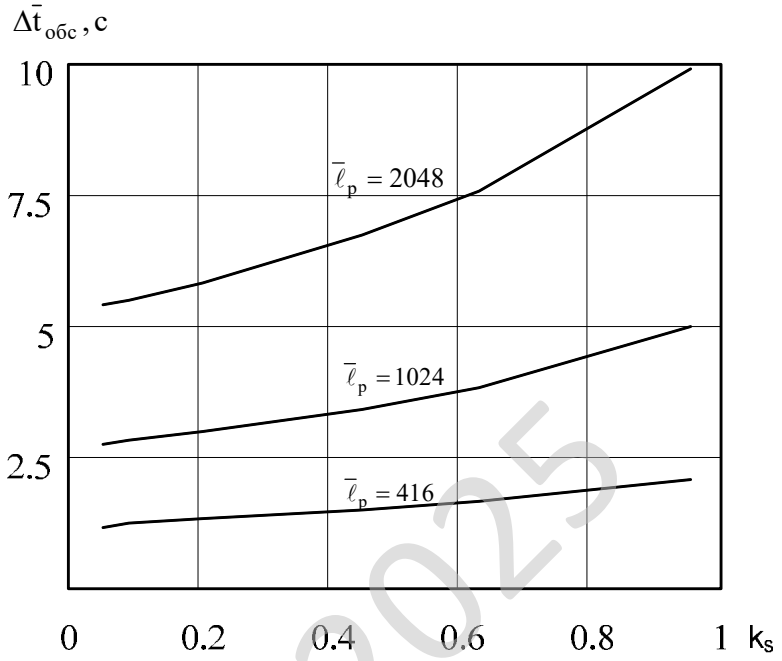


Рисунок 3.1 – Залежність $\Delta\bar{t}_{обс}$ від завантаження системи

Для визначення обмежень використання багатошляхової маршрутизації розроблена математична модель процесу доставки інформації в комп'ютерних мережах систем критичного застосування. Провівши перетворення Лапласа-стілтьєса ймовірність $Q^{(сч)}$ доставки інформації до одержувача за час, що не перевищує допустиме значення, розрахуємо як:

$$Q^{(сч)} = Q_{осн}^{(\mathrm{i,j})} \cdot P_{осн}^{(вм)} + Q_m^{(i,j)} \cdot \left(1 - P_{осн}^{(вм)}\right), m=M\text{-}1, \qquad (3.3)$$

де $Q_{осн}^{(\mathrm{i,j})} = \frac{1}{1+\gamma} \cdot \frac{1}{1+\mu} \cdot \frac{\rho_e - \lambda_{\mathrm{i,j}}}{\rho_e - \lambda_{\mathrm{i,j}} + \nu_e}$ – ймовірність доставки інформації до одержувача за час, якій не перевищує допустиме значення, за маршрутом, що складається тільки з

одного каналу зв'язку (i, j); $Q_m^{(i,j)} = \frac{1}{1+\gamma} \cdot \frac{1}{1+\mu} \cdot \left(\frac{e^{-\nu_e /(m\cdot\rho_э)} \left(1 - \frac{\lambda_m^{(i,j)}}{m \cdot \rho_e}\right)}{1 - \frac{\lambda_m^{(i,j)}}{\nu_e}\left(1 - e^{-\nu_e /(m\cdot\rho_e)}\right)} \right)^n$ –

ймовірність доставки інформації до одержувача за час, якій не перевищує допустиме значення, по іншим маршрутам, що складаються в середньому з n вузлів зв'язку в кожному; $p_{осн}^{(вм)}$ – ймовірність вибору основного маршруту; $\rho_e = \rho_s^{(c)} \cdot k_{\text{тот}}$ – експлуатаційна пропускна спроможність c-го каналу зв'язку s-го маршруту; $\lambda_{i,j} = k_{i,j} \cdot \lambda$ – інтенсивність потоку інформації в каналі зв'язку (i, j); $\nu_e = \nu \cdot \left(\frac{1 + \rho_e k_{\text{пр}}}{\nu k_{\text{тот}} + d} \right)$ – еквівалентна інтенсивність старіння; $\nu = 1 / T_{доп}$ – інтенсивність старіння; d – інтенсивність відновлення каналів зв'язку; $k_{гот}$ – коефіцієнт готовності каналу зв'язку; $\gamma = \nu / r$ – відносна інтенсивність старіння за інтенсивністю комутації; $\mu = \nu / z$ – відносна інтенсивність старіння за інтенсивністю розповсюдження; $k_{\text{пр}} = 1 - k_{\text{тот}}$ – коефіцієнт простою каналу зв'язку.

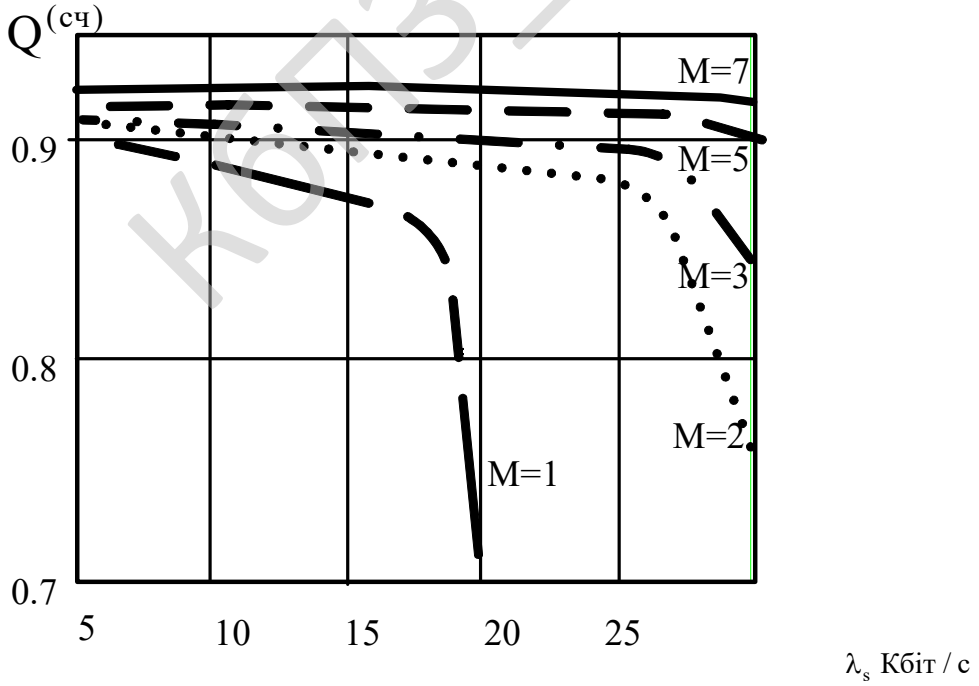


Рисунок 3.2 – Залежність ймовірності $Q^{(сч)}$ від кількості M маршруті передачі інформації та інтенсивності λ

Проведене моделювання процесу передачі інформації та аналіз його результатів показали зменшення ймовірності $Q^{(сч)}$ доставки інформації до одержувача за час, який не перевищує допустиме значення, при зменшенні кількості маршрутів передачі інформації та збільшенні інтенсивності потоку інформації в каналі зв'язку, що ілюструє рисунок 3.2. Таким чином, проведене моделювання визначило необхідність застосування багатошляхової маршрутизації при високому навантаженні на комп'ютерні мережі систем критичного застосування.

Аналіз виразу (3.2) та врахування того, що основним параметром, який впливає на своєчасність передачі цифрової інформації про повітряну обстановку, є середній час доставки інформаційного пакету, дозволяють вибрати цільову функцію завдання маршрутизації і розподілу інформаційних потоків у вигляді

$$\max_{s}\left\{k_s\sum_{c=1}^{\Psi_s}\left(\frac{\ell_p\cdot k_s^{(c)}}{k_s^{(c)}\cdot\rho_s^{(c)}-\lambda\cdot\ell_p\cdot k_s}\right)\right\}. \qquad (3.4)$$

Запропонуємо метод забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування.

Джерелом інформації в комп'ютерних мережах систем критичного застосування є *i*-й вузол зв'язку (ВЗ), щодо якого розглянуті такі множини: $U=\{u_\alpha \mid \aleph(u_\alpha)\subset\aleph\}$ – рівні ієрархії на дереві допустимих маршрутів; $\aleph_{баз}=\bigcup_{u_\alpha=1}^{|U|}\aleph(u_\alpha)$ – найдені шляхи передачі інформації, де u_α – номер рівня ієрархії. Формування множини $\aleph_{баз}$ для кожного вузла «*i*» є ітераційним процесом покрокового додавання вузлів з відповідним рівнем ієрархії із множини U, що збільшує число альтернативних шляхів передачі інформації з вузла «*i*» та створює передумову мінімізації середнього часу $T_{срд}$ доставки інформаційних пакетів в комп'ютерних мережах систем критичного застосування. Але збільшення числа рівнів ієрархії призводить до ускладнення алгоритмів розрахунку топології, що, в свою чергу, викликає зростання часу формування множини $\aleph_{баз}$. Тому після кожного кроку додавання вузлів відповідного рівня ієрархії доцільно перевіряти можливість забезпечення своєчасності передачі інформації знайденими на цьому кроці

маршрутами з урахуванням коефіцієнта $k_s^{(баз)} = (k_1^{(баз)},...,k_M^{(баз)})$. Це дозволить розподілити інформаційні пакети в залежності від «відстаней» $T_{i,j} = (T_{i,j}^{(1)},...,T_{i,j}^{(M)})$ кожного з маршрутів (i, j). Припущення, які були зроблені, дозволяють сформулювати оптимізаційну задачу пошуку значень $\aleph(u_\alpha)$:

$$U = \left\{ u_\alpha \mid \aleph(u_\alpha) \subset \aleph \right\} \to min; \tag{3.5}$$

$$\aleph_{баз} = \bigcup_{u_\alpha=1}^{|U|} \aleph(u_\alpha), \quad |U| \geq 1, \quad |U| < \max_{\eta_m \in \aleph} |\eta_m|; \tag{3.6}$$

$$k_s^{(баз)} = T_{i,j}^{(s)} / \sum_{s=1}^{M} T_{i,j}^{(s)}; \tag{3.7}$$

$$T_{\text{срд}}\left(\bigcup_{u_\alpha' \leq u_\alpha} \aleph(u_\alpha') \right) \leq T_{доп} \quad \forall u_\alpha \in U; \tag{3.8}$$

$$Q^{(сч)}\left(\bigcup_{u_\alpha' \leq u_\alpha} \aleph(u_\alpha') \right) \geq Q_{доп}. \tag{3.9}$$

Незважаючи на переваги (використання множини шляхів передачі інформації, пропорційний розподіл потоку інформації по каналах зв'язку), такий підхід до розподілу мережевого ресурсу має ряд недоліків, зокрема відсутність врахування ймовірності спотворення інформації на базовій множині маршрутів, що росте із збільшенням $|\aleph_{баз}|$ і структурних особливостей вибраних маршрутів. Для усунення вказаних недоліків на наступному етапі рішення задачі забезпечення своєчасності необхідно знайти оптимальну множину маршрутів передачі інформації.

Встановлено, що для каналу $c \in \Im$ маршруту $s \in \aleph_{баз}$ ймовірність спотворення одного біта рівно $q_s^{(c)}$, тобто ймовірність неспотворення біта дорівнює $p_s^{(c)} = 1 - q_s^{(c)}$. Тоді для пуассонівського потоку інформації інтенсивністю $\lambda \cdot k_s$ з пакетом довжиною ℓ_p, що проходить *c*-м каналом зв'язку *s*-го маршруту, за час Δt ймовірність неспотворення інформації дорівнює $p_s^{(c)}(\Delta t) = (1 - q_s^{(c)})^{\lambda k_s \ell_p \Delta t}$. Відповідно, ймовірність $p_s(\Delta t)$ неспотворення інформації при передачі її *s*-м маршрутом за час Δt дорівнює $p_s(\Delta t) = \prod_{c \in \eta_s} (1 - q_s^{(c)})^{\lambda k_s \ell_p \Delta t}$, тобто при передачі інформаційного потоку інтенсивністю λ з використанням багатошляхової маршрутизації на базовій

множині $\aleph_{баз}$ маршрутів ймовірність $p(\aleph_{баз}, \Delta t)$ неспотворення дорівнює $p(\aleph_{баз}, \Delta t) = \prod_{s \in \aleph_{баз}} \prod_{c \in \eta_s} (1 - q_s^{(c)})^{\lambda k_s \ell_p \Delta t}$ і, відповідно, ймовірність $q(\aleph_{баз}, \Delta t)$ спотворення за тих самих умов дорівнює

$$q(\aleph_{баз}, \Delta t) = 1 - \prod_{s \in \aleph_{баз}} \prod_{c \in \eta_s} (1 - q_s^{(c)})^{\lambda k_s \ell_p \Delta t}. \quad (3.10)$$

Враховуючи, що в запропонованому підході знаходження базової множини маршрутів при багатошляховій маршрутизації необхідно понизити рівень спотворення передачі інформації про повітряну обстановку, задача оптимізації розподілу потоку інтенсивністю λ в середовищі базової множини маршрутів $\aleph_{баз}$ формулюється таким чином:

$$q(\aleph_{баз}, \Delta t) \xrightarrow{\aleph_{вб}} min; \quad (3.11)$$

при

$$k_s = \frac{1 - t_s \Big/ \sum_{s \in \aleph_{вб}} t_s}{|\aleph_{вб}| - 1}, \quad \aleph_{вб} \subset \aleph_{баз}; \quad (3.12)$$

$$T_{срд}(\aleph_{баз}) \le T_{доп}; \quad (3.13)$$

$$Q^{(сч)}(\aleph_{баз}) \ge Q_{доп}; \quad (3.14)$$

$$p_s(\Delta t) \ge p_{доп}(\Delta t); \quad (3.15)$$

де $\aleph_{вб}$ – множина маршрутів передачі інформації, вибраних з множини $\aleph_{баз}$ для зменшення ймовірності $q(\aleph_{баз}, \Delta t)$; t_s – «відстань» від джерела до адресата на s-му маршруті; $p_{доп}(\Delta t)$ – допустима ймовірність неспотворення біта інформації переданого маршрутом за час Δt.

При незнаходженні жодного розподілу з множиною $\aleph_{баз}$, що задовольняє обмеженням (3.14) – (3.18), розширюємо $\aleph_{баз}$ шляхом його об'єднання з множиною маршрутів наступного рівня ієрархії відповідно до виразів (3.8) – (3.12).

3.2 Розробка структурної схеми

Структурна схема розробленої системи зображена на рисунку 3.3. Розглянемо її більш детально. Розроблене, у ході виконання бакалаврської роботи, програмне забезпечення відноситься до техніки зв'язку й може використовуватися як система зв'язку між комп'ютерною мережею в літальному апарату й комп'ютерною мережею на землі. Технічний результат складається в підвищенні ефективності використання каналів зв'язку. Для цього система містить засоби, виконані з можливістю встановлення мережного з'єднання між комп'ютерною мережею на землі й комп'ютерною мережею в літальному апарату через, щонайменше, один носій зв'язку в режимі синхронного зв'язку.

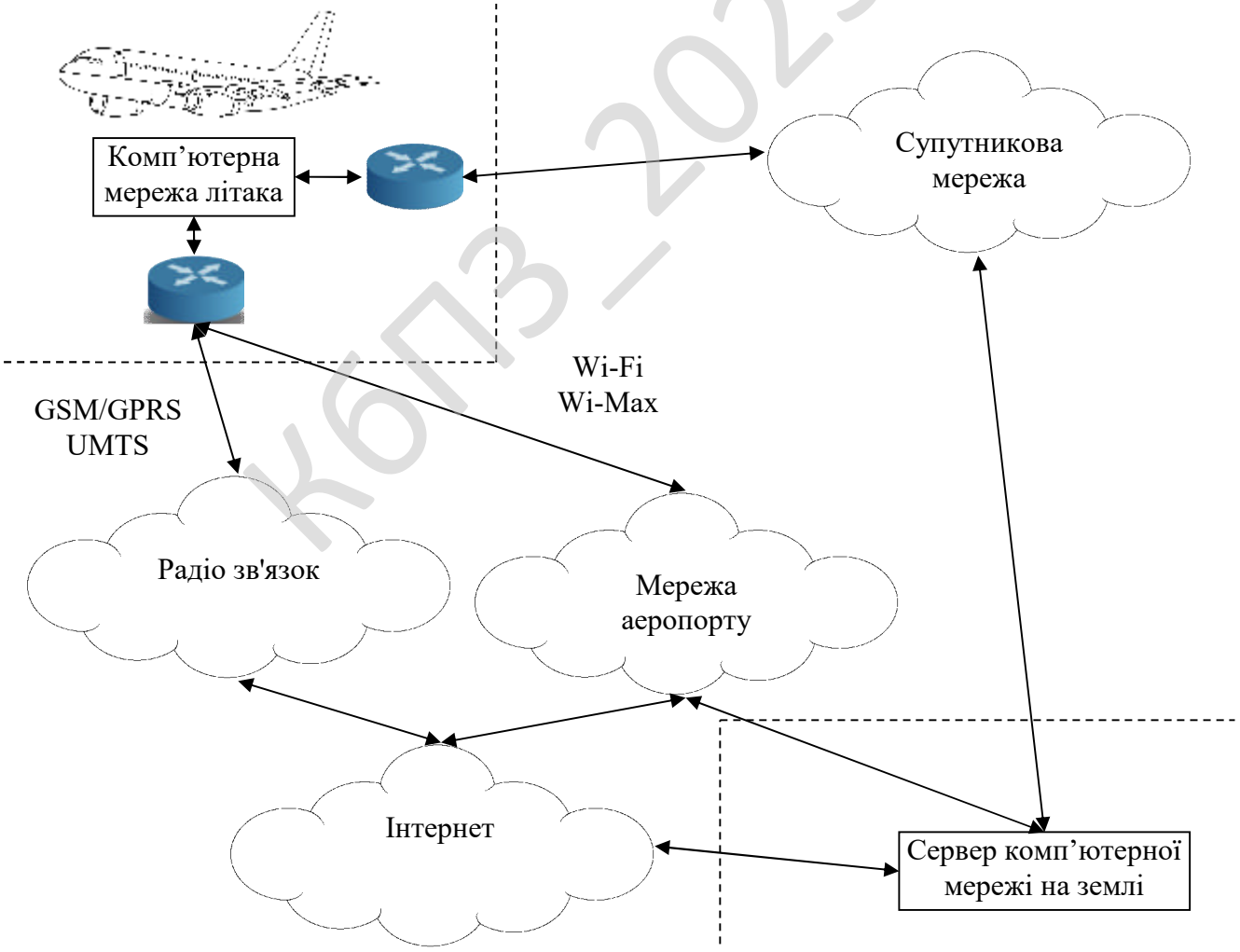


Рисунок 3.1 – Структурна схема системи

Система авіоніки в цей час містить сукупність бортових інструментів і баз даних, призначених для інформатизації інструментів, зокрема, інструментів діагностики, обслуговування, і документи, такі як інструкції з діагностики несправностей або інструкції для експлуатації літака. Інструменти використовуються в цей час, наприклад, прикладними програмами або базами даних.

В основному можна розглядати два основних типи обслуговування.

Насамперед, варто розглядати обслуговування, що виконують або на основній базі технічного обслуговування літака, або за межами цієї бази і яке включає операції, обмежені регулюванням, забезпеченням безпеки й підготовкою літака до польоту, називаною також диспетчеризацією, без затримки або в обмежений термін.

Потім варто розглядати обслуговування, виконуване на основній базі технічного обслуговування літака, де здійснюють додаткові операції обслуговування, такі як обслуговування, проведене через регулярні інтервали.

Операції обслуговування, виконувані в літаку й на наземній базі обслуговування відповідно до відомого рішення, виконують за допомогою системи, зокрема, центрального комп'ютера обслуговування («Central Maintenance Computer»), що збирає, узагальнює й виводить у вигляді звіту несправності змінних блоків LRU літака («Line Replacable Unit») з метою надання допомоги екіпажу й обслуговуючому персоналу в процесі обслуговування.

Несправності змінних блоків літака є об'єктом керування тривожною сигналізацією за допомогою комп'ютера.

Центральний комп'ютер обслуговування передає в компанію, що експлуатує літак, зокрема, у центр контролю обслуговування MCC (скорочення від «Maintenance Control Center») повідомлення обслуговування.

З комп'ютером керування тривожною сигналізацією з'єднаний екран для індикації несправностей змінних блоків літака.

Сукупність несправностей або подій, що відбуваються протягом одного циклу експлуатації літака, заноситься в бортовий журнал, називаний «logbook». Цей бортовий журнал літака ведеться або пілотами («technical logbook»), або екіпажем обслуговування салону («Cabin Logbook» в англосаксонській термінології).

Для цього екіпаж записує від руки виявлені несправності в бортовий журнал, а також польотні умови, у яких ці несправності відбулися.

Коли літак перебуває на землі, бортовий журнал вилучається в літаку й перевіряється на землі центром МСС контролю обслуговування. Після цього обслуговуючий технік піднімається на борт літака, щоб проаналізувати виявлені несправності й зробити діагностику.

Потім технік вертається на наземну базу обслуговування, щоб одержати процедуру по ізолюванню несправності.

Із цією інструкцією, називаної також TSM (скорочення від troubleshooting manuel») технік вертається на борт літака, щоб здійснити цю процедуру по ізолюванню несправностей.

По завершенню ізолювання несправностей технік вертається на наземну базу, щоб одержати інструкцію з ремонту й, якщо буде потреба, замовити запасну частину на складі запчастин.

Потім обслуговуючий технік знову вертається на літак, щоб здійснити процедуру ремонту.

Після цього роблять тести, щоб перевірити роботу після ремонту, і здійснюють процедуру приймання, що підтверджує готовність літака до польоту.

Нарешті, це приймання відзначають у бортовому журналі.

Як можна легко зрозуміти із усього вищесказаного, цей операційний цикл обслуговування вимагає більших витрат і приводить до значної затримки літака на землі.

Інше відоме рішення складається в збереженні в пам'яті в бортових запам'ятовувальних носіях (бази даних) сукупності процедур ізолювання

несправностей і сукупності процедур ремонту, що дозволяє уникнути переміщень обслуговуючого техніка між літаком і наземною базою обслуговування.

Разом з тим, сукупність процедур ізолювання несправностей і сукупність процедур ремонту являють собою величезний обсяг даних, що може становити трохи гігабайт інформації.

Крім того, всі інструменти, дані й документи необхідно регулярно обновляти, щоб екіпаж літака й, зокрема, пілот і обслуговуючий технік завжди мали під рукою останню версію інструментів і документів.

Для цього інструменти й документація завантажуються в комп'ютер або комп'ютери літака техніком, відповідальним за підтримку відновлень цих інструментів і документів (або за синхронізацію бортових баз даних, що містять ці документи, з наземними базами даних). Для цього технік має портативний комп'ютер, що містить у пам'яті останню версію інструментів і даних, і піднімається на борт літака, щоб зробити завантаження й відновлення інструментів і даних.

Однак, з огляду на, що ці інструменти й документація займають великий обсяг інформації, а саме трохи гігабайт, це відновлення забирає багато часу й приводить до відносно тривалої затримки літака на землі.

Це ж відбувається, коли технік використовує портативний комп'ютер з Wi-Fi, за допомогою якого він завантажує дані й обновляє інструменти й дані, що зберігаються в пам'яті в мережі літака, на підставі даних, завантажених у його портативний комп'ютер.

Крім того, авіаційна компанія звичайно експлуатує великий парк літаків, що відбивається на вартості обслуговування інструментів і документів літаків її парку, а також у керуванні конфігурацією великого обсягу даних на землі, які необхідно завантажити на борті літака.

Тому підтримка відновлення такого обсягу утруднено. Як результат, технік, що обслуговує,, опираючись на ці процедури, що зберігаються в пам'яті на літаку, може одержати інформацію, що стосується майбутніх процедур

ізолювання й ремонту, що може виявитися застарілої й навіть помилковою. Крім того, якщо дані за рішенням проблем перебувають на борті, це не звільняє обслуговуючого техніка від необхідності зв'язуватися зі складом запасних частин.

Програмне забезпечення розроблене, у ході виконання бакалаврської роботи, покликано усунути, щонайменше, один з недоліків відомих технологій і методів. Для цього розроблене, у ході виконання бакалаврської роботи, програмне забезпеченням дозволяє, зокрема, скоротити витрати по обслуговуванню, прискорити уведення в лад літака, обновляти дані й інструменти літального апарата в умовах захищеності без необхідності втручання техніка.

Відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, щонайменше, одна система авіоніки зв'язана в режимі реального часу й безупинно з наземною інфраструктурою. Наземна інфраструктура й бортова система авіоніки спільно використовують, щонайменше, один інформативний інструмент. Цей інструмент забезпечує доступ у наземну систему, бортову систему й у комп'ютерній мережі й дозволяє робити дії на відстані між землею й бортом літака. Його може використовувати єдиний оператор, що перебуває в певнім фіксованому місці.

Операцію обслуговування й одночасне відновлення баз даних можна здійснювати синхронно й з координацією всього за один раз, завдяки використанню загального інформативного інструмента. Операція обслуговування може включати тест і звертання до документації літака.

Система передбачає одну операцію обслуговування, скоординовану в режимі реального часу між землею й бортом літака. Вона передбачає одну послідовність операцій обслуговування, скоординованих у режимі реального часу між землею й бортом літака, і дозволяє ідентифікувати, ремонтувати й стежити (відслідковувати) у базах даних здійснювані дії.

Зв'язок може здійснюватися, наприклад, по захищеному протоколу IP. Координація й синхронізація баз даних відбувається в режимі реального часу.

Літак може перебувати в польоті, і оператор на землі може робити тестування системи під час польоту.

Альтернативно літак може перебувати на землі, і оператор може перебувати на борті літака або на землі в центрі обслуговування.

Розроблене, у ході виконання бакалаврської роботи, програмне забезпечення надає систему зв'язку між комп'ютерною мережею в літальному апарату й комп'ютерній мережі на землі за рахунок установлення мережного з'єднання в режимі синхронного зв'язку, щоб створити безперервність зв'язку комп’ютерної мережі літального апарата з комп'ютерною мережею на землі.

Крім того, ця система дозволяє робити відновлення даних, що зберігаються в пам'яті мережі літального апарата, для мережі комп'ютерів на землі й навпаки.

Крім того, відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, можна здійснювати інтерактивну навігацію в даних, що зберігаються в пам'яті в наземній інфраструктурі, а також у сайтах документів, що містять, наприклад, документацію літака (TSM або іншу).

При цьому немає необхідності в перевірці або в операції синхронізації баз даних між землею й бортом літака. Система забезпечує можливість виконання операцій на землі з борта літака (спільно використовувані інструменти земля/борт) або на борті літака із землі, завдяки синхронному зв'язку.

Середовищем зв'язку є, наприклад, комп'ютерна мережа мобільної телефонії, комп'ютерна мережа бездротового зв'язку, комп'ютерна мережа супутникового зв'язку й/або лінія дротовий зв'язка.

Відповідно до ознаки, система містить засоби відновлення даних, що зберігаються в пам'яті мережі комп'ютерів літального апарата на підставі даних, що зберігаються в пам'яті мережі комп'ютерів на землі.

Відповідно до іншої ознаки, система містить засоби передачі даних, що зберігаються в пам'яті мережі комп'ютерів літального апарата, у комп'ютерній мережі на землі.

Відповідно до варіанта виконання, комп'ютерна мережа в літальному апарату й комп'ютерній мережі на землі з'єднані через віртуальну приватну комп'ютерна мережу.

Об'єктом програмного забезпечення розробленого, у ході виконання бакалаврської роботи, є також комп'ютерна мережа літального апарата, що відрізняється тим, що містить засоби, виконані з можливістю встановлення мережного з'єднання з комп'ютерною мережею на землі через, щонайменше, одне середовище зв'язку в режимі синхронного зв'язку.

Цей пристрій має ті ж переваги, що й коротко описана вище система зв'язку.

Об'єктом програмного забезпечення розробленого, у ході виконання бакалаврської роботи, є також комп'ютерна мережа на землі, що відрізняється тим, що містить засоби, виконані з можливістю встановлення мережного з'єднання з комп'ютерною мережею літального апарата через, щонайменше, одне середовище зв'язку в режимі синхронного зв'язку.

Цей пристрій має ті ж переваги, що й коротко описана вище система зв'язку.

Інші переваги, завдання й відмітні ознаки програмного забезпечення розробленого, у ході виконання бакалаврської роботи, будуть більше очевидні з нижченаведеного докладного опису.

Відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, на борту літака встановлена електронна система обслуговування, виконана з можливістю здійснення операцій обслуговування, зокрема, призначена замінити паперовий процес електронним процесом.

Ця система обслуговування заснована на бортовій інфраструктурі літака, тобто на системі авіоніки, що містить, зокрема, сукупність функціональних блоків літака, наприклад, змінних блоків літака, що містять додатки для екіпажа й для обслуговування, на наземній інфраструктурі для підготовки, індивідуалізації й керування даними, які повинні використовуватися на борті, наприклад, для

здійснення операцій обслуговування або для одержання даних з літака для їхнього використання на землі, і на інфраструктурі з'єднання для обміну даними між наземною інфраструктурою й бортовою інфраструктурою й для відновлення інструментів і даних, що зберігаються в пам'яті в бортовій інфраструктурі.

Наземна інфраструктура перебуває, наприклад, на базі обслуговування авіаційної компанії, що експлуатує літак.

Ця наземна інфраструктура містить, зокрема, сукупність блоків обробки, з'єднаних один з одним через телекомунікаційну комп'ютерна мережу. Ця комп'ютерна мережа містить також з'єднання, наприклад, типу Інтернету для з'єднання із серверами заводів-виробників або з будь-якою третьою особою.

Наземна інфраструктура з'єднана також через комп'ютерну мережу зв'язку (інфраструктура з'єднання) з комп'ютерною мережею авіоніки літаків. Комп'ютерна мережа зв'язку заснована, наприклад, на середовищі бездротового зв'язку, наприклад, Wi-Fi або Wi-Max, на середовищі мобільного телефонного зв'язку, наприклад, GSM/GPRS або UMTS або на середовищі супутникового зв'язку. Крім того, літак може з'єднуватися із землею через дротовий зв'язок у випадку здійснення ремонту при неприступності радіозв'язку.

Так, комп'ютерна мережа наземної інфраструктури містить, зокрема, сервер, виконаний з можливістю передачі даних на літак і прийому даних від літака по супутниковому зв'язку, і сервер, виконаний з можливістю передачі даних на літак і прийому даних від літака з використанням середовища бездротового зв'язку або мобільного телефонного зв'язку.

Крім того, можна використовувати портативний носій, такий як портативний комп'ютер, ключ USB («Universal Serial Bus»), CD/DVD для обміну даними з літаком.

Відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, інфраструктура літака є мобільною комп'ютерною мережею, виконаною з можливістю встановлення зв'язку з наземною

інфраструктурою авіакомпанії таким чином, щоб забезпечувати безперервність між бортовою інфраструктурою й наземною інфраструктурою.

Згідно із одним із варіантів виконання, бортова інфраструктура зв’язується з наземною інфраструктурою в режимі синхронного зв'язку, причому цей тип зв'язку дозволяє вести інтерактивну навігацію в сайтах документів, що містять, наприклад, документацію літака.

Синхронний зв'язок складається у встановленні зв'язку або каналу зв'язку між системою авіоніки й наземною інфраструктурою, спеціально призначеного для зв'язку між ними, тобто він є вільним, коли, наприклад, необхідно звернутися до даних у наземній інфраструктурі з борта літального апарата або одержати інформацію, що зберігається в пам'яті в наземній інфраструктурі.

Таким чином, немає необхідності у встановленні зв'язку або каналу зв'язку щораз, коли необхідно здійснити зв'язок.

Отже, зв'язок між літальному апаратом і наземною інфраструктурою надійно забезпечений, оскільки не залежить від зайнятості або незайнятості каналу зв'язку.

Оскільки інфраструктура літака стає нерозривним продовженням наземної інфраструктури, можна робити відновлення й операції обслуговування синхронно між землею й бортом літака.

Крім того, зв'язок можна ініціювати через бортову інфраструктуру або через наземну інфраструктуру.

Відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, комп'ютерна мережа зв'язку, що з'єднує бортову інфраструктуру літака й наземну інфраструктуру, дозволяє відмовитися від установки всіх програмних інструментів і даних на борті літака, а встановлювати тільки основні інструменти, при цьому інші дані можна одержувати через з'єднання, коли це необхідно. Таким чином, технік у літаку, що обслуговує, може одержати доступ до даних, що зберігається в пам'яті в наземній інфраструктурі й

які дозволяють йому робити операції обслуговування без переміщень між літаком і базою обслуговування.

Крім того, технік у літаку, що обслуговує, може робити відновлення інструментів і даних, що зберігаються в пам'яті в інфраструктурі літака.

Крім того, технік, що обслуговує, може робити відновлення інструментів і даних у літаку із землі в ході операції, називаної дистанційним відновленням («remote update»). Наприклад, технік, що обслуговує, може обновляти зміст бортового журналу літака після обслуговування.

Точно так само, пілот або обслуговуючий оператор може звернутися в наземні сервери в режимі реального часу, щоб одержати доступ до сукупності серверів компанії, що експлуатує літак, і одночасно обновити дані й інструменти на борті за допомогою операцій, які теж називаються дистанційними («remote operations» в англосаксонській термінології).

Нарешті, технік на землі може здійснити тести на системі авіоніки до виконання операцій обслуговування шляхом передачі команд через комп'ютерну мережа зв'язку. Таким чином, технік, що обслуговує, може, наприклад, ще до посадки літака здійснити тести з метою ідентифікації несправних змінних блоків літака.

Згідно із одним із варіантів виконання, у середовищі зв'язку між бортовою інфраструктурою й наземною інфраструктурою, зокрема, у бездротовій мережі або в мережі мобільної телефонії створюють протокол інкапсуляції, називаний також туннелізацією («tunneling»), що можуть інкапсулювати передані дані в зашифрованому виді. Цю створювану комп'ютерна мережу називають віртуальною приватною комп'ютерною мережею (позначуваної RPV або VPN від «Virtual Private Network» в англосаксонській термінології). Цю комп'ютерна мережу називають віртуальною, тому що вона з'єднує дві фізичні мережі за допомогою не обов'язково надійного середовища зв'язку, і приватної, тому що доступ до даних можуть одержувати тільки комп'ютери мереж із двох сторін віртуальної приватної мережі. Крім того, вона забезпечує захист обмінів на не обов'язково надійному середовищі зв'язку.

Таким чином, при менших витратах створюють захищену лінію зв'язку.

Існує наступний можливий варіант застосування цієї системи відповідно до сьогодення, з розробленим у ході виконання бакалаврської роботи, програмним забезпеченням.

Відповідно до цього варіанта застосування, сервер авіаційної компанії за межами літака, у цьому випадку на землі, з'єднаний із сервером зв'язку бортової інфраструктури літака через віртуальну комп'ютерна мережу. Сервер літака містить мережний сервер ANSU («Aircraft Network Server Unit»), теж з'єднаний із сервером зв'язку.

Із сервером ANSU з'єднані, зокрема, блок інтерфейсу сервера, різні бортові термінали за допомогою електронного мережного блоку маршрутизації ESU («Ethernet Switch Unit»).

Згідно із одним із варіантів виконання винаходу, електронний блок зберігання інформації з'єднаний з комп'ютерною мережею супутникового зв'язку типу Satcom, що, у свою чергу, може бути з'єднана із сервером авіаційної компанії.

Сервер зв'язку виконаний з можливістю з'єднання через мережне з'єднання, наприклад, через віртуальну приватну комп'ютерна мережу, із сервером авіаційної компанії з використанням різних середовищ зв'язку, зокрема, мережі мобільної телефонії, наприклад, мережі GSM («Global System for Mobile Communication» в англосаксонській термінології)/EDGE/UMTS («Universal Mobile Telecommunications System»)/HSDPA («High Speed Downlink Packet Access») або мережі бездротового зв'язку, наприклад, мережі Wi-Fi 802.11 a/b/g або мережі супутникового зв'язку, наприклад, мережі HSD («High Speed Data Satcom»).

Таким чином, комп'ютерна мережа літака з'єднана з наземною комп'ютерною мережею авіаційної компанії, що експлуатує літак.

Під час установлення мережного з'єднання між комп'ютерною мережею літака й комп'ютерною мережею на землі середовище вибирають із безлічі

наявностей, що є в наявності зв'язку, зокрема, залежно від незайнятості середовищ зв'язку або від швидкості передачі інформації середовищ зв'язку.

Сервери й роблять інкапсуляцію й декапсуляцію даних через механізми шифрування й кодування.

Ці середовища зв'язку виконані з можливістю підвищеної швидкості передачі, щоб забезпечувати передачу більших мас даних між наземною інфраструктурою й бортовою інфраструктурою літака за прийнятний час і, зокрема, щоб дозволяти робити завантаження останніх версій інструментів, документів і даних з наземної інфраструктури авіаційної компанії в комп'ютери літака, причому операцію завантаження може робити по команді технік на борті літака або технік на землі з наземної інфраструктури.

Обслуговуючий технік на борті літака може також одержувати доступ до даних обслуговування й до центральних інструментів керування інформацією авіаційної компанії («Maintenance information server» або «Flight Ops Information server»), що зберігається в пам'яті в наземній інфраструктурі.

Крім того, цей тип з'єднання, завдяки з'єднанням Інтернет, дозволяє, з літака, входити в сервери, з'єднані з наземною інфраструктурою авіакомпанії, такі як сервер виробника літака або певного основного встаткування літака або його салону.

Крім того, відповідно до цієї архітектури, технік, що обслуговує, на борті літака може одержувати доступ до постачальників, наприклад, щоб звернутися до польотних даних або документації обслуговування або щоб зв'язатися із сервісними підприємствами на землі, які підтримують операції обслуговування літака.

За допомогою такої архітектури обслуговування літака, що складається в усуненні несправностей, підтримці літака в гарному польотному стані й у ремонті літака, здійснюють у самі короткі строки й найбільше оптимально, тому що всі наземні інструменти обслуговування літака обновляються, зокрема, у момент видачі дозволу на диспетчеризацію літака.

Крім того, відповідно до програмного забезпечення розробленого, у ході виконання бакалаврської роботи, електронне обслуговування дозволяє усувати несправності й підтримувати літак у гарному польотному стані в будь-який момент і незалежно від його місцезнаходження.

Для цього в літак завантажують мінімум даних інформації, таких як інструмент діагностики, електронний бортовий журнал, мінімальний список устаткування MEL («Minimum Equipment List»), або навіть частину цих даних.

Потім, через комп'ютерну мережу зв'язку обслуговуючий технік на борті літака за допомогою з'єднання, називаного дистанційним з'єднанням («remote access»), зокрема, захищеного з'єднання одержує доступ до даних, наявним у наземній інфраструктурі компанії, таким як посібник з ремонту TSM, посібник з обслуговування AMM (скорочення від «Aircraft Maintenance Manuel») або каталог IPC (скорочення від «Identification Part Catalogue» в англо-саксонської термінології), що дозволяє ідентифікувати номер деталі, яку треба замінити, і замовити її на складі запчастин.

Таким чином, через комп’ютерну мережу зв’язку, зокрема, використовуючи захищений канал типу VPN, технік одержує доступ до керівництва, що зберігається в пам'яті в наземній інфраструктурі, причому ці керівництва представлені своїми останніми версіями, як показано на фіг.4, не прибігаючи до переміщень між літаком і наземною інфраструктурою обслуговування.

Існує варіант, коли технік на борті літака шляхом дистанційних команд, зокрема, команд по консультації одержує доступ до процедури ізолювання діагністованої несправності, називаної також збоєм у роботі, а також до процедури ремонту ізольованої несправності й, якщо буде потреба, до складу запчастин через середовище зв'язку.

Згідно із одним із варіантів виконання, це мережне з'єднання є з'єднанням синхронного зв'язку.

Відповідно до іншого варіанта виконання, перед прибуттям літака на аеродром технік на землі може передати команди через комп'ютерну мережу зв'язку на бортову інфраструктуру, щоб зробити певне число тестів з метою діагностики, ізолювання й наступного усунення несправностей.

Відповідно до варіанта виконання, інструменти, зокрема, інструменти діагностики, і дані можуть бути завантажені в бортову інфраструктуру літака через комп’ютерну мережу зв’язку, що виконана з можливістю здійснення обмінів між бортовою інфраструктурою й наземною інфраструктурою за допомогою високошвидкісного засобу зв'язку.

Для цього комп'ютерна мережа зв'язку можна виконати з можливістю встановлення зв'язку між сервером зв'язку й сервером компанії через комп’ютерну мережу мобільної телефонії й/або через комп’ютерну мережу бездротового зв'язку, зокрема, з використанням захищеного каналу типу VPN.

Відповідно до варіанта сценарію, наявності несправності встаткування довідаються, завдяки збереженню несправності в бортовому журналі (logbook). Оператор на землі зв'язується з літаком із центра обслуговування (MCC) на землі.

Якщо в результаті випробування виявляється, що несправністю встаткування є «spurious message» (помилкове повідомлення), оператор може, перебуваючи у своєму службовому приміщенні, вирішити, що встаткування є робочим і послати статус «ОК» на борт літака (відновлення бортової бази даних) одночасно з відновленням наземної бази даних.

Існує тільки один інструмент земля/борт, що забезпечує обслуговування літака. Мова йде про інструмент, якому можна використовувати з борта літака або із центра обслуговування.

Існує ще варіант архітектури застосування сервера зв'язку в літаку, виконаного з можливістю встановлення зв'язку через комп’ютерну мережу мобільної телефонії або через комп’ютерну мережу бездротового зв'язку.

Сервер зв'язку містить модуль бездротового зв'язку TWLU («Terminal Wireless LAN Unit»), виконаний з можливістю встановлення зв'язку, наприклад,

відповідно до стандартів Wi-Fi a/b/g або Wi-Max, і модуль мобільної телефонії, такий як модуль GSM/GPRS або UMTS, причому обоє ці модуля з'єднані з модулем-тріплексором, з'єднаним з антеною.

У модулі мобільної телефонії встановлена експлуатаційна система, у якій є присутнім маршрутизатор, виконаний з можливістю маршрутизації зв'язку або в напрямку модуля бездротового зв'язку TWLU, або прямо в напрямку модуля-тріплексора, щоб використовувати протокол мобільної телефонії.

Зв'язком сервера літака із сервером авіаційної компанії управляє модуль VPN.

Крім того, на вході модуля VPN між даними, що надходять від мережного сервера ANSU, і модулем VPN установлений захисний модуль («firewall») для захисту сервера від проникнення.

Розглянемо наступний варіант установлення зв'язку між комп'ютерною мережею, що утворить, щонайменше, частину бортової інфраструктури літака, і комп'ютерною мережею, що утворить, щонайменше, частину наземної інфраструктури авіаційної компанії, відповідно до програмного забезпечення, розробленим у ході виконання бакалаврської роботи, заснований на архітектурі утримуючий бездротовий зв'язок і мобільний телефонний зв'язок.

Як було зазначено вище, на літаку встановлений сервер ANSU і сервер зв'язку, що містить у даному прикладі модуль бездротового зв'язку TWLU і модуль мобільної телефонії.

Що стосується мережі авіаційної компанії, з якої встановиться зв'язок сервер літака, то вона містить проксі-сервер («proxy server», називаний також «уповноваженим сервером») типу RADIUS («Remote Authentification Dial-In User Service»), виконаний з можливістю прийому й передачі запитів і даних через антену 610.

Проксі-сервер є машиною, що виконує функцію посередника між комп'ютерами локальної мережі авіаційної компанії й другою комп'ютерною мережею, комп'ютерною мережею літака.

Проксі-сервер з'єднаний через локальну комп'ютерна мережу з іншими серверами RADIUS. Дійсно, необхідно відзначити, що сервер RADIUS може виконувати функцію вповноваженого посередника, тобто передавати запити від клієнта на інші сервери RADIUS.

Сервер RADIUS дозволяє встановлювати зв'язок між функціями ідентифікації й базою користувачів, забезпечуючи стандартизовану передачу даних по автентифікації.

Щоб здійснювати обміни даними між сервером літака й локальною комп'ютерною мережею авіаційної компанії, сервер ANSU створює сертифікат літака й передає його на модуль бездротового зв'язку через модуль мобільної телефонії, як було зазначено вище.

Модуль бездротового зв'язку видає запит у локальну комп'ютерна мережу авіаційної компанії по протоколі EAP-TLS («Extensible Authentication Protocol – Transport Layer Security» в англосаксонській термінології), щоб зробити обмін сертифікатами й створити, таким чином, захищений тунель між комп'ютерною мережею літака й локальною комп'ютерною мережею авіаційної компанії. Створена в такий спосіб комп'ютерна мережа є віртуальною приватною комп'ютерною мережею.

Для цього протокол EAP-TLS використовує два сертифікати для створення захищеного тунелю, що потім забезпечує ідентифікацію: з боку сервера й з боку клієнта.

Цей протокол використовує інфраструктуру з відкритими ключами («Public Key Infrastructure») для захисту повідомлень ідентифікації між клієнтами, а саме між серверами літаків авіаційної компанії й серверами RADIUS авіаційної компанії.

Після цього ідентифікацію роблять, зокрема, шляхом передачі запиту типу DHCP («Dynamic Host Configuration Protocol») на проксі-сервер локальної мережі авіаційної компанії, щоб себе ідентифікувати.

Розглянемо різні віртуальні приватні мережі, які можуть бути створені між комп'ютерною мережею літака й комп'ютерною мережею на землі, зокрема, комп'ютерною мережею авіаційної компанії.

Показано створення віртуальної приватної мережі на основі середовища мобільного телефонного зв'язку, а саме мережі GSM/GPRS або UMTS. Разом з тим можна використовувати будь-який тип мережі мобільної телефонії як середовище зв'язку для віртуальної приватної мережі відповідно до програмного забезпечення, розробленим у ході виконання бакалаврської роботи.

Цей тип віртуальної приватної мережі, що забезпечує зв'язок мережі комп'ютерів літака з наземною комп'ютерною мережею, реалізують через провайдеру мережі радіозв'язку в пакетному режимі й комп'ютерній мережі Інтернет або приватну локальну комп'ютерна мережу.

Крім того, можливе створення віртуальної приватної мережі на основі середовища бездротового зв'язку, наприклад, мережі Wi-Fi або Wi-Max, що є комп'ютерною мережею аеропорту. Цю віртуальну приватну комп'ютерна мережу реалізують також через комп’ютерну мережу Інтернет або приватну локальну комп'ютерна мережу.

Крім того, віртуальну приватну комп'ютерна мережу можна створювати між комп'ютерною мережею літака й наземною комп'ютерною мережею, коли літак перебуває в польоті, зокрема, використовуючи супутниковий зв'язок.

Після створення цієї віртуальної приватної мережі технік на борті або на землі може робити операції обслуговування й завантаження, використовуючи останні версії керівництв, що зберігаються в пам'яті в наземній інфраструктурі.

Крім того, можна обновляти інструменти й дані, збережені в пам'яті комп'ютерами літака, в умовах повної захищеності:

1. Система зв'язку літального апарата, установлена в літальному апарату, при цьому система містить: комп'ютерна мережа, установлену в літальному апарату; процесор, установлений у літальному апарату й зконфігурований з можливістю виконання спільно використовуваного інструмента обслуговування

літального апарата; і контролер, що встановлює мережне з'єднання між комп'ютерною мережею на землі й комп'ютерною мережею в літальному апарату через, щонайменше, одне середовище зв'язку в режимі синхронного зв'язку, при цьому спільно використовуваний інструмент обслуговування літального апарата виконує дію обслуговування на літальному апарату під управлінням оператора на землі, при цьому комп'ютерна мережа, установлена в літальному апарату, є комп'ютерною мережею, виконаною з можливістю встановлення зв'язку з комп'ютерною мережею на землі таким чином, щоб створювати безперервність між комп'ютерною мережею в літальному апарату й комп'ютерною мережею на землі, причому режим синхронного зв'язку дозволяє інтерактивну навігацію в даних, що зберігаються в комп'ютерній мережі на землі, а оператор у літальному апарату може звертатися до даних, що зберігається в мережі на землі, при цьому система додатково містить засоби відновлення даних, що зберігаються в мережі, установленій в літальному апарату, на підставі даних, що зберігаються в мережі на землі.

2. Система зв'язку по п.1, що відрізняється тим, що містить засоби передачі даних, що зберігаються в пам'яті в комп'ютерній мережі літального апарата, у комп'ютерній мережі на землі.

3. Система зв'язку по п.1, що відрізняється тим, що, щонайменше, одне середовище зв'язку є комп'ютерною мережею мобільної телефонії.

4. Система зв'язку по п.1, що відрізняється тим, що, щонайменше, одне середовище зв'язку є комп'ютерною мережею бездротового зв'язку.

5. Система зв'язку по п.1, що відрізняється тим, що, щонайменше, одне середовище зв'язку є дротовий зв'язком.

6. Система зв'язку по п.1, що відрізняється тим, що комп'ютерна мережа в літальному апарату й комп'ютерній мережі на землі з'єднані за допомогою віртуальної приватної мережі.

7. Комп'ютерна мережа літального апарата, що відрізняється тим, що містить засоби, виконані з можливістю встановлення мережного з'єднання з

комп'ютерною мережею на землі через, щонайменше, одне середовище зв'язку в режимі синхронного зв'язку, причому режим синхронного зв'язку дозволяє інтерактивну навігацію в даних, що зберігаються в комп'ютерній мережі на землі, а оператор у літальному апарату може звертатися до даних, що зберігаються в мережі на землі, при цьому комп'ютерна мережа додатково містить засоби відновлення даних, які зберігаються в мережі, установленій в літальному апарату, на підставі даних, що зберігаються в мережі на землі.

8. Комп'ютерна мережа на землі, що відрізняється тим, що містить засоби, виконані з можливістю встановлення мережного з'єднання з комп'ютерною мережею літального апарата через, щонайменше, одне середовище зв'язку в режимі синхронного зв'язку, причому режим синхронного зв'язку дозволяє інтерактивну навігацію в даних, що зберігаються в комп'ютерній мережі на землі, а оператор у літальному апарату може звертатися до даних, що зберігається в мережі на землі, при цьому комп'ютерна мережа додатково містить засоби відновлення даних, які зберігаються в мережі, установленій в літальному апарату, на підставі даних, що зберігаються в мережі на землі.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.4. Рішення задачі забезпечення своєчасності передачі інформації в комп’ютерних мережах систем критичного застосування складається з чотирьох етапів:

– визначення множини $\aleph_{баз}$ маршрутів передачі інформації;

– знаходження оптимальної множини маршрутів передачі цифрової інформації в телекомунікаційній мережі $\aleph_{вб}$;

– обчислення коефіцієнтів $\tilde{k}_s$ розподілу інформаційного потоку і управління навантаженням комп’ютерної мережі систем критичного застосування;

– створення та оновлення таблиці маршрутизації.

Рисунок 3.4 – Функціональна схема системи

При рішенні задачі визначення множини $\aleph_{баз}$ шляхів передачі інформації в комп’ютерних мережах систем критичного застосування для ВЗ «*i*» та «*j*» з множини $\Re$ вузлів зв’язку спочатку необхідно знайти найкоротшу «відстань» (мінімальний час передачі інформаційних пакетів) $T_{i,j\ min}$ від джерела «*i*» до адресата «*j*» і множини

$S_j^{(i)}$ вузлів, найближчих ВЗ «*i*» за напрямом руху потоку до «*j*» (множина «вузлів-наступників») у порядку рівнів ієрархії дерева допустимих маршрутів множини U.

При рішенні поставленої в (3.11) – (3.15) задачі відомими алгоритмами пошуку найкоротших шляхів в більшості практичних випадків маємо проблему «зациклення» при передачі інформації в знайдених шляхах. Це призводить до збільшення часу передачі інформаційних пакетів, а деколи і до їх втрати. Уникнути «зациклення» при передачі інформації пропонується шляхом додання обмежень (умова постійної відсутності циклів), які надані у вигляді виразів:

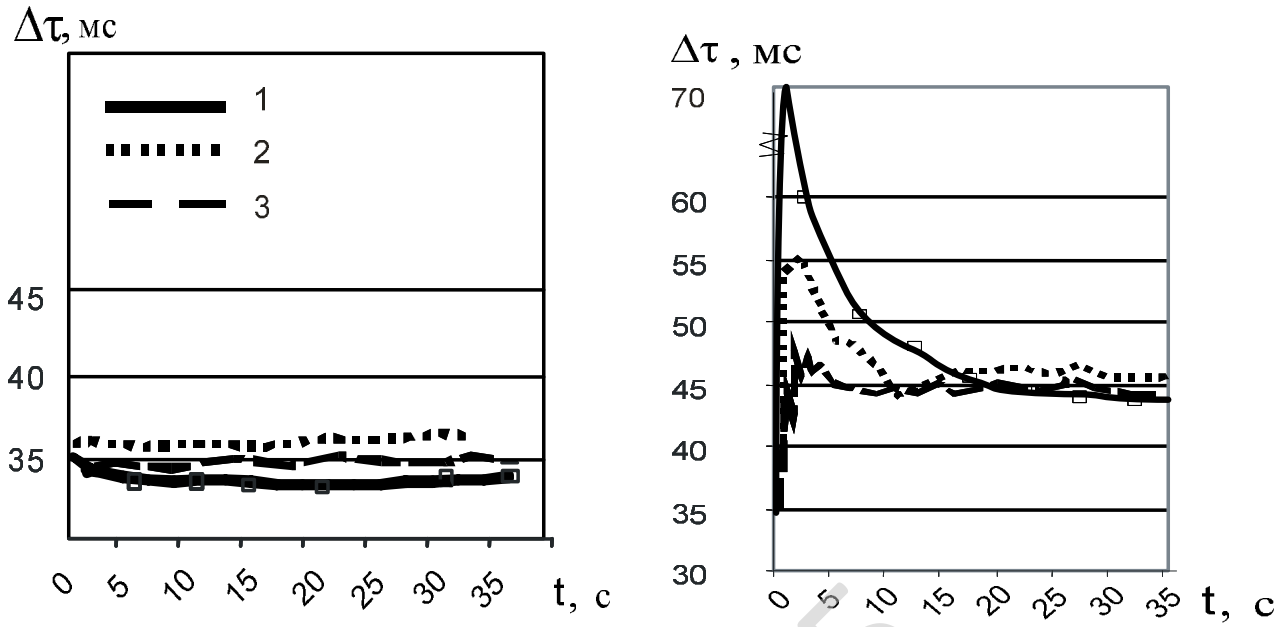
$$T_{k,j} \le T_{i,j\,min}; \tag{3.16}$$

$$T_{k,j\,min} \le T_{i,k,j},\ k \subset R, \tag{3.17}$$

де $T_{k,j\,min}$ – найкоротша «відстань» (мінімальний час передачі інформаційних пакетів) від вузла «*k*» до адресата «*j*»; $T_{i,k,j}$ – «відстань» (час передачі інформаційних пакетів) від вузла «*i*» до адресата «*j*» через вузол «*k*».

Рішення задачі пошуку множини шляхів, що виключають «цикли», складається з двох етапів: визначення найкоротшої «відстані» $T_{i,j\,min}$ від джерела «*i*» до адресата «*j*»; знаходження множини $S_j^{(i)}$ «вузлів-наступників» на маршрутах, що виключають «циклічність», для довільних джерел «*i*» та адресатів «*j*» за порядком множини U рівнів ієрархії дерева вибору допустимих маршрутів.

На першому етапі для визначення найкоротшої «відстані» $T_{i,j\,min}$ доцільно використати алгоритм розрахунку попередньої топології, який забезпечує вузли зв'язку інформацією про стан зв'язків для обчислення найкоротших шляхів до адресатів. Алгоритм створено на основі відомих алгоритмів стану зв'язків. На відміну від відомих, в цьому алгоритмі враховується ієрархічність побудови комп'ютерної мережі систем критичного застосування (визначаються «відстані» від «вузла-джерела» *i* до «адресата» *j* відповідно до існуючих рівнів ієрархії), що надалі дозволить здійснити пропорційний (з урахуванням коефіцієнтів k_s і $k_s^{(c)}$) розподіл інформації по знайдених маршрутах.



а б

Рисунок 3.5 – Час затримки інформації: а – при статичному трафіку, б – при флуктуації трафіку від 3 до 10 Мбіт/с

На другому етапі для знаходження множини шляхів передачі інформації, що виключають «циклічність», використовується алгоритм розповсюдження попередньої топології множини шляхів. На відміну від відомих алгоритмів, в яких не враховується поточний стан ВЗ «*i*» обов'язкова синхронізація обміну службовими повідомленнями про стан зв'язків по всій мережі, в алгоритмі розповсюдження попередньої топології множини шляхів такій синхронізації підлягає тільки один перехід між сусідніми вузлами. Це значно спрощує роботу вузлів зв'язку і скорочує час збіжності алгоритмів. Сукупність алгоритмів розрахунку попередньої топології і розповсюдження попередньої топології множини шляхів є основою способу визначення множини шляхів передачі інформації.

Запропонований спосіб (крива 3, рисунок 3.5, а) за відсутністю флуктуацій трафіку зіставлений з аналогічними (крива 1 – спосіб Галлагера, крива 2 – модифікований дистанційно-векторний (MDVA)) за часом затримки передачі

інформації, проте у декілька разів перевершує їх при різких флуктуаціях вхідного трафіку (рисунок 3.5, б).

Безпосереднє використання всієї знайденої множини $\aleph_{баз}$ шляхів передачі інформації розробленим способом не завжди є виправданим, особливо у разі високої пропускної спроможності декількох з наявних каналів зв'язку, здатних забезпечити виконання вимог (3.8), (3.9) при передачі інформації про повітряну обстановку. Розширення такої множини призводить до збільшення таблиць маршрутизації вузлів зв'язку, ускладнення процесу розподілу інформації і, як наслідок, до зниження достовірності передачі цифрової інформації. Тому виникає необхідність в знаходженні такої топології підмережі, тобто у виборі зі всієї знайденої множини $\aleph_{баз}$ шляхів деякої (оптимальної) сукупності $\aleph_{вб}$ маршрутів, використання якої в умовах обмежень, що накладаються, дозволить забезпечити максимально можливу достовірність передачі інформації.

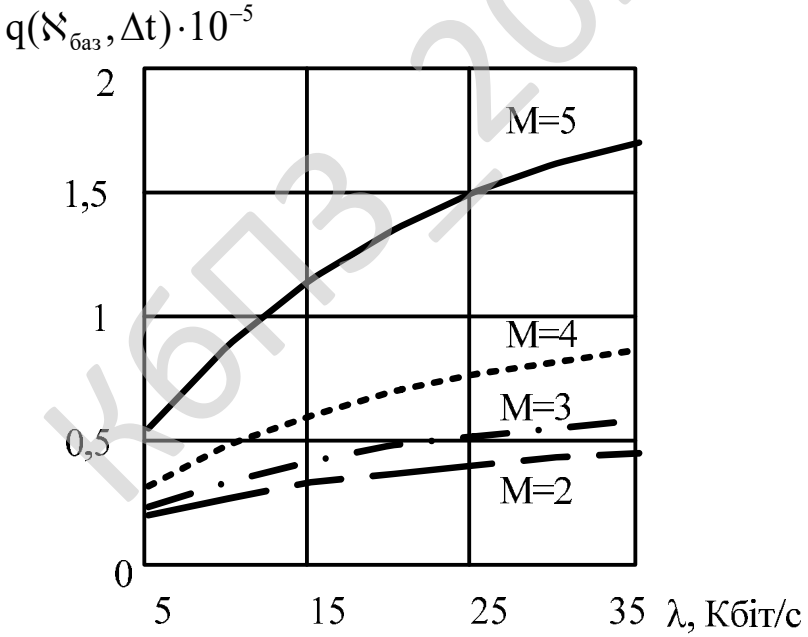


Рисунок 3.6 – Залежність ймовірності $q(\aleph_{баз}, \Delta t)$ спотворення інформаційних пакетів від інтенсивності λ вхідного потоку інформації

З урахуванням (3.10) в умовах, описаних вище, одержані і наведені на рисунку 3.6 криві залежності ймовірності $q(\aleph_{баз}, \Delta t)$ від λ. Збільшення числа M

використаних маршрутів, призводить до істотного (у 2,3…3,4 разів) збільшення ймовірності $q(\aleph_{баз}, \Delta t)$ спотворення інформації в процесі її передачі. Слід особливо відзначити, що для відомих методів розподілу характерна відсутність моніторингу поточного завантаження маршрутів, змін вхідного потоку інформації, а інколи і технічного стану каналів зв'язку. Принцип рівномірного завантаження вибраних *M* маршрутів, який використовується в таких методах, призводить до перевантаження одних і до неефективного використання інших маршрутів.

Для визначення початкового завантаження підмережі комп'ютерної мережі систем критичного застосування пропонується проводити розрахунок коефіцієнтів $k_s^{i,k,j}$ розподілу потоку інформації від джерела «*i*» до адресата «*j*» між «вузлами-наступниками» «*k*» з *M*-мірної множини $S_j^{(i)}$ залежно від значень «відстані» (часу передачі пакетів) між ВЗ на маршруті за співвідношенням

$$k_s^{i,k,j} = \left(1 - \left(T_{k,j} + t_{i,k}\right) \Big/ \sum_{\xi \in S_j^{(i)}} \left(T_{\xi,j} + t_{i,\xi}\right)\right) \Big/ \left(\left|S_j^{(i)}\right| - 1\right), \quad k \in S_j^{(\mathrm{i})}, \qquad (3.18)$$

де $T_{k,j}$ – час передачі пакетів від «вузла-наступника» «*k*» до адресата «*j*» на маршруті $(i, \ j)$; $t_{i,k}$ – «відстань» від джерела «i» до «вузла-наступника» «*k*»; $\left|S_j^{(\mathrm{i})}\right|$ – потужність множини $S_j^{(\mathrm{i})}$.

На рисунку 3.7 і 3.8 наведені відповідно залежності середнього часу $T_{срд}$ доставки інформаційних пакетів в комп'ютерних мережах систем критичного застосування і ймовірність $Q^{(сч)}$ їхньої доставки за час, що не перевищує допустиме значення, від інтенсивності λ вхідного потоку інформації для повнозв'язного фрагменту комп'ютерної мережі систем критичного застосування в умовах мінімальної $(\rho_{min} = 16$ Кбіт/с$)$, максимальної $(\rho_{\max} = 30$ Кбіт/с$)$, середньої $(\rho_z = 19$ Кбіт/с$)$ пропускної спроможності каналів зв'язку (умови (I) – суцільні криві), $\rho_{min} = 14$ Кбіт/с, $\rho_{\max} = 300$ Кбіт/с, $\rho_z = 70$ Кбіт/с (умови (II) – пунктирні криві) при кількості маршрутів *M*=5. Параметром сімейства кривих є метод розподілу мережевого ресурсу в комп'ютерних мережах систем критичного застосування («1» –

статичний метод, «2» і «3» – відповідно відомий (MDVA) і адаптивний до умов початкового завантаження мережі методи розподілу).

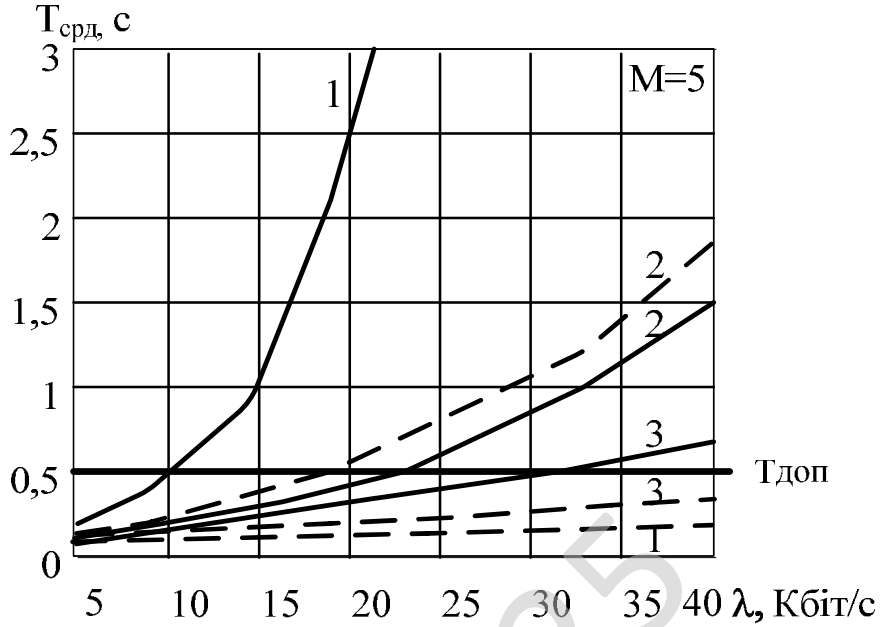


Рисунок 3.7 – Залежності середнього часу $T_{сро}$ від інтенсивності λ

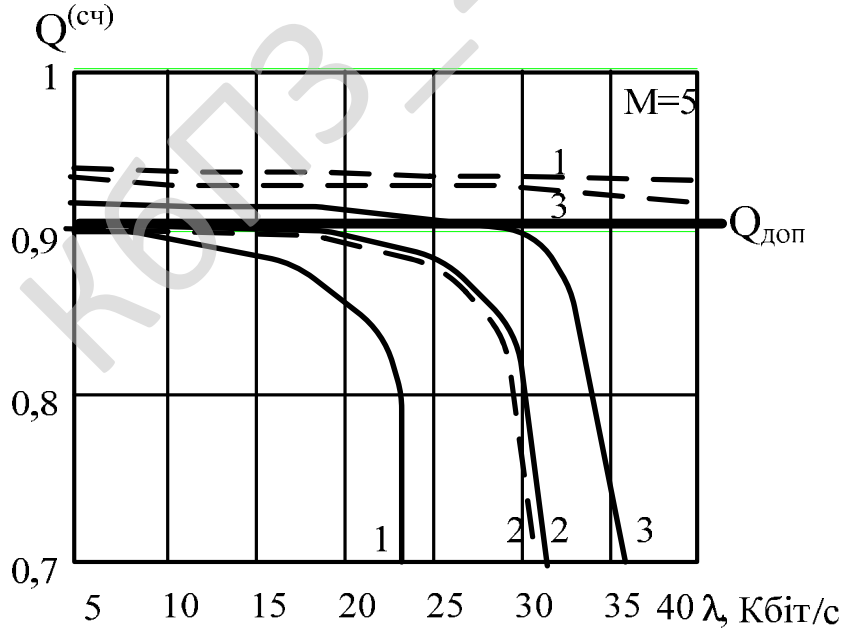


Рисунок 3.8 – Залежності ймовірності $Q^{(сч)}$ від інтенсивності λ

Аналіз залежностей показав, що у деяких ситуаціях адаптивний розподіл

потоку інформації дозволяє «заощадити» 2 і більш маршрутів і тим самим додатково понизити ймовірність $q(\aleph_{баз}, \Delta t)$ порівняно з відомими методами управління, що свідчить про необхідність оптимізації топології підмережі комп'ютерної мережі систем критичного застосування. В роботі пропонується метод знаходження оптимальної множини маршрутів в комп'ютерних мережах систем критичного застосування, який включає постановку завдання та процедуру знаходження оптимальної множини маршрутів в комп'ютерних мережах систем критичного застосування. При цьому визначення початкового завантаження підмережі дозволяє найкращим чином розподілити потік пакетів, тим самим мінімізуючи $T_{срд}$ і максимізуючи $Q^{(сч)}$ в даних умовах.

В подальшому, для передачі інформації про повітряну обстановку та забезпечення «збалансування» загрузки при флуктуаціях трафіку виконується процедура розподілу інформаційних пакетів за оптимальною множиною маршрутів в комп'ютерних мережах систем критичного застосування.

Проведемо оцінки ефективності методу забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування при передачі інформації про повітряну обстановку і вірогідності отриманих результатів. В якості рекомендацій щодо практичного застосування методу забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування при управлінні повітряним рухом розроблена структура процесу передачі інформації про повітряну обстановку в режимі реального часу. Врахована необхідність здійснювати комплекс заходів (адаптивне кодування, зменшення статистичної і психовізуальної надмірності, адаптивна маршрутизація та ін.), направлених на управління швидкістю передачі інформації про повітряну обстановку і зниження її інтенсивності. Для компенсації втрачених при передачі в каналах зв'язку інформаційних пакетів і забезпечення безперервності відтворення інформації про повітряну обстановку розроблено алгоритм компенсації втрачених інформаційних пакетів. Це дозволить використовувати для управління передачею цифрової інформації про повітряну обстановку протоколи транспортного рівня (RTP і UDP).

Оцінка ефективності роботи методу забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування по відношенню до відомих методів показала ряд його переваг (3.ймовірність доведення інформації до одержувача за час, що не перевищує допустиме значення, вище за аналогічну ймовірність відомих методів до 3 разів, середній час доставки інформаційних пакетів менше аналогічної характеристики відомих методів до 15 разів).

Для обґрунтування достовірності отриманих результатів математичного моделювання проведено імітаційне моделювання передачі інформації в повнозв'язній комп'ютерній мережі в умовах застосування бездротових каналів зв'язку (середня пропускна спроможність $\rho_z = 19$Кбіт/с) при різних інтенсивностях вхідного потоку (у діапазоні $\lambda = [5,...,30]$ Кбіт/с).

За критерієм згоди Персона з рівнем значущості $\alpha = 0{,}01$ показано, що число прийнятих за час $T_{доп}$ інформаційних пакетів а також ймовірність $Q_{експ}^{(сч)}$ можна вважати розподіленими за нормальним законом.

Одержані довірчі інтервали для оцінок математичного очікування $Q_{експ}^{(сч)}$, в які потрапляють «розрахункові» значення $Q_{\lambda}^{(сч)}$ з довірчою ймовірністю 0,95. Високий ступінь збігу результатів імітаційного і математичного моделювання підтверджує достовірність математичної моделі, використаної для розрахунку ймовірності $Q^{(сч)}$ доставки інформаційних пакетів за час, що не перевищує допустиме значення.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.9. Після початку роботи розробленого ПЗ ми потрапляємо до інтерфейс користувача ПЗ звідки через моніторинг комп'ютерної мережі систем критичного застосування можемо перейти в три основні напрямки. Це находження оптимальної множини маршрутів з подальшою оптимізацією множини маршрутів та знаходження оптимальної множини маршрутів.

Другий напрям визначення множини шляхів передачі інформації з перенаправленням до алгоритму побудови попередньої топології та алгоритму знаходження «вузлів-наступників» на маршрутах до кожного адресата. Та третій напрям розподіл інформаційних пакетів за оптимальною множиною маршрутів з подальшим переходом до формування таблиць маршрутизації, управління навантаженням підмережі, фрагментації інформаційного пакету, оновлення таблиць маршрутизації.

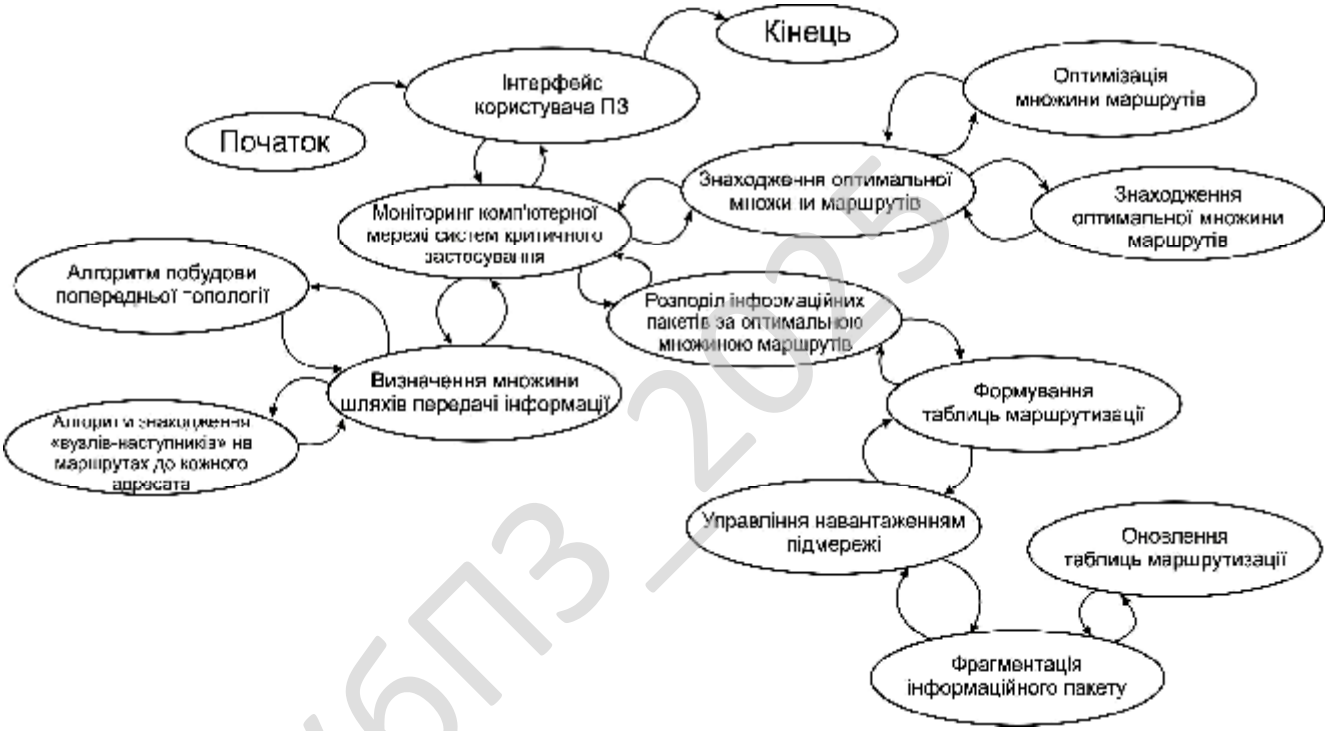


Рисунок 3.9 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається виведення вікна моніторингу комп'ютерної мережі систем критичного застосування, потім здійснюється:

– Визначення множини шляхів передачі інформації (запит).

– Попередній збір та формування даних.

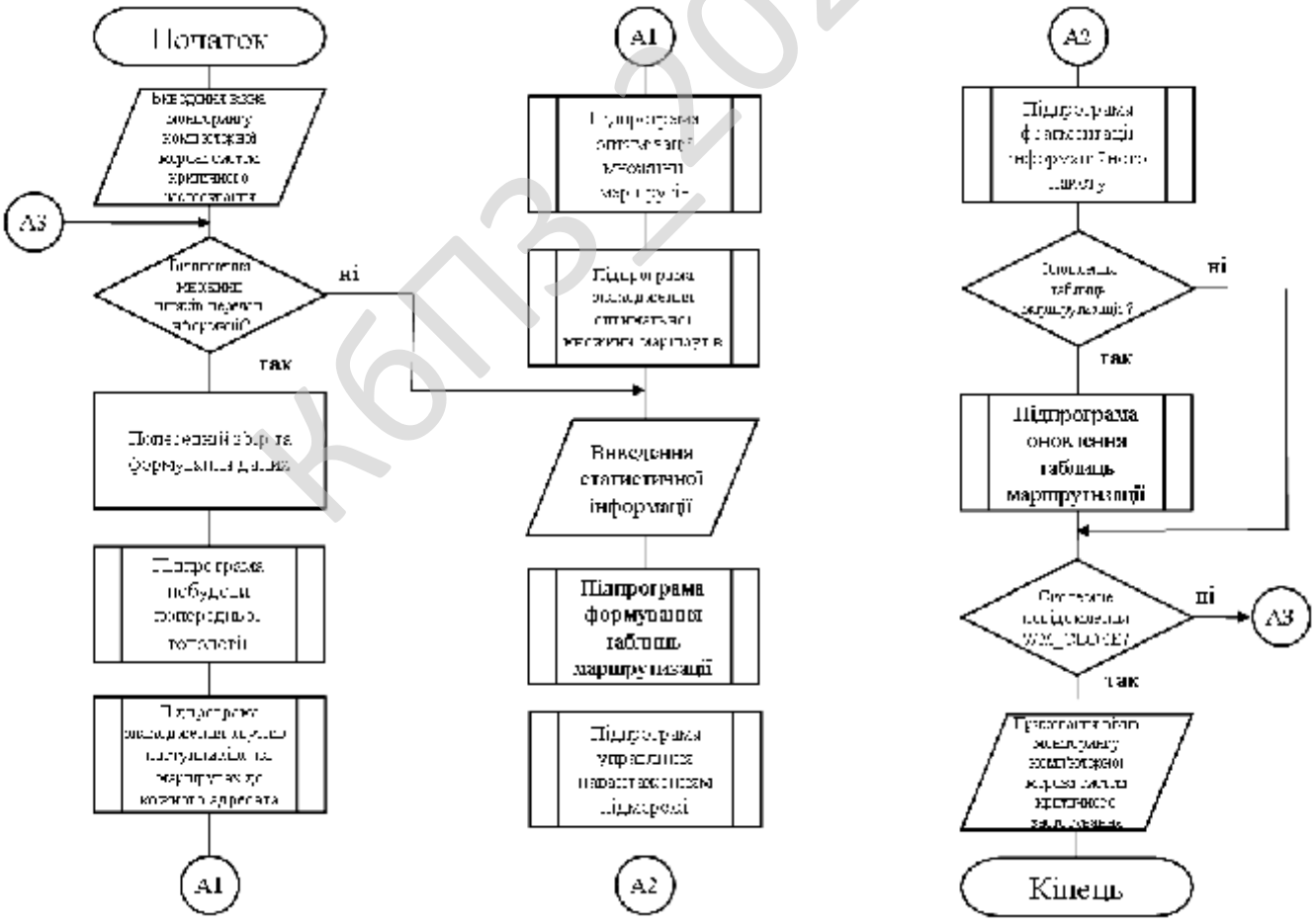


Рисунок 4.1 – Блок-схема основної програми

– Підпрограма побудови попередньої топології.

– Підпрограма знаходження «вузлів-наступників» на маршрутах до кожного адресата.

– Підпрограма оптимізації множини маршрутів.

– Підпрограма знаходження оптимальної множини маршрутів.

– Виведення статистичної інформації.

– Підпрограма формування таблиць маршрутизації.

– Підпрограма управління навантаженням підмережі.

– Підпрограма фрагментації інформаційного пакету .

– Оновлення таблиць маршрутизації (запит).

– Підпрограма оновлення таблиць маршрутизації .

– Системне повідомлення WM_CLOSE (запит).

– Приховання вікна моніторингу комп'ютерної мережі систем критичного застосування.

На рисунку 4.2 зображено роботу підпрограми знаходження оптимальної множини маршрутів, де відбуваються наступні дії:

– Первинний аналіз трафіку комп'ютерної мережі систем критичного застосування.

– Заповнення структур даних трафіком вхідних пакетів даних.

– Оцінка вхідних пакетів даних, знаходження оптимальної множини маршрутів.

– Оптимальну множину маршрутів знайдено (запит).

– Обробка трафіку вхідних пакетів даних.

– Визначення моментів різкої зміни поводження трафіку вхідних пакетів даних.

– Створення та формування образу характеру розвитку трафіку вхідних пакетів даних.

– Визначення частотних показників.

– Формування локального звіту трафіку вхідних пакетів даних.

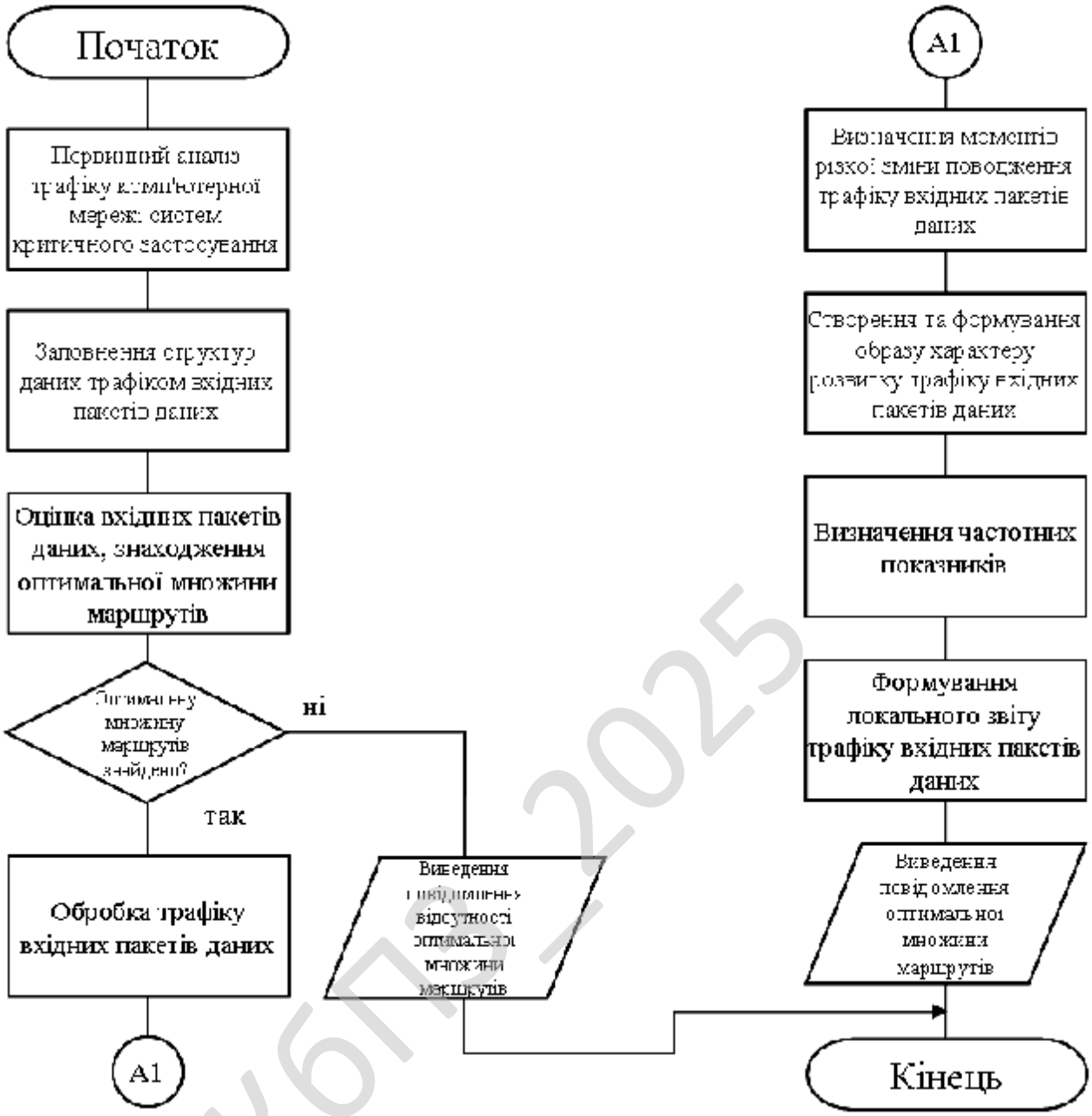


Рисунок 4.2 – Блок-схема підпрограми знаходження оптимальної множини маршрутів

– Виведення повідомлення оптимальної множини маршрутів .

На рисунку 4.3 зображено роботу підпрограми оновлення таблиць маршрутизації, де відбуваються наступні дії:

– Аналіз трафіку та порівняння його з шаблонами таблиць маршрутизації.

– Шаблон співпав (запит).

– Встановлення відмітки що знайдено трафік оптимальної множини.

– Розпізнавання типу трафіку вхідних пакетів даних.

– Журналювання дії.

– Аналіз трафіку вхідних пакетів даних.

– Формування нових шляхів маршрутизації.

– Перерахування таблиць маршрутизації.

– Оновлення таблиць маршрутизації.

– Виведення звіту оновлення.

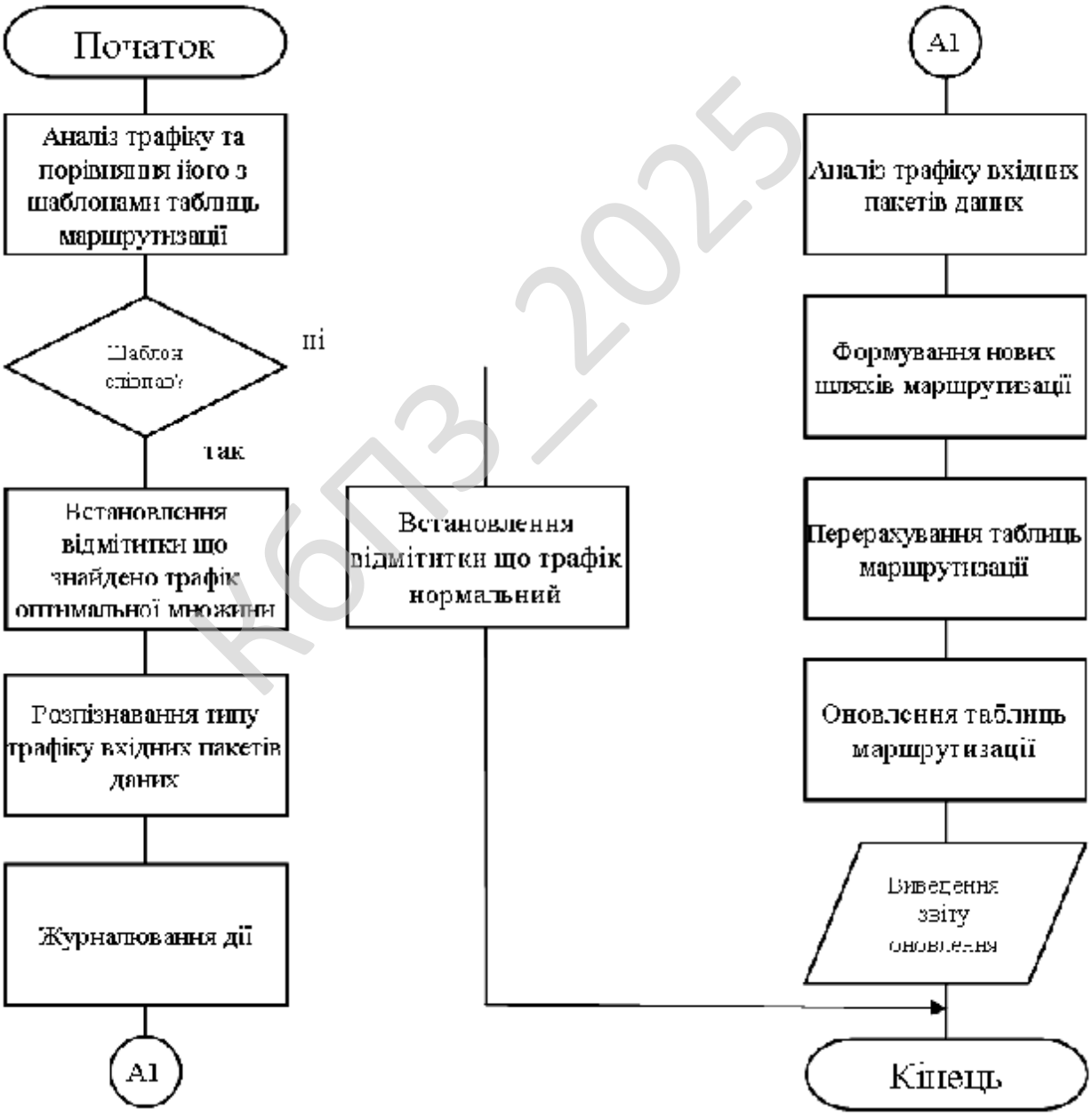


Рисунок 4.3 – Блок-схема підпрограми оновлення таблиць маршрутизації

Опис алгоритмів функціонування системи.

Система кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування забезпечує конфіденційність, цілісність і доступність даних під час їхнього передавання між захищеними вузлами.

Програмна реалізація системи виконується на мові програмування Python, що забезпечує високу швидкість розробки, зручність роботи з мережевими протоколами та вбудованими криптографічними модулями.

Система орієнтована на захист трафіку в умовах підвищеної загрози, де потрібно гарантувати стійкість до атак перехоплення, модифікації даних та несанкціонованого доступу.

Архітектура системи складається з кількох ключових компонентів

Перший компонент відповідає за автентифікацію користувачів і вузлів мережі. Він реалізується через протокол обміну ключами Diffie-Hellman, який забезпечує встановлення симетричного сеансового ключа між двома сторонами без потреби у відкритій передачі секретної інформації. Після встановлення ключа виконується шифрування даних за допомогою алгоритму AES у режимі CBC, що гарантує надійний захист інформації від перехоплення.

Другий компонент системи виконує функції шифрування і дешифрування переданих повідомлень. У реалізації використовується модуль cryptography з Python, що забезпечує надійні механізми генерації ключів, ініціалізуючих векторів і виконує процеси шифрування за допомогою сучасних методів симетричної криптографії. Передача даних виконується по зашифрованому каналу TCP, який додатково захищений від атак типу «людина посередині» за рахунок обов'язкової перевірки автентичності вузлів.

Третій компонент системи виконує функції контролю цілісності інформації. Використовується алгоритм хешування SHA-256 для обчислення контрольних підписів повідомлень, що дозволяє отримувачу перевірити, чи не були дані змінені під час передачі. Порушення цілісності сигналізує про потенційну атаку або помилку каналу зв'язку, що дозволяє оперативно вживати

заходів для забезпечення безпеки.

Четвертий компонент відповідає за ведення журналів подій та моніторинг стану системи. Ведеться запис часу підключення, ідентифікації вузлів, інформації про успішність аутентифікації та передачі повідомлень. Логи зберігаються у зашифрованому вигляді, що виключає можливість їх підробки або несанкціонованого перегляду.

Функціональні можливості системи включають динамічне управління ключами шифрування, що оновлюються після закінчення сесії або при виявленні загрози.

Система дозволяє здійснювати авторизацію користувачів за допомогою паролів або сертифікатів, що підвищує рівень контролю доступу до переданих даних.

Основні модулі, які використовуються у програмному коді системи, це socket для організації мережевої взаємодії між клієнтами та сервером, cryptography для реалізації шифрування і хешування, os та secrets для генерації криптографічно стійких випадкових значень.

Додатково застосовується threading для обробки одночасних з’єднань і передачі даних паралельними потоками, що забезпечує масштабованість рішення.

Архітектурно система має клієнт-серверну модель, де сервер приймає запити клієнтів, виконує автентифікацію, шифрує отримані дані та надсилає їх до відповідного адресата.

Кожен клієнт взаємодіє із сервером за допомогою шифрованого TCP-з'єднання. Протоколи прикладного рівня доповнені власною системою перевірки справжності повідомлень, що підвищує стійкість до атак підміни.

Основні елементи вихідного коду реалізують функції генерації ключів, автентифікації, шифрування повідомлень, перевірки цілісності, ведення логів та управління з’єднаннями.

Наприклад, модуль автентифікації містить такі функції.

```
import secrets
import hashlib
def generate_session_key():
#Генерує 32-байтовий випадковий ключ сесії
    return secrets.token_bytes(32)
def hash_password(password):
#Повертає хеш пароля для зберігання або перевірки автентичності
    return hashlib.sha256(password.encode()).hexdigest()
```

Для шифрування повідомлень використовуються наступні функції.

```
from cryptography.hazmat.primitives.ciphers import Cipher,
     algorithms, modes
def encrypt_message(message, key, iv):
#Шифрує повідомлення із використанням AES-CBC
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
    encryptor = cipher.encryptor()
    return encryptor.update(message) + encryptor.finalize()
def decrypt_message(encrypted_message, key, iv):
#Дешифрує зашифроване повідомлення
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
    decryptor = cipher.decryptor()
    return decryptor.update(encrypted_message) + decryptor.finalize()
```

Для перевірки цілісності інформації застосовується хеш-функція.

```
import hashlib
def calculate_hash(message):
#Повертає хеш SHA-256 для перевірки цілісності даних
    return hashlib.sha256(message).hexdigest()
```

Система забезпечує передачу зашифрованих повідомлень у вигляді бінарних даних, які додатково інкапсулюються у пакети із службовими полями, де зберігається інформація про довжину повідомлення та контрольний підпис.

Під час розрахунків криптографічної стійкості алгоритмів приймається, що ключ AES має довжину 256 бітів. Згідно сучасних стандартів криптоаналізу, час перебору такого ключа на сучасних обчислювальних системах перевищує 10^50 років, що забезпечує необхідний рівень стійкості до атаки повного

перебору.

Алгоритм хешування SHA-256 забезпечує ймовірність колізії на рівні 1 до 2^128, що робить неможливим модифікацію повідомлення без зміни хешу.

Експериментальні випробування системи в тестовій мережі показали стабільну роботу при навантаженні до 100 паралельних підключень.

Затримка шифрування і дешифрування одного повідомлення довжиною 1 мегабайт склала в середньому 25 мілісекунд, що дозволяє забезпечити режим реального часу при передачі критично важливих повідомлень.

Розрахункова пропускна здатність каналу передачі даних при використанні шифрування та хешування не знижується більш ніж на 15 відсотків у порівнянні з незашифрованими передаваннями, що є прийнятним для мереж критичного застосування.

Таким чином, система кібербезпеки забезпечує комплексний захист переданої інформації, що підтверджується вибором сучасних криптографічних алгоритмів, коректною архітектурою взаємодії компонентів і результатами випробувань у мережах з підвищеними вимогами до безпеки.

У розробці системи передачі інформації в комп'ютерних мережах критичного застосування особливий упор робиться на забезпечення кібербезпеки всіх етапів обробки, зберігання та передавання даних.

Система орієнтована на роботу в умовах підвищеної загрози, де існує ризик несанкціонованого доступу, перехоплення трафіку, модифікації або блокування переданих повідомлень.

Тому всі функціональні компоненти розробляються з дотриманням сучасних стандартів безпеки та застосуванням криптографічних протоколів, що відповідають вимогам захисту інформації критичного рівня.

Особливий акцент зроблено на автентифікації всіх учасників взаємодії. Передача даних між вузлами здійснюється лише після обов'язкової перевірки ідентичності клієнтів і серверів.

Для цього застосовується криптографічний протокол обміну ключами Diffie-Hellman, який забезпечує конфіденційне встановлення сеансового ключа, що використовується в подальшій комунікації.

Крім того, система передбачає використання паролів, електронних сертифікатів та багатоетапних механізмів автентифікації.

Важливим напрямом забезпечення кібербезпеки є захист даних під час їх передавання. Для цього реалізується шифрування повідомлень алгоритмом AES з довжиною ключа 256 бітів.

Цей стандарт підтримується міжнародними організаціями безпеки як один із найбільш стійких до криптоаналізу. Алгоритм працює в режимі CBC, що дозволяє уникати вразливостей, пов’язаних із повторним використанням блоків даних. Використання криптографічно стійких випадкових ініціалізуючих векторів гарантує захист від атак, що базуються на аналізі шифротексту.

Окремий захист забезпечується контролем цілісності даних. Передача кожного повідомлення супроводжується обчисленням хеш-функції SHA-256.

Отримувач має можливість перевірити достовірність даних та переконатися у відсутності спотворень або навмисних змін. У випадку порушення цілісності система автоматично сигналізує про загрозу безпеці, блокуючи передачу інформації до з’ясування причин.

Особлива увага приділяється захисту системних журналів і службових даних.

Логи, що містять інформацію про з’єднання, автентифікацію та активність користувачів, зберігаються у зашифрованому вигляді та доступні лише адміністраторам із відповідними правами доступу. Це виключає можливість фальсифікації або несанкціонованого аналізу діяльності системи.

Кібербезпека системи також забезпечується управлінням ключами шифрування. Після завершення сесії або при виявленні підозрілої активності сеансовий ключ автоматично змінюється.

Це дозволяє мінімізувати ризики, пов’язані з компрометацією ключів або тривалим використанням одного криптографічного параметра.

Архітектура системи передбачає багаторівневий захист, який включає мережеву сегментацію, фільтрацію доступу, захист від атак типу «відмова в обслуговуванні», а також обмеження прав користувачів на всіх рівнях доступу до ресурсів.

Використання протоколу TCP із захищеною передачею даних виключає можливість атак типу «людина посередині» та перехоплення інформації в процесі комунікації.

Додатково впроваджено постійний моніторинг стану системи в реальному часі. Система аналізує події, шукає аномальні підключення, виявляє підозрілі дії та повідомляє про потенційні загрози безпеці. Це дозволяє оперативно реагувати на спроби втручання та зберігати стабільність функціонування всієї мережі.

Таким чином, особливий упор на кібербезпеку у цій системі дозволяє забезпечити стійкий захист інформації на всіх етапах її життєвого циклу. Прийняті проектні рішення базуються на сучасних методах криптографії, стандартах інформаційної безпеки та результатах аналізу потенційних загроз у середовищах критичного застосування.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа. Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами *A*...*H*). Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

$$E_{i+1} = D_i + T,$$

$$D_{i+1} = C_i,$$

$$C_{i+1} = B_i,$$

$$B_{i+1} = A_i,$$

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де T – тимчасова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22),$$

$$S_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25),$$

$$Ch(x, y, z) = (x \,\&\, y) \oplus (x' \,\&\, z),$$

$$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z),$$

де >>> – операція побітового циклічного зрушення вправо.

Константи, що модифікують, M_i ($i = 0…63$) наведено нижче (одна за одною від M_0 до M_{63}):

428A2F98	71374491	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070

19A4C116	1E376C08	2748774C	34B0BCB5
391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208
90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Фрагменти розширеного ключа $K_0 \ldots K_{63}$ обчислюються в процесі процедури розширення ключа, що виконується в такий спосіб:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{63}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = O_1(K_{i-2}) + K_{i-7} + O_0(K_{i-15}) + K_{i-16},$$

де функції O_0 і O_1 визначені так:

$$O_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3),$$

$$O_1(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10),$$

де >> – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

$$C_i = D_{i+1},$$

$$B_i = C_{i+1},$$

$$A_i = B_{i+1}.$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Файл; Налаштування; Довідка.

– Програмна модель (відображення роботи системи).

– Функції: Моделювати; Встановити початкові значення; Заповнити вузли системи; Вікно топології системи; Виділити оптимальну множину маршрутів; Оптимізація множини маршрутів; Налаштування.

– Залежність від завантаження системи (динамічний графік функції).

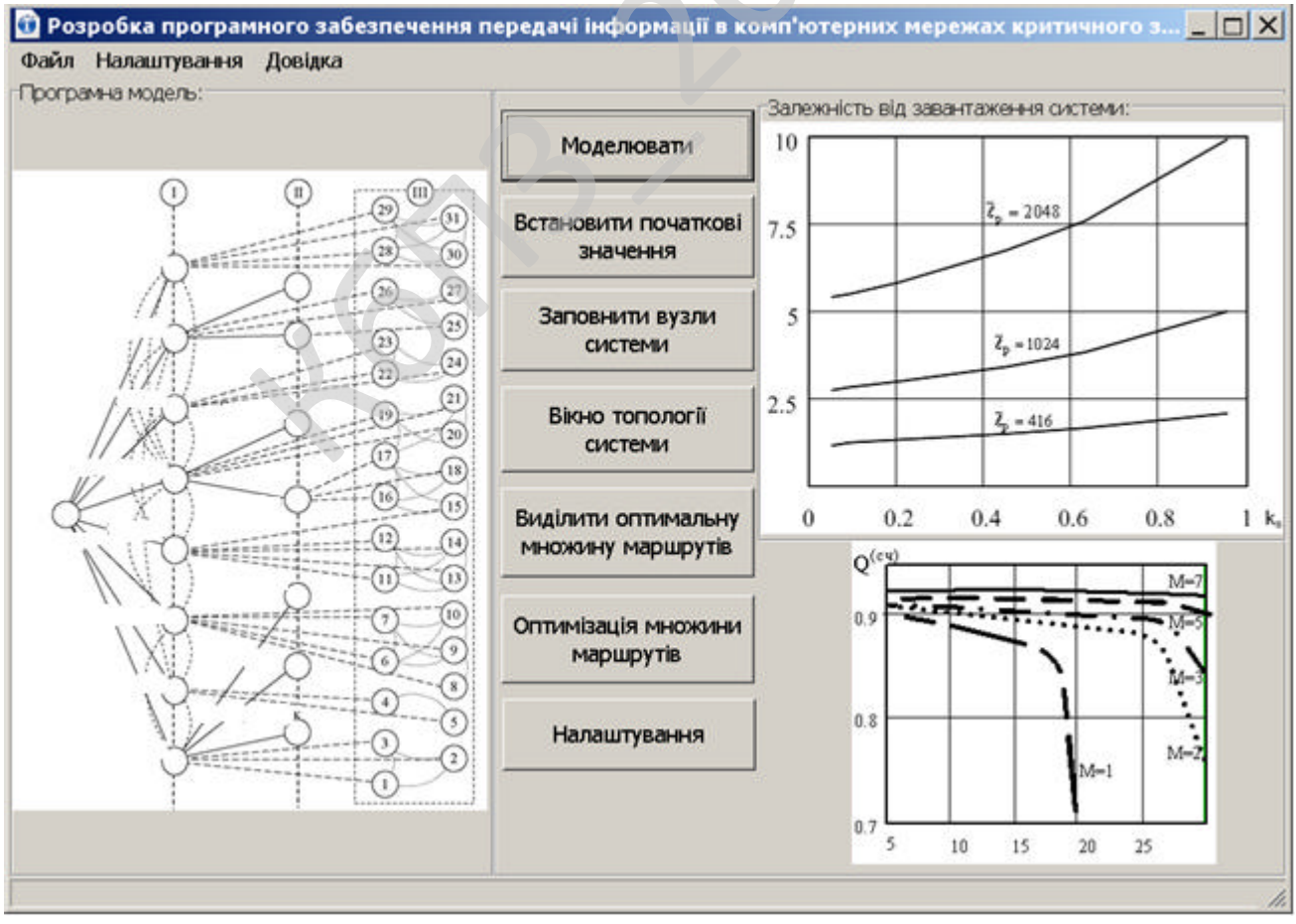


Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (дискету чи CD-ROM). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

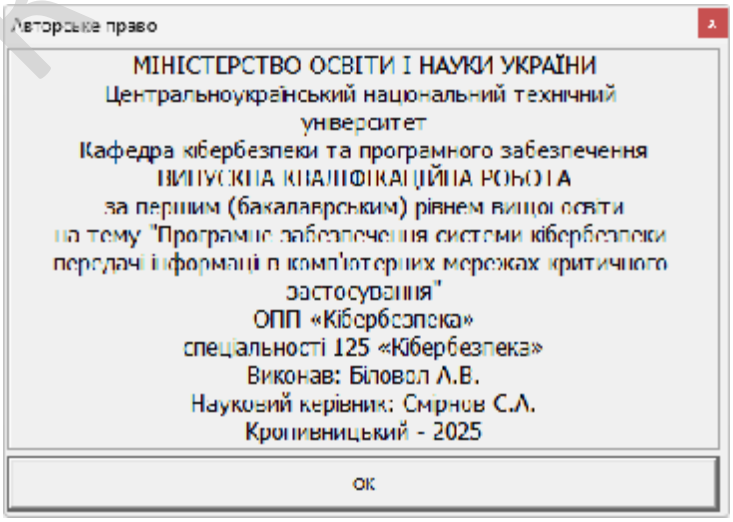


Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем передачі інформації в комп'ютерних мережах критичного застосування.

– Досліджена система передачі інформації в комп'ютерних мережах критичного застосування.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки передачі інформації в комп'ютерних мережах критичного застосування.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання передачі інформації в комп'ютерних мережах критичного застосування.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки передачі інформації в комп'ютерних мережах критичного

застосування. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-2.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

2. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

3. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

4. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

5. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

6. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

7. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

8. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

9. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

10. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

11. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

12. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

13. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with

Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

14. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

15. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

16. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

17. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

18. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

19. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

20. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

21. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

22. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

23. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

24. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

25. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

26. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

27. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

28. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

29. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

30. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

31. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

32. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

33. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

34. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

35. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

36. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-

quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

37. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

38. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

39. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

40. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

41. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

42. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

43. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

44. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

45. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

46. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

47. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

48. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

49. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

50. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

51. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

52. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

53. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 77 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки передачі інформації в комп’ютерних мережах критичного застосування

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
# Програмне забезпечення системи кібербезпеки передачі інформації в комп'ютерних
мережах критичного застосування
# Розроблено з урахуванням високих стандартів безпеки
# та стабільності роботи системи.

import socket
import threading
import logging
import time
import sys
import random
import os
import hashlib
import json
from datetime import datetime

# Налаштування системного логування
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s', filename='cybersecurity.log', filemode='a')

def print_banner():
# Відображення банеру системи при запуску програми
    banner = "**************************************\n" + \
" Система кібербезпеки передачі інформації\n" + \
"**************************************"
    print(banner)

class IntrusionDetection:
# Клас для виявлення та обробки спроб несанкціонованого доступу
    def __init__(self, threshold=3):
# Ініціалізація системи виявлення вторгнення з пороговим значенням
        self.failed_attempts = {}
# Ініціалізація словника для зберігання кількості невдалих спроб
        self.threshold = threshold
# Встановлення порогового значення для блокування

    def record_failed_attempt(self, ip):
# Реєстрація невдалої спроби доступу від певного IP
        if ip in self.failed_attempts:
            self.failed_attempts[ip] += 1
        else:
            self.failed_attempts[ip] = 1
# Логування невдалої спроби
        logging.warning("Невдала спроба доступу з IP: %s", ip)

    def is_ip_blocked(self, ip):
# Перевірка, чи IP заблоковано
        if ip in self.failed_attempts and self.failed_attempts[ip] >=
self.threshold:
            return True
        return False

    def reset_attempts(self, ip):
# Скидання лічильника спроб для IP
        if ip in self.failed_attempts:
            self.failed_attempts[ip] = 0

class EncryptionModule:
# Клас для шифрування та дешифрування даних
```

```
    def __init__(self, key):
# Ініціалізація модуля шифрування з заданим ключем
        self.key = key.encode('utf-8')
# Перетворення ключа у байтове представлення

    def encrypt(self, plaintext):
# Шифрування вхідного тексту
        plaintext_bytes = plaintext.encode('utf-8')
# Перетворення тексту у байтове представлення
        encrypted_bytes = self._xor_cipher(plaintext_bytes)
# Виконання XOR-шифрування
        return encrypted_bytes.hex()
# Повернення шифрованих даних у вигляді шістнадцяткового рядка

    def decrypt(self, ciphertext):
# Дешифрування шифрованих даних
        ciphertext_bytes = bytes.fromhex(ciphertext)
# Перетворення шістнадцяткового рядка у байти
        decrypted_bytes = self._xor_cipher(ciphertext_bytes)
# Виконання XOR-розшифрування
        return decrypted_bytes.decode('utf-8')
# Повернення розшифрованого тексту

    def _xor_cipher(self, data):
# Реалізація алгоритму XOR для шифрування та дешифрування
        result = bytearray()
# Ініціалізація масиву для зберігання результату
        for i in range(len(data)):
# Проходження по кожному байту вхідних даних
            result.append(data[i] ^ self.key[i % len(self.key)])
# Застосування XOR з відповідним байтом ключа
        return bytes(result)
# Повернення результату у вигляді байтового рядка

class SecureServer:
# Клас для реалізації безпечного сервера
    def __init__(self, host, port, encryption_module, intrusion_detection,
auth_token):
# Ініціалізація сервера з параметрами хоста, порту, модуля шифрування, системи
виявлення вторгнення та токеном автентифікації
        self.host = host
# Встановлення хоста сервера
        self.port = port
# Встановлення порту сервера
        self.encryption_module = encryption_module
# Призначення модуля шифрування
        self.intrusion_detection = intrusion_detection
# Призначення системи виявлення вторгнення
        self.auth_token = auth_token
# Встановлення токена автентифікації
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Створення сокет-сервера для роботи з TCP протоколом
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# Налаштування сокету для повторного використання адрес
        self.server_socket.bind((self.host, self.port))
# Прив'язка сокету до хоста та порту
        self.server_socket.listen(5)
# Прослуховування до 5 підключень одночасно
        logging.info("Сервер запущено на %s:%s", self.host, self.port)
# Логування запуску сервера

    def start_server(self):
# Метод для запуску сервера та обробки підключень
```

```
        while True:
# Нескінченний цикл для прийому підключень
            client_socket, client_address = self.server_socket.accept()
# Прийом нового підключення
            ip = client_address[0]
# Отримання IP адреси клієнта
            if self.intrusion_detection.is_ip_blocked(ip):
# Перевірка, чи не заблоковано IP адрес клієнта
                logging.error("Заблоковане підключення з IP: %s", ip)
# Логування спроби підключення з заблокованого IP
                client_socket.close()
# Закриття з'єднання з клієнтом
                continue
            client_thread = threading.Thread(target=self.handle_client,
args=(client_socket, client_address))
# Створення окремого потоку для обробки клієнта
            client_thread.daemon = True
# Встановлення потоку як демон
            client_thread.start()
# Запуск потоку для обробки підключення

    def handle_client(self, client_socket, client_address):
# Метод для обробки взаємодії з клієнтом
        ip = client_address[0]
# Отримання IP адреси клієнта
        logging.info("Нове підключення від %s", ip)
# Логування нового підключення
        authenticated = self.authenticate_client(client_socket, ip)
# Виконання автентифікації клієнта
        if not authenticated:
# Якщо автентифікація не пройшла успішно
            client_socket.close()
# Закриття з'єднання з клієнтом
            return
        while True:
# Цикл для прийому даних від клієнта
            try:
                data = client_socket.recv(1024)
# Отримання даних від клієнта
                if not data:
# Якщо дані не отримано
                    break
                simulate_network_latency()
# Симуляція затримки мережі перед обробкою даних
                decrypted_data =
self.encryption_module.decrypt(data.decode('utf-8'))
# Розшифрування отриманих даних
                logging.info("Отримано дані від %s: %s", ip, decrypted_data)
# Логування отриманих даних
                response = self.process_request(decrypted_data)
# Обробка запиту та формування відповіді
                encrypted_response = self.encryption_module.encrypt(response)
# Шифрування відповіді
                client_socket.send(encrypted_response.encode('utf-8'))
# Відправка зашифрованої відповіді клієнту
            except Exception as e:
# Обробка виключень під час взаємодії з клієнтом
                logging.error("Помилка під час обробки даних від %s: %s", ip,
str(e))
# Логування помилки
                break
        client_socket.close()
# Закриття з'єднання після завершення обробки
```

```
    def authenticate_client(self, client_socket, ip):
# Метод для автентифікації клієнта
        try:
            auth_data = client_socket.recv(1024)
# Отримання даних автентифікації від клієнта
            received_token =
self.encryption_module.decrypt(auth_data.decode('utf-8'))
# Розшифрування отриманого токена автентифікації
            if received_token.strip() == self.auth_token:
# Перевірка відповідності токена автентифікації

client_socket.send(self.encryption_module.encrypt("AUTH_SUCCESS").encode('utf-
8'))
# Відправка підтвердження успішної автентифікації
                return True
            else:
                self.intrusion_detection.record_failed_attempt(ip)
# Реєстрація невдалої спроби автентифікації

client_socket.send(self.encryption_module.encrypt("AUTH_FAILED").encode('utf-
8'))
# Відправка повідомлення про невдалу автентифікацію
                return False
        except Exception as e:
# Обробка виключень під час автентифікації
            logging.error("Помилка автентифікації для %s: %s", ip, str(e))
# Логування помилки автентифікації
            self.intrusion_detection.record_failed_attempt(ip)
# Реєстрація невдалої спроби
            return False

    def process_request(self, data):
# Метод для обробки запиту від клієнта та формування відповіді
        logging.debug("Обробка запиту: %s", data)
# Логування детальної інформації про запит
        response = "Запит оброблено успішно"
# Формування базової відповіді
        if data.lower() == "ping":
# Перевірка запиту на відповідність "ping"
            response = "pong"
# Формування відповіді "pong" для запиту "ping"
        return response
# Повернення сформованої відповіді

class SecureClient:
# Клас для реалізації безпечного клієнта
    def __init__(self, server_host, server_port, encryption_module, auth_token):
# Ініціалізація клієнта з параметрами сервера, модуля шифрування та токена
автентифікації
        self.server_host = server_host
# Встановлення хоста сервера
        self.server_port = server_port
# Встановлення порту сервера
        self.encryption_module = encryption_module
# Призначення модуля шифрування
        self.auth_token = auth_token
# Встановлення токена автентифікації для клієнта
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Створення сокету для підключення до сервера

    def connect_to_server(self):
# Метод для встановлення з'єднання з сервером
```

```
        try:
            self.client_socket.connect((self.server_host, self.server_port))
# Підключення до сервера за вказаною адресою та портом
            encrypted_token = self.encryption_module.encrypt(self.auth_token)
# Шифрування токена автентифікації
            self.client_socket.send(encrypted_token.encode('utf-8'))
# Відправка токена автентифікації на сервер
            auth_response = self.client_socket.recv(1024)
# Отримання відповіді від сервера на автентифікацію
            response_text =
self.encryption_module.decrypt(auth_response.decode('utf-8'))
# Розшифрування відповіді автентифікації
            if response_text.strip() == "AUTH_SUCCESS":
# Перевірка відповіді на успішну автентифікацію
                logging.info("Автентифікація пройшла успішно")
# Логування успішної автентифікації
                return True
            else:
                logging.error("Автентифікація не пройшла")
# Логування невдалої автентифікації
                return False
        except Exception as e:
# Обробка виключень під час підключення до сервера
            logging.error("Помилка підключення до сервера: %s", str(e))
# Логування помилки підключення
            return False

    def send_secure_message(self, message):
# Метод для відправки зашифрованого повідомлення на сервер
        try:
            encrypted_message = self.encryption_module.encrypt(message)
# Шифрування повідомлення перед відправкою
            self.client_socket.send(encrypted_message.encode('utf-8'))
# Відправка зашифрованого повідомлення на сервер
        except Exception as e:
# Обробка виключень під час відправки повідомлення
            logging.error("Помилка при відправці повідомлення: %s", str(e))
# Логування помилки відправки

    def receive_response(self):
# Метод для отримання та розшифрування відповіді від сервера
        try:
            response_data = self.client_socket.recv(1024)
# Отримання даних відповіді від сервера
            decrypted_response =
self.encryption_module.decrypt(response_data.decode('utf-8'))
# Розшифрування отриманої відповіді
            return decrypted_response
# Повернення розшифрованої відповіді
        except Exception as e:
# Обробка виключень під час отримання відповіді
            logging.error("Помилка при отриманні відповіді: %s", str(e))
# Логування помилки отримання
            return None

def simulate_network_latency():
# Симуляція затримки в мережі для реалістичності передачі даних
    delay = random.uniform(0.1, 1.0)
# Випадкова затримка між 0.1 та 1.0 секунд
    time.sleep(delay)
# Виконання затримки

def check_system_integrity():
```

```
# Симуляція перевірки цілісності системи
    integrity_status = True
# Припущення, що система у нормальному стані
    logging.info("Перевірка цілісності системи виконана успішно")
# Логування успішної перевірки
    return integrity_status

def query_security_database(query):
# Симуляція запиту до бази даних безпеки
    logging.debug("Запит до бази даних: %s", query)
# Логування запиту до бази даних
    dummy_results = {"status": "ok", "data": []}
# Повернення демо-результатів запиту
    return dummy_results

def log_security_event(event):
# Функція для логування подій безпеки
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
# Формування часової мітки для події
    logging.info("Подія безпеки [%s]: %s", timestamp, event)
# Логування події безпеки

def system_monitor():
# Функція для моніторингу системних параметрів
    while True:
# Нескінченний цикл моніторингу
        integrity = check_system_integrity()
# Перевірка цілісності системи
        if not integrity:
# Перевірка в разі порушення цілісності
            log_security_event("Порушення цілісності системи")
# Логування події порушення цілісності
        time.sleep(5)
# Затримка перед наступною перевіркою

def periodic_security_query():
# Функція для періодичного виконання запитів до бази даних безпеки
    while True:
# Нескінченний цикл для запитів до бази даних безпеки
        results = query_security_database("SELECT * FROM security_logs")
# Виконання запиту до бази даних безпеки
        logging.debug("Результати запиту до бази даних: %s", results)
# Логування результатів запиту
        time.sleep(10)
# Затримка перед наступним запитом

def detailed_logging():
# Функція для детального логування системних подій
    logging.info("Початок детального логування системних подій")
# Логування початку детального логування
    for i in range(3):
# Цикл для симуляції додаткових логів
        logging.debug("Детальний лог події номер %d", i+1)
# Логування інформації для кожного елемента циклу
        time.sleep(0.5)
# Затримка для симуляції часу обробки
    logging.info("Завершення детального логування системних подій")
# Логування завершення детального логування

def perform_system_backup():
# Функція для симуляції резервного копіювання системних даних
    logging.info("Початок резервного копіювання системних даних")
# Логування початку процесу резервного копіювання
```

```
    time.sleep(2)
# Затримка для симуляції процесу резервного копіювання
    logging.info("Резервне копіювання завершено успішно")
# Логування успішного завершення резервного копіювання

def main_menu():
# Основне меню для вибору режиму роботи програми
    print_banner()
# Відображення банеру при запуску програми
    while True:
# Нескінченний цикл для роботи основного меню
        print("\nОберіть режим роботи:")
# Виведення меню вибору режиму
        print("1. Запустити сервер")
# Варіант для запуску сервера
        print("2. Запустити клієнта")
# Варіант для запуску клієнта
        print("3. Вийти з програми")
# Варіант для виходу з програми
        choice = input("Ваш вибір: ")
# Отримання вибору користувача
        if choice == "1":
# Перевірка вибору для запуску сервера
            host = input("Введіть IP-адресу сервера (наприклад, 127.0.0.1): ")
# Отримання IP-адреси для сервера
            port = int(input("Введіть порт сервера (наприклад, 9000): "))
# Отримання номера порту для сервера
            auth_token = input("Введіть токен автентифікації: ")
# Отримання токена автентифікації для сервера
            key = input("Введіть ключ шифрування: ")
# Отримання ключа для шифрування даних
            encryption_module = EncryptionModule(key)
# Створення модуля шифрування з використанням вказаного ключа
            intrusion_detection = IntrusionDetection()
# Створення системи виявлення вторгнення
            server = SecureServer(host, port, encryption_module,
intrusion_detection, auth_token)
# Ініціалізація безпечного сервера з заданими параметрами
            log_security_event("Запуск сервера")
# Логування події запуску сервера
            server_thread = threading.Thread(target=server.start_server)
# Створення окремого потоку для запуску сервера
            server_thread.daemon = True
# Встановлення потоку як демон
            server_thread.start()
# Запуск потоку сервера
            print("Сервер запущено. Натисніть Enter для повернення до меню.")
# Повідомлення про успішний запуск сервера
            input()
# Очікування натискання клавіші для повернення до меню
        elif choice == "2":
# Перевірка вибору для запуску клієнта
            server_host = input("Введіть IP-адресу сервера для підключення: ")
# Отримання IP-адреси сервера для клієнта
            server_port = int(input("Введіть порт сервера для підключення: "))
# Отримання номера порту для підключення до сервера
            auth_token = input("Введіть токен автентифікації: ")
# Отримання токена автентифікації для клієнта
            key = input("Введіть ключ шифрування: ")
# Отримання ключа шифрування для клієнта
            encryption_module = EncryptionModule(key)
# Створення модуля шифрування для клієнта
```

```
            client = SecureClient(server_host, server_port, encryption_module,
auth_token)
# Ініціалізація безпечного клієнта з заданими параметрами
            if client.connect_to_server():
# Перевірка успішного підключення до сервера
                log_security_event("Клієнт успішно підключився до сервера")
# Логування успішного підключення клієнта
                while True:
# Цикл для взаємодії клієнта з сервером
                    message = input("Введіть повідомлення для відправки (або
'exit' для виходу): ")
# Отримання повідомлення від користувача
                    if message.lower() == "exit":
# Перевірка, чи користувач бажає вийти з режиму клієнта
                        break
                    simulate_network_latency()
# Симуляція затримки мережі перед відправкою повідомлення
                    client.send_secure_message(message)
# Відправка зашифрованого повідомлення на сервер
                    response = client.receive_response()
# Отримання відповіді від сервера
                    if response:
# Перевірка отримання відповіді
                        print("Відповідь від сервера: ", response)
# Виведення отриманої відповіді від сервера
                    else:
                        print("Не вдалося отримати відповідь від сервера")
# Повідомлення про відсутність відповіді від сервера
            else:
                print("Не вдалося підключитися до сервера")
# Повідомлення про невдале підключення до сервера
        elif choice == "3":
# Перевірка вибору для виходу з програми
            print("Вихід з програми. До побачення!")
# Повідомлення про вихід з програми
            sys.exit(0)
# Завершення роботи програми
        else:
            print("Невірний вибір. Спробуйте ще раз.")
# Повідомлення про неправильний вибір

if __name__ == '__main__':
# Точка входу в програму
    monitor_thread = threading.Thread(target=system_monitor)
# Створення потоку для моніторингу системи
    monitor_thread.daemon = True
# Встановлення потоку як демон
    monitor_thread.start()
# Запуск потоку моніторингу системи
    query_thread = threading.Thread(target=periodic_security_query)
# Створення потоку для періодичних запитів безпеки
    query_thread.daemon = True
# Встановлення потоку як демон
    query_thread.start()
# Запуск потоку періодичних запитів
    main_menu()
# Виклик основного меню для взаємодії з користувачем
```

Файл MultiLevelEncryption.py

```
import base64
import socket
import json
import threading
import time
import random
import statistics
import numpy as np

class MultiLevelEncryption:
    def __init__(self, key, shift):
        self.key = key
        self.shift = shift
    def _xor_encrypt(self, data):
        result = []
        for i in range(len(data)):
            result.append(chr(ord(data[i]) ^ ord(self.key[i % len(self.key)])))
        return ''.join(result)
    def _caesar_encrypt(self, data):
        result = []
        for char in data:
            result.append(chr((ord(char) + self.shift) % 256))
        return ''.join(result)
    def _reverse(self, data):
        return data[::-1]
    def _base64_encode(self, data):
        encoded = base64.b64encode(data.encode('utf-8'))
        return encoded.decode('utf-8')
    def encrypt(self, plaintext):
        step1 = self._xor_encrypt(plaintext)
        step2 = self._caesar_encrypt(step1)
        step3 = self._reverse(step2)
        step4 = self._base64_encode(step3)
        return step4
    def _base64_decode(self, data):
        decoded = base64.b64decode(data.encode('utf-8'))
        return decoded.decode('utf-8', errors='ignore')
    def _caesar_decrypt(self, data):
        result = []
        for char in data:
            result.append(chr((ord(char) - self.shift) % 256))
        return ''.join(result)
    def decrypt(self, ciphertext):
        step1 = self._base64_decode(ciphertext)
        step2 = self._reverse(step1)
        step3 = self._caesar_decrypt(step2)
        step4 = self._xor_encrypt(step3)
        return step4

class SIEMIntegration:
    def __init__(self, siem_host, siem_port):
        self.siem_host = siem_host
        self.siem_port = siem_port
        self.log_queue = []
        self.lock = threading.Lock()
    def add_log(self, log_entry):
        with self.lock:
            self.log_queue.append(log_entry)
    def flush_logs(self):
        with self.lock:
            logs_to_send = list(self.log_queue)
```

```
            self.log_queue = []
        if logs_to_send:
            self._send_to_siem(logs_to_send)
    def _send_to_siem(self, logs):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((self.siem_host, self.siem_port))
            payload = json.dumps({"logs": logs})
            s.sendall(payload.encode('utf-8'))
            response = s.recv(1024)
            s.close()
            return response.decode('utf-8')
        except Exception as e:
            return None
    def start_periodic_flush(self, interval):
        def flush_loop():
            while True:
                self.flush_logs()
                time.sleep(interval)
        t = threading.Thread(target=flush_loop)
        t.daemon = True
        t.start()

class NetworkTrafficAnalyzer:
    def __init__(self):
        self.traffic_data = []
    def simulate_packet(self):
        packet = {"src": self._random_ip(), "dst": self._random_ip(), "size":
random.randint(20,1500), "timestamp": time.time()}
        return packet
    def _random_ip(self):
        return ".".join(str(random.randint(0,255)) for _ in range(4))
    def collect_traffic(self, count):
        for i in range(count):
            packet = self.simulate_packet()
            self.traffic_data.append(packet)
            time.sleep(0.01)
    def analyze_traffic(self):
        sizes = [packet["size"] for packet in self.traffic_data]
        if sizes:
            avg = statistics.mean(sizes)
            med = statistics.median(sizes)
            stdev = statistics.stdev(sizes) if len(sizes) > 1 else 0
            anomalies = [p for p in self.traffic_data if p["size"] > avg +
2*stdev or p["size"] < avg - 2*stdev]
            return {"average": avg, "median": med, "stdev": stdev, "anomalies":
anomalies}
        else:
            return {"average": 0, "median": 0, "stdev": 0, "anomalies": []}
    def clear_data(self):
        self.traffic_data = []

class CentralizedAccessControl:
    def __init__(self):
        self.users = {}
        self.roles = {}
        self.permissions = {}
    def add_user(self, username, password):
        self.users[username] = {"password": password, "roles": []}
    def remove_user(self, username):
        if username in self.users:
            del self.users[username]
    def add_role(self, role_name):
```

```
            if role_name not in self.roles:
                self.roles[role_name] = []
        def assign_role_to_user(self, username, role_name):
            if username in self.users and role_name in self.roles:
                if role_name not in self.users[username]["roles"]:
                    self.users[username]["roles"].append(role_name)
        def add_permission(self, permission_name):
            if permission_name not in self.permissions:
                self.permissions[permission_name] = []
        def assign_permission_to_role(self, role_name, permission_name):
            if role_name in self.roles and permission_name in self.permissions:
                if permission_name not in self.roles[role_name]:
                    self.roles[role_name].append(permission_name)
        def check_permission(self, username, permission_name):
            if username not in self.users:
                return False
            user_roles = self.users[username]["roles"]
            for role in user_roles:
                if role in self.roles and permission_name in self.roles[role]:
                    return True
            return False
        def list_users(self):
            return list(self.users.keys())
        def list_roles(self):
            return list(self.roles.keys())
        def list_permissions(self):
            return list(self.permissions.keys())

    class MLLogAnalyzer:
        def __init__(self, learning_rate=0.01, epochs=1000):
            self.learning_rate = learning_rate
            self.epochs = epochs
            self.weights = None
            self.bias = 0
        def _sigmoid(self, z):
            return 1 / (1 + np.exp(-z))
        def train(self, X, y):
            n_samples, n_features = X.shape
            self.weights = np.zeros(n_features)
            self.bias = 0
            for epoch in range(self.epochs):
                linear_model = np.dot(X, self.weights) + self.bias
                y_predicted = self._sigmoid(linear_model)
                dw = (1 / n_samples) * np.dot(X.T, (y_predicted - y))
                db = (1 / n_samples) * np.sum(y_predicted - y)
                self.weights -= self.learning_rate * dw
                self.bias -= self.learning_rate * db
        def predict_prob(self, X):
            linear_model = np.dot(X, self.weights) + self.bias
            return self._sigmoid(linear_model)
        def predict(self, X, threshold=0.5):
            probabilities = self.predict_prob(X)
            return np.array([1 if p >= threshold else 0 for p in probabilities])
        def vectorize_logs(self, logs):
            vocab = {}
            for log in logs:
                tokens = log.split()
                for token in tokens:
                    if token not in vocab:
                        vocab[token] = len(vocab)
            X = np.zeros((len(logs), len(vocab)))
            for i, log in enumerate(logs):
                tokens = log.split()
```

```
                for token in tokens:
                    if token in vocab:
                        X[i][vocab[token]] += 1
        return X, vocab
    def evaluate(self, X, y):
        predictions = self.predict(X)
        accuracy = np.sum(predictions == y) / len(y)
        return accuracy

def main():
    mlencrypt = MultiLevelEncryption("secretkey", 3)
    plaintext = "Sensitive data for encryption"
    encrypted_text = mlencrypt.encrypt(plaintext)
    decrypted_text = mlencrypt.decrypt(encrypted_text)
    print("Original:", plaintext)
    print("Encrypted:", encrypted_text)
    print("Decrypted:", decrypted_text)
    siem = SIEMIntegration("127.0.0.1", 5000)
    for i in range(10):
        siem.add_log({"event": "event" + str(i), "value": i, "timestamp":
time.time()})
    siem.flush_logs()
    siem.start_periodic_flush(5)
    nta = NetworkTrafficAnalyzer()
    nta.collect_traffic(100)
    analysis_results = nta.analyze_traffic()
    print("Traffic Analysis:", analysis_results)
    nta.clear_data()
    cac = CentralizedAccessControl()
    cac.add_user("alice", "alicepwd")
    cac.add_user("bob", "bobpwd")
    cac.add_role("admin")
    cac.add_role("user")
    cac.assign_role_to_user("alice", "admin")
    cac.assign_role_to_user("bob", "user")
    cac.add_permission("read")
    cac.add_permission("write")
    cac.assign_permission_to_role("admin", "read")
    cac.assign_permission_to_role("admin", "write")
    cac.assign_permission_to_role("user", "read")
    print("Alice can write:", cac.check_permission("alice", "write"))
    print("Bob can write:", cac.check_permission("bob", "write"))
    print("Users:", cac.list_users())
    print("Roles:", cac.list_roles())
    print("Permissions:", cac.list_permissions())
    logs = [
        "Error occurred in module1",
        "Warning in module2",
        "Normal operation module3",
        "Error in module4",
        "Critical failure in module5",
        "Info: module6 running",
        "Error: module7 timeout",
        "Warning: module8 latency high",
        "Info: module9 recovered",
        "Critical error module10"
    ]
    ml_analyzer = MLLogAnalyzer(learning_rate=0.1, epochs=500)
    X, vocab = ml_analyzer.vectorize_logs(logs)
    y = np.array([1 if "Error" in log or "Critical" in log else 0 for log in
logs])
    ml_analyzer.train(X, y)
    predictions = ml_analyzer.predict(X)
```

```
    accuracy = ml_analyzer.evaluate(X, y)
    print("ML Log Predictions:", predictions)
    print("ML Log Accuracy:", accuracy)

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

Файл register_user.py

```
import os
import socket
import threading
import time
import random
import hashlib
import json
import base64
import shutil

class BiometricAuthentication:
    def __init__(self):
        self.registered_users = {}
    def register_user(self, username, biometric_data):
        hashed_data = hashlib.sha256(biometric_data.encode('utf-8')).hexdigest()
        self.registered_users[username] = hashed_data
    def authenticate(self, username, biometric_data):
        if username not in self.registered_users:
            return False
        hashed_data = hashlib.sha256(biometric_data.encode('utf-8')).hexdigest()
        return self.registered_users[username] == hashed_data
    def simulate_scan(self):
        sample_data = "fingerprint" + str(random.randint(1000, 9999))
        return sample_data
    def list_users(self):
        return list(self.registered_users.keys())
    def update_biometric(self, username, biometric_data):
        if username in self.registered_users:
            hashed_data = hashlib.sha256(biometric_data.encode('utf-
8')).hexdigest()
            self.registered_users[username] = hashed_data
            return True
        return False
    def remove_user(self, username):
        if username in self.registered_users:
            del self.registered_users[username]
            return True
        return False

class VPNIntegration:
    def __init__(self, server, port):
        self.server = server
        self.port = port
        self.connected = False
        self.connection_socket = None
    def connect(self):
        try:
            self.connection_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
            self.connection_socket.connect((self.server, self.port))
            self.connected = True
        except Exception as e:
            self.connected = False
    def disconnect(self):
```

```
        if self.connection_socket:
            self.connection_socket.close()
            self.connected = False
    def send_data(self, data):
        if not self.connected:
            return None
        try:
            encoded = data.encode('utf-8')
            self.connection_socket.sendall(encoded)
            response = self.connection_socket.recv(2048)
            return response.decode('utf-8')
        except Exception as e:
            return None
    def simulate_vpn_tunnel(self):
        time.sleep(random.uniform(0.5, 2.0))
        return True
    def is_connected(self):
        return self.connected
    def reconnect(self):
        self.disconnect()
        time.sleep(1)
        self.connect()

class AutomatedBackup:
    def __init__(self, source_dir, backup_dir, interval):
        self.source_dir = source_dir
        self.backup_dir = backup_dir
        self.interval = interval
        self.running = False
    def start_backup(self):
        self.running = True
        t = threading.Thread(target=self._backup_loop)
        t.daemon = True
        t.start()
    def stop_backup(self):
        self.running = False
    def _backup_loop(self):
        while self.running:
            self.create_backup()
            time.sleep(self.interval)
    def create_backup(self):
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        dest = os.path.join(self.backup_dir, "backup_" + timestamp)
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
        shutil.copytree(self.source_dir, dest)
    def list_backups(self):
        if not os.path.exists(self.backup_dir):
            return []
        return [f for f in os.listdir(self.backup_dir) if
f.startswith("backup_")]
    def restore_backup(self, backup_name):
        backup_path = os.path.join(self.backup_dir, backup_name)
        if os.path.exists(backup_path):
            for item in os.listdir(self.source_dir):
                item_path = os.path.join(self.source_dir, item)
                if os.path.isfile(item_path):
                    os.remove(item_path)
                elif os.path.isdir(item_path):
                    shutil.rmtree(item_path)
            for item in os.listdir(backup_path):
                s = os.path.join(backup_path, item)
                d = os.path.join(self.source_dir, item)
```

```
                if os.path.isdir(s):
                    shutil.copytree(s, d)
                else:
                    shutil.copy2(s, d)
            return True
        return False
    def backup_status(self):
        backups = self.list_backups()
        return {"total_backups": len(backups), "last_backup": backups[-1] if
backups else None}

class DDoSProtection:
    def __init__(self, threshold, window):
        self.threshold = threshold
        self.window = window
        self.requests = {}
    def log_request(self, ip):
        current_time = time.time()
        if ip not in self.requests:
            self.requests[ip] = []
        self.requests[ip].append(current_time)
        self.requests[ip] = [t for t in self.requests[ip] if current_time - t <=
self.window]
    def is_ddos(self, ip):
        if ip not in self.requests:
            return False
        return len(self.requests[ip]) > self.threshold
    def reset_ip(self, ip):
        if ip in self.requests:
            self.requests[ip] = []
    def block_ip(self, ip):
        self.requests[ip] = [float('inf')]
    def unblock_ip(self, ip):
        if ip in self.requests and float('inf') in self.requests[ip]:
            self.requests[ip] = []
    def simulate_request(self, ip, count):
        for _ in range(count):
            self.log_request(ip)
            time.sleep(0.1)
    def monitor_ips(self):
        monitored = {}
        for ip, times in self.requests.items():
            monitored[ip] = len(times)
        return monitored

class VulnerabilityScanner:
    def __init__(self, target):
        self.target = target
        self.open_ports = []
        self.vulnerabilities = {}
    def scan_ports(self, start_port, end_port):
        self.open_ports = []
        for port in range(start_port, end_port + 1):
            try:
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s.settimeout(0.1)
                result = s.connect_ex((self.target, port))
                if result == 0:
                    self.open_ports.append(port)
                s.close()
            except Exception as e:
                continue
        return self.open_ports
```

```
    def check_vulnerabilities(self):
        self.vulnerabilities = {}
        for port in self.open_ports:
            if port == 21:
                self.vulnerabilities[port] = "FTP vulnerability detected"
            elif port == 22:
                self.vulnerabilities[port] = "SSH vulnerability detected"
            elif port == 80:
                self.vulnerabilities[port] = "HTTP vulnerability detected"
            elif port == 443:
                self.vulnerabilities[port] = "HTTPS vulnerability detected"
            else:
                self.vulnerabilities[port] = "No known vulnerability"
        return self.vulnerabilities
    def detailed_scan(self):
        details = {}
        for port in self.open_ports:
            detail = {"banner": self.get_banner(port), "status": "open",
"vulnerability": self.vulnerabilities.get(port, "Unknown")}
            details[port] = detail
        return details
    def get_banner(self, port):
        try:
            s = socket.socket()
            s.settimeout(1)
            s.connect((self.target, port))
            banner = s.recv(1024).decode('utf-8', errors='ignore')
            s.close()
            return banner.strip()
        except Exception as e:
            return ""
    def full_scan(self, start_port, end_port):
        self.scan_ports(start_port, end_port)
        self.check_vulnerabilities()
        return self.detailed_scan()

def main():
    bio = BiometricAuthentication()
    bio.register_user("user1", bio.simulate_scan())
    bio.register_user("user2", bio.simulate_scan())
    result1 = bio.authenticate("user1", bio.simulate_scan())
    result2 = bio.authenticate("user2", bio.simulate_scan())
    print("Biometric auth user1:", result1)
    print("Biometric auth user2:", result2)
    vpn = VPNIntegration("127.0.0.1", 8080)
    vpn.connect()
    vpn.simulate_vpn_tunnel()
    vpn_response = vpn.send_data("Test VPN data")
    vpn.disconnect()
    print("VPN response:", vpn_response)
    backup = AutomatedBackup("source_dir", "backup_dir", 10)
    if not os.path.exists("source_dir"):
        os.makedirs("source_dir")
    with open("source_dir/data.txt", "w") as f:
        f.write("Backup test data")
    backup.create_backup()
    backups_list = backup.list_backups()
    backup_status = backup.backup_status()
    print("Backups list:", backups_list)
    print("Backup status:", backup_status)
    ddos = DDoSProtection(5, 2)
    ddos.simulate_request("192.168.1.100", 7)
    ddos_status = ddos.is_ddos("192.168.1.100")
```

```
    monitored_ips = ddos.monitor_ips()
    print("DDoS status for 192.168.1.100:", ddos_status)
    print("Monitored IPs:", monitored_ips)
    scanner = VulnerabilityScanner("127.0.0.1")
    open_ports = scanner.scan_ports(20, 100)
    vulnerabilities = scanner.check_vulnerabilities()
    detailed = scanner.detailed_scan()
    full = scanner.full_scan(20, 100)
    print("Open ports:", open_ports)
    print("Vulnerabilities:", vulnerabilities)
    print("Detailed scan:", detailed)
    print("Full scan:", full)

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

Файл verify.py

```
import time, hmac, base64, hashlib, random, string, threading, json, socket, os

class MultifactorAuthentication:
    def __init__(self):
        self.users = {}
    def register_user(self, username, password, phone, email):
        secret = ''.join(random.choices(string.ascii_letters + string.digits,
k=16))
        self.users[username] = {'password': password, 'phone': phone, 'email':
email, 'secret': secret}
    def verify_password(self, username, password):
        if username in self.users and self.users[username]['password'] ==
password:
            return True
        return False
    def generate_otp(self, username):
        if username not in self.users:
            return None
        secret = self.users[username]['secret']
        timestep = int(time.time() // 30)
        msg = timestep.to_bytes(8, 'big')
        key = secret.encode('utf-8')
        h = hmac.new(key, msg, hashlib.sha1).digest()
        o = h[19] & 15
        otp = (int.from_bytes(h[o:o+4], 'big') & 0x7fffffff) % 1000000
        return str(otp).zfill(6)
    def send_otp_sms(self, username):
        otp = self.generate_otp(username)
        if otp:
            return "SMS sent to {}: {}".format(self.users[username]['phone'],
otp)
        return "User not found"
    def send_otp_email(self, username):
        otp = self.generate_otp(username)
        if otp:
            return "Email sent to {}: {}".format(self.users[username]['email'],
otp)
        return "User not found"
    def verify_otp(self, username, otp_input):
        otp = self.generate_otp(username)
        return otp == otp_input

class ExtendedEventLogging:
    def __init__(self):
        self.logs = []
    def log_event(self, event_type, message, severity, timestamp=None):
        if timestamp is None:
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        entry = {"type": event_type, "message": message, "severity": severity,
"timestamp": timestamp}
        self.logs.append(entry)
    def get_logs(self):
        return self.logs
    def filter_logs(self, severity=None, event_type=None):
        result = []
        for log in self.logs:
            if severity and log["severity"] != severity:
                continue
            if event_type and log["type"] != event_type:
                continue
            result.append(log)
```

```
        return result
    def export_logs_to_file(self, filename):
        with open(filename, "w") as f:
            json.dump(self.logs, f)
    def import_logs_from_file(self, filename):
        if os.path.exists(filename):
            with open(filename, "r") as f:
                self.logs = json.load(f)
    def clear_logs(self):
        self.logs = []

class NetworkEventMonitor:
    def __init__(self):
        self.events = []
    def simulate_event(self, device_id, event_code, description):
        event = {"device_id": device_id, "event_code": event_code,
"description": description, "timestamp": time.time()}
        self.events.append(event)
    def generate_random_events(self, count):
        for i in range(count):
            device_id = "Device-" + str(random.randint(1, 10))
            event_code = random.choice(["ERROR", "WARNING", "INFO", "DEBUG"])
            description = "Event {} from {}".format(i, device_id)
            self.simulate_event(device_id, event_code, description)
            time.sleep(0.01)
    def get_events(self):
        return self.events
    def filter_events_by_code(self, event_code):
        return [event for event in self.events if event["event_code"] ==
event_code]
    def clear_events(self):
        self.events = []
    def alert_if_critical(self, threshold):
        critical_events = [e for e in self.events if e["event_code"] == "ERROR"]
        if len(critical_events) > threshold:
            return "Critical alert: {} error
events".format(len(critical_events))
        return "Normal"
    def export_events(self, filename):
        with open(filename, "w") as f:
            json.dump(self.events, f)

class AutomatedIncidentResponse:
    def __init__(self):
        self.incidents = []
    def record_incident(self, incident_id, description, severity):
        incident = {"id": incident_id, "description": description, "severity":
severity, "status": "new", "timestamp": time.time()}
        self.incidents.append(incident)
    def list_incidents(self):
        return self.incidents
    def escalate_incident(self, incident_id):
        for incident in self.incidents:
            if incident["id"] == incident_id:
                incident["severity"] = "High"
                incident["status"] = "escalated"
    def resolve_incident(self, incident_id):
        for incident in self.incidents:
            if incident["id"] == incident_id:
                incident["status"] = "resolved"
    def auto_respond(self):
        for incident in self.incidents:
```

```
            if incident["status"] == "new" and incident["severity"] in ["High",
"Critical"]:
                incident["status"] = "auto-responded"
                self.trigger_response_actions(incident)
    def trigger_response_actions(self, incident):
        actions = []
        if incident["severity"] == "Critical":
            actions.append("Shutdown affected systems")
            actions.append("Notify all admins")
        else:
            actions.append("Restart affected services")
            actions.append("Send notification to support")
        incident["actions"] = actions
    def export_incidents(self, filename):
        with open(filename, "w") as f:
            json.dump(self.incidents, f)
    def import_incidents(self, filename):
        if os.path.exists(filename):
            with open(filename, "r") as f:
                self.incidents = json.load(f)

class CloudServicesIntegration:
    def __init__(self):
        self.providers = {}
    def add_provider(self, name, config):
        self.providers[name] = config
    def list_providers(self):
        return list(self.providers.keys())
    def upload_file(self, provider_name, file_path, destination):
        if provider_name not in self.providers:
            return "Provider not found"
        time.sleep(random.uniform(0.1, 0.5))
        return "File {} uploaded to {} on provider {}".format(file_path,
destination, provider_name)
    def download_file(self, provider_name, file_path, destination):
        if provider_name not in self.providers:
            return "Provider not found"
        time.sleep(random.uniform(0.1, 0.5))
        return "File {} downloaded from provider {} to {}".format(file_path,
provider_name, destination)
    def list_files(self, provider_name, directory):
        if provider_name not in self.providers:
            return []
        files = ["file" + str(i) + ".txt" for i in range(1, random.randint(3,
10))]
        return files
    def create_instance(self, provider_name, instance_type):
        if provider_name not in self.providers:
            return "Provider not found"
        instance_id = provider_name + "_" +
''.join(random.choices(string.ascii_lowercase + string.digits, k=8))
        return "Instance created: " + instance_id
    def delete_instance(self, provider_name, instance_id):
        if provider_name not in self.providers:
            return "Provider not found"
        return "Instance {} deleted from provider {}".format(instance_id,
provider_name)
    def get_instance_status(self, provider_name, instance_id):
        if provider_name not in self.providers:
            return "Provider not found"
        status = random.choice(["running", "stopped", "terminated"])
        return "Instance {} is {}".format(instance_id, status)
```

```
def main():
    mfa = MultifactorAuthentication()
    mfa.register_user("john", "pass123", "+1234567890", "john@example.com")
    mfa.register_user("jane", "pass456", "+0987654321", "jane@example.com")
    password_check = mfa.verify_password("john", "pass123")
    otp_sms = mfa.send_otp_sms("john")
    otp_email = mfa.send_otp_email("jane")
    otp_john = mfa.generate_otp("john")
    otp_verify = mfa.verify_otp("john", otp_john)
    print("Password check:", password_check)
    print("OTP SMS:", otp_sms)
    print("OTP Email:", otp_email)
    print("OTP Verify:", otp_verify)
    logger = ExtendedEventLogging()
    logger.log_event("LOGIN", "User john logged in", "INFO")
    logger.log_event("ACCESS", "User jane accessed secure area", "WARNING")
    logger.log_event("ERROR", "Failed login attempt for user john", "ERROR")
    all_logs = logger.get_logs()
    filtered_logs = logger.filter_logs(severity="ERROR")
    logger.export_logs_to_file("logs.json")
    logger.clear_logs()
    logger.import_logs_from_file("logs.json")
    print("All Logs:", all_logs)
    print("Filtered Logs:", filtered_logs)
    monitor = NetworkEventMonitor()
    monitor.generate_random_events(50)
    events = monitor.get_events()
    error_events = monitor.filter_events_by_code("ERROR")
    alert_status = monitor.alert_if_critical(5)
    monitor.export_events("events.json")
    print("Total Events:", len(events))
    print("Error Events:", len(error_events))
    print("Alert Status:", alert_status)
    incident = AutomatedIncidentResponse()
    incident.record_incident("INC001", "Data breach detected", "Critical")
    incident.record_incident("INC002", "Unauthorized access attempt", "High")
    incident.record_incident("INC003", "Minor service disruption", "Low")
    incident.export_incidents("incidents.json")
    incidents_list = incident.list_incidents()
    print("Incidents:", incidents_list)
    cloud = CloudServicesIntegration()
    cloud.add_provider("AWS", {"region": "us-east-1", "api_key": "AKIA...",
"secret": "SECRET"})
    cloud.add_provider("Azure", {"region": "eastus", "client_id": "CLIENTID",
"secret": "SECRET"})
    providers = cloud.list_providers()
    upload_result = cloud.upload_file("AWS", "data.txt", "/backup/")
    download_result = cloud.download_file("Azure", "report.pdf", "/local/")
    files_list = cloud.list_files("AWS", "/data/")
    instance_id = cloud.create_instance("AWS", "t2.micro")
    status = cloud.get_instance_status("AWS", instance_id)
    delete_result = cloud.delete_instance("AWS", instance_id)
    print("Cloud Providers:", providers)
    print("Upload Result:", upload_result)
    print("Download Result:", download_result)
    print("Files List:", files_list)
    print("Instance ID:", instance_id)
    print("Instance Status:", status)
    print("Delete Result:", delete_result)

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