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 СКУД підприємства з використанням proximity-карт”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Осипенко Д.С.
«____» _____ 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

Осипенку Даніелю Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт*

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 СКУД підприємства з використанням proximity-карт*

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
_____ Осипенко Д.С.
(прізвище та ініціали)

АНОТАЦІЯ

Осипенко Д.С. Програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт. 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 СКУД підприємства з використанням proximity-карт.

Метою розробки є програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт.

Результат роботи – програмна реалізація системи кібербезпеки СКУД підприємства з використанням proximity-карт.

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, СКУД

ABSTRACT

Osypenko D.S. Software for the enterprise ACS cybersecurity system using proximity cards. 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 that is intended for the enterprise ACS cybersecurity system using proximity cards.

The purpose of the development is software for the enterprise ACS cybersecurity system using proximity cards.

The result of the work is the software implementation of the enterprise ACS cybersecurity system using proximity cards.

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, ACS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ – автоматизована система управління

ДСТ – держстандарт

ЕМЗ – електромагнітний замок

ЕОМ – електроно-обчислювальна машина

ІСО – міжнародний стандарт

ОПС – охоронно-пожежна сигналізація

ПЗ – програмне забезпечення

ПЗП – постійний запам’ятовуючий пристрій

ПК – програмний комплекс

СКУД – система контролю та управління доступом

СУД – система управління доступом

PIN – personal identification cod – персональний ідентифікаційний код

RTE – Request To Exit – кнопка «Вихід»

ВСТУП

Актуальність теми. Контроль доступу є важливим компонентом інформаційних технологій (ІТ) і кібербезпеки. Це механізм, який регулює, хто або що може переглядати, використовувати або отримувати доступ до певного ресурсу в обчислювальному середовищі. Основною метою є мінімізація ризиків безпеки, гарантуючи, що лише авторизовані користувачі, системи чи служби мають доступ до необхідних ресурсів.

Контроль доступу — це не лише дозвіл або заборона доступу. Це включає в себе ідентифікацію особи або системи, автентифікацію їхньої особистості, авторизацію для доступу до ресурсу та перевірку їх шаблонів доступу. Цей процес мінімізує ризик несанкціонованого доступу, захищаючи конфіденційну інформацію та системи.

Сучасна ІТ-інфраструктура та моделі роботи створюють нові виклики контролю доступу. Такі тенденції, як використання хмарних обчислень, зростання використання мобільних пристроїв на робочому місці та перехід до видалення роботи, означають, що кількість точок доступу до організації зростає експоненціально. Нові технології, такі як керування ідентифікацією та доступом (IAM), і такі підходи, як нульова довіра, допомагають керувати цією складністю та запобігати неавторизованому доступу

У даній роботі для реалізації СКУД пропонується використовувати проксіміті (proximity) карти (безконтактні карти доступу).

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт.

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

– Огляд існуючих систем СКУД підприємства з використанням proximity-карт.

– Дослідження системи кібербезпеки СКУД підприємства з використанням proximity-карт.

– Програмна реалізація системи кібербезпеки СКУД підприємства з використанням proximity-карт.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі СКУД підприємства з використанням proximity-карт.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт, 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

Деякі закордонні компанії, намагаючись заповнити поки ще вільну нішу українського ринку, часом проявляють несумлінність у рекламі, у наданні повної інформації про технічні й функціональні можливості систем, про особливості їхньої експлуатації в порівняно складних кліматичних умовах і т.п. Найчастіше постачальники й продавці заради прибутку пропонують замовникові апаратуру низької якості й некваліфіковані послуги. Повсюдно й самі покупці не мають достатнього досвіду в цій сфері. У результаті на важливих об'єктах можна зустріти непрофесійно спроектовані системи СКУД, у яких навіть технічні характеристики не відповідають умовам експлуатації в Україні. Крім усього цього серед вітчизняних користувачів систем не вистачає висококваліфікованих фахівців, здатних на високому рівні здійснювати технічне обслуговування й у стислий термін усувати дефекти встаткування.

1.2 Область застосування

Кіберзлочинці стають все більш досвідченими, використовуючи передові методи для зламу систем безпеки та отримання несанкціонованого доступу до ресурсів.

Контроль доступу – це проактивний захід безпеки, який допомагає стримувати, виявляти та запобігати неавторизованому доступу. Контролюючи, хто або що має доступ до ресурсу, він гарантує, що лише ті, хто має необхідні дозволи, можуть отримати доступ до даних або служби. Це значно знижує ризик порушення безпеки як зовнішніми зловмисниками, так і внутрішніми загрозами.

Крім того, контроль доступу в безпеці має вирішальне значення для дотримання різноманітних нормативних вимог. Такі нормативні акти, як

Загальний регламент захисту даних (GDPR) і Закон про перенесення та підзвітність медичного страхування (HIPAA), вимагають від організацій застосовувати суворі заходи контролю доступу для захисту персональних даних. Невиконання вимог може призвести до серйозних штрафів і шкоди репутації.

Ось загальний процес забезпечення доступу та управління контролем доступу в організації:

1. Автентифікація. Автентифікація є першим кроком у контролі доступу. Він передбачає перевірку особи користувача або системи, яка запитує доступ. Зазвичай це робиться шляхом зіставлення наданих облікових даних із збереженою інформацією. Методи автентифікації включають автентифікацію на основі пароля, біометрії та сертифіката.

2. Авторизація: Авторизація слідує за успішною автентифікацією. Він передбачає надання або заборону доступу на основі привілеїв користувача або системи. Привілеї попередньо визначені та визначають, до яких ресурсів користувач або система може отримати доступ і в якому обсязі. Авторизація допомагає підтримувати принцип найменших привілеїв, гарантуючи, що користувачі та системи мають лише необхідний доступ.

3. Доступ: Доступ стосується фактичного використання або взаємодії з ресурсом. Це може включати перегляд, зміну чи видалення даних або використання служби. Ступінь доступу визначається процесом авторизації. Доступ відстежується та контролюється для запобігання несанкціонованим діям.

4. Керувати: Управління контролем доступу передбачає підтримку та оновлення системи контролю доступу. Це включає визначення та оновлення політик доступу, керування обліковими даними користувачів, підключення та відключення користувачів, а також підтримку апаратного та програмного забезпечення контролю доступу. Ефективне управління гарантує, що система контролю доступу залишається надійною та оновленою.

5. Аудит: Аудит є важливою складовою контролю доступу. Це передбачає моніторинг і запис моделей доступу та дій. Аудит допомагає виявити будь-які

незвичайні або підозрілі дії та допомагає в судово-медичних розслідуваннях. Регулярні перевірки можуть виявити вразливі місця в безпеці та допомогти покращити систему контролю доступу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки СКУД підприємства з використанням proximity-карт, 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

СКУД Castle

Професійна система контролю й управління доступом (СКУД) Castle дозволяє організувати контроль доступу в приміщення, будинки або групи будинків через необмежене число дверей, турнікетів необмеженій кількості співробітників і відвідувачів. Дана система контролю доступу являє собою апаратно-програмний комплекс, що складається з різноманітних асортиментів контролерів і додаткового програмного забезпечення, має широкі інтеграційні здатності, що дозволяє здійснювати інтеграцію встаткування Castle, приміром, із системами ITV, і може використовуватися для створення розподілених систем безпеки, що включають у себе відеоспостереження, ОПС, СКУД з використанням цифрових відеореєстраторів і відеосерверів. На сьогоднішній день Castle – рішення з унікальною технологією синхронізації баз даних, що забезпечує створення класичних і біометричних СКУД для окремих і територіально вилучених об'єктів однієї компанії.

СКУД Castle – це:

– Головні достоїнства системи – якість і ціни, а також модульність. Система Castle відповідає всім вимогам сучасних систем контролю й управління доступом, при цьому ціни на контролери й ПЗ залишаються на низькому рівні.

– Широкий вибір контролерів для рішення будь-яких завдань. Контролери Castle є мережними. Всі одержувані події зберігаються в енергонезалежній пам'яті контролера. Контролери можуть підключатися через TCP/IP, RS485, USB.

– ПЗ Castle має модульну архітектуру. Всі встановлені модулі інтегруються в єдиний користувальницький інтерфейс. Додаткові модулі можуть у будь-який момент додаватися у вже встановлену систему.

– Гнучка цінова політика. Устаткування завжди в наявності на складі, тому поставка можлива в мінімальний термін.

– Повна інформаційна підтримка партнерів по реалізації.

Система контролю доступу (СКУД) призначена для забезпечення санкціонованого проходу в приміщення, які перебувають під охороною, контролю доступу в дані приміщення й запобігання несанкціонованого проникнення. Система контролю доступу дозволяє організувати прохід співробітників на охоронюваний об'єкт за допомогою проксіміті карт – безконтактних карт, біометричних зчитувачів (зчитувачі відбитка пальців), організувати прохід співробітників на підприємстві або в офісі через двері, турнікети, шлагбауми й хвіртки.

СКУД Castle повністю виключає вплив людського фактора на пропускний режим, тому що контроль доступу покладає на встаткування. При цьому Castle дозволяє розмежувати доступ у приміщення, тобто дозволити даному співробітникові прохід тільки в певні приміщення відповідно до допусків.

Система дозволяє знімати приміщення з охорони й ставити їх на охорону. У випадку виникнення позаштатної ситуації система видасть тривогу. Всі події, тривоги, постановка, зняття з охорони фіксуються в журналі подій і надовго зберігаються в системі, а також у пам'яті контролерів.

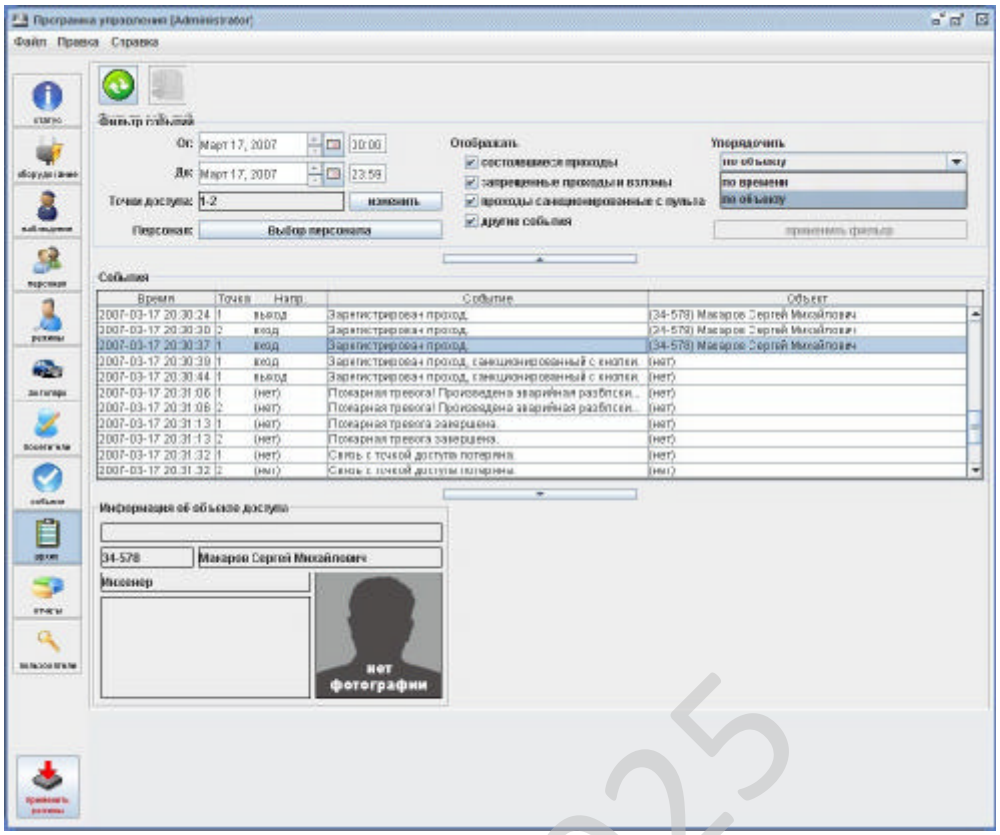

Рисунок 2.1 – Інтерфейс користувача СКУД Castle

Система контролю доступу дозволяє фіксувати дату й час проходу співробітників у приміщення, здійснювати облік робочого часу й ідентифікацію співробітника за допомогою візуального порівняння особи, які скористувалися картою, з оригінальною фотографією власника даної карти, а також за даними, отриманим з камери відеоспостереження, здійснювати контроль переміщення співробітників по охоронюваному об'єкті, забезпечувати захист від передачі картки іншій особі за допомогою системи контролю за повторним входом.

СКУД дозволяє заборонити прохід співробітників у святкові, вихідні дні й після закінчення робочого дня, організувати базу даних для кожного співробітника або відвідувача, одержувати в реальному часі інформацію про різні позаштатні й тривожні ситуації.

СКУД Castle може бути інтегрована із системами безпеки ITV на базі відеосерверів "Videосервер". Грамотна інтеграція СКУД, приміром, із системою

відеоспостереження «Інтелект» дозволяє повністю контролювати ситуацію на об'єкті й у випадку виникнення позаштатної ситуації дана сукупність систем дозволяє в найкоротший термін виявити порушника. Завдяки наявності тривожних входів і виходів СКУД може бути інтегрована із системою охоронної сигналізація. Це дозволяє при проникненні в приміщення включати сирену, лампу або інший сигнал. Також система контролю доступу може бути інтегрована із системою пожежної сигналізації (ОПС), що дозволяє розблокувати двері, відкрити ворота, опустити турнікети у випадку пожежі для евакуації персоналу.

СКУД «Castle» може запропонувати безліч варіантів реалізації, контролю доступу від офісу з п'ятьома співробітниками до забезпечення безпеки підприємства з десятками тисяч співробітників.

Основні переваги СКУД «Castle»

Висока надійність за рахунок:

– Найвищого рівня захисту від перешкод. Кожний контролер має гальванічну розв'язку й повноцінний грозозахист лінії зв'язку, захист входів/виходів від перенапруги, перевантаження й переполюсовки, що мінімізує ризик виходу з ладу встаткування при спробах злому, помилках монтажу й забезпечується безперебійна робота в умовах сильних зовнішніх перешкод.

– Вся пам'ять контролерів енергонезалежна.

– Годинники реального часу мають додаткове джерело резервного живлення (на випадок відключення акумулятора).

– Контролери «Castle» мають великий обсяг автономної пам'яті.

– Всі контролери приймають рішення автономно, не очікуючи реакції сервера або якогось іншого компонента системи.

Переваги при автоматизації турнікетів:

– До системи підключаються будь-які типи турнікетів.

– Система «Castle» фіксує всі проходи, навіть чинені при механічно разблокованому турнікеті.

– Турнікет автоматично розблокується при пожежі (якщо до контролерів «Castle» підключена ОПС або механічна кнопка аварійного розблокування).

– Час очікування проходу, що налаштовується.

Переваги при автоматизації дверей:

– У системі «Castle» існує режим фіксації проходів при відкритих дверях, що корисні для обліку робочого часу при використанні точок доступу, обладнаних дверима.

– Двері при пожежі відмикається автоматично, якщо до контролерів «Castle» підключена лінія пожежної сигналізації, або вручну при натисканні кнопки аварійного розблокування.

– Час очікування проходу, що налаштовується.

– При втриманні дверей у відкритому стані довше певного часу, система сигналізує про цей факт індикацією зчитувачів.

Переваги при автоматизації воріт:

– Існує можливість підключення одного зчитувача замість двох. Напрямок проїзду при цьому визначається по датчиках присутності автомобілів.

– Можливе підключення як до стороннього контролера воріт, так і прямо до моторів приводів (у такому випадку сторонній контролер не потрібний).

– Система «Castle» забезпечує безпека автомобілів, що перебувають у зоні проїзду (при підключенні до неї датчиків присутності автомобілів).

Топологія мережі СКУД:

– Система «Castle» використовує звичайну локальну комп'ютерну мережу або стандартний RS485 інтерфейс для зв'язку з контролерами.

– Довжина одного шлейфа RS485 може становити до 1200 метрів. Можливо використання повторювачів RS485 (включаючи сторонні розробки). Кожний повторювач збільшує дальність лінії зв'язку на 1200 метрів.

– До одного шлейфа RS485 підключається до 255 контролерів.

– До одного сервера системи «Castle» можна підключити до 16 шлейфів RS485.

– Існує можливість побудови системи з декількома серверами.

– Для підключення шлейфів RS485 до серверів системи використовується перетворювач RS485 в USB.

Гнучкість часових зон і графіків, облік робочого часу:

– У системі «Castle» часові зони є динамічними й можуть мати періодичність повторення від 1 до 31 днів.

– Можливо задати будь-яку кількість свят і інших виключень із правил. Всі виключення із правил по допусках і робочих графіках застосовуються до довільної групи об'єктів доступу.

– У межах кожного дня часової зони може бути задана будь-яка кількість інтервалів доступу на вхід і на вихід.

– Контролери «Castle» можуть працювати в спеціальному режимі, коли вони використовуються тільки для обліку робочого часу без управління якимось пристроєм, що перепиняє.

Спеціальні режими роботи:

– Можливий режим підвищеного контролю, коли для дозволу доступу система вимагає додаткового набору особистого цифрового коду.

– Підтримується підключення збірників пропусків відвідувачів.

– Підтримується режим фіксації проходів у невизначений час за рішенням охоронця.

Відновлення програмного забезпечення:

– Виробник публікує безкоштовні відновлення як ПЗ, так і мікропрограм контролерів.

– Мікропрограми обновляються без відключення контролерів від виконавчих механізмів з будь-якого клієнтського місця системи «Castle». Час простою точки доступу при цьому обчислюється максимум декількома секундами.

Можливості інтеграції:

– Із системою пожежної сигналізації для розблокування точок проходу у випадку виникнення пожежі.

– Із системою відеоспостереження «Інтелект» для здійснення відеозапису фактів проходів і проїздів автомобілів.

– З додатками компаній Microsoft для завантаження списків персоналу, передачі табелів обліку робочого часу в кадрові й бухгалтерські програми, перегляду створюваних звітів у додатках MS Office.

– З іншим устаткуванням і програмним забезпеченням для будь-яких потреб підприємства через убудований відкритий інтерфейс.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість

дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив’язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп’ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

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 СКУД підприємства з використанням proximity-карт.

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

Історично склалося так, що інтенсивність прогресу в секторі контролю доступу істотно уступала суміжним областям – зокрема, відеоспостереженню. Однак останнім часом на хвилі інтеграції й з падінням цін на бездротове встаткування й біометрію в СКУД стало спостерігатися досить помітне «ворушіння».

Одним з найбільш помітних зрушень у контролі доступу за останні пару років став перехід від інтеграції до уніфікації – зокрема, до уніфікованих платформ для побудови систем. При цьому платформи є загальними для СКУД і відеоспостереження, що дозволяє істотно знизити вартість володіння й розширити функціональність систем за рахунок впровадження більше складної аналітики, більше гнучких можливостей для формування звітів і зрощування рішень із системами управління робочими процесами організацій-користувачів.

Ще одним важливим явищем, що змінило конфігурацію ринку СКУД, став ріст популярності замків з бездротовим управлінням, що дозволило розраховувати на значне зростання прибутку, утвореного в даному секторі. Бездротові замки до цього моменту випускалися протягом 5-7 років, однак їхня вартість була занадто високою, що й визначило межі їхнього поширення. Серед технічних переваг, що обумовили популярність замків з бездротовим управлінням – набагато більш висока функціональність, ніж у традиційних механічних замків, і при цьому – можливість повноцінного контролю й моніторингу фізичного доступу. Витрати на встаткування одних дверей механічним замком промислового класу становлять від 2500 до 6000 доларів. А оскільки бездротової замок дозволяє варіювати права фізичного доступу для різних категорій користувачів, це дає можливість знизити витрати на одні двері

до менш ніж тисячі доларів. Приклади такого роду точок доступу – двері в лікарняну аптеку або серверну.

Хмари й СКУД

Переклад СКУД, як і інших корпоративних систем, з фізичних серверів на віртуальні, називаний віртуалізацією, в останні кілька років став досить повсякденною справою. Тут багато чого залежить від так званого гіпервайзера – платформи, використовуваної для створення й підтримки роботи віртуальних серверів. Багато приватних компаній віддають «хмари» на аутсорсинг таким провайдерам, як Amazon або Google. При цьому існують побоювання, що стосуються захисту даних, а також експлуатаційної готовності віртуальних сервісів: гарантувати її трохи складніше, ніж забезпечити підтримку працездатності фізичного сервера, що перебуває на території самої організації.

Можливо, саме тому темпи росту «хмарного» сектора в СКУД поки ще не занадто високі. Куди швидше росте попит на системи корпоративного рівня, обумовлений тенденцією до скорочення кількості джерел інформації. Відповідно до цього тренда, інтеграція стає обов'язковою вимогою. Це виливається, зокрема, у використання однієї й тієї ж бази даних для зберігання інформації із систем відеоспостереження й контролю доступу.

Крім урядових і великих корпоративних структур, замовниками інтегрованих систем всі частіше стають підприємства середнього бізнесу. По суті, будь-яка організація, що має власну територіально розподілену мережу, здатна виявитися у виграші від впровадження мережного рішення контролю доступу, оскільки рішення зможе повною мірою використовувати розподілену архітектуру мережі.

Цілий ряд питань, що визначають майбутнє хмарних СКУД, поки ще залишається без відповіді. Чи порахують влади доцільним використання комерційних сервісів? Чи з'являться загальноприйняті стандарти сервісу? Чи не позначиться на доходах сектора падіння цін на СКУД?

Експерти дослідницької компанії IHS уважають, що в довгостроковій перспективі сектор хмарного контролю доступу (ACaaS) неминуче виявиться у виграші. Фактори, що сприяють успіху, – віртуалізація сервісів і масовий перехід систем на хмарне управління. Цей глобальний процес стосується не тільки систем безпеки, мова йде про послуги взагалі. У першій половині 2014 року був відзначений істотний ріст продажів хмарних сервісів, націлених на масового клієнта – у середньому по світовому ринку ACaaS він склав 12,7%. Коммодизація апаратних засобів приводить до того, що фокус уваги галузі зміщається саме на сервісну сторону, а також на показники окупності вкладень. У результаті постачальники послуг усе більшою мірою звертають увагу на інформаційні технології, інтеграцію систем і можливості вилученого доступу до них з метою моніторингу й управління. Це дозволяє надіятись на те, що зусилля, спрямовані на зміцнення сектору ACaaS, уже у швидкий час принесуть плоди.

Незважаючи на те, що галузь контролю доступу традиційно пручається впровадженню нових технологій, перебороти інерційність мислення кінцевиків – ще тільки півділа. Більшості інсталяторів і інтеграторів, чий бізнес зложився до появи «хмар», має бути розстатися з думкою про переваги продажу «заліза» і коробкових рішень. Майбутнє – за продажами послуг, опцій і концепцій. Крім цього, у щоденній практиці значно зростає роль фахівців з IT; це дозволяє оперативно вирішувати питання резервування даних, забезпечення сертифікації, захисту від хакерів і вірусів і т.п. Незважаючи на те, що процес переходу може виявитися хворобливим, рух у цю сторону вже, по суті, почалося: мережі готелів установлюють електронні ключі на Bluetooth-Зв'язки, бездротові замки з'являються в штаб-квартирах банків, а в житла громадян неухильно проникають мобільні рішення контролю доступу.

На всі ключові питання із приводу майбутнього хмарних СКУД є одна універсальна відповідь: так. Контроль доступу неминуче «полетить у хмари», однак, можливо, це відбудеться не так незабаром, як сподівалися деякі галузеві гравці, що поквапилися зробити ставку на ACaaS. На сьогоднішній день

переважна більшість провайдерів хмарних СКУД не мають на цьому вузької спеціалізації. У результаті хмарні системи контролю доступу займають досить скромне місце в загальному спектрі ділової активності моніторингових компаній і інтеграторів. Одна із причин цьому – те, що на ранніх стадіях розвитку сервіси ACaaS приносять досить скромні прибутки, оскільки справжня вигода цих систем починається після висновку певної маси користувальницьких контрактів. Тому багато компаній змушені пригальмовувати розвиток даного напрямку, навіть маючи його позначеним у клієнтській пропозиції.

Важливо відзначити, що, як правило, сьогоднішні провайдери послуг хмарного управління СКУД використовують модель, строго говорячи, що не є по-справжньому «хмарної». Справа в тому, що для реалізації сервісу вони використовують лише обмежену кількість серверів, і на даний момент цього досить. Ситуація швидше за все зміниться, як тільки з'явиться масовий попит на хмарні системи контролю доступу.

Експерти IHS висловлюють надію на те, що інтерес до ACaaS збережеться й у державних організацій, однак характер прийняття рішень у цих структурах робить процеси впровадження відносно повільними й дорогими. У США вже запущена ініціатива milCloud, а також ряд інших; у Євросоюзі утворене Європейське хмарне партнерство (European Cloud Partnership), також називане Trusted Cloud, з яким у певній мірі буде зв'язане впровадження хмарних СКУД у практику бюджетних організацій.

У цілому, сектор ACaaS проявляє всі ознаки здорового розвитку, і заставою його успішного росту є значно більша гнучкість пропонованих клієнтам рішень, ніж у традиційних «прив'язані до місця» СКУД. Приміром, кінцевикам може бути надана можливість самостійно конфігурувати свої системи, а догляд за інфраструктурною її складовою можуть здійснювати сторонні виконавці. Як варіант, за додаткову плату провайдер може взяти на себе й здійснення функцій моніторингу, печатки звітів, виготовлення пропусків, надання прав доступу й т.п. Існують і схеми, що дозволяють провайдерам брати на себе певний обсяг

виконання поточних завдань, і при цьому повністю або частково «розчинити» витрати клієнтів на придбання встаткування в щомісячних платежах – це дозволяє знизити ціну «вхідного квитка» для клієнтів.

Є й фактори, що стримують ріст хмарних СКУД. Насамперед це триваюче зниження середньостатистичних питомих витрат на одну точку доступу, а також міркування схоронності персональних даних, необхідність виставляння унікальних рахунків кожному із клієнтів, принципова схильність «хмар» хакерським атакам, а також ряд інших проблем, істотна частина яких – питання, характерні для галузі безпеки в цілому.

Розвиток біометрії

Очікування галузі від ставшої в останні роки цілком доступної за ціною біометрії залишаються досить високими. Урядові й інші особливо важливі об'єкти сьогодні обладнаються рішеннями біометричного контролю доступу чи ледве не за замовчуванням. При цьому за державний рахунок були проведені порівняльні дослідження технологій персональної ідентифікації по біометричних параметрах, у ході яких було встановлено, зокрема, що сканування відбитків пальців є найшвидшим способом перевірки дійсності користувачів, а сканування райдужної оболонки око досить роздратовуючи діє на людей, що запитують фізичний доступ – і це приводить до ще більших затримок при зчитуванні.

З'являються й виробники, що забезпечують у всіх своїх продуктах зчитування біометрії. Для таких вертикальних ринків, як авіаперевезення, це є ключовою перевагою: багато аеропортів впровадили в практику програми добровільної реєстрації пасажирів, що часто користуються послугами повітряного транспорту. Це дозволяє істотно скоротити час, затрачуваний на перед польотний огляд, однак вимагає зчитування біометричних характеристик пасажирів. На урядових об'єктах і критично важливих інфраструктурах існують вимоги до багатофакторної ідентифікації, відповідності яким компанія, що спеціалізується на традиційних СКУД, зможе досягти лише при наявності партнера-постачальника біометрії.

Найбільш перспективними різновидами біометрії сьогодні вважаються технології безконтактного зчитування, однак на даний момент показники їхньої надійності залишають бажати кращого. Ближче всього до масового впровадження підійшли розпізнавання по рисунку вен на долоні або пальці, а також розпізнавання осіб. Можливо, першими корпоративними користувачами такого роду рішень стануть компанії-технологічні лідери начебто Facebook або Google.

Кроки стандартизації

Форум ONVIF, натхненний успішним впровадженням специфікацій стандарту відкритих мережних систем в області відеоспостереження, в останні роки обертає все більшу увагу на контроль доступу. Недавно це вилилося в опублікування специфікацій профілю Із цього стандарту, що мають безпосереднє відношення до СКУД. Аналогічні зусилля по досягненню сумісності вживають і інші галузеві альянси – зокрема, з'явився відкритий протокол OSDP, розробляє норми по СКУД і PSIA.

Роль стандартизації з погляду замовників систем – забезпечити безпечне вкладення засобів. Виграють від неї й системні інтегратори, і проектувальники рішень. Однак головне сьогодні полягає в тому, що до стандартизації систем і їхніх компонентів стали позитивно ставитися виробники.

Роль IT-служби кінцевика

Цікаво, що в прийнятті рішень про придбання систем контролю доступу ключова роль стала належати службам інформаційних технологій, а не службам безпеки компаній-користувачів. Зрештою, оскільки все більша кількість нових систем використовують стандартні промислові сервери, кому, як не «айтишнікам», має бути їх обслуговувати? До того ж IT-фахівців цілком логічно залучити до консультацій із приводу можливого придбання мережної системи контролю доступу – тільки вони можуть адекватно оцінити, приміром, необхідні для роботи СКУД резерви пропускної здатності мережі.

В умовах щільного контролю з боку IT-фахівців представникам системного інтегратора залишається лише встановлювати й налаштовувати

кінцеві пристрої – зчитувачі й дверні контролери. Але навіть при цьому IP-адреси прийдеться одержувати у айтишників.

Двофакторна авторизація в СКУД

Для мінімізації ризику в системах контролю доступу сьогодні повсюдно рекомендується застосовувати двофакторну ідентифікацію – комбінацію двох і більше методів авторизації користувача («щось вам відоме», «щось у вас наявне» і «щось властиве лише вам»).

З одним з найпростіших прикладів двофакторної авторизації ви зіштовхуєтеся, знімаючи готівку в банкомату: для цього потрібна пластикова карта («щось у вас наявне») і пін-код («щось вам відоме»). В установах охорони здоров'я факторами ідентифікації найчастіше є проксіміті-карта, використовувана медпрацівникам для входу в будинок лікарні, і персональний код або пароль. Тобто, доктор прикладає до зчитувача карту, а потім уводить на клавіатурі набір цифр.

Це просто й відносно швидко. Однак немає чи тут підступу – чи не довелося пожертвувати безпекою заради цієї простоти?

Проксіміті-карта під пильним поглядом

На жаль, використання карт доступу спільно з уведенням паролів не є настільки захищеним методом контролю доступу, яким він здається на перший погляд. Звичайно, така комбінація є більше безпечною, ніж просте уведення пароля, однак про серйозний ступінь надійності тут говорити вже не доводиться. Проксіміті-карти використовуються в системах контролю фізичного доступу на об'єкти протягом більш ніж трьох десятиліть, а в цей час із їхньою допомогою здійснюється також управління доступом користувачів у комп'ютерні мережі. Однак чи гарні проксіміті карти для контролю доступу на об'єкти охорони?

У кожну проксіміті-карту «зашитий» фіксований ідентифікаційний номер, називаний серійним номером карти (card serial number, CSN) – а передача його на зчитувач здійснюється у відкритому, незашифрованому виді. Серійний номер карти прив’язується до особистості користувача. Іншими словами, фіксований

ідентифікатор CSN відіграє роль ім'я користувача, і його використання в парі з паролем або PIN-кодом дозволяє провести в рамках СКУД операцію, аналогічну входу в комп'ютерну мережу.

Нові карти радіочастотної ідентифікації, крім зберігання серійного номера, дозволяють записувати й зберігати на карті й інші дані, шифрувати інформацію при записі й передачі, а також дозволяти безпечний обмін даними зі зчитувачами. Однак все це використовується лише для контролю фізичного доступу й не застосовується при забезпеченні доступу в комп'ютерні мережі. Міри підвищеної захищеності повинні впроваджуватися за узгодженням з виробником карт, до того ж їхнє застосування знижує швидкість обміну інформацією при ідентифікації користувача; не все просто тут є і із сумісністю карт із різним устаткуванням аналогічного типу. Тому в більшості програмних рішень персональної ідентифікації використовується тільки лише фіксований код, не прив'язаний до конкретного технологічного виконання карт.

Виходить, що у всіх радіочастотних карт є якісь «загальний знаменник» – серійний номер карти, робота з яким здійснюється швидко й без зайвих турбот із сумісністю. Однак при відсутності шифрування номер цей досить просто перехопити, а саму карту клонувати. Таким чином, карткова система є всього лише варіацією комбінації логін/пароль.

У більшості рішень із використанням проксіміті-карт процедура уведення пароля фактично опущена. Але навіть там, де вона передбачена, виробники дозволяють кінцевику використовувати «спрощену» процедуру входу (grace period), коли уведення пароля потрібно не при кожній спробі авторизації. Тобто, на початку робочого дня потрібно пред'явити карту й увести PIN-код, а в наступні 4-8 годин пред'явник карти може проходити через точку доступу безперешкодно. Зрозуміло, що відбудеться, якщо в цей інтервал часу карта буде загублена або украдена: неї зможе використовувати хто завгодно, навіть не знаючи пароля.

На жаль, у звичайному житті питання безпеки прийняті відсувати на другий план. Проксіміті-карти споконвічно не призначалися для захисту комп'ютерних мереж, додатків і важливих даних, однак є безліч організацій, які «для зручності» користуються цією технологією для захисту критично важливих активів.

Є чи в цього методу альтернатива? Вона повинна бути не менш зручною, ніж використання карти в сполученні з паролем, і при цьому система повинна «дізнаватися» користувача, що запитує дозвіл на доступ. Те, ніж співробітник організації в принципі здатний поділитися з ким-небудь ще, не дає нам достовірне подання про те, хто він, властиво, такий. Те, що може бути з легкістю скопійоване – наприклад, серійний номер карти – також не дозволяє гарантувати ідентичність користувача. Виходить, що єдиний вихід у таких випадках – покластися на біометрію.

3.2 Розробка структурної схеми

У даному проекті використовується proximity карта Em-Marine ISO 125КГц – безконтактна RFID карта. Електронні пропуски-ідентифікатори – карта тонка Em-Marine ISO 125 КГц під пряму печатку. Пластиковий пропуск використовує proximity технологію й призначений для ідентифікації персоналу в системах контролю доступу й обліку робочого часу.

Поява в СКУД технології Proximity (у дослівному перекладі "близький", "ближній") викликало природне бажання збільшити дальність зчитування коду карти. У результаті "народилися" системи Hands free ("вільні руки"). Але оскільки природу обдурити неможливо, для збільшення дальності довелося оснащувати ідентифікатори (карти) малогабаритною літієвою батарейкою. Зате тепер для живлення мікросхеми карти не була потрібна велика потужність випромінювання зчитувача, і дальність подібних систем перевищила один метр. Історично лідером

тут стала англійська компанія Cotag. Її активні карти мають строк життя не менш п'яти років.

Технологія активних ідентифікаторів одержала подальший розвиток у системах упізнавання автомобілів. Такий ідентифікатор (уже не у вигляді карти, а виді невеликого блоку, що кріпиться до кузова автомобіля й одержує живлення від його бортової мережі) працював, як правило, зі зчитувачем, антена якого являла собою дротову петлю, що закривається в дорожнє полотно.

Контроль доступу до власності охоплює регулювання та керування входом і виходом до власності. Це гарантує, що уповноважені люди можуть швидко отримати доступ до власності та її охоронюваних зон, не допускаючи сторонніх осіб.

У цю цифрову епоху менеджери нерухомості навряд чи можуть покладатися на традиційні методи контролю доступу, такі як замки та ключі. Досконаліші технології, такі як мобільний доступ і біометричні сканери (наприклад, відбитки пальців або розпізнавання обличчя), навіть замінюють звичайні системи доступу, такі як коди введення з клавіатури, ключ-картки та брелоки.

Більшість сучасних систем доступу до власності автентифікують і записують особистість людей, які входять до власності, і застосовують обмеження доступу на основі попередньо визначених правил і дозволів, встановлених адміністратором.

Як керувати доступом до власності?

Сучасні системи доступу до власності пропонують гнучкість і налаштування, і ви повинні повністю скористатися цими перевагами, щоб запобігти порушенням безпеки. Ви можете почати з визначення груп користувачів, рівнів доступу та дозволів на основі ролей або обов’язків. Використовуйте « Принцип найменших привілеїв », щоб авторизовані люди мали доступ лише до тих просторів, які їм потрібні.

Використовуйте глобальне централізоване керування доступом, щоб оптимізувати операції під час керування кількома властивостями. Використовуйте інформаційну панель вашої системи , щоб керувати всіма властивостями, відмикати двері та розглядати підозрілі події доступу, навіть на льоту.

Щоб підтримувати відповідність, підзвітність і пом’якшення, відстежуйте події доступу та встановлюйте сповіщення про критичні.

Інтегруйте контроль доступу з іншими системами безпеки та управління будівлею для комплексної, покращеної безпеки, покращеного керування відвідувачами та спрощених операцій.

Найпоширеніші типи систем контролю доступу до власності

Найпоширенішими типами систем контролю доступу є:

– Фізичні замки та системи доступу з ключем: рідко використовуються для контролю доступу до власності через обмежену функціональність, недостатню гнучкість і відсутність можливостей журналу аудиту.

– Системи доступу з клавіатури : прості у використанні та зазвичай економічно ефективні, вони вимагають від користувачів введення правильного PIN-коду на клавіатурі для входу. Можливості спільного доступу до PIN-коду або легкість для правопорушників просто спостерігати за PIN-кодом роблять клавіатурні системи варіантом із низьким рівнем безпеки.

– Системи доступу до карток і брелоків : користувачі пред’являють свою проксіміті-картку, смарт-картки або брелок, що містить електронні облікові дані до картки, зчитувальному пристрою, який автентифікує облікові дані та надає доступ, якщо він авторизований. Вони пропонують можливість відстежувати дії користувачів за допомогою журналів аудиту, але не зручні та легкі в управлінні, оскільки деякі користувачі схильні забувати, губити або втрачати свої облікові дані.

– Біометричні системи доступу: використовуючи унікальні біометричні ознаки, як-от відбитки пальців, долоні, обличчя та райдужну оболонку ока, для

ідентифікації осіб, біометричні системи забезпечують високий рівень безпеки, але часто є дорожчими для впровадження.

– Системи доступу на основі мобільних пристроїв : використовуючи смартфони або переносні пристрої як цифрові облікові дані, системи мобільного доступу пропонують користувачам зручність, водночас дозволяючи адміністраторам віддалено керувати доступом і гнучко налаштовувати систему відповідно до своїх потреб.

– Інтегровані системи доступу: поєднання багатьох методів доступу, таких як мобільні пристрої, пристрої для зчитування карток і введення з клавіатури, в уніфіковане рішення, ці системи забезпечують підвищену безпеку, гнучкість і масштабованість.

Власності зі спільним входом

Більшість комерційних об’єктів, як-от офісні будівлі, флекс-простори та коворкінги, торгові центри та бізнес-парки, мають спільні входи. Це загальні точки входу, які надають доступ до кількох підприємств або підрозділів у власності.

Виконуючи функції централізованих вузлів доступу, якими часто керує власник нерухомості або компанія з управління нерухомістю, спільні входи дозволяють орендарям, відвідувачам і клієнтам діставатися до різних офісів, торгових приміщень або об’єктів у межах власності. Зазвичай, служачи початковою точкою безпеки, ці входи використовують системи контролю доступу для забезпечення авторизованого входу та підтримки безпечного та контрольованого середовища.

Доступ орендодавця до власності

Власники майна та орендодавці мають права та привілеї доступу до орендованих приміщень. Угоди про комерційну оренду надають більшу гнучкість для узгодження умов доступу, ніж житлова нерухомість.

Положення про доступ орендодавця, які часто визначаються в договорі оренди, охоплюють конкретні операційні потреби орендарів і питання

конфіденційності. Оскільки орендодавці відповідають за утримання місць загального користування та проведення необхідного ремонту, вони обирають і керують системою спільного контролю доступу.

З іншого боку, орендарі мають право на безперебійне користування орендованими ними приміщеннями. Зважаючи на те, що вони часто встановлюють власну систему контролю доступу, щоб захистити свій простір, зручний потік доступу буває рідко, враховуючи, що їм потрібні два різні набори облікових даних для входу на робоче місце.

Приватна власність з відкритим доступом

Незважаючи на те, що вони належать і контролюються приватними особами, деякі об’єкти допускають публічний доступ для різних видів діяльності. Наприклад, деякі офісні будівлі мають зони загального доступу, як-от роздрібні магазини, ресторани чи інші підприємства, розташовані всередині будівлі, зазвичай розташовані на першому поверсі.

У цьому випадку власники або менеджери нерухомості відповідають за обслуговування та безпеку відвідувачів. Встановлення балансу між публічним доступом та інтересами та безпекою власника та орендаря є вирішальним для успіху цих просторів.

Автомобільний доступ до об'єкта

Ефективний доступ транспортного засобу є важливим для підвищення безпеки, безперебійної роботи, позитивного досвіду користувачів і відповідності комерційним об’єктам. Щоб забезпечити безперебійний в’їзд і виїзд авторизованих транспортних засобів на територію, менеджери нерухомості використовують пристрої контролю доступу транспортних засобів, щоб полегшити швидкий доступ і часто перевіряти особи людей.

Ворота та фізичні бар’єри, зазвичай обладнані електронними системами контролю доступу, є найпоширенішим вибором. Інші заходи включають виділену парковку для відвідувачів, системи розпізнавання номерних знаків, персонал служби безпеки та управління потоком руху.

Належний контроль доступу транспортних засобів сприяє безпеці, ефективності та безперебійному досвіду для власників нерухомості, орендарів і відвідувачів.

Як контролювати доступ до власності зі спільним входом

Незважаючи на те, що системи контролю доступу є важливими, вони часто викликають запитання щодо безпеки та зручності від орендарів у помешканнях зі спільними входами. Більшість компаній хочуть підвищити безпеку та забезпечити своїм співробітникам безперешкодне та приємне перебування в офісі. Носити два набори облікових даних лише для того, щоб увійти в офіс, незручно та може призвести до інцидентів.

Якщо орендодавці та менеджери нерухомості прагнуть прийняти орендарів і запровадити систему контролю доступу за їхнім вибором на спільному вході, вони можуть мати десятки різних зчитувачів доступу біля вхідних дверей. Що ще важливіше, вони не матимуть повного контролю над людьми, які входять до їхньої власності, за безпеку якої вони відповідають.

Як вибрати правильне рішення щодо власності доступу?

Вибір правильного рішення для контролю доступу до власності вимагає ретельного розгляду ваших налаштувань і потреб.

Інтуїтивно зрозуміла інформаційна панель із централізованим керуванням віддаленим доступом, мобільний додаток із найвищим рейтингом, налаштовуємі групи користувачів і розклади доступу, розширена документація та підтримка продукту, а також різноманітні методи доступу та інтеграції.

Різні варіанти розгортання забезпечують підвищену безпеку та перспективність вашої системи, одночасно забезпечуючи рентабельну міграцію. Вони також легко вирішують завдання спільного входу.

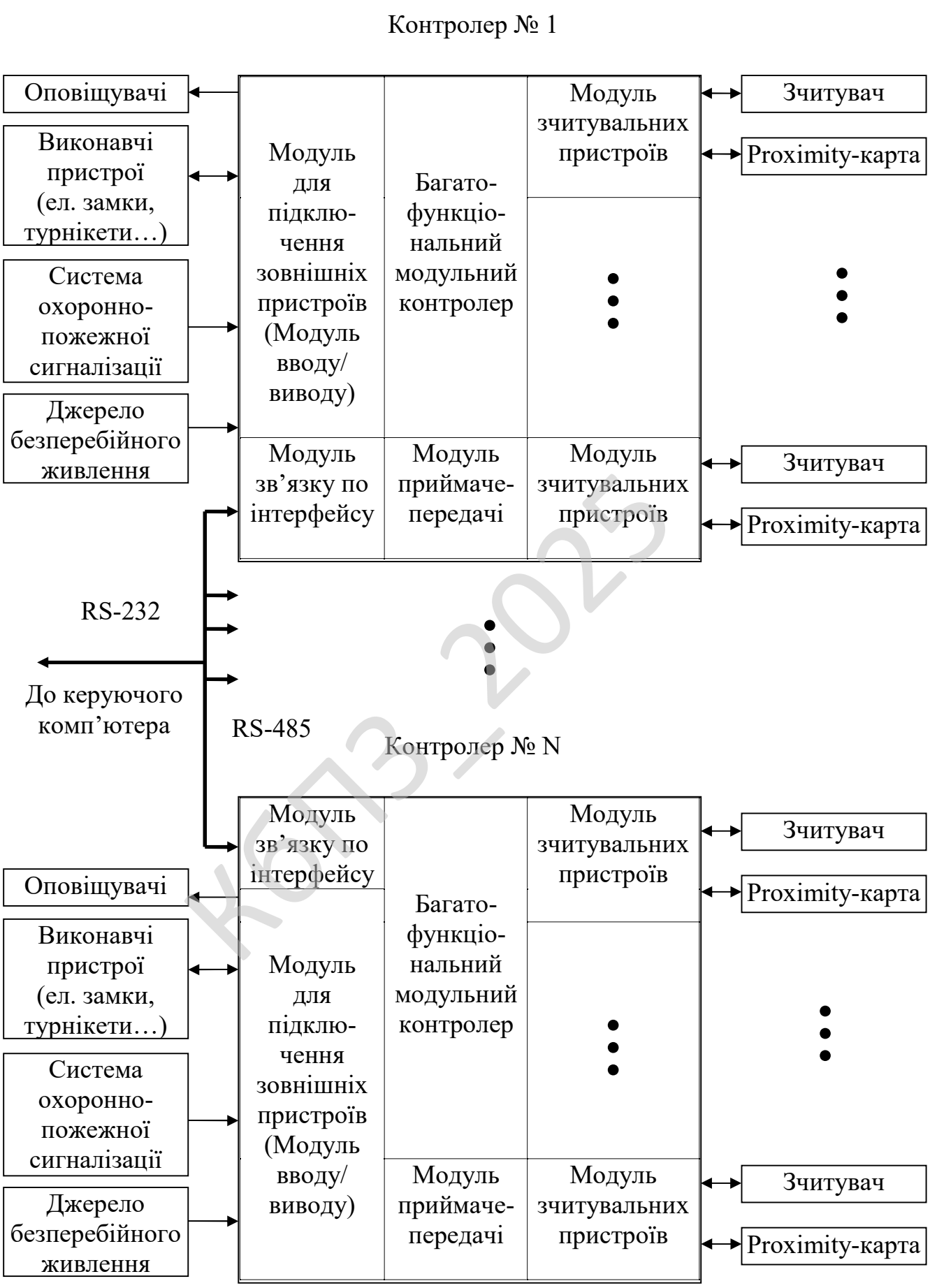

Рисунок 3.1 – Структурна схема системи

Класи СКУД

До СКУД 1-го класу відносяться малофункціональні системи малої ємності, що працюють в автономному режимі. Такі системи застосовуються у випадку, якщо замовникові необхідно забезпечити контрольований доступ співробітників і відвідувачів, що мають відповідний ідентифікатор.

Автономна система складається з контролера звичайно об'єднаного зі зчитувачем і виконавчого елемента. Як правило, використовуються магнітні (рідше безконтактні) картки, електронні ключі "TouchMemory". Залежно від типу контролера або замка кількість осіб у списках може досягати від 60 до 2800 чоловік. Автономні системи забезпечуються резервним живленням і мають механічний ключ для відкривання замка в аварійних ситуаціях.

СКУД 2-го класи також монофункціональні системи, але в них уже є можливість розширення й включення їх або їхніх складових частин у загальну лінію зв'язку (мережний режим). Дані системи мають ряд додаткових функцій. На об'єктах, обладнаних засобами й системами ОПС (охоронно-пожежна сигналізація), СКУД 2-го класи застосовуються як самостійні системи, і вони часто розглядаються тільки як засоби посилення режиму забезпечення безпеки об'єкта.

СКУД 3-го й 4-го класів звичайно називаються мережними, тому що контролери об'єднані в локальну мережу, що працюють у реальному часі й ведучі безперервний діалог з периферійними пристроями, із провідним контролером або з керуючим комп'ютером, розташованим у пункті охорони. Системи цих класів це великі й багаторівневі системи, розраховані на велику кількість користувачів (1500 чоловік і більше).

Подібні системи застосовуються у випадку, коли необхідно контролювати час проходу співробітників і відвідувачів на об'єкт і в приміщення. При цьому застосовуються більше складні електронні ідентифікатори (Proxmity картки, біометричний контроль або їхні сполучення). Час проходу на щодня тижня й для кожного власника електронної картки задається адміністратором системи.

Системи 3-го класи звичайно інтегруються із системами ОПС і ТСВ на релейному рівні. Релейний рівень припускає наявність додаткового модуля в контролері (або додаткових входів/виходів у контролері), до якого підключаються охоронні або пожежні оповіщувачі, і релейні виходи для управління телекамерами й іншими пристроями. Подібна інтеграція застосовується, в основному, на малих об'єктах. На таких об'єктах кількість взаємодій між системами невелико, і всі вони можуть бути враховані в процесі проектування системи безпеки. Цей рівень інтеграції є простим, універсальним і досить надійним.

Системи 4-го класи це багаторівневі системи великої ємності. Відмінні риси більших систем – наявність розвиненого програмного забезпечення, що дозволяє реалізовувати велика кількість функціональних можливостей і високий ступінь інтеграції на програмному (системному) рівні з іншими системами охорони й безпеки.

Програмний рівень припускає об'єднання різних систем на основі єдиної програмно-апаратної платформи, з єдиним комунікаційним протоколом і загальною базою даних.

СКУД для автономного режиму роботи

СКУД 1-го й 2-го класів, що працюють в автономному режимі, звичайно обладнаються: квартири, котеджі, невеликі офіси, магазини, аптеки, готелі й т.п. і мало значимі зони на важливих об'єктах. Це дозволяє раціонально зменшити число каналів, що обслуговуються дорогими СКУД 3-го й 4-го класів. Дані СКУД це невеликі й недорогі системи, що обслуговують, як правило, до 8-мі пристроїв загородження (дверей, воріт, турнікетів і т.п.). СКУД 1-го й 2-го класів можна застосовувати й на важливих об'єктах або приміщеннях, якщо необхідний рівень безпеки забезпечується системами охоронної сигналізації й відеоконтролю.

Наявність описаних модулів багатофункціонального контролера створює більші можливості по управлінню різноманітною периферією системи. Як контрольовані точки можуть виступати головки, що зчитують, PIN-Клавіатури,

замкнуті/розімкнуті контакти кнопок, реле, вихідні контакти різних об'ємних або поверхневих оповіщувачів. Як виконавчі пристрої можуть використовуватися електрозамки дверей, виконавчі пристрої шлагбаумів, турнікетів, пристрою тривожного оповіщення й висвітлення, телевізійні камери й т.д.

Логічний пристрій (процесор) контролера дозволяє робити необхідну установку параметрів доступу в кожній контрольній точці за допомогою програмного забезпечення, тобто конфігурувати систему. Системний програміст може задавати параметри (замкнуте/розімкнутий стан контактів реле або кнопок, стан і режим роботи лічильників, стан флатових регістрів, часові інтервали реєстраторів подій і т.д.) прямо із клавіатури комп'ютера. Це дає можливість реалізовувати різні варіанти організації контролю й управління доступом, гнучко міняючи їх відповідно до поточних вимог.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Вибір СКУД для встаткування об'єкта

Обстеження об'єкта

Вибір варіанта встаткування об'єкта засобами СКУД варто починати з його обстеження. При обстеженні визначаються характеристики значимості приміщень об'єкта, його будівельні й архітектурно-планувальні рішення, умови експлуатації, режими роботи, обмеження або, навпаки, розширення права доступу окремих співробітників, параметри встановлених (або передбачуваних до установки на даному об'єкті) пристроїв, що входять у СКУД. За результатами обстеження визначаються тактичні характеристики й структура СКУД, технічні характеристики її компонентів, а також складається технічне завдання на встаткування об'єкта СКУД.

Рисунок 3.2 – Функціональна схема системи

У технічному завданні вказується:

– призначення системи, технічне обґрунтування й опис системи;

– розміщення складових частин системи;

– умови експлуатації складових частин системи;

– основні технічні характеристики такі як:

– пропускна здатність в охоронювані зони особливо в годину-пік;

– максимально можливе число користувачів на один зчитувач;

– максимальне число й види карток-пропусків;

– вимоги до маскування й захисту складових частин СКУД від вандалізму;

– оповіщення про тривожні й аварійні ситуації й вживання відповідних заходів по їхньому припиненню або попередженню;

– можливість роботи й збереження даних без комп'ютера або при його відмові;

– програмне забезпечення системи;

– вимоги до безпеки;

– вимоги до електроживлення;

– обслуговування й ремонт системи;

– вимоги до можливості включення системи СКУД в інтегровану систему безпеки.

Архітектурно-планувальні й будівельні рішення

Шляхом вивчення креслень, обходу й огляду об'єкта, а також проведення необхідних вимірів визначаються:

– кількість входів/виходів і їхні геометричні розміри (площа, лінійні розміри, пропускна здатність і т.п.);

– матеріал будівельних конструкцій;

– кількість окремо вартих будинків, їхня поверховість;

– кількість відкритих площадок;

– кількість опалювальних і неопалюваних приміщень і їхнє розташування.

Умови експлуатації

Ураховувати шкідливий вплив навколишнього середовища треба лише для виконавчих пристроїв, зчитувачів і контролерів (сполучених зі зчитувачами в одному конструктивному блоці), призначених для роботи поза опалювальними закритими приміщеннями або в особливих умовах (запиленість, підвищена вологість, негативна температура й т.п.). Для надійної роботи СКУД на об'єкті необхідно враховувати вплив електромагнітних перешкод, перепади напруги живлення, далекість зчитувачів і контролерів від керуючого центра, заземлення складових частин системи й т.п.

Вимоги до електроживлення

Основне електроживлення СКУД повинне здійснюватися від мережі змінного струму частотою 50 Гц із номінальною напругою 220 В.

СКУД повинні зберігати працездатність при відхиленнях напруги мережі від мінус 15 до +10 % і частоти до ±1 Гц від номінального значення.

Електроживлення окремих СКУД допускається здійснювати від інших джерел з іншими параметрами вихідних напруг вимоги до яких установлюються в нормативних документах на конкретні типи систем.

Електропостачання технічних засобів СКУД здійснюється від вільної групи щита чергового висвітлення. При відсутності на об'єкті щита чергового висвітлення або вільної групи на ньому, замовник установлює самостійний щит електроживлення на відповідну кількість груп. Щит електроживлення, установлюваний поза охоронюваним приміщенням, повинен розміщатися в металевій шафі, що замикається, і заблокований на відкривання.

СКУД повинні мати резервне електроживлення при проваллі основного електроживлення. Номінальна напруга резервного джерела живлення повинне бути 12 або 24 В. Перехід на резервне живлення й назад повинен відбуватися автоматично без порушення встановлених режимів роботи й функціонального стану СКУД.

СКУД повинні зберігати працездатність при відхиленнях напруги резервного джерела живлення від мінус 15 до плюс 10 % від номінального значення.

Резервне джерело живлення повинен забезпечити функціонування системи при проваллі напруг у мережі на час не менш 8 ч.

При використанні як джерело резервного живлення акумулятора, повинен виконуватися автоматичну зарядку акумулятора.

Акумуляторні батареї (за винятком що не обслуговуються), як правило, розміщаються в спеціальних акумуляторних приміщеннях на стелажах або

полках шафи, відповідно до вимог ТУ 4-ДО.610.236-87 у піддонах, стійких до впливу агресивних середовищ.

Свинцеві акумулятори ємністю не більше 72 А/год і лужні акумуляторні батареї ємністю не більше 100 А/год і напругою до 60 У можуть установлюватися в загальних виробничих невибухо- і непожежнонебезпечних приміщеннях у металевих шафах з відособленої приточно-витяжною вентиляцією.

Акумуляторні установки повинні бути обладнані відповідно до вимог ППЕ "Правила пристрою електроустановок".

При використанні як джерела резервного живлення, акумулятора або сухих батарей, повинна бути передбачена індикація розряду акумулятора або батареї нижче припустимої межі. Для автономних систем індикація розряду повинна бути світлова або звукова, для мережних систем сигнал розряду акумулятора повинен передаватися на центральний пульт.

Хімічні джерела струму (батарейки), убудовані в активні ідентифікатори або забезпечуючи схоронність даних повинні забезпечувати працездатність засобів контролю й управління доступом протягом часу, не менш 5 років.

Програма надає більші сервісні можливості операторові, виводячи різноманітну інформацію на екран. Наприклад, на дисплеї комп'ютера можна мати план одного або декількох приміщень із позначеними на ньому контрольованими точками, індикацію несанкціонованих проникненні (якщо потрібно – зі звуковим супроводом). На екран можуть виводитися численні повідомлення, наприклад, повні або короткі звіти про зареєстровані події з можливістю їхньої роздруківки на принтері.

Програмне забезпечення повинне забезпечує:

– ініціалізацію ідентифікаторів (занесення кодів ідентифікаторів до пам’яті системи);

– завдання характеристик контрольованих точок;

– установку часових інтервалів доступу (вікон часу);

– установку рівнів доступу для користувачів;

– протоколювання поточних подій;

– ведення баз даних;

– збереження даних і установок при аваріях і збоях у системі.

Загальний принцип роботи будь-який RFID системи досить простий. У системі завжди є два основних компоненти: це зчитувач і ідентифікатор (карта, мітка, брелок). Зчитувач випромінює в навколишній простір електромагнітну енергію. Ідентифікатор приймає сигнал від зчитувача й формує відповідний сигнал, що приймається антеною зчитувача й обробляється його електронним блоком.

Рівень доступу – сукупність часових інтервалів доступу (вікон часу) і місць проходу (маршрутів переміщення), які призначаються певній особі або групі осіб, яким дозволений доступ у задані охоронювані зони в задані часові інтервали).

Програмне забезпечення повинне бути стійко до випадкових і навмисних впливів наступного виду:

– відключення керуючого комп'ютера;

– програмне скидання керуючого комп'ютера;

– апаратне скидання керуючого комп'ютера;

– натискання на клавіатурі випадковим образом клавіш;

– випадковий перебір пунктів меню програми.

Після зазначених впливів і після перезапуску програми повинна зберігатися працездатність системи й схоронність установлених даних. Зазначені впливи не повинні приводити до відкривання пристроїв загородження й зміні діючих кодів доступу.

Програмне забезпечення повинне бути захищене від навмисних впливів з метою зміни установок у системі.

Вид і ступінь захисту повинні бути встановлені в паспортах на конкретні види засобів або систем. Відомості наведені в технічній документації не повинні розкривати таємність захисту.

Програмне забезпечення при необхідності повинне бути захищене від несанкціонованого копіювання.

Програмне забезпечення повинне бути захищене від несанкціонованого доступу за допомогою паролів. Кількість рівнів доступу по паролях повинне бути не менш 3.

Рівні доступу, що рекомендуються, по типу користувачів:

– перший ("адміністрація") – доступ до всіх функцій контролю й доступу;

– другий ("оператор") – доступ тільки до функцій поточного контролю;

– третій ("системний програміст") – доступ до функцій конфігурації програмного забезпечення, без доступу до функцій, що забезпечують управління виконавчих пристроїв.

При уведенні пароля на екрані дисплея не повинні відображатися вводяться знаки, що.

Число символів пароля повинне бути не менш 5.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування ПЗ.

– Обробник помилок.

– Моніторинг системи зчитування смарт-карт.

– Отримання даних смарт-карти за технологією RFID.

– БД користувачів.

– Перевірка ступені захищеності.

– Сканування ресурсів системи.

– Модуль аналізу даних.

– Взаємодія зі службами.

– Система архівації даних.

Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

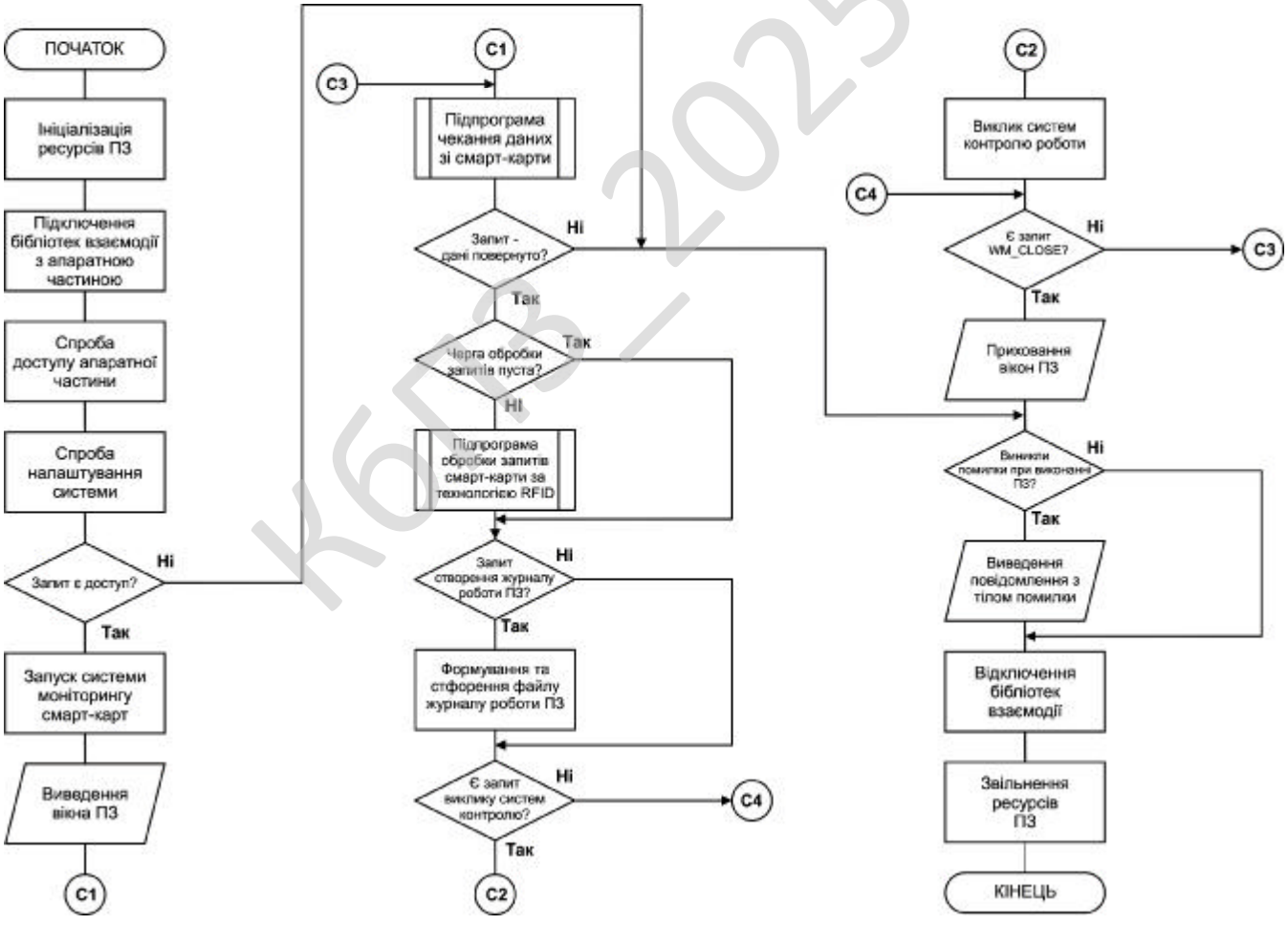

Рисунок 4.1 – Блок схема основної програми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

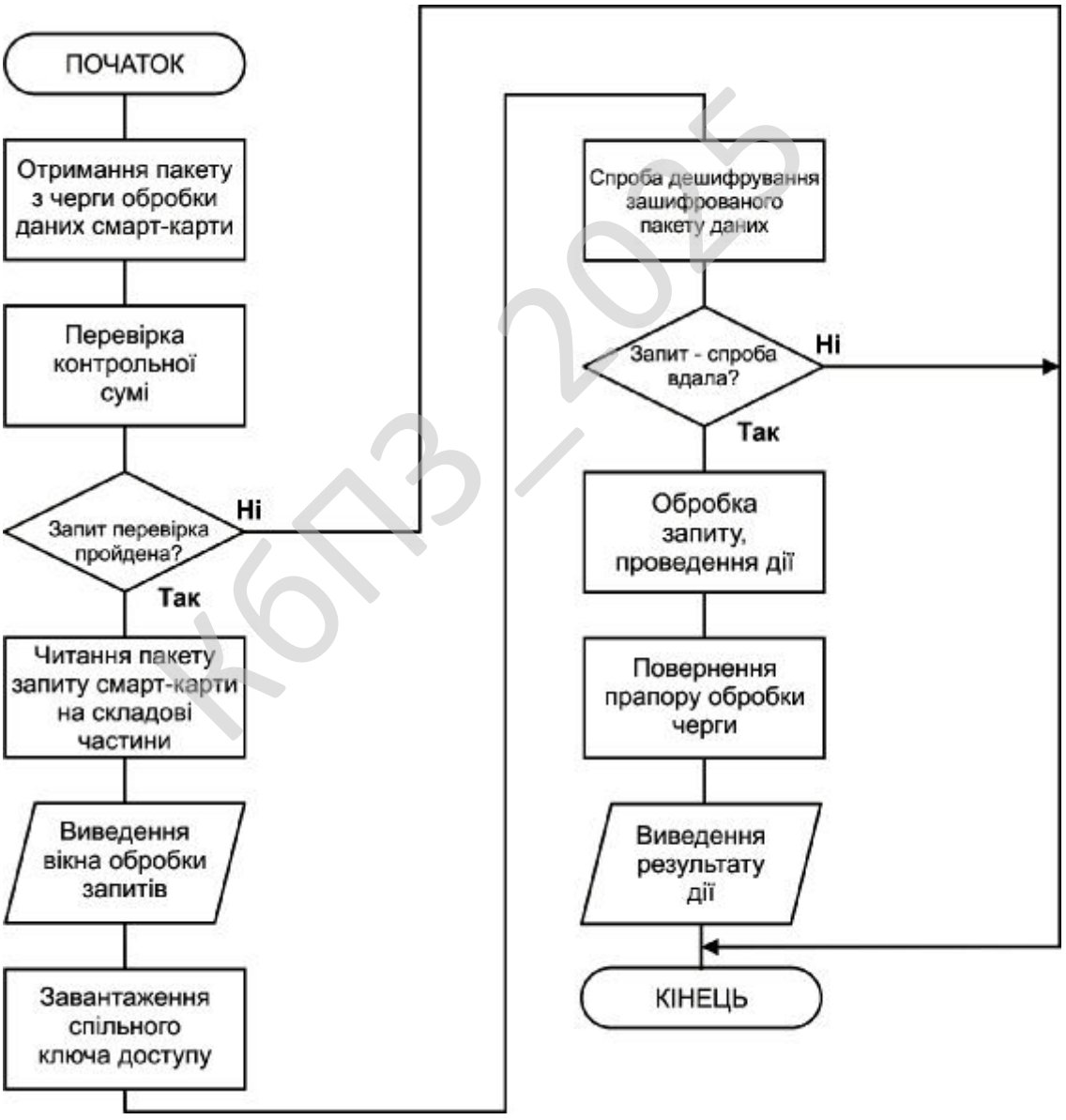

Рисунок 4.2 – Блок схема підпрограми

Опис алгоритмів функціонування системи

Перед розглядом алгоритмів функціонування системи необхідно розглянути процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом.

Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

У системі кібербезпеки СКУД підприємства використовується сучасна технологія proximity карт для контролю доступу до приміщень підприємства і забезпечення високого рівня захисту інформації та матеріальних ресурсів установи.

Система працює у режимі реального часу і використовує спеціалізовані пристрої для зчитування даних з карток що видаються кожному користувачу з метою ідентифікації.

Система сприймає дані зчитування ідентифікаторів карток і передає їх до централізованого сервера де проводиться аналіз відповідності даних з інформацією в базі користувачів.

Сервер перевіряє права доступу на основі збережених даних і виконує розрахунки для аналізу пропускної здатності системи з урахуванням середнього часу обробки запитів і кількості звернень протягом встановленого періоду часу.

Результати обчислень підтверджують правильність обраних проектних рішень що забезпечують оперативне реагування на події та високий рівень безпеки підприємства в умовах підвищеної навантаженості.

Архітектура системи складається з модуля зчитування карток який забезпечує первинну обробку даних, модуля перевірки доступу який порівнює отримані ідентифікатори з даними в базі, модуля логування подій що реєструє кожну спробу доступу і виконує статистичний аналіз даних а також модуля розрахунків продуктивності що оцінює час відгуку системи і пропускну здатність у режимі реального часу.

Програма написана мовою Python і виконує завдання по модульній обробці подій доступу з використанням об'єктно орієнтованого підходу для забезпечення масштабованості та розширюваності розробки.

У програмному коді присутні функції для зчитування даних з карток, перевірки прав доступу користувачів, генерації логів подій і виконання математичних розрахунків для аналізу ефективності роботи системи.

Логіка роботи системи реалізована шляхом об'єднання кількох алгоритмів що дозволяють проводити як аутентифікацію користувачів так і аналіз статистичних даних для визначення середнього часу відповіді системи і її пропускної здатності.

Обчислення проводяться з урахуванням середнього часу обробки кожного запиту і загальної кількості проведених заходів що дозволяє об'єктивно оцінити ефективність обраних проектних рішень і їх відповідність вимогам безпеки підприємства.

Програма має можливість моніторингу всієї роботи системи і генерує звіти для аналізу ефективності роботи обчислювальних алгоритмів що сприяє оперативному виявленню можливих загроз безпеці.

Нижче наведено вихідний текст програми на мові Python з поясненням функцій і елементів коду

```
#import необхідних бібліотек
import time
```

```
import random

#define клас системи контролю доступу
class AccessControlSystem:
    def __init__(self, valid_cards):
        self.valid_cards = valid_cards
        self.logs = []

    def read_card(self, card_id):
        #симулюється процес зчитування картки з невеликим затриманням
        time.sleep(0.1)
        return card_id

    def check_access(self, card_id):
        #перевіряється відповідність картки даним з бази
        if card_id in self.valid_cards:
            return True
        else:
            return False

    def log_event(self, card_id, access_granted, response_time):
        #реєструється подія доступу з параметрами картки результату перевірки і
часу відповіді
        event = {
            "card": card_id,
            "access": access_granted,
            "time": response_time
        }
        self.logs.append(event)

    def calculate_response_time(self, start_time):
        #обчислюється час відповіді від початку обробки події до завершення
алгоритму
        return time.time() - start_time

    def simulate_access(self, card_id):
        #симулюється повний процес обробки події доступу з обчисленням часу
відповіді
        start_time = time.time()
        read_id = self.read_card(card_id)
        access = self.check_access(read_id)
        response_time = self.calculate_response_time(start_time)
```

```
        self.log_event(read_id, access, response_time)
        return access, response_time

#define функцію розрахунку пропускної здатності системи на основі загальної
кількості подій і часу моделювання
def calculate_throughput(total_events, total_time):
    #обчислюється показник пропускної здатності як співвідношення кількості
оброблених подій до часу моделювання
    if total_time == 0:
        return 0
    return total_events / total_time

#створюється база даних дійсних карток для ідентифікації користувачів
valid_cards = ["A12345", "B67890", "C54321"]

#створюється екземпляр системи контролю доступу з переданою базою дійсних карток
acs = AccessControlSystem(valid_cards)

#симулюється серія подій доступу для оцінки роботи системи в умовах високої
навантаженості
access_events = []
start_simulation = time.time()
for i in range(50):
    #випадковим чином обирається ідентифікатор картки з бази або випадкова
недійсна картка
    card = random.choice(valid_cards + ["X00000"])
    result, resp_time = acs.simulate_access(card)
    access_events.append(resp_time)

total_simulation_time = time.time() - start_simulation
throughput = calculate_throughput(len(access_events), total_simulation_time)

#print результати моделювання для підтвердження правильності проектних рішень
print("Середній час відповіді системи", sum(access_events) / len(access_events))
print("Пропускна здатність системи", throughput)
```

У цьому програмному коді реалізовано модуль зчитування картки який симулює фізичну взаємодію пристрою з proximity карткою при зверненні користувача до системи доступу.

Модуль перевірки доступу проводить порівняння отриманого ідентифікатора з переліком дійсних значень що зберігаються у базі даних користувачів установи.

Модуль логування подій фіксує кожну спробу доступу з інформацією про ідентифікатор картки результат перевірки і час відповіді системи що дозволяє проводити статистичний аналіз роботи системи за визначеним періодом часу.

Модуль розрахунків продуктивності обчислює показник пропускної здатності на основі кількості оброблених подій і загального часу моделювання роботи системи що дозволяє об'єктивно оцінити ефективність алгоритмів і обраних проектних рішень.

Дані розрахунків засвідчують коректність обраної архітектури системи і забезпечують високий рівень кібербезпеки підприємства.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа і-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний.

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3].

Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Висновок

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Розділ який відображає всі наявні proximity-карти у вигляді дерево образної структури. Як приклад це можуть бути зали у магазині. Це дозволяє швидко проводити сортування міток.

– Розділ який відповідає за основний дії програми: Створити proximity-карту; Видалити proximity-карту; Змінити proximity-карту; Додати групу; Видалити групу; Змінити групу.

– Розділи меню: Файл; Система; Налаштування; Довідка.

– Розділ поточного стану.

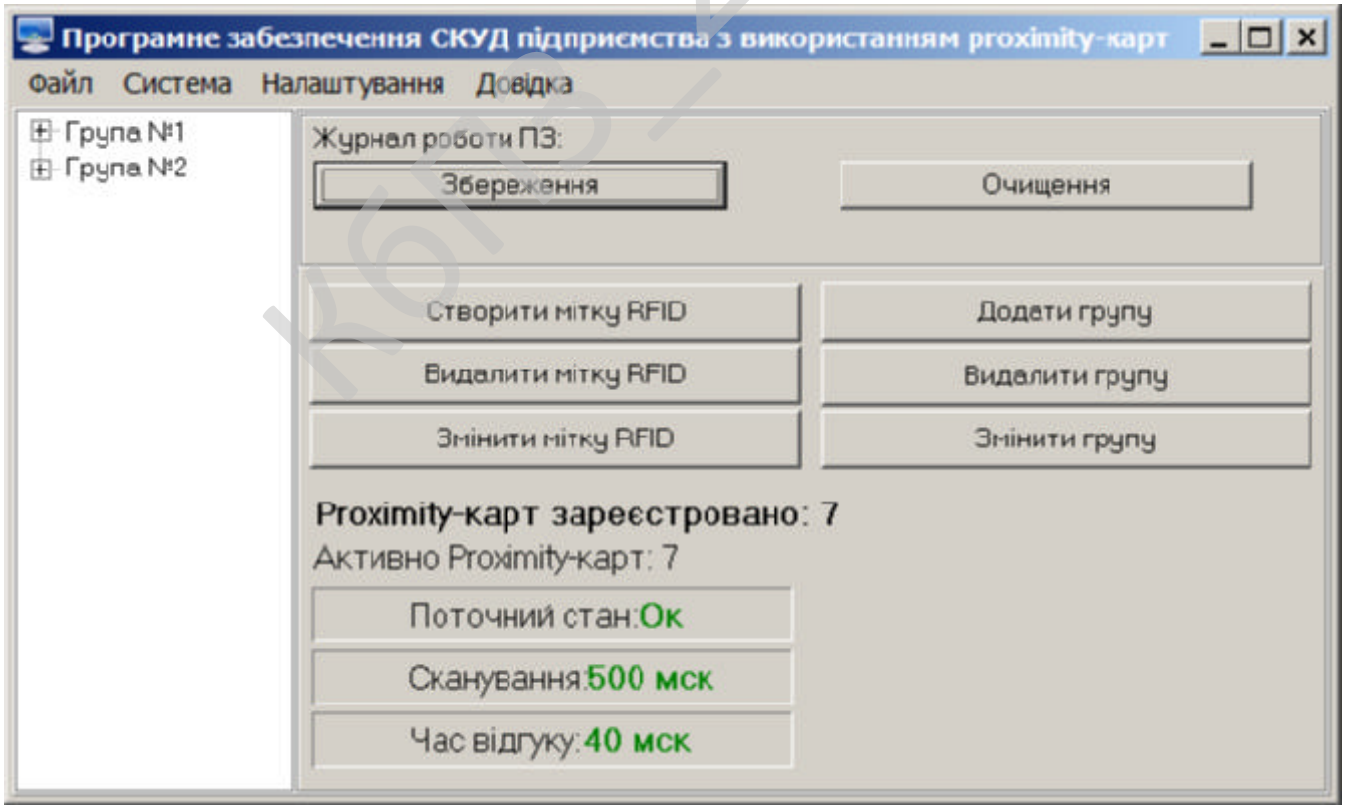

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

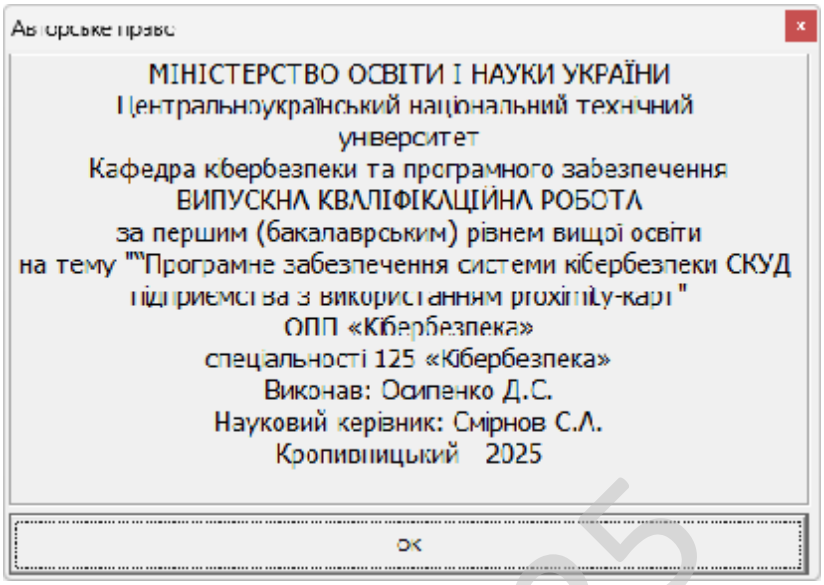

Рисунок 5.2 – Авторське право

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 (не повнофункціональна 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). 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.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки СКУД підприємства з використанням proximity-карт.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем СКУД підприємства з використанням proximity-карт.

– Досліджена система СКУД підприємства з використанням proximity-карт.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки СКУД підприємства з використанням proximity-карт.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання СКУД підприємства з використанням proximity-карт.

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 СКУД підприємства з використанням proximity-карт. 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 MISTY1.

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

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

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

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

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

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

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

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

9. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

10. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

11. 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.

12. 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.

13. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

14. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

15. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

16. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

17. 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,

18. 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.

19. 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.

20. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

21. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

22. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

23. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

24. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

25. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

26. 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

27. 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.

28. 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.

29. 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.*

30. 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.

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

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки СКУД підприємства з використанням proximity-карт.

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 СКУД підприємства з використанням proximity-карт.

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 СКУД підприємства з використанням proximity-карт;

– 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 – 66 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 5.06.2025 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 СКУД підприємства з використанням proximity-карт

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпорт необхідних модулів
import sqlite3
import datetime
import hashlib
import random
import time
import threading
import sys

#Оголошення глобальних налаштувань для системи
DATABASE_NAME = ":memory:"
SECURITY_LEVEL = "HIGH"
ACCESS_LOG_FILE = "access_log.txt"
SIMULATED_NETWORK_DELAY = 0.5
CRYPTO_SALT = "S3cUr3S@lt"

#Функція для генерації криптографічного хешу з використанням даних та ключа
def generate_hash(data, key):
#Генеруємо хеш з використанням SHA256
    combined = f"{data}{key}{CRYPTO_SALT}"
#Обчислюємо хеш значення
    return hashlib.sha256(combined.encode('utf-8')).hexdigest()

#Функція симуляції мережевого пакету з затримкою
def simulate_network_packet(data):
#Створюємо мережеву затримку для симуляції пересилки даних
    time.sleep(SIMULATED_NETWORK_DELAY)
#Повертаємо дані, ніби вони прийшли через мережу
    return data

#Клас, що описує об’єкт proximity-карти
class ProximityCard:
#Конструктор класу для ініціалізації картки
    def __init__(self, card_id):
#Ініціалізація ідентифікатора картки
        self.card_id = card_id
#Створення контрольного числа для картки
        self.security_token = generate_hash(card_id, "default")
#Випадкове згенероване число для додаткової безпеки
        self.random_seed = random.randint(1000, 9999)
#Метод повернення інформації про картку
    def get_card_info(self):
#Повертаємо словник з даними картки
        return {"card_id": self.card_id, "token": self.security_token, "seed":
self.random_seed}

#Клас, що реалізує систему контролю доступу з використанням proximity-карт
class AccessControlSystem:
#Конструктор класу для ініціалізації системи
    def __init__(self, db_name=DATABASE_NAME):
#Ініціалізація з'єднання з базою даних
        self.conn = sqlite3.connect(db_name, check_same_thread=False)
#Створення курсору для виконання SQL запитів
        self.cursor = self.conn.cursor()
#Виклик методу ініціалізації бази даних
        self.initialize_db()
#Встановлення блокування для багатопотоковості
        self.db_lock = threading.Lock()
#Створення списку для зберігання сесійних ключів
        self.session_keys = []
```

```
#Ініціалізація активного статусу системи
        self.active = True
#Метод для створення необхідних таблиць у базі даних
    def initialize_db(self):
#Створення таблиці співробітників
        self.cursor.execute("""CREATE TABLE IF NOT EXISTS employees (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            card_id TEXT NOT NULL,
            name TEXT NOT NULL,
            access_level TEXT NOT NULL,
            encryption_key TEXT NOT NULL
        )""")
#Створення таблиці логів доступу
        self.cursor.execute("""CREATE TABLE IF NOT EXISTS access_logs (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT NOT NULL,
            card_id TEXT NOT NULL,
            access_result TEXT NOT NULL,
            event_message TEXT NOT NULL
        )""")
#Коміт змін у базі даних
        self.conn.commit()
#Метод для додавання нового співробітника у систему
    def add_employee(self, card_id, name, access_level, encryption_key):
#Закриття доступу до бази даних через блокування
        with self.db_lock:
#Виконання SQL запиту для додавання запису співробітника
            self.cursor.execute("INSERT INTO employees (card_id, name,
access_level, encryption_key) VALUES (?, ?, ?, ?)",
                                (card_id, name, access_level, encryption_key))
#Коміт змін після додавання співробітника
            self.conn.commit()
#Метод для запису події до журналу доступу
    def log_event(self, card_id, access_result, event_message):
#Отримання поточного часу для запису логів
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
#Закриття доступу до БД для багатопотокового запису
        with self.db_lock:
#SQL запит для запису інформації про подію
            self.cursor.execute("INSERT INTO access_logs (timestamp, card_id,
access_result, event_message) VALUES (?, ?, ?, ?)",
                                (current_time, card_id, access_result,
event_message))
#Коміт змін після запису події
            self.conn.commit()
#Запис події також в текстовий лог
        with open(ACCESS_LOG_FILE, "a") as log_file:
            log_file.write(f"{current_time} | Card: {card_id} | Result:
{access_result} | Message: {event_message}\n")
#Метод перевірки дійсності картки та доступу співробітника
    def check_access(self, card: ProximityCard):
#Отримання інформації про картку
        card_info = card.get_card_info()
#Закриття доступу до бази даних через блокування
        with self.db_lock:
#SQL запит для вибору співробітника за card_id
            self.cursor.execute("SELECT name, access_level, encryption_key FROM
employees WHERE card_id = ?", (card_info["card_id"],))
#Отримання запису співробітника
            employee = self.cursor.fetchone()
#Перевірка чи співробітник знайдений
        if employee is None:
#Логування події з відмовою у доступі
```

```
            self.log_event(card_info["card_id"], "DENIED", "Невідомий card_id -
доступ заборонено")
#Повернення результату з відмовою
            return False
        else:
#Розшифровка даних співробітника та перевірка токена
            name, access_level, encryption_key = employee
#Генерація контрольного токена для порівняння
            expected_token = generate_hash(card_info["card_id"], encryption_key)
#Перевірка відповідності токенів
            if expected_token == card_info["token"]:
#Логування події з дозволом на доступ
                self.log_event(card_info["card_id"], "ALLOWED", f"Доступ
дозволено для {name} з рівнем доступу {access_level}")
#Повернення результату з дозволом
                return True
            else:
#Логування події з відмовою через невідповідність токену
                self.log_event(card_info["card_id"], "DENIED", f"Невідповідність
токену для співробітника {name}")
#Повернення результату з відмовою
                return False
#Метод симуляції запиту до бази даних з використанням кількох SQL запитів
    def perform_security_queries(self, card: ProximityCard):
#Отримання інформації про картку
        card_info = card.get_card_info()
#SQL запит для вибору всіх співробітників
        with self.db_lock:
            self.cursor.execute("SELECT * FROM employees")
#Отримання всіх записів співробітників
            all_employees = self.cursor.fetchall()
#SQL запит для перевірки логів доступу
            self.cursor.execute("SELECT * FROM access_logs WHERE card_id = ?",
(card_info["card_id"],))
#Отримання логів, пов'язаних з карткою
            card_logs = self.cursor.fetchall()
#Додатковий SQL запит для підрахунку кількості успішних спроб доступу
            self.cursor.execute("SELECT COUNT(*) FROM access_logs WHERE
access_result = 'ALLOWED' AND card_id = ?", (card_info["card_id"],))
#Отримання кількості дозволених доступів
            allowed_count = self.cursor.fetchone()[0]
#Повернення результатів запитів як словник
        return {"all_employees": all_employees, "card_logs": card_logs,
"allowed_count": allowed_count}
#Метод для симуляції періодичного оновлення протоколів безпеки
    def update_security_protocols(self):
#Циклічне оновлення протоколів безпеки поки система активна
        while self.active:
#Генерація нового криптографічного ключа для сесії
            new_session_key = generate_hash(str(random.randint(10000, 99999)),
"session")
#Додавання нового ключа до списку сесійних ключів
            self.session_keys.append(new_session_key)
#Логування події оновлення протоколу
            self.log_event("SYSTEM", "INFO", "Оновлено протокол безпеки з новим
сесійним ключем")
#Очікування наступного циклу оновлення
            time.sleep(5)
#Метод для симуляції періодичної перевірки системної безпеки
    def run_security_check(self):
#Безкінечний цикл виконання перевірок
        while self.active:
#Перевірка журналу логів на невідповідності
```

```
            with self.db_lock:
#SQL запит для вибору останніх 5 подій безпеки
                self.cursor.execute("SELECT * FROM access_logs ORDER BY id DESC
LIMIT 5")
#Отримання останніх записів журналу
                recent_logs = self.cursor.fetchall()
#Обхід кожного запису для аналізу подій
            for log in recent_logs:
#Аналізуємо запис на наявність підозрілих подій
                if "DENIED" in log[3]:
#Логування події в консолі системи
                    print(f"Підозріла подія: {log}")
#Очікування перед наступною перевіркою
            time.sleep(10)
#Метод для симуляції процесу сканування картки
    def scan_card(self, card: ProximityCard):
#Отримання інформації картки
        card_info = card.get_card_info()
#Відправка даних картки через мережу з затримкою
        received_data = simulate_network_packet(card_info)
#Розшифровка отриманих даних (симуляція, тому просто передаємо дані)
        processed_data = received_data
#Перевірка доступу за допомогою методу check_access
        result = self.check_access(card)
#Вивід результату сканування картки
        if result:
            print(f"Доступ дозволено для картки {card_info['card_id']}")
        else:
            print(f"Доступ заборонено для картки {card_info['card_id']}")
#Виклик додаткових SQL запитів для аналізу безпеки
        query_results = self.perform_security_queries(card)
#Вивід результатів аналізу в консоль
        print("Кількість дозволених спроб доступу:",
query_results["allowed_count"])
#Метод для безпечного завершення роботи системи
    def shutdown_system(self):
#Встановлення статусу неактивності
        self.active = False
#Закриття з'єднання з базою даних
        with self.db_lock:
            self.conn.close()
#Логування події завершення роботи
        self.log_event("SYSTEM", "INFO", "Система контролю доступу вимкнена")
#Додаткова функція для генерації випадкових карток для тестування
def generate_random_card():
#Генерація випадкового ідентифікатора картки
    random_id = f"CARD{random.randint(1000, 9999)}"
#Повернення нового об'єкта картки
    return ProximityCard(random_id)
#Функція для попереднього заповнення бази даних співробітниками
def preload_employees(system: AccessControlSystem):
#Список співробітників для поповнення бази
    employees = [
        {"card_id": "CARD1234", "name": "Олександр", "access_level": "ADMIN",
"encryption_key": "key1234"},
        {"card_id": "CARD5678", "name": "Вікторія", "access_level": "USER",
"encryption_key": "key5678"},
        {"card_id": "CARD9012", "name": "Михайло", "access_level": "USER",
"encryption_key": "key9012"},
        {"card_id": "CARD3456", "name": "Світлана", "access_level": "MANAGER",
"encryption_key": "key3456"}
    ]
#Обхід списку співробітників та додавання кожного в базу
```

```
    for emp in employees:
        system.add_employee(emp["card_id"], emp["name"], emp["access_level"],
emp["encryption_key"])
#Функція для симуляції роботи системи контролю доступу
def simulate_system_operation():
#Створення об'єкта системи контролю доступу
    system = AccessControlSystem()
#Попереднє завантаження співробітників у базу
    preload_employees(system)
#Створення потоку для оновлення протоколів безпеки
    protocol_thread = threading.Thread(target=system.update_security_protocols)
    protocol_thread.daemon = True
    protocol_thread.start()
#Створення потоку для періодичної перевірки системної безпеки
    security_thread = threading.Thread(target=system.run_security_check)
    security_thread.daemon = True
    security_thread.start()
#Список тестових карток, включаючи відомі та невідомі
    test_cards = [
        ProximityCard("CARD1234"),
        ProximityCard("CARD5678"),
        ProximityCard("UNKNOWN001"),
        ProximityCard("CARD9012"),
        generate_random_card(),
        generate_random_card()
    ]
#Циклічний запуск сканування карток
    for card in test_cards:
#Сканування кожної картки із затримкою між запитами
        system.scan_card(card)
#Додаткове очікування для симуляції реального часу
        time.sleep(2)
#Виконання додаткових SQL запитів для тестування бази даних
    extra_query = system.perform_security_queries(ProximityCard("CARD1234"))
    print("Всі співробітники:", extra_query["all_employees"])
    print("Логи для CARD1234:", extra_query["card_logs"])
#Завершення роботи системи після симуляції
    system.shutdown_system()
#Точка входу у програму
if __name__ == "__main__":
#Запуск основної функції симуляції роботи системи
    simulate_system_operation()
#Кінець файлу з програмою
```

Файл RoleManager12.py

```
import sqlite3
import datetime
import hashlib
import random
import time
import threading
import smtplib
import tkinter as tk
from tkinter import scrolledtext
from flask import Flask, jsonify, request, render_template_string

DATABASE_NAME = ":memory:"
LOG_FILE = "system_access.log"
SYSTEM_STATUS = {"active": True, "last_update":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}

class RoleManager:
    def __init__(self):
        self.roles = {}
    def add_role(self, card_id, role, permissions):
        self.roles[card_id] = {"role": role, "permissions": permissions}
    def remove_role(self, card_id):
        if card_id in self.roles:
            del self.roles[card_id]
    def check_permission(self, card_id, permission):
        if card_id in self.roles:
            return permission in self.roles[card_id]["permissions"]
        return False
    def list_roles(self):
        return self.roles

class MultiFactorAuth:
    def __init__(self):
        self.pending = {}
    def send_verification(self, email):
        code = str(random.randint(100000, 999999))
        self.pending[email] = code
        print("Sending verification code to", email, "code:", code)
        return code
    def verify_code(self, email, code):
        if email in self.pending and self.pending[email] == code:
            del self.pending[email]
            return True
        return False

class AdvancedEncryption:
    def __init__(self):
        self.algorithm = "xor"
    def xor_encrypt(self, data, key):
        encrypted = ''.join(chr(ord(c) ^ ord(key[i % len(key)])) for i, c in
enumerate(data))
        return encrypted
    def xor_decrypt(self, encrypted, key):
        decrypted = ''.join(chr(ord(c) ^ ord(key[i % len(key)])) for i, c in
enumerate(encrypted))
        return decrypted
    def caesar_encrypt(self, data, shift):
        encrypted = ''
        for c in data:
            if c.isalpha():
                base = ord('A') if c.isupper() else ord('a')
```

```
                encrypted += chr((ord(c) - base + shift) % 26 + base)
            else:
                encrypted += c
        return encrypted
    def caesar_decrypt(self, encrypted, shift):
        return self.caesar_encrypt(encrypted, -shift)
    def encrypt(self, data, key):
        encrypted1 = self.xor_encrypt(data, key)
        encrypted2 = self.caesar_encrypt(encrypted1, 3)
        return encrypted2
    def decrypt(self, encrypted, key):
        decrypted1 = self.caesar_decrypt(encrypted, 3)
        decrypted2 = self.xor_decrypt(decrypted1, key)
        return decrypted2

class ExtendedAccessControlSystem:
    def __init__(self, db_name=DATABASE_NAME):
        self.conn = sqlite3.connect(db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.cursor.execute("CREATE TABLE IF NOT EXISTS employees (id INTEGER
PRIMARY KEY AUTOINCREMENT, card_id TEXT NOT NULL, name TEXT NOT NULL,
access_level TEXT NOT NULL, encryption_key TEXT NOT NULL)")
        self.cursor.execute("CREATE TABLE IF NOT EXISTS access_logs (id INTEGER
PRIMARY KEY AUTOINCREMENT, timestamp TEXT NOT NULL, card_id TEXT NOT NULL,
access_result TEXT NOT NULL, event_message TEXT NOT NULL)")
        self.conn.commit()
        self.db_lock = threading.Lock()
        self.session_keys = []
        self.active = True
        self.role_manager = RoleManager()
        self.mfa = MultiFactorAuth()
        self.encryption_module = AdvancedEncryption()
    def add_employee(self, card_id, name, access_level, encryption_key):
        with self.db_lock:
            self.cursor.execute("INSERT INTO employees (card_id, name,
access_level, encryption_key) VALUES (?, ?, ?, ?)", (card_id, name,
access_level, encryption_key))
            self.conn.commit()
            self.role_manager.add_role(card_id, access_level, ["access"])
    def log_event(self, card_id, access_result, event_message):
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with self.db_lock:
            self.cursor.execute("INSERT INTO access_logs (timestamp, card_id,
access_result, event_message) VALUES (?, ?, ?, ?)", (current_time, card_id,
access_result, event_message))
            self.conn.commit()
        with open(LOG_FILE, "a") as f:
            f.write(f"{current_time} | Card: {card_id} | Result: {access_result}
| Message: {event_message}\n")
    def check_access(self, card_id, token):
        with self.db_lock:
            self.cursor.execute("SELECT name, access_level, encryption_key FROM
employees WHERE card_id = ?", (card_id,))
            employee = self.cursor.fetchone()
        if employee is None:
            self.log_event(card_id, "DENIED", "Unknown card_id")
            return False
        name, access_level, encryption_key = employee
        expected_token = hashlib.sha256((card_id + encryption_key).encode('utf-
8')).hexdigest()
        if expected_token == token:
            self.log_event(card_id, "ALLOWED", f"Access granted for {name}")
            return True
```

```
        self.log_event(card_id, "DENIED", f"Token mismatch for {name}")
        return False
    def perform_security_queries(self, card_id):
        with self.db_lock:
            self.cursor.execute("SELECT * FROM employees")
            all_employees = self.cursor.fetchall()
            self.cursor.execute("SELECT * FROM access_logs WHERE card_id = ?",
(card_id,))
            card_logs = self.cursor.fetchall()
            self.cursor.execute("SELECT COUNT(*) FROM access_logs WHERE
access_result = 'ALLOWED' AND card_id = ?", (card_id,))
            allowed_count = self.cursor.fetchone()[0]
        return {"all_employees": all_employees, "card_logs": card_logs,
"allowed_count": allowed_count}
    def update_security_protocols(self):
        while self.active:
            new_session_key = hashlib.sha256(str(random.randint(10000,
99999)).encode('utf-8')).hexdigest()
            self.session_keys.append(new_session_key)
            self.log_event("SYSTEM", "INFO", "Security protocol updated with new
session key")
            time.sleep(5)
    def run_security_check(self):
        while self.active:
            with self.db_lock:
                self.cursor.execute("SELECT * FROM access_logs ORDER BY id DESC
LIMIT 5")
                recent_logs = self.cursor.fetchall()
            for log in recent_logs:
                if "DENIED" in log[3]:
                    print("Suspicious event detected:", log)
            time.sleep(10)
    def shutdown_system(self):
        self.active = False
        with self.db_lock:
            self.conn.close()
        self.log_event("SYSTEM", "INFO", "Access control system shutdown")
    def scan_card(self, card_id, token):
        result = self.check_access(card_id, token)
        if result:
            print(f"Access allowed for card {card_id}")
        else:
            print(f"Access denied for card {card_id}")
        query_results = self.perform_security_queries(card_id)
        print("Allowed access count:", query_results["allowed_count"])
        return result

app = Flask(__name__)
system_instance = ExtendedAccessControlSystem()

@app.route("/")
def index():
    with system_instance.db_lock:
        system_instance.cursor.execute("SELECT * FROM access_logs ORDER BY id
DESC LIMIT 20")
        logs = system_instance.cursor.fetchall()
    html = "<html><head><title>Remote Monitoring</title></head><body><h1>Access
Logs</h1><table border='1'><tr><th>ID</th><th>Timestamp</th><th>Card
ID</th><th>Result</th><th>Message</th></tr>"
    for log in logs:
        html +=
f"<tr><td>{log[0]}</td><td>{log[1]}</td><td>{log[2]}</td><td>{log[3]}</td><td>{l
og[4]}</td></tr>"
```

```
    html += "</table></body></html>"
    return html

@app.route("/status")
def status():
    return jsonify(SYSTEM_STATUS)

@app.route("/add_employee", methods=["POST"])
def add_employee():
    data = request.json
    card_id = data.get("card_id")
    name = data.get("name")
    access_level = data.get("access_level")
    encryption_key = data.get("encryption_key")
    system_instance.add_employee(card_id, name, access_level, encryption_key)
    return jsonify({"status": "Employee added"})

@app.route("/get_roles")
def get_roles():
    roles = system_instance.role_manager.list_roles()
    return jsonify(roles)

class DashboardGUI:
    def __init__(self, system):
        self.system = system
        self.root = tk.Tk()
        self.root.title("Interactive Dashboard")
        self.text_area    =    scrolledtext.ScrolledText(self.root,    width=100,
height=30)
        self.text_area.pack()
        self.update_interval = 5000
        self.refresh_dashboard()
    def refresh_dashboard(self):
        with self.system.db_lock:
            self.system.cursor.execute("SELECT  *  FROM  access_logs  ORDER  BY  id
DESC LIMIT 20")
            logs = self.system.cursor.fetchall()
        self.text_area.delete("1.0", tk.END)
        for log in logs:
            self.text_area.insert(tk.END, f"{log}\n")
        self.root.after(self.update_interval, self.refresh_dashboard)
    def run(self):
        self.root.mainloop()

def preload_employees(system):
    employees = [
        {"card_id":  "CARD1001",  "name":  "Anand",  "access_level":  "ADMIN",
"encryption_key": "adminkey1"},
        {"card_id":  "CARD1002",  "name":  "Bhavna",  "access_level":  "USER",
"encryption_key": "userkey2"},
        {"card_id":  "CARD1003",  "name":  "Chandra",  "access_level":  "MANAGER",
"encryption_key": "mankey3"},
        {"card_id":  "CARD1004",  "name":  "Deepak",  "access_level":  "USER",
"encryption_key": "userkey4"}
    ]
    for emp in employees:
        system.add_employee(emp["card_id"],  emp["name"],  emp["access_level"],
emp["encryption_key"])

def run_flask_app():
    app.run(port=5000)

def run_tkinter_dashboard():
```

```
    dashboard = DashboardGUI(system_instance)
    dashboard.run()

def run_security_protocol_thread():
    system_instance.update_security_protocols()

def run_security_check_thread():
    system_instance.run_security_check()

def simulate_mfa():
    email = "user@example.com"
    code_sent = system_instance.mfa.send_verification(email)
    print("MFA verification for", email, "code sent:", code_sent)
    time.sleep(2)
    if system_instance.mfa.verify_code(email, code_sent):
        print("MFA verified for", email)
    else:
        print("MFA verification failed for", email)

def simulate_card_scanning():
    cards = [
        {"card_id": "CARD1001", "encryption_key": "adminkey1"},
        {"card_id": "CARD1002", "encryption_key": "userkey2"},
        {"card_id": "CARD9999", "encryption_key": "wrongkey"},
        {"card_id": "CARD1003", "encryption_key": "mankey3"}
    ]
    for card in cards:
        token           =           hashlib.sha256((card["card_id"]           +
card["encryption_key"]).encode('utf-8')).hexdigest()
        system_instance.scan_card(card["card_id"], token)
        time.sleep(3)

def simulate_encryption():
    data = "SensitiveAccessData"
    key = "SecretKey123"
    encrypted = system_instance.encryption_module.encrypt(data, key)
    print("Encrypted data:", encrypted)
    decrypted = system_instance.encryption_module.decrypt(encrypted, key)
    print("Decrypted data:", decrypted)

def main():
    preload_employees(system_instance)
    threading.Thread(target=run_security_protocol_thread, daemon=True).start()
    threading.Thread(target=run_security_check_thread, daemon=True).start()
    threading.Thread(target=run_flask_app, daemon=True).start()
    threading.Thread(target=run_tkinter_dashboard, daemon=True).start()
    simulate_mfa()
    simulate_card_scanning()
    simulate_encryption()
    time.sleep(60)
    system_instance.shutdown_system()

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

Файл LogRotatorX.py

```
import os
import time
import threading
import datetime
import random
import smtplib
from email.mime.text import MIMEText

class LogRotator:
    def __init__(self, log_file, max_size=1024):
        self.log_file = log_file
        self.max_size = max_size
        self.rotation_interval = 10
        self.running = True
        self.thread = threading.Thread(target=self.rotate_log)
        self.thread.daemon = True
    def start(self):
        self.thread.start()
    def rotate_log(self):
        while self.running:
            if os.path.exists(self.log_file):
                size = os.path.getsize(self.log_file)
                if size > self.max_size:
                    timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
                    new_name = self.log_file + "." + timestamp
                    os.rename(self.log_file, new_name)
                    open(self.log_file, "w").close()
            time.sleep(self.rotation_interval)
    def stop(self):
        self.running = False
        self.thread.join()

class BiometricScanner:
    def __init__(self):
        self.fingerprint_db = {"user1": "fp_hash_1", "user2": "fp_hash_2"}
        self.face_db = {"user1": "face_hash_1", "user2": "face_hash_2"}
    def scan_fingerprint(self, user_id):
        simulated_hash = "fp_hash_" + str(random.randint(1, 2))
        return self.fingerprint_db.get(user_id, None) == simulated_hash
    def scan_face(self, user_id):
        simulated_hash = "face_hash_" + str(random.randint(1, 2))
        return self.face_db.get(user_id, None) == simulated_hash

class VideoSurveillance:
    def __init__(self):
        self.camera_running = False
        self.snapshots = []
        self.capture_interval = 3
        self.thread = threading.Thread(target=self.capture_loop)
        self.thread.daemon = True
        self.running = False
    def start_camera(self):
        self.camera_running = True
        self.running = True
        self.thread.start()
    def capture_loop(self):
        while self.running:
            if self.camera_running:
                snapshot = "Snapshot_" +
datetime.datetime.now().strftime("%Y%m%d%H%M%S") + "_" + str(random.randint(100,
999))
```

```
                self.snapshots.append(snapshot)
            time.sleep(self.capture_interval)
    def stop_camera(self):
        self.camera_running = False
    def get_snapshots(self):
        return self.snapshots
    def clear_snapshots(self):
        self.snapshots = []

class AnomalyDetector:
    def __init__(self, log_file):
        self.log_file = log_file
        self.denied_threshold = 5
        self.anomalies = []
    def analyze_logs(self):
        if os.path.exists(self.log_file):
            with open(self.log_file, "r") as f:
                lines = f.readlines()
            denied_count = 0
            for line in lines:
                if "DENIED" in line:
                    denied_count += 1
            if denied_count >= self.denied_threshold:
                anomaly = "High number of denied accesses: " + str(denied_count)
                self.anomalies.append(anomaly)
                return anomaly
        return None
    def get_anomalies(self):
        return self.anomalies

class NotificationSystem:
    def __init__(self, smtp_server="localhost", smtp_port=25,
sender_email="noreply@example.com"):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.sender_email = sender_email
        self.recipients = []
    def add_recipient(self, email):
        self.recipients.append(email)
    def send_notification(self, subject, message):
        msg = MIMEText(message)
        msg["Subject"] = subject
        msg["From"] = self.sender_email
        for recipient in self.recipients:
            msg["To"] = recipient
            try:
                server = smtplib.SMTP(self.smtp_server, self.smtp_port)
                server.sendmail(self.sender_email, recipient, msg.as_string())
                server.quit()
            except Exception as e:
                print("Error sending email to", recipient, ":", e)

def simulate_log_rotation():
    rotator = LogRotator("system_access.log", max_size=512)
    rotator.start()
    for i in range(20):
        with open("system_access.log", "a") as f:
            f.write("Log entry " + str(i) + "\n")
        time.sleep(0.5)
    rotator.stop()

def simulate_biometric_scans():
    scanner = BiometricScanner()
```

```
    results = {}
    for user in ["user1", "user2", "user3"]:
        fp_result = scanner.scan_fingerprint(user)
        face_result = scanner.scan_face(user)
        results[user] = {"fingerprint": fp_result, "face": face_result}
    print("Biometric scan results:", results)

def simulate_video_surveillance():
    surveillance = VideoSurveillance()
    surveillance.start_camera()
    time.sleep(10)
    surveillance.stop_camera()
    snaps = surveillance.get_snapshots()
    print("Video surveillance snapshots:", snaps)
    surveillance.clear_snapshots()

def simulate_anomaly_detection():
    detector = AnomalyDetector("system_access.log")
    with open("system_access.log", "w") as f:
        for i in range(10):
            if i % 2 == 0:
                f.write("20250101010101 | CARD1001 | DENIED | Access denied
simulation\n")
            else:
                f.write("20250101010101 | CARD1001 | ALLOWED | Access allowed
simulation\n")
    anomaly = detector.analyze_logs()
    if anomaly:
        print("Anomaly detected:", anomaly)
    else:
        print("No anomaly detected")

def simulate_notifications():
    notifier = NotificationSystem(smtp_server="localhost", smtp_port=1025,
sender_email="alert@example.com")
    notifier.add_recipient("admin@example.com")
    notifier.add_recipient("security@example.com")
    notifier.send_notification("Security Alert", "An anomaly has been detected
in the access logs.")
    print("Notification sent")

def run_all_new_features():
    simulate_log_rotation()
    simulate_biometric_scans()
    simulate_video_surveillance()
    simulate_anomaly_detection()
    simulate_notifications()

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

Файл X Interface.py

```
import sqlite3
import datetime
import hashlib
import random
import time
import threading
import json
import csv
import os
from flask import Flask, jsonify, request, render_template_string

class MobileInterface:
    def __init__(self):
        self.app = Flask("MobileInterface")
        self.setup_routes()
    def setup_routes(self):
        @self.app.route("/mobile/dashboard")
        def dashboard():
            data = {"status": "ok", "date":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"), "message": "Mobile
dashboard data"}
            return jsonify(data)
        @self.app.route("/mobile/scan", methods=["POST"])
        def scan():
            data = request.get_json()
            card_id = data.get("card_id", "")
            response = {"card_id": card_id, "access": "granted" if
random.choice([True, False]) else "denied", "time":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
            return jsonify(response)
    def run(self, port=5001):
        self.app.run(port=port)

class ElectronicLock:
    def __init__(self):
        self.door_status = "locked"
        self.alarm_status = "off"
    def lock(self):
        self.door_status = "locked"
        return self.door_status
    def unlock(self):
        self.door_status = "unlocked"
        return self.door_status
    def trigger_alarm(self):
        self.alarm_status = "on"
        return self.alarm_status
    def reset_alarm(self):
        self.alarm_status = "off"
        return self.alarm_status
    def get_status(self):
        return {"door_status": self.door_status, "alarm_status":
self.alarm_status}

class AuditReportGenerator:
    def __init__(self, db_path="access_control.db"):
        self.db_path = db_path
        self.report_file = "audit_report.txt"
        self.create_dummy_logs()
    def create_dummy_logs(self):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
```

```
        c.execute("CREATE TABLE IF NOT EXISTS access_logs (id INTEGER PRIMARY
KEY AUTOINCREMENT, timestamp TEXT, card_id TEXT, result TEXT, message TEXT)")
        for i in range(1, 21):
            timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            card_id = "CARD" + str(1000 + i)
            result = random.choice(["ALLOWED", "DENIED"])
            message = "Simulated log entry " + str(i)
            c.execute("INSERT INTO access_logs (timestamp, card_id, result,
message) VALUES (?, ?, ?, ?)", (timestamp, card_id, result, message))
        conn.commit()
        conn.close()
    def generate_report(self):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        c.execute("SELECT * FROM access_logs")
        rows = c.fetchall()
        report_lines = []
        for row in rows:
            line = "ID: " + str(row[0]) + " | Time: " + row[1] + " | Card: " +
row[2] + " | Result: " + row[3] + " | Message: " + row[4]
            report_lines.append(line)
        conn.close()
        with open(self.report_file, "w") as f:
            for line in report_lines:
                f.write(line + "\n")
        return self.report_file

class CardDataBackup:
    def __init__(self, db_path="employee_data.db"):
        self.db_path = db_path
        self.backup_file = "card_backup.json"
        self.create_dummy_employees()
    def create_dummy_employees(self):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS employees (id INTEGER PRIMARY KEY
AUTOINCREMENT, card_id TEXT, name TEXT, access_level TEXT, encryption_key
TEXT)")
        employees = [
            ("CARD2001", "Ravi", "ADMIN", "key2001"),
            ("CARD2002", "Sita", "USER", "key2002"),
            ("CARD2003", "Amit", "USER", "key2003"),
            ("CARD2004", "Priya", "MANAGER", "key2004")
        ]
        c.executemany("INSERT INTO employees (card_id, name, access_level,
encryption_key) VALUES (?, ?, ?, ?)", employees)
        conn.commit()
        conn.close()
    def backup_data(self):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        c.execute("SELECT * FROM employees")
        rows = c.fetchall()
        data_list = []
        for row in rows:
            data_list.append({"id": row[0], "card_id": row[1], "name": row[2],
"access_level": row[3], "encryption_key": row[4]})
        conn.close()
        with open(self.backup_file, "w") as f:
            json.dump(data_list, f, indent=4)
        return self.backup_file

class ImprovedUI:
```

```
    def __init__(self):
        self.app = Flask("ImprovedUI")
        self.setup_routes()
    def setup_routes(self):
        @self.app.route("/ui/dashboard")
        def dashboard():
            template = """
            <html>
            <head>
                <title>Improved Dashboard</title>
                <style>
                    body {font-family: Arial, sans-serif; background-color:
#f0f0f0; margin: 0; padding: 0;}
                    .header {background-color: #4CAF50; color: white; padding:
20px; text-align: center;}
                    .content {padding: 20px;}
                    table {width: 100%; border-collapse: collapse;}
                    th, td {border: 1px solid #dddddd; padding: 8px; text-align:
left;}
                    tr:nth-child(even) {background-color: #dddddd;}
                </style>
            </head>
            <body>
                <div class="header">
                    <h1>Improved Dashboard</h1>
                </div>
                <div class="content">
                    <h2>Access Log Summary</h2>
                    <table>
                        <tr>
                            <th>ID</th>
                            <th>Timestamp</th>
                            <th>Card ID</th>
                            <th>Result</th>
                            <th>Message</th>
                        </tr>
                        {% for log in logs %}
                        <tr>
                            <td>{{ log[0] }}</td>
                            <td>{{ log[1] }}</td>
                            <td>{{ log[2] }}</td>
                            <td>{{ log[3] }}</td>
                            <td>{{ log[4] }}</td>
                        </tr>
                        {% endfor %}
                    </table>
                </div>
            </body>
            </html>
            """
            conn = sqlite3.connect("access_control.db")
            c = conn.cursor()
            c.execute("SELECT * FROM access_logs ORDER BY id DESC LIMIT 20")
            logs = c.fetchall()
            conn.close()
            return render_template_string(template, logs=logs)
    def run(self, port=5002):
        self.app.run(port=port)

def simulate_mobile_interface():
    mobile = MobileInterface()
    threading.Thread(target=mobile.run,                    kwargs={"port":5001},
daemon=True).start()
```

```
    time.sleep(1)

def simulate_improved_ui():
    ui = ImprovedUI()
    threading.Thread(target=ui.run, kwargs={"port":5002}, daemon=True).start()
    time.sleep(1)

def simulate_electronic_lock():
    lock = ElectronicLock()
    lock.unlock()
    time.sleep(1)
    lock.lock()
    time.sleep(1)
    lock.trigger_alarm()
    time.sleep(1)
    lock.reset_alarm()
    print("ElectronicLock status", lock.get_status())

def simulate_audit_report():
    audit = AuditReportGenerator()
    report_file = audit.generate_report()
    with open(report_file, "r") as f:
        content = f.read()
    print("Audit report generated:\n", content)

def simulate_card_backup():
    backup = CardDataBackup()
    backup_file = backup.backup_data()
    with open(backup_file, "r") as f:
        content = f.read()
    print("Card backup data:\n", content)

def main():
    simulate_mobile_interface()
    simulate_improved_ui()
    simulate_electronic_lock()
    simulate_audit_report()
    simulate_card_backup()
    while True:
        time.sleep(1)

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

Файл Xlog.py

```
import threading
import binascii
import logging
import errno
import ndef
import nfc


log = logging.getLogger(__name__)


class HandoverServer(threading.Thread):
    """ NFC Forum Connection Handover server
    """
    def __init__(self, llc, request_size_limit=0x10000,
                 recv_miu=1984, recv_buf=15):
        socket = nfc.llcp.Socket(llc, nfc.llcp.DATA_LINK_CONNECTION)
        recv_miu = socket.setsockopt(nfc.llcp.SO_RCVMIU, recv_miu)
        recv_buf = socket.setsockopt(nfc.llcp.SO_RCVBUF, recv_buf)
        socket.bind('urn:nfc:sn:handover')
        log.info("handover server bound to port {0} (MIU={1}, RW={2})"
                 .format(socket.getsockname(), recv_miu, recv_buf))
        socket.listen(backlog=2)
        threading.Thread.__init__(self, name='urn:nfc:sn:handover',
                                  target=self.listen, args=(llc, socket))

    def listen(self, llc, socket):
        log.debug("handover listen thread started")
        try:
            while True:
                client_socket = socket.accept()
                client_thread = threading.Thread(target=self.serve,
                                                 args=(client_socket,))
                client_thread.start()
        except nfc.llcp.Error as error:
            (log.debug if error.errno == errno.EPIPE else log.error)(error)
        finally:
            socket.close()
            log.debug("handover listen thread terminated")

    def serve(self, socket):
        peer_sap = socket.getpeername()
        log.info("serving handover client on remote sap {0}".format(peer_sap))
        send_miu = socket.getsockopt(nfc.llcp.SO_SNDMIU)
        try:
            while socket.poll("recv"):
                request = bytearray()
                while socket.poll("recv"):
                    request += socket.recv()

                    if len(request) == 0:
                        continue  # need some data

                    try:
                        list(ndef.message_decoder(request, 'strict', {}))
                    except ndef.DecodeError:
                        continue  # need more data

                    response = self._process_request_data(request)

                    for offset in range(0, len(response), send_miu):
```

```
                        fragment = response[offset:offset + send_miu]
                        if not socket.send(fragment):
                            return  # connection closed

        except nfc.llcp.Error as error:
            (log.debug if error.errno == errno.EPIPE else log.error)(error)
        finally:
            socket.close()
            log.debug("handover serve thread terminated")

    def _process_request_data(self, octets):
        log.debug("<<< %s", binascii.hexlify(octets).decode())
        try:
            records = list(ndef.message_decoder(octets, 'relax'))
        except ndef.DecodeError as error:
            log.error(repr(error))
            return b''

        if records[0].type == 'urn:nfc:wkt:Hr':
            records = self.process_handover_request_message(records)
        else:
            log.error("received unknown request message")
            records = []

        octets = b''.join(ndef.message_encoder(records))
        log.debug(">>> %s", binascii.hexlify(octets).decode())
        return octets

    def process_handover_request_message(self, records):
        """Process a handover request message. The *records* argument holds a
        list of :class:`ndef.Record` objects decoded from the received
        handover request message octets, where the first record type is
        ``urn:nfc:wkt:Hr``. The method returns a list of :class:`ndef.Record`
        objects with the first record typ ``urn:nfc:wkt:Hs``.

        This method should be overwritten by a subclass to customize
        it's behavior. The default implementation returns a
        :class:`ndef.HandoverSelectRecord` with version ``1.2`` and no
        alternative carriers.

        """
        log.warning("default process_request method should be overwritten")
        return [ndef.HandoverSelectRecord('1.2')]
```