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 для біоідентифікації особи за райдужною оболонкою ока”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Філіпенко М.С.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.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

Філіпенку Микиті Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 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 для біоідентифікації особи за райдужною оболонкою ока*

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 О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Філіпенко М.С.
(прізвище та ініціали)

АНОТАЦІЯ

Філіпенко М.С. Програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока. 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 для біоідентифікації особи за райдужною оболонкою ока.

Метою розробки є програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

Результат роботи – програмна реалізація системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

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 Visual C++, HTML, ASP, Jscript.

Ключові слова: кібербезпека, біоідентифікація

ABSTRACT

Filipenko M.S. Software for a cybersecurity system for bioidentification of a person by the iris. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a cybersecurity system for bioidentification of a person by the iris.

The purpose of the development is the software for a cybersecurity system for bioidentification of a person by the iris.

The result of the work is the software implementation of a cybersecurity system for bioidentification of a person by the iris.

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 was developed in Visual C++, HTML, ASP, Jscript.

Keywords: cybersecurity, bioidentification

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM – електронно–обчислювальна машина

IT – інформаційні технології

API – прикладний програмний інтерфейс

AppWizard – засоби автоматизованого створення додатків

CEBIT – комп’ютерна виставка

FAR – False Acceptance Rate

FRR – False Rejection Rate

ISO/IEC – стандарт

JTC1 – міжнародний комітет зі стандартизації в області IT

Md5 – алгоритм шифрування

MFC – Microsoft Foundation Class library

NIST – національний інститут стандартизації

OLE – технологія зв'язування й вбудовування об'єктів

PIN – personal identification number

SC37 – спеціальний біопараметричний комітет

SSL – Secure Sockets Layer

USB – universal serial bus

ВСТУП

Актуальність теми. Ми вже чули про ідентифікацію особистості за відбитками пальців (дактилоскопія) – найпоширеніший спосіб, розпізнавання за особою, за долонями рук, за ДНК, за ходою, за голосом. Думаю, не новина й розпізнавання за райдужною оболонкою ока. Де в повсякденному житті ми можемо вже сьогодні зустрітися із цим унікальним методом?

Райдужна оболонка ока є унікальною характеристикою людини. Дана технологія розпізнавання заснована на розходженнях у рисунку райдужної оболонки ока у видимому (або інфрачервоному) світлі.

Основою для ідентифікації є видимі розподіли райдужної оболонки на радіальні сектори, кільця, борозни, ластовиння, що для кожної людини є індивідуальні й неповторним як відбитки пальців. Щільність інформації, що витягається, така, що можна говорити, що райдужна оболонка має 266 унікальних «крапок» ідентифікації в порівнянні з 10-60 крапками для інших біометричних методів.

Дослідники з університету Карнегі Меллон зараз випробовують технологію, що дозволяє зробити автентифікацію людини за райдужною оболонкою ока на відстані до 12 м. Подібно іншим способам біометричної ідентифікації система аналізує отримане з відеокамери або фотокамери зображення, порівнюючи рисунок зі зразками, наявними в базі даних.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

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

– Огляд існуючих систем для біоідентифікації особи за райдужною оболонкою ока.

– Дослідження системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

– Програмна реалізація системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для біоідентифікації особи за райдужною оболонкою ока.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока, 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

Виявилося, що за допомогою камери високого дозволу можливо ідентифікувати людину на великій відстані, аналізуючи світло ближньої інфрачервоної області спектра.

Система здатна розпізнати людини в різних світлових умовах. Вона може використовуватися навіть для аналізу зображення, відбитого в дзеркалі. Фахівці опублікували відео, на якому людина була успішно ідентифікована за райдужною оболонкою ока під час типової перевірки поліцією автомобіля. Система розпізнала людину по відбитому в бічному дзеркалі заднього виду зображенню. Така система знайде застосування при перевірці безпеки в місцях масового скупчення людей – в аеропортах, на вокзалах або стадіонах. Крім цього, її можуть використовувати як міра безпеки в урядових закладах.

1.2 Область застосування

Компанія EyeTicket і Міжнародна асоціація повітряного транспорту тестують нову технологію – цифрова камера буде робити знімок райдужної оболонки ока пасажира. Спеціальне програмне забезпечення за рисунком радужки буде встановлювати номер паспорта часто літаючого пасажира, і комп'ютери аеропорту й авіаліній будуть автоматично «дізнаватися» людини. Тепер туристи, що нудяться в довгих чергах чекаючи паспортного контролю й огляду багажу, незабаром зможуть уникнути цієї неприємної процедури. Крок до вдосконалення закордонних паспортів – оснащення їх безконтактним мікрочипом, на якому будуть закодовані біометричні дані власника (наприклад, форма особи, райдужна оболонка ока, відбитки пальців, геометрія руки).

Китайська компанія ZTE, що займається виробництвом електроніки, випустила унікальний смартфон, який можна розблокувати й одержати доступ до інформації за допомогою погляду. Щоб ідентифікувати власника, цей пристрій застосовує новітній біометричний чип. Споконвічно комп'ютер запам'ятовує унікальний судинний рисунок очного яблука. Після ж цього, для виходу з режиму блокування, смартфон необхідно піднести до ока.

Люди, які першими змогли випробувати нововведення, затверджують, що такий тип розблокування набагато зручніше, ніж введення пароля, малювання геометричного рисунка на тачскріні або ж відбиток пальця. Час, необхідний для запам'ятовування судинного рисунка, становить близько 8 секунд. Наступна ідентифікація відбувається протягом секунди.

Подібний біометричний метод набагато дешевше, ніж аналогічний з відбитком пальця, тому що для цього не потрібно обладнати смартфон якими-небудь додатковими пристроями.

Визначати користувача по індивідуальному судинному рисунку можна в будь-яких електронних пристроях, постачених камерою. Необхідно тільки встановити на нього відповідну програму.

Ринок біометрії активно розвивається, що пов'язане із проблемою безпеки в усьому світі. Можливо, у швидкому майбутньому ідентифікація за райдужною оболонкою ока стане повсюдної.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока, 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

BioID

Платформа BioID надає широкі можливості біометричних технологій у розпорядження компаній і підприємств, оптимізуючих бізнес-процеси й орієнтованих на ефективне обслуговування своїх замовників, клієнтів, покупців, відвідувачів.

Внутрішня діяльність компанії і її взаємодія із зовнішнім оточенням вимагають надійної, швидкої й зручної ідентифікації всіх учасників даних процесів (включаючи постійних співробітників і знову, що прибувають відвідувачів), і ці найважливіші вимоги реалізує платформа BioID.

Ідентифікація користувачів здійснюється по їхніх біометричних параметрах (відбиткам пальців). Розпізнавання проходить максимально комфортно (за 1-2 секунди) і не вимагає пред'явлення яких-небудь карт, талонів і т.п. Біометричні ідентифікатори унікальні для кожної людини, тому не доводиться сумніватися в результатах розпізнавання.

Платформа BioID включає готові засоби наочного подання інформації про ці результати. Зазначена інформація відразу ж надається зацікавленим у ній менеджерам і вищому керівництву компанії.

Сфери застосування BioID**:**

– підприємства, що діють в індустрії відпочинку й гостинності – реєстрація відвідувачів, у т.ч. речі, що здає, у гардероб (камеру схову) і/або на час покидаючих будинок з наміром повернутися в нього, прокат інвентарю;

– спортивні й фізкультурно-оздоровчі комплекси й центри – обслуговування власників абонементів, розмежування доступу в спортивні зали, басейни й т.д. відповідно до оплачених видів послуг;

– установи культури, навчальні й дошкільні заклади – ідентифікація абітурієнтів при проходженні вступних випробувань, організація іспитів і дистанційного навчання, обслуговування в бібліотеках, їдалень, кафетеріях; ідентифікація батьків, інших родичів, опікунів і інших дорослих, що мають право супроводжувати дитину, що залишає школу або дитячий садок;

– установи охорони здоров'я – реєстрація й обслуговування пацієнтів, розмежування доступу до медичних інформаційних систем, електронним історіям хвороби, організація видачі фармацевтичних препаратів;

– підприємства торгівлі й сфери послуг – реєстрація постійних покупців і клієнтів, учасників програм лояльності й інших осіб, яким надаються додаткові бонуси й знижки;

– транспортні підприємства – супровід часто подорожуючих пасажирів і осіб, що обслуговуються по VIP -програмах, прискорення передполітного контролю;

– фінансові й страхові компанії, банки – ідентифікація постійних клієнтів, претендентів на одержання кредиту й/або висновок страхового договору.

Сценарії використання BioID**:**

– негайне застосування готових засобів реєстрації користувачів, генерації звітів і форм виводу інформації, що входять у комплект поставки платформи BioID;

– швидка інтеграція функцій біометричної ідентифікації в інші інформаційні й управлінські системи, що функціонують у компанії;

– розробка (з мінімальними витратами часу й сил) власних прикладних біометричних рішень на базі BioID.

Алгоритм дії BioID:

– Внесення даних про користувачів: ПІБ (при необхідності), цифрова фотографія (можуть застосовуватися будь-які цифрові камери й апарати),

біометричні ідентифікатори, контактні відомості, будь-яка інша додаткова інформація. Конкретний состав і обсяг даних про користувачів визначаються в процесі експлуатації платформи.

– Реєстрація чергового відвідування користувача. Здійснюється за результатами біометричної ідентифікації (сканування відбитка пальця). Супроводжується виводом інформації на екран монітора: фотографії користувача, його ПІБ, інших відомостей, внесених на першому етапі. Результати реєстрації автоматично заносяться також у звіти.

– Аналіз звітів і/або трансляція даних про реєстрацію відвідування в інші інформаційні системи (наприклад, оповіщення лікаря про прихід пацієнта, оцінка про явку студента на іспит і т.п.).

Клієнт-серверна архітектура. Компоненти BioID

У платформі BioID реалізований клієнт-серверні технології: ефективні, відказостійкі, масштабовані.

Відомості про користувачів зберігаються в базі даних, кількість користувачів не обмежено. Для керування базою даних застосовується Microsoft SQL Server.

Запити до бази даних, що надходять від клієнтських додатків, обробляються спеціалізованим біометричним сервером – BioLink Server.

Клієнтські додатки:

– BioID Manager – з його допомогою вводяться дані про користувачів, формуються звіти, здійснюється керування функціонуванням платформи;

– BioID CheckPoint – реєструє відвідування користувачів, виводить на екран монітора інформацію про фіксуєму подію;

– BioID Configurator – управляє розташуванням на екрані монітора форм виводу інформації й дозволяє редагувати ці форми

– Функціонуванням платформи управляє BioLink Server Manager. Він дозволяє набудовувати підключення до бази даних і здійснювати резервне копіювання й відновлення бази даних.

Кастомізація. Розширені можливості

Кожне із клієнтських додатків BioID складається програмної оболонки й модулів, що підключаються до неї. У комплект поставки входять основні модулі список реєструємих подій, вікно уведення даних про користувача й т.д. У процесі експлуатації BioID можна легко й швидко формувати нові модулі, що відбивають логіку бізнес-процесів у конкретній компанії, або доповнювати існуючі моделі новими елементами. Наприклад, можна виводити на екран список послуг, доступних даному відвідувачеві фітнес-центра, час прийому й кабінети лікарів, які працюють із конкретним пацієнтом, розклад і результати здачі іспитів даним студентом і т.п.

Замовникові надається повний контроль над кількістю, місцем розташування й інтерфейсом модулів. У розпорядженні замовника перебувають як готові блоки, доступні до застосування відразу ж після установки BioID, так і зручний конструктор нових форм, що повністю відповідають потребам роботи в даній компанії.

Ліцензійна політика

Стартовий комплект платформи BioID включає ліцензії на 100 користувачів і наступні продукти:

– сервер BioLink;

– BioLink Server Manager;

– BioID Manager;

– BioID CheckPoint;

– BioID Configurator.

BioLink Server Manager, BioID Manager, BioID CheckPoint, що входять у стартовий комплект, призначені для установки на одному комп'ютері.

Поставляються також додаткові ліцензії на BioID Manager, CheckPoint, BioID Configurator.

Ці додаткові ліцензії на клієнтські додатки здобуваються в наступних випадках:

– якщо необхідно розділити уведення даних про користувачів, реєстрацію їхніх наступних відвідувань, діяльність по зміні (доповненню) форм виводу інформації;

– якщо реєстрацію відвідувань необхідно вести одночасно в декількох пунктах (наприклад, на декількох прохідних, у кожній з навчальних аудиторій і т.д.).

Якщо кількість користувачів перевищує 100 чоловік, поставляються додаткові ліцензії на користувачів.

Таким чином, замовникові надається повна воля підбора оптимальної конфігурації BioID – як за складом клієнтських додатків, так і за кількістю користувачів.

У стартовий комплект поставки BioID входить СУБД Microsoft SQL у розповсюджуваній безкоштовно редакції – Microsoft SQL Server 2005 Express Edition. У даній редакції обсяг бази даних обмежений 4 GB ; допускається її використання на однопроцесорному комп'ютері з оперативною пам'яттю до 1 GB. Якщо потреби замовника перевищують зазначені параметри, необхідна додаткова закупівля Microsoft SQL Server інших редакцій.

Апаратні засоби біометричної ідентифікації отримуються по окремому замовленню. Для роботи додатка BioID CheckPoint необхідний сканер відбитків пальців серії BioLink U-Match.

BioTime

Із системою BioTime працювати зручно й легко. Дружній, продуманий інтерфейс можна швидко набудовувати під конкретні завдання, велика номенклатура звітів задовольняє максимум потреб керівників, співробітників кадрових служб і відділів безпеки, бухгалтерів і менеджерів оперативної й тактичної ланки. Інтеграція BioTime у корпоративну інформаційну систему також здійснюється вільно й плавно, і не випадково керівники IT -служб і системних адміністраторів одними з перших підтримують впровадження BioTime.

Система BioTime адаптована під потреби підприємств, що діють у різних масштабах.

Директори компаній малого й середнього бізнесу по достоїнству оцінюють ефективність і економічність BioTime, простоту експлуатації цієї системи, розумні системні вимоги, які виконуються навіть у самому невеликому офісі.

Керівників великих підприємств залучає можливість застосовувати BioTime у мережі територіально розподілених філій і відділень – при тім, що повністю забезпечені централізоване керування системою й можливості її масштабування при успішному розвитку й подальшому росту компанії. BioTime ураховує управлінську ієрархію, дозволяє набудовувати права й повноваження менеджерів, чітко регламентуючи можливості доступу кожного з них до необхідної інформації.

Безупинно нарощується функціонал системи BioTime у частині контролю фізичного доступу. Впроваджувати біометричні технології для рішення цих завдань можна й з «чистого аркуша» (при в'їзді в новий офіс або зміні робочого приміщення), і у вже існуючі системи контролю доступу (доповнюючи, наприклад, наявні карткові зчитувачі ідентифікації за відбитками пальців при доступі в особливо важливі приміщення).

Біометрія визнана експертами самою перспективної й швидко, що розвивається технологією, контролю доступу, що використовують у найбільших банках, державних установах, на підприємствах і виробництвах особливої важливості (аеропорти, вокзали, порти, електростанції, машинобудівна, хімічна, оборонна промисловість, підприємства аерокосмічної галузі й т.д.).

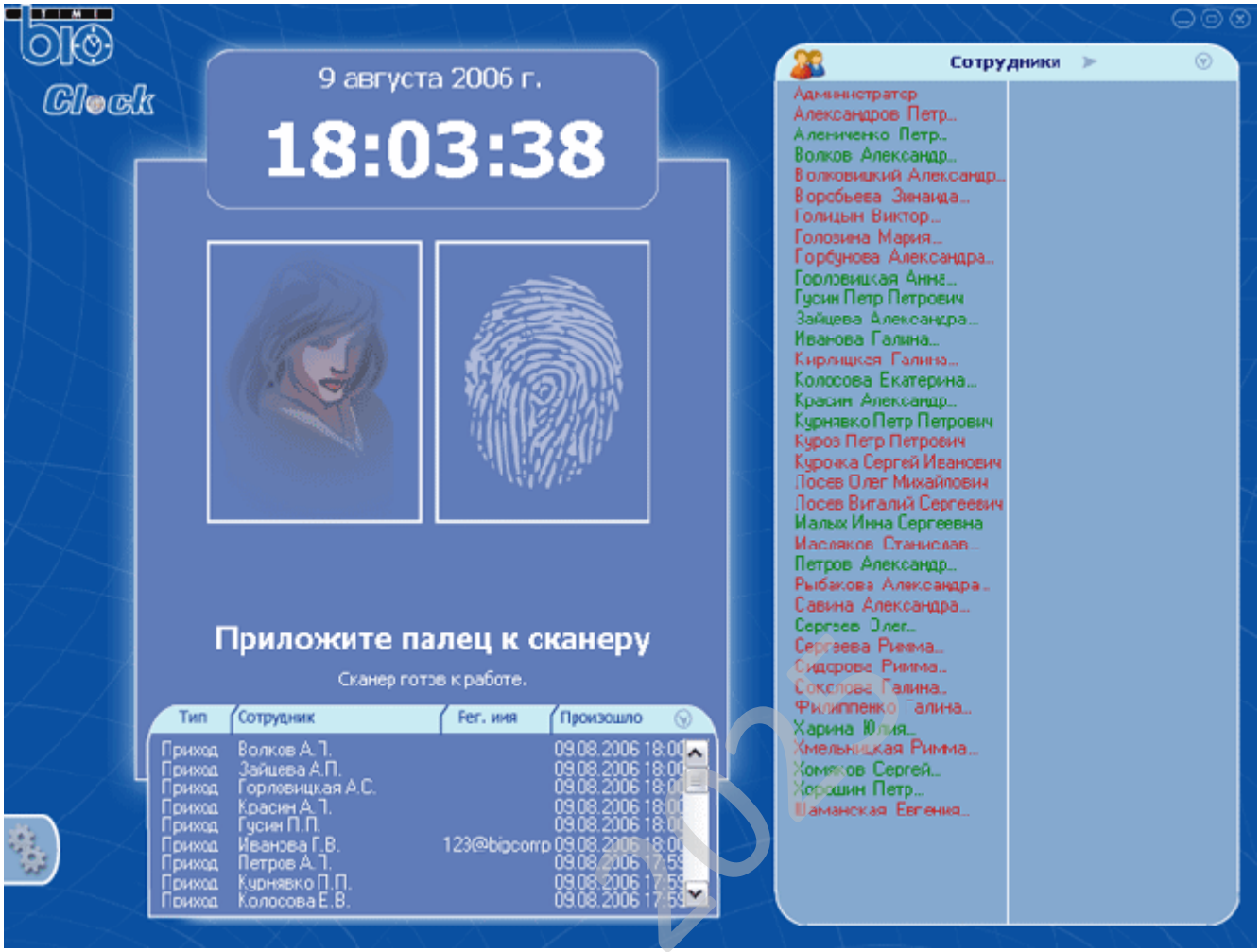


Рисунок 2.1 – Інтерфейс користувача BioTime

BioLink AMIS

Боротьба з тероризмом, злочинністю й нелегальною міграцією, введення паспортно-візових документів нового покоління, забезпечення безпеки суспільства й держави, транспортних і фінансових потоків...

У рішенні цих завдань особливу роль грає впровадження біометричних технологій за допомогою інформаційної системи автоматичної мультибіометричної ідентифікації BioLink AMIS (Automated Multi-biometric Information System).

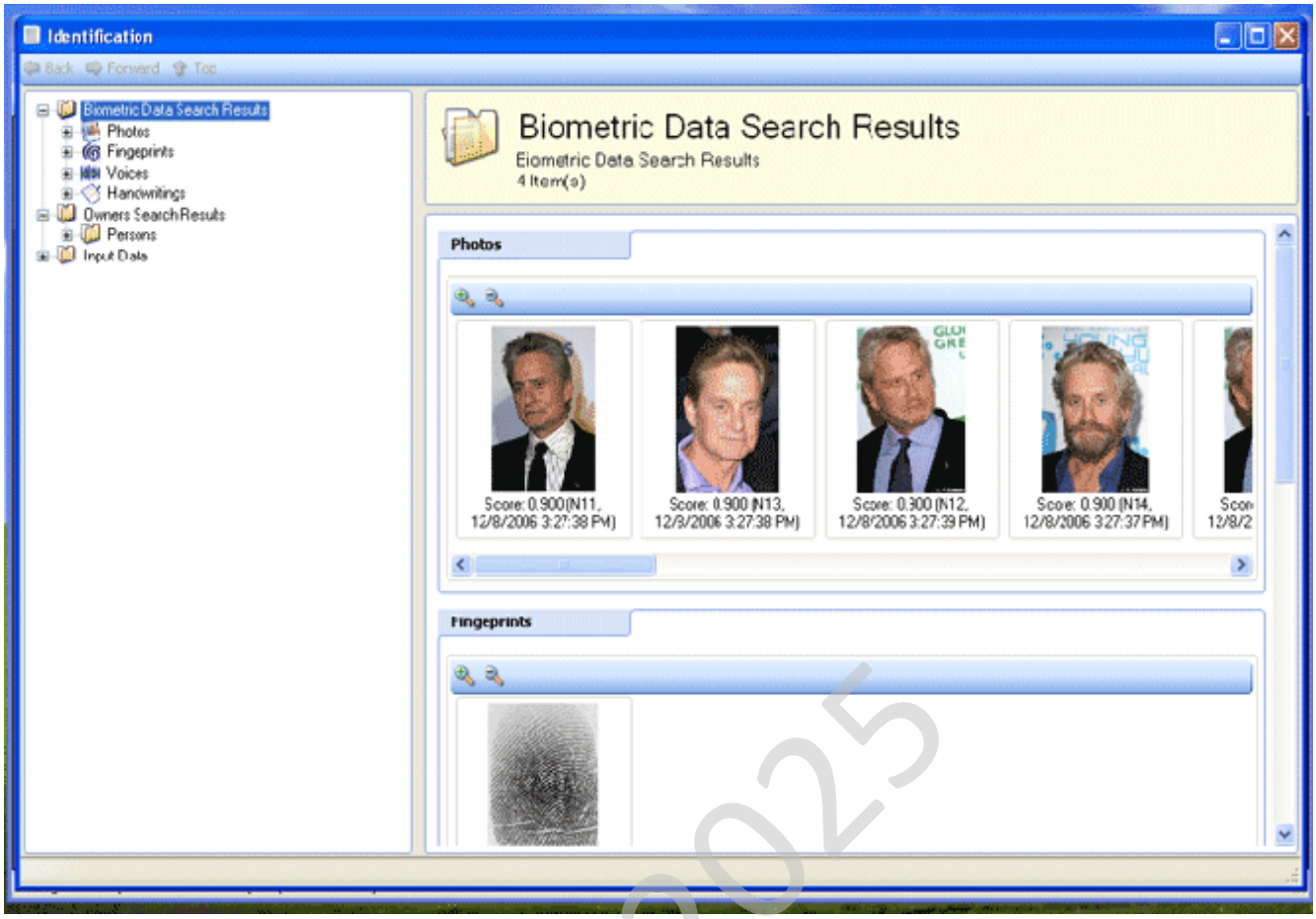


Рисунок 2.2 – Інтерфейс користувача BioLink AMIS

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка додатків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично

весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, ніж звичайні виклики функцій. За рахунок цього значно спрощується розробка додатків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки додатків Windows дозволяють автоматизувати процес створення додатка. Для цього використовуються генератори додатків. Програміст відповідає на питання генератора додатків і визначає властивості додатка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор додатків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої додатки. Подібні засоби автоматизованого створення додатків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики додатка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні додатки, а також додатки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину додатка програмістові прийдеться розробляти самостійно. Вихідний текст додатка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон додатка – це вже половина всієї роботи. Вихідні тексти додатків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти додатків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої додатки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених

для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-додатки на базі об'єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-додатків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

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 для біоідентифікації особи за райдужною оболонкою ока.

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

Безпека комп'ютерних систем, але не тільки, є одним з найважливіших питань сучасної інформатики. Існує багато рішень для захисту даних, а також для ідентифікації та автентифікації користувачів. Їх можна розділити на три категорії:

– щось, що ви знаєте, наприклад, паролі;

– щось у вас є, наприклад жетони;

– щось, що ви є – біометричні особливості.

В даний час також викликає інтерес дво- або багатофакторна автентифікація, тобто комбінація декількох методів автентифікації.

У цьому документі увага зосереджена на біометричних методах розпізнавання людей на основі загальних, унікальних, постійних і вимірюваних фізичних або поведінкових характеристик людини. Унікальність риси означає, що вона повинна бути присутня у всіх людей. Дана риса має бути унікальною в масштабі людської популяції, і не буде інших, хто міг би використати її риси, щоб видати себе за іншу людину. Постійність має гарантувати, що ознака залишається незмінною протягом усього життя людини, незалежно від старіння людини чи хвороб. Вимірність функції має гарантувати, що її можна виміряти за допомогою доступних технологій і методів оцінки.

Серед біометричних ознак, які використовуються для ідентифікації людей, можна назвати відбитки пальців, форму обличчя, сітківку ока та райдужну оболонку. Використання фізичних має недолік, який, незважаючи на складність підробки, може бути фальсифікований за допомогою нових технологій. Це пов’язано з тим, що до них відносно легко отримати доступ. Наприклад, відбитки пальців можуть бути зняті з дверних ручок або окулярів і використані неавторизованою особою для проникнення в захищену систему. Крім того, навіть

за характеристиками голосу сучасні технології дозволяють з високою точністю записати та відтворити голос людини, яка має доступ до ресурсу. Такий запис можна потім застосувати для зламу біометричної безпеки, яка використовує людський голос для ідентифікації. Системи, що використовують сканери райдужної оболонки ока, також можна обдурити фотографією райдужної оболонки ока з високою роздільною здатністю [1].

Ці недоліки зумовлювали пошук особливостей поведінки, які є унікальними для кожної людини, і в той же час, які було б важко скопіювати або повторити. Такі риси, як набір тексту та ходьба, відповідають цим умовам, а також керування мишею та рухами очей. Останні із зазначених ознак були взяті до уваги в цьому дослідженні.

Рух людського ока здебільшого не плавний і ґрунтується на швидких і раптових стрибках – саккадах. Між стрибками відбуваються фіксації, фокусування зору на предметі, під час яких мозок збирає та обробляє інформацію. Фіксації тривають в середньому 200-300 мс, але їх тривалість може коливатися від десятків мілісекунд до декількох секунд. Фокусування на предметі не означає, що око залишається нерухомим. Під час фіксації можна спостерігати легкі рухи очного яблука, які спрямовані на утримання видимого об’єкта в центрі сітківки, запобігання згасанню сцени або стабілізацію зору під час руху голови. Ці мікрорухи включають мікросакади, дрейфи та тремори. Мікросакади – це рухи, що відбуваються з типовою частотою 1–2 Гц з невеликою амплітудою (менше 0.5°), висока швидкість порівняно з дрейфами або підземними поштовхами (15–50°/с) і тривалістю 10–30 мс. Дрифти – це повільні рухи (з амплітудою менше 0.13°) зі швидкістю приблизно рівною 0.5°/с, що виконується на частоті нижче 40 Гц. Останній з типів фіксаційних рухів – мікротремор малої амплітуди (прибл. 0.00017°) і частотний діапазон 40–100 Гц, досягаючи максимальної швидкості 20°/с [2, 3]. Через наявність цих трьох рухів очі ніколи не бувають стабільними, а дві додаткові величини описують фіксацію: дисперсія та швидкість ока.

Саккади – це швидкі рухи очей, спрямовані на встановлення ока на наступну точку фіксації. Під час саккад інформація не збирається й не обробляється. Саккаду можна описати наступними параметрами: амплітуда, швидкість і тривалість. Амплітуда саккад коливається від 4 до20°, а швидкість досягає 300–500°/с. Тому вони вважаються найшвидшими рухами, які може зробити людське тіло. Тривалість саккад найчастіше становить 30–80 мс.

Деякі дослідження, присвячені застосуванню рухів очей як ознак біометричної ідентифікації, були проведені раніше. Одне з перших досліджень у цій галузі описано в роботі [4]. У цьому рішенні використовується парадигма «точки стрибка». На екрані протягом 8 с відображалися 9 точок у вигляді матриці 3×3. В експерименті брали участь дев'ять учасників, для яких було записано 30 випробувань для класифікації, що дало 270 сеансів у наборі даних. Перші 15 кепстральних коефіцієнтів були витягнуті з горизонтальних і вертикальних сигналів обох очей. У результаті був отриманий 60-вимірний вектор для кожного випробування. Алгоритм k -найближчих сусідів (k NN) для $k = 3$ і $k = 7$, наївний Байєс, дерево рішень і машина опорних векторів (SVM) використовувалися як класифікатори. Класифікацію проводили за допомогою 10-кратної стратифікованої перехресної перевірки. Найкращі результати було виявлено для алгоритму k NN для $k = 3$. Він досягав середнього FAR (частка помилкових прийомів) 1,48% і FRR (частка помилкових відхилень) 22,59%.

Подальші дослідження з біометричної ідентифікації описані в роботі [5]. В експерименті взяли участь 12 осіб, завданням яких було спостерігати на екрані статичний хрест. Для кожного учасника було отримано приблизно 47 рядків із частотою дискретизації 50 Гц. Такі характеристики, як швидкість ока або різниця в діаметрі зіниці ока, були розраховані на основі зібраних даних. Комбіновані ознаки та кожен компонент окремо перевіряли як вектор ознак. Для класифікації було обрано k NN алгоритм. Найкращі результати, які досягли точності класифікації 90%, стосувалися статичної ознаки – відстані між очима. Для динаміки ока була отримана точність на рівні 50%–60% для різниці діаметра

зіниці ока.

Протягом 2012–2015 років було проведено три конкурси з перевірки та ідентифікації рухів очей. Перший [6] був проведений у 2012 р. Було використано чотири типи наборів даних: два некаліброваних і два каліброваних. Стимул був представлений у вигляді стрибаючої точки з використанням різних макетів. За винятком необробленого позиційного сигналу ока, розглядалися лише ознаки, пов'язані з першою та другою похідною (тобто швидкість і прискорення, відповідно), а *k* NN, SVM та байєсовська мережа використовувалися як класифікатори. Результати експерименту показали, що певний унікальний шум, присутній у некаліброваних наборах даних, може позитивно вплинути на точність класифікації. Іншим фактором, що впливає на результати, є час розділення сеансів запису. Чим менший часовий інтервал, тим кращі результати. Другий конкурс [7], організований у 2014 році, надав дані, зібрані протягом двох сесій на основі спостереження за обличчями. Результати, отримані учасниками, показали, що не було кореляції між рівнем розпізнавання та довжиною вибірки. Те саме стосується знайомства обличчя та самих зображень. Однак результати цього конкурсу підтвердили, що інтервал часу між записами зразків мав значний вплив на точність класифікації. Третій конкурс [8], який проходив у 2015 році, надав чотири різні набори даних, щоб дозволити перевірити різні параметри: різні візуальні стимули (випадкові крапки та текст) і різні часові інтервали між записами (три сеанси – в середньому 30 хвилин і 12 місяців). Записуючий пристрій, який використовувався для фіксації рухів очей, працював із частотою дискретизації, що дорівнює 1000 Гц. Учасники змагань в основному витягували статистичні характеристики з профілів фіксації та саккади: положення, напрямок, швидкість та прискорення. Серед методів, використаних для класифікації, були нейронні мережі, SVM і *k* NN. Результати конкурсу також показали, що старіння шаблону мало більш значний вплив на точність розпізнавання, ніж тип візуального стимулу.

У [9] автори провели біометричне дослідження на основі руху очей у три

сеанси з інтервалом у 30 хвилин та один рік відповідно. Дані про рухи очей з експерименту були розділені на фіксації та саккади, а їх статистичні ознаки були використані для цілей класифікації. Серед них були профілі положення, швидкості та прискорення, а також тривалість, дисперсія, довжина шляху та спільність. Набір даних було зареєстровано з частотою дискретизації 1000 Гц, а потім зменшено до 250 Гц. Два типи стимулів: випадкова точка, що з'являється на екрані, і текст, використовувалися в експерименті за участю 153 суб'єктів для першого і другого сеансу і 37 – для третього. У процесі класифікації дві різні мережі Гаусса RBF навчалися окремо, і оцінки, отримані від обох мереж, використовувалися для ідентифікації суб’єкта. Продуктивність становила приблизно від 80% до 90%, залежно від використовуваного набору даних.

Існують також дослідження [10], в яких автори досліджували використання саккад для верифікації користувача серед інших суб’єктів. В якості стимулу використовувався jumping dot, для чого було отримано 30 великих горизонтальних амплітудних саккад для 109 суб'єктів. Із сигналів було виділено вісім характеристик: амплітуда, точність, затримка та максимальна кутова швидкість, час до досягнення максимальної швидкості, максимальне кутове прискорення та максимальне уповільнення. Було зібрано чотири сигнали від кожного суб'єкта. Одного з них взяли на тестування, а трьох інших на навчання. Згодом для класифікації використовувалися MLP і мережі радіальної базисної функції, опорні векторні машини та логістичний дискримінантний аналіз. Залежно від використовуваного методу вони отримали точність приблизно від 63% до 85%.

Інші дослідження на цю тему описані в [11]. Автори збирали активність очей під час читання тексту. В експерименті взяли участь 40 учасників, розділених на дві групи по 20 осіб. Учасникам першої групи були представлені витяги зі статей Washington Post News, кожна з яких містила шість текстів, різні для кожної людини. Решта 20 осіб зачитували шість уривків із паперів, які були однаковими для кожного учасника. Дані, отримані від айтрекера, записувалися з

частотою дискретизації 1000 Гц. Виділені ознаки були розділені на кілька типів: ознаки фіксації (наприклад, загальна тривалість і частота фіксацій), саккадні (наприклад, середня тривалість саккад, середня швидкість саккад, середня горизонтальна амплітуда саккад), особливості реакції зіниці при читанні статті, частота зміни діаметра зіниці. Моделі класифікації були побудовані з використанням таких класифікаторів, як нейронна мережа (багатошаровий перцептрон), випадковий ліс і LMT (дерево логістичної моделі). Їх вивчали та перевіряли за допомогою 10-кратної перехресної перевірки. Отримана точність, сукупність співпадаючих класів з різних класифікаторів, склала 95,31%, а EER (Equal Error Rate) на рівні 2,03%. Повний огляд пов'язаних робіт із біометричної ідентифікації на основі руху очей можна знайти в [1].

Вищеописані дослідження дали багатообіцяючі результати, які, однак, потребують подальшого вдосконалення з точки зору ефективності ідентифікації. Крім того, у деяких дослідженнях набори даних були зібрані під час одноденного експерименту [10, 11, 12], що могло полегшити розпізнавання суб'єктів. В інших випадках дані реєструвалися з тижневими чи однорічними інтервалами, але були отримані незадовільні результати [6, 7, 8], або використовувалися складні налаштування класифікації та стимули [9, 11]. Це було мотивуючим фактором для дослідження нових можливостей у цій галузі.

Внесок дослідження, представленого в цій статті, полягає в представленні нового підходу для визначення векторів ознак. Запропоноване рішення поєднує раніше використовувані величини, такі як швидкість руху очей і прискорення, з характеристиками, оціненими за допомогою аналізу нелінійної динамічної системи. Наскільки відомо авторам, таке рішення раніше не застосовувалося в цій галузі.

Матеріали та методи

Представлені на даний момент дослідження були зосереджені на розробці нового підходу для перевірки особистості користувачів на основі сигналу руху очей. Завдання було реалізовано в кілька етапів. Перший з них полягав у

визначенні векторів ознак, які використовуються для опису зареєстрованих зразків руху очей для кожного учасника. Згодом деякі вектори, отримані таким чином, були застосовані для навчання класифікатора вивчати поведінку рухів очей суб'єктів. Частина векторів, що залишилася, була використана для перевірки ефективності обраних методів, що означає належне розпізнавання зразків як належних чи ні ідентифікованого користувача. Детальний опис кожного етапу дослідження було представлено в наступних розділах.

Набір даних

Дослідження проводилося з використанням даних, зібраних під час експерименту за участю 24 суб'єктів. Їхнє завдання полягало в тому, щоб стежити очима за точкою стрибка, яка з’являлася в 29 положеннях на екрані. 9 та 29 позиції мали однакові координати. Розташування стимулів було розроблено таким чином, щоб забезпечити рівномірне покриття екрану та отримати різну довжину саккадичних рухів. Стимулом було темне коло розміром0.5×0.5°пульсуючий – коливався у розмірі – на екрані, щоб привернути увагу учасників. Одночасно точка була видна лише в одному місці і відображалася протягом 3 с. Перерв між наступними презентаціями пункту не було. Після зникнення в одному місці, воно відразу проявилося в наступному.

Схема розташування точок стрибка

Експеримент складався з двох сеансів, під час яких точка відображалася в однаковому, попередньо визначеному порядку. Сеанси проводилися з інтервалом у два місяці, щоб уникнути ефекту навчання, який виникає, коли учасник може передбачити наступну позицію балів. Обидва сеанси проводилися в однаковому середовищі: в одній кімнаті, в той самий день тижня та в той самий час доби. Під час обох сесій всі учасники мали однакові умови. Кімната була обладнана штучним освітленням, що забезпечувало відповідний фон для реєстрації рухів очей. Перед кожним експериментом учасникам повідомляли про загальну мету тестів, після чого вони підписували форму згоди. У результаті було отримано 48 файлів, по два на кожного учасника.

Дані були зібрані за допомогою айтрекера Ober Consulting Jazz-Novo, який записує положення очей із частотою дискретизації 1000 Гц [13]. В експериментах використовувався плоский екран 1280 × 1024 (370 мм × 295 мм). Відстань око-екран дорівнювала 500 мм, вертикальний і горизонтальний кути огляду були рівними40°і32°відповідно. Голову стабілізували за допомогою підставки для підборіддя.

3.2 Розробка структурної схеми

Технологія ідентифікації райдужної оболонки ока розроблена для розробників та інтеграторів біометричних систем. Ця технологія включає багато власних рішень, які забезпечують надійну реєстрацію райдужної оболонки за різних умов і швидке зіставлення райдужної оболонки в режимах 1-до-1 і 1-до-багатьох.

Доступний як набір для розробки програмного забезпечення, який дозволяє розробляти автономні та мережеві рішення на платформах Microsoft Windows, Linux, macOS, iOS та Android.

Особливості та можливості

– Швидка та точна ідентифікація райдужної оболонки, перевірена NIST IREX.

– Надійне розпізнавання навіть із відведеними очима або звуженими повіками.

– Оригінальний запатентований алгоритм усуває обмеження та недоліки існуючих найсучасніших алгоритмів.

– Виявлення живості райдужної оболонки може запобігти підробці зображень райдужної оболонки на фотографіях або контактних лінзах.

– Доступний як багатоплатформний SDK, який підтримує кілька мов програмування.

– Розумні ціни, гнучке ліцензування та безкоштовна підтримка клієнтів.

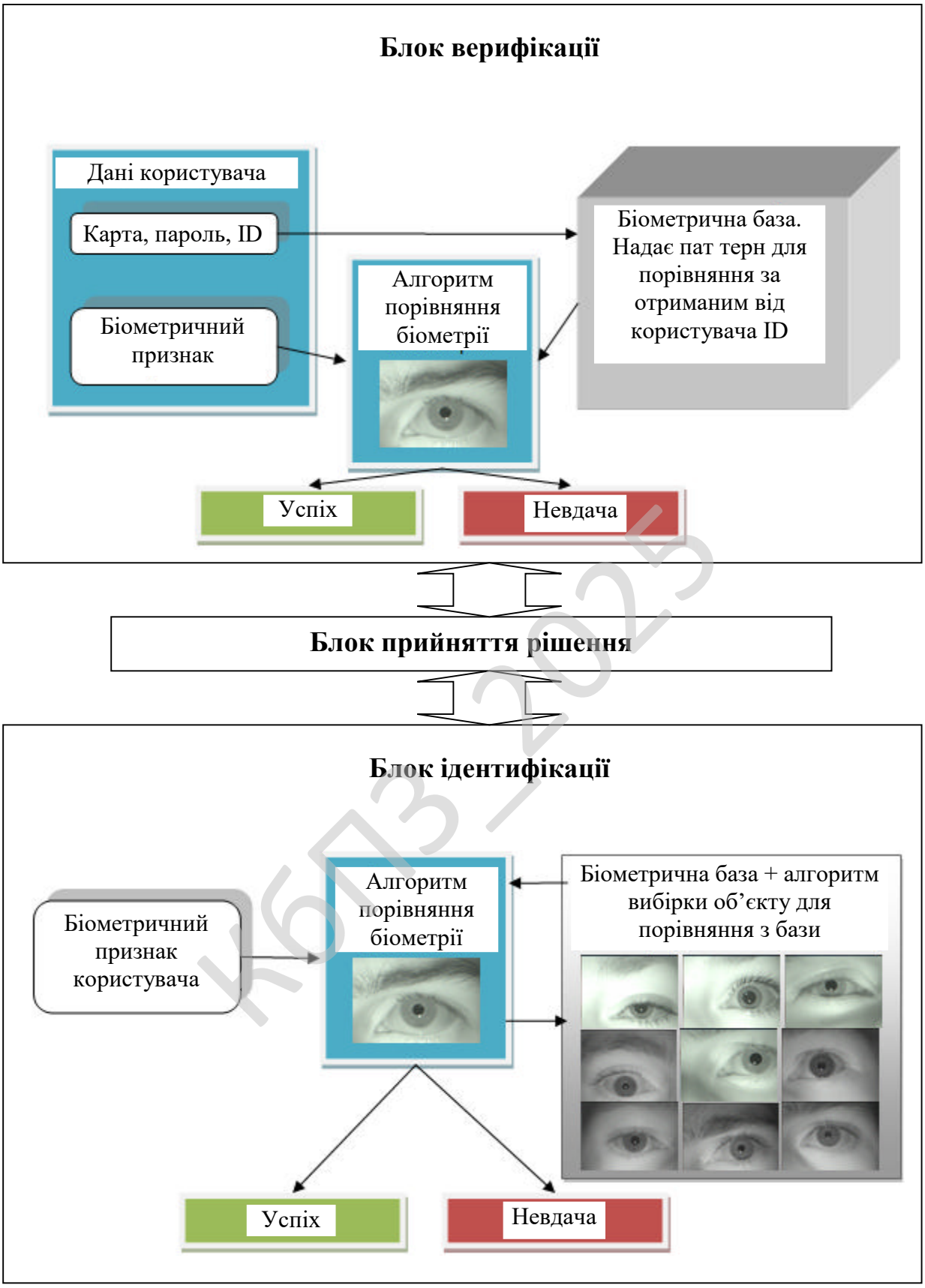


Рисунок 3.1 – Структурна схема системи

Алгоритм усуває обмеження та недоліки існуючих найсучасніших алгоритмів. Реалізується розширену сегментацію райдужної оболонки, реєстрацію та зіставлення за допомогою надійних алгоритмів обробки цифрових зображень:

– Надійне виявлення райдужної оболонки. Райдужка виявляється навіть за наявності перешкод для зображення, візуального шуму та/або різного рівня освітлення. Усуваються відблиски світла, засмічення повік і вій. Також приймаються зображення зі звуженими повіками або очима, які дивляться вбік.

– Автоматичне виявлення та корекція чергування забезпечує максимальну якість шаблонів рис райдужної оболонки із рухомих зображень райдужної оболонки.

– Очі, що дивляться вбік, правильно розпізнаються на зображеннях, сегментуються та трансформуються так, ніби вони дивляться прямо в камеру

– Правильна сегментація райдужки досягається навіть за таких умов:

- Ідеальних кіл не виходить. VeriEye використовує моделі активної форми, які точніше моделюють контури ока, оскільки межі райдужної оболонки не моделюються ідеальними колами.
- Центри внутрішніх і зовнішніх меж райдужної оболонки різні. Внутрішня межа райдужної оболонки та її центр позначені червоним кольором, зовнішня межа та її центр – зеленим кольором.
- Межі райдужної оболонки – це точно не кола і навіть не еліпси, особливо на зображеннях райдужної оболонки, що дивляться вбік.
- Межі райдужки здаються ідеальними колами. Якість розпізнавання ще можна покращити, якщо визначити межі точніше. Зверніть увагу на ці невеликі недоліки порівняно з ідеальними круглими білими контурами.
- Райдужка частково закрита повіками. Верхня і нижня повіки позначені червоним і зеленим відповідно.

– Визначення якості зображення райдужної оболонки. Оцінку якості зображення можна використовувати під час реєстрації райдужної оболонки, щоб

переконатися, що лише шаблон райдужної оболонки найкращої якості зберігатиметься в базі даних. Кут повороту можна визначити за зображенням райдужної оболонки для прийняття подальших рішень щодо прийняття зображення для реєстрації. Також до зарахування можуть бути відхилені райдужні оболонки, закриті косметичними (декоративними) контактними лінзами з художніми зображеннями або зміною кольору.

– Виявлення живості. Захоплену райдужну оболонку можна проаналізувати, чи є вона «живою» чи підробкою, щоб запобігти порушенню безпеки, якщо розмістити фотографію перед камерою або надіти контактні лінзи з підробленою текстурою райдужної оболонки.

– Автоматичне визначення положення діафрагми. Алгоритм здатний розділяти зображення лівої та правої райдужки.

– Надійність. Алгоритм показав чудову точність розпізнавання під час оцінок NIST IREX.

3.3 Розробка функціональної схеми

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

Функціональна схема – це схема, що описує саму суть роботи пристрою, програми, взаємодії й має більш узагальнений рівень, ніж структурна.

У функціональній схемі розглянута загальна взаємодія між клієнтом і сервером (схему варто переглядати знизу догори).

На стороні клієнта формується модель, що складається з біометричного ідентифікаційного запису, криптографічного додавання й відправляється на сервер IIS – це Інформаційний Інтернет-сервер компанії Microsoft; веб-сервер, що запускається в операційних системах Windows. Там відбувається перевірка криптографічного додавання за допомогою модуля COM. При проходженні

перевірки відбувається добування з бази даних Microsoft Access еталонного паспорта й порівняння моделі з паспортом.

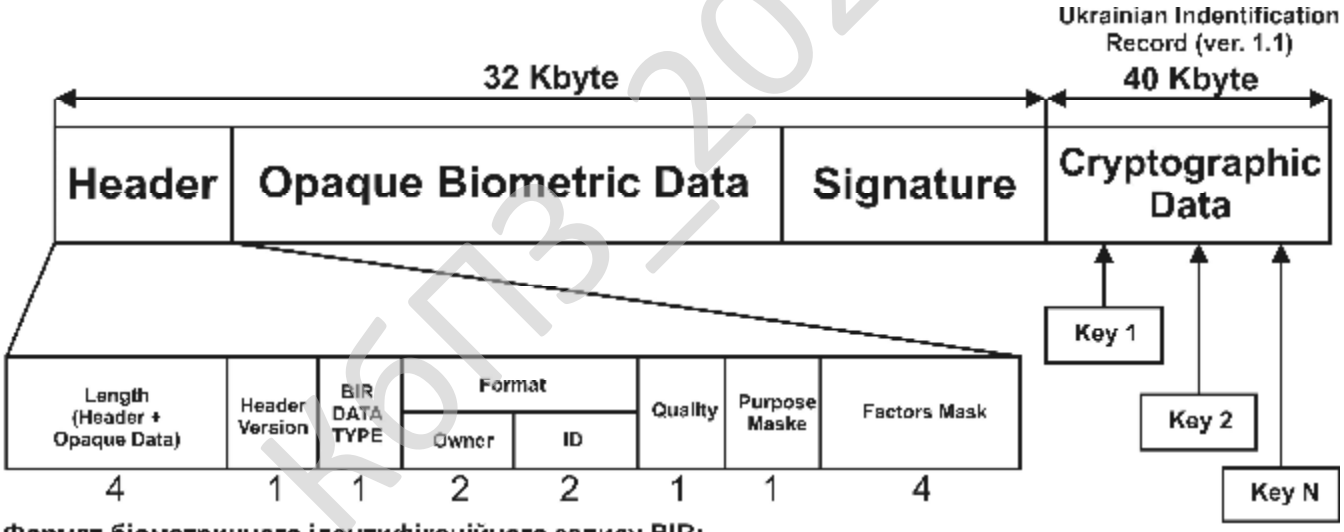
На даній схемі можна чітко собі представити загальну функціональну взаємодію й можливості розробленої системи.

Рисунок 3.2 – Функціональна схема системи

Розглянемо формат передачі даних, які складаються з біометричного ідентифікаційного запису й українського криптографічного додавання, зображених на рисунку 3.3.

Основним терміном інтерфейсу BioAPI є біометричний ідентифікаційний запис – BIR (Biometric Identification Record), яку можна визначити як набір біометричних даних, з яким працюють додатки й провайдери послуг. У загальному BIR – це єдиний формат, пропонований для заміни й об'єднання самих різних форматів подання біометричних даних устаткуванням і програмним забезпеченням різних виробників.

Формат BIR, його структура й состав, затверджені Національним інститутом стандартів і технології США NIST, України й інших країн, а також затверджено Common Biometric Exchange File Format (CBEFF, узагальнений формат обміну біометричними даними). BIR складається із трьох секцій даних: заголовка, біометричних даних і електронного цифрового підпису. Далі розглянемо пояснення до полів BIR по специфікації BioAPI 1.1.



Формат біометричного ідентифікаційного запису BIR:

Length - довжина заголовка й біометричних даних;
Header Version - версія заголовка BIR;
BIR Data Type - тип даних BIR;
Format (Owner/ID) - опис формату біометричних даних;
Quality - якість (необхідно для деяких типів біометричних даних);
Purpose Mask - ціль: верифікація, ідентифікатор по імені користувача, ідентифікація, реєстрація для ідентифікації, аудит)
Factors Mask - використовуваний метод біометрії;
Opaque Biometric Data - біометричні дані;
Signature - поле електронно-цифрового підпису;
Record Data Type - тип біометричних даних;
Creation Date - дата створення BIR, час і день одержання біометричних даних;
Creator - ідентифікатор творця BIR.

Рисунок 3.3 – Структурна схема формату передачі даних

Сектор Заголовок BIR (Header) містить у собі наступні поля:

– Length – довжина, сума розміру заголовка й біометричних даних;

– Header Version – версія заголовка BIR;

– BIR Data Type – тип даних BIR. Насправді по стандарті CBEFF формування даного заголовка це поле носить інша назва, більш точно його визначальне: Опції захисту (Security options). У цьому полі вказується тип захисту біометричних даних (1 з 4 значень): тільки біометрія (немає захисту), криптографія, ЕЦП, криптографія й ЕЦП;

– Format (Owner/ID) – опис формату біометричних даних, представлених після заголовка BIR. Всі формати біометричних даних, підтримуючих BioAPI, реєструються в Міжнародній промисловій біометричній асоціації IBIA (International Biometric Industry Association).

Поле складається із двох підполей:

– Власник формату, тобто компанія, що зареєструвала даний формат подання біометричних даних, формат може бути декількох типів. У Додатку наведений список всіх зареєстрованих форматів подання біометричних даних.

– Ідентифікатор формату – номер використовуваного формату із зареєстрованих власником.

– Quality – якість, для деяких типів біометричних даних необхідно вказувати дане значення. Якість вказується в проміжку від 0 до 100. Якщо зазначено «-1» то якість не задана, якщо «-2», те в даному біометричному методі таке поняття як «якість» не використовується.

– Purpose Mask – ціль, з якої будуть використані біометричні дані. У даному полі вказується одне із шести значень:

–верифікація (порівняння «один до одного», користувач називає своє ім'я й пред'являє ідентифікатор, по імені користувача шукається шаблон його ідентифікатора й вони рівняються);

–ідентифікація (порівняння «один до багатьох», користувач пред'являє свій ідентифікатор, і в базі шаблонів шукається найбільш близький до нього);

–реєстрація;

–реєстрація тільки для верифікації;

–реєстрація тільки для ідентифікації;

–аудит.

– Factors Mask – використовуваний метод біометрії. У цей час зареєстровано 19 методів розпізнавання (можливе розширення даного списку при твердженні наступної версії формату):

– комбінація методів;

– форма особи;

– голос;

– відбиток пальця;

– сітківка ока;

– райдужна оболонка;

– геометрія кисті руки;

– динаміка підпису;

– динаміка клавіатурного набору;

– рух губ;

– термографія особи;

– термографія руки;

– хода;

– запах тіла;

– ДНК;

– форма вуха;

– геометрія пальця;

– геометрія долоні;

– малюнок вен на руці.

Сектор Біометричні дані (Opaque Biometric Data) – містить безпосередньо біометричні дані, використовувані для розпізнавання людини, відповідно до зареєстрованого формату, зазначеним у заголовку BIR у поле Format.

ЕЦП (Signature) – поле, у яке заноситься електронний цифровий підпис. Поле є опціональним. Якщо ЕЦП використовується, то підписуються разом і заголовок BIR, і біометричні дані.

Крім цього в заголовку BIR існує також декілька опціональних полів, актуальних не для всіх біометричних методів розпізнавання:

– Record Data Type – тип біометричних даних, що втримуються. Передані в складі BIR біометричні дані можуть бути трьох типів: неопрацьовані, преоброблені, оброблені;

– Creation Date – дата створення BIR, час і день одержання біометричних даних;

– Creator – ідентифікатор творця BIR.

В інтерфейсі BioAPI визначені два рівні:

– верхній, визначальний інтерфейс клієнтського й серверного додатків, що викликає функції автентифікації;

– нижній, визначальний інтерфейс взаємодії із провайдером біометричних послуг (Biometric Service Provider), що виконують виклики верхнього рівня.

Верхній рівень (у версії 1.0 має назву «H», high) визначає три основних функції, необхідних додатку для проведення біометричному автентифікації:

– Enroll (реєстрація). Вимір з біометричного пристрою, що зчитує, обробляються в придатну для використання форму, з якої формується шаблон, що повертається додатку;

– Verify (верифікація, порівняння «один до одного»). Одна або більша кількість вимірів знімається з біометричного пристрою, обробляється в придатну для використання форму й потім рівняється з відповідним шаблоном. Результати порівняння вертаються додатку;

– Identify (ідентифікації, порівняння «один до багатьох»). Одна або більша кількість вимірів знімається з біометричного пристрою, обробляється в придатну для використання форму й рівняється з набором шаблонів. Як результат

вертається список, що показує, наскільки близько виміру збігаються з найближчими кандидатами на ідентифікацію з набору шаблонів.

Нижній рівень, SPI (service provider interface) – визначає інтерфейс до провайдеру біометричних послуг (BSP – Biometric Service Provider), у якості якого можуть виступати практично будь-які підтримуючі цей інтерфейс біометричні системи, пристрої або програмні продукти . Функцією SPI є відображення «один до одного» викликів верхнього рівня у виклики до BSP.

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. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).
З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи біоідентифікації особи за райдужною оболонкою ока.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких 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.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм.

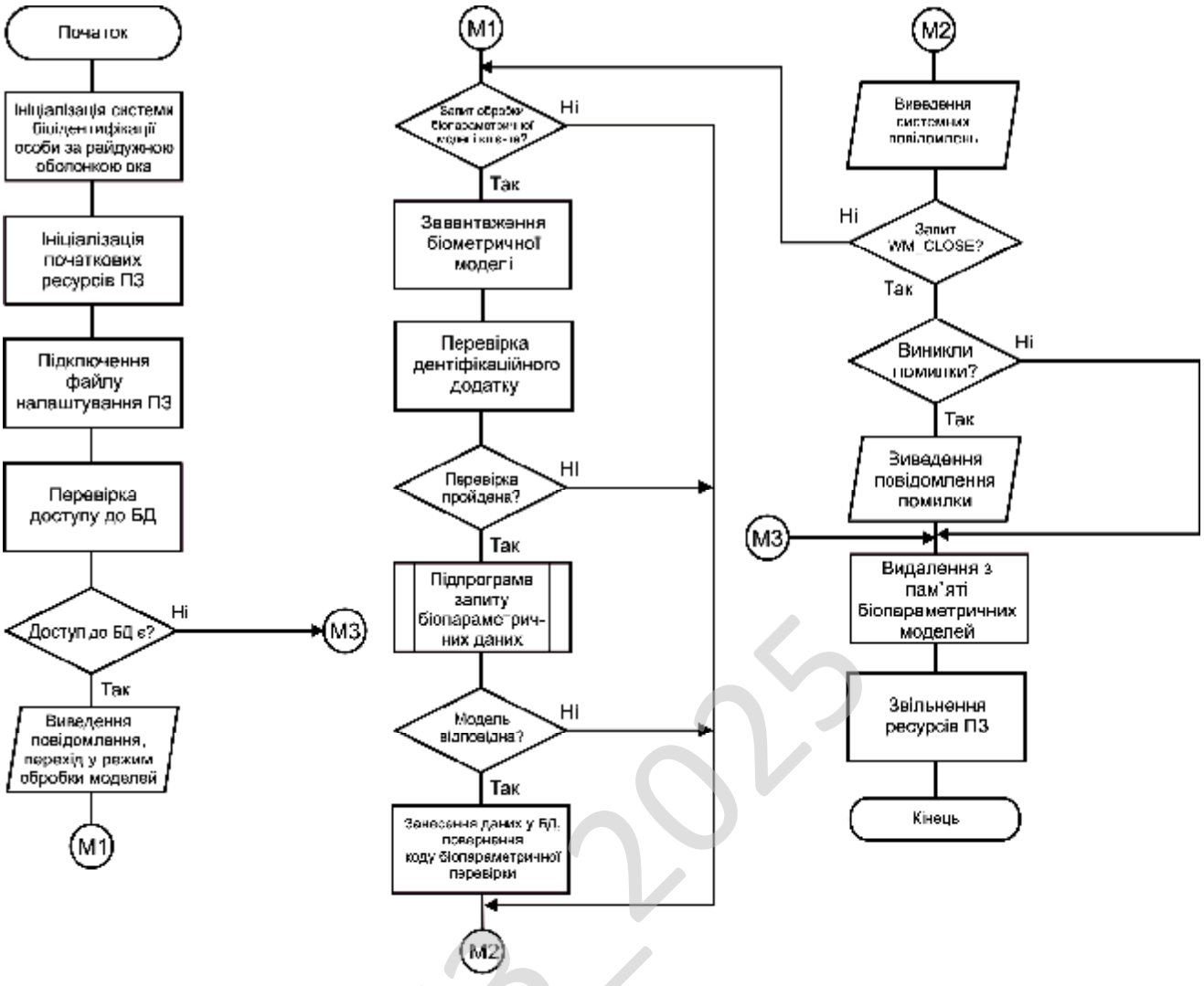


Рисунок 4.1 – Блок-схема основної програми

Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

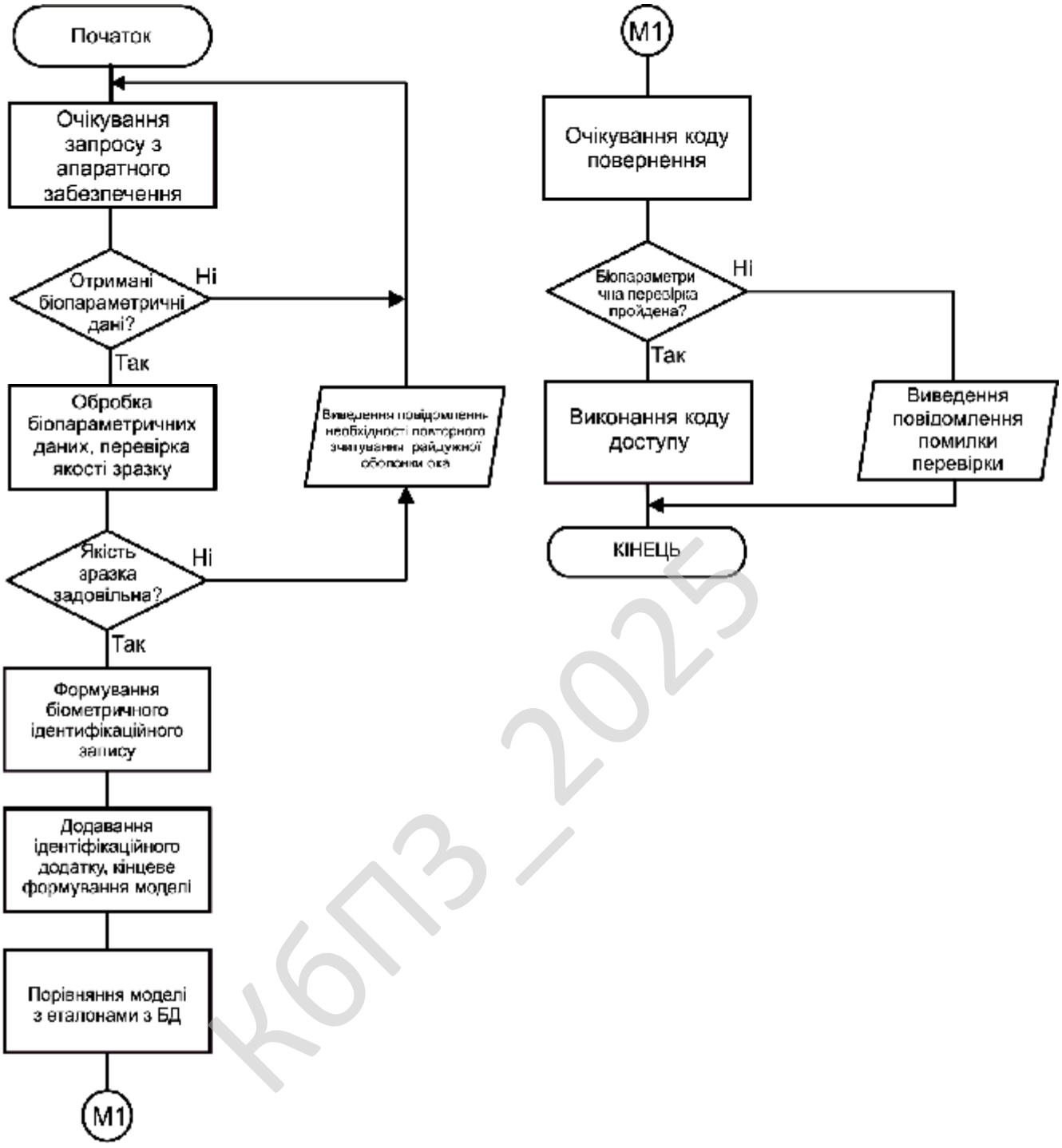


Рисунок 4.2 – Блок-схема роботи підпрограми

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми

автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання

діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на

одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази.Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить

посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з’являться квантові комп’ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення системи біоідентифікації особи за райдужною оболонкою ока складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Налаштування; Авторське право.

– Функції представлені у графічному вигляді.

– Підрозділу виведення даних з ВЕБ камери.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ: Оновлення та налаштування чутливості.

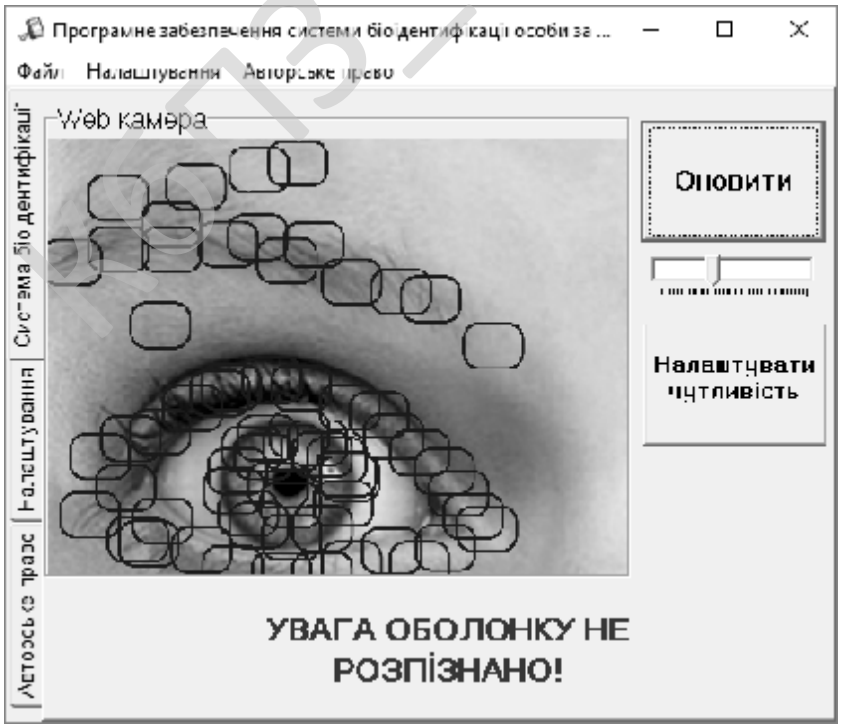


Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

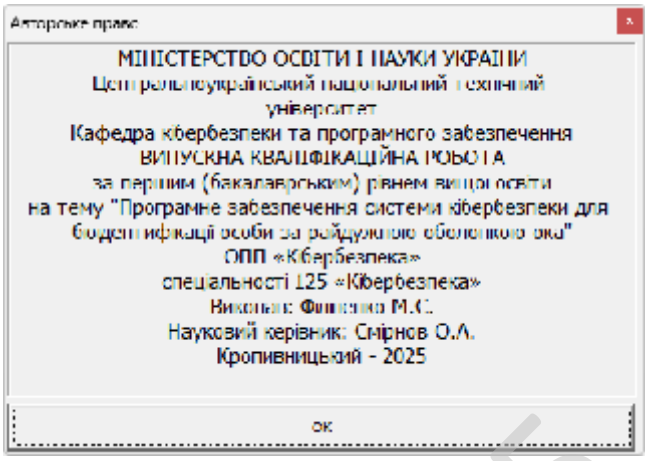


Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись. Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат ПЗ

тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення. Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок. Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

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). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для біоідентифікації особи за райдужною оболонкою ока.

– Досліджена система для біоідентифікації особи за райдужною оболонкою ока.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для біоідентифікації особи за райдужною оболонкою ока.

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 Visual C++, HTML, ASP, Jscript. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для біоідентифікації особи за райдужною

оболонкою ока. 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 ДСТУ 9041:2020.

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

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

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

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

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

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

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

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

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

10. Kuznetsov, O., Smirnov, O., Mormul, M., Kotukh, Y., Zvieriev, V. «Comparative Research on Cryptocurrency Efficiency: An Objective Analysis of Key Metrics». *International Journal of Computing* 23(4), 2024. pp. 563-573.

11. Kuznetsov O., Frontoni E., Kuznetsova K., Smirnov O., Kostenko V. «Blockchain applications in metaverse environments: new horizons». *Advanced Metaverse Wireless Communication Systems.* pp. 255-293. 2024.

12. Kuznetsov, O., Frontoni, E., Chevardin, V., Smirnov, O., Imoize, A.L. «Advancing metaverse security with cryptographic innovations». *Advanced Metaverse Wireless Communication Systems.* pp 351-386. 2024.

13. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

14. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

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

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

17. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

18. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

19. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

20. 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.

21. 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.

22. 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.

23. 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,

24. 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.

25. 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.

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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока.

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. № 51-02 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 для біоідентифікації особи за райдужною оболонкою ока.

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 для біоідентифікації особи за райдужною оболонкою ока;

– 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 Visual C++, HTML, ASP, Jscript.

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 – 62 аркуші.

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 6.06.2025 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 О.А.

Програмне забезпечення системи кібербезпеки для біоідентифікації особи за райдужною оболонкою ока

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2025 року

Для правильної роботи системи необхідна наявність наступних бібліотек Windows:

- **btWebRecognitionServer.dll** - COM-модуль, що реалізує інтерфейс до математичних бібліотек
- **BIOHFW32.DLL, BTBSPCln.DLL, BTBSPCTL.DLL, BTBSPSRV.DLL, DSPSTM32.DLL** бібліотеки математичної підтримки, що входять до складу BioWebSDK.

Ім'я серверної частини (Com модуля)- **bioSERVERAutoidentificathion**

Ім'я клієнтської частини (ActiveX)- **bioClient**

Серверна частина файл BioIDENT.cpp

```
// Copyright (C) 2025, Філіпенко Микита Сергійович, КБ-22-МБ  All Rights
Reserved.
//
// Name: BioIDENT.cpp
//
// Description:
//    Magister work. Bioparameters Identificathion

#include "precomp.h"
extern BOOLEAN   gbInitialized;
extern PFWPasport gpFWPasport;
NTSTATUS SERVACTIVEXSKFilter(SKPacketBioIDENTPtr pFilterFunction, BOOLEAN Add)
{
PDEVICE_OBJECT pDeviceObject = NULL;
PFILE_OBJECT   pFileObject   = NULL;
SK_SET_BioIDENT_SERVACTIVEX_INFO SERVACTIVEXInfo;
UNICODE_STRING FilterName;
KEVENT NotificationHandle;
IO_STATUS_BLOCK ioStatusBlock;
PIRP Irp;
RtlInitUnicodeString(&FilterName, DD_SK_DEVICE_NAME);
NTStatus=IoGetDeviceObjectPointer(&FilterName,STANDARD_RIGHTS_ALL,&pFileObject
, &pDeviceObject);
if(NT_SUCCESS(NTStatus))
{
SERVACTIVEXInfo.BioIDENTPtr = pFilterFunction;
SERVACTIVEXInfo.Priority = 2;
SERVACTIVEXInfo.Add = Add;
KeInitializeEvent(&NotificationHandle, NotificationEvent, FALSE);
Irp=IoBuildDeviceIoControlRequest(IOCTL_SK_SET_BioIDENT_SERVACTIVEX,pDeviceObj
ect,(PVOID) &SERVACTIVEXInfo, sizeof(SK_SET_BioIDENT_SERVACTIVEX_INFO),
NULL,0,FALSE, &NotificationHandle,&ioStatusBlock);

if(Irp != NULL)
{
NTStatus = IoCallDriver(pDeviceObject, Irp);

if (NTStatus == STATUS_PENDING)
{
IOStatus = KeWaitForSingleObject(&NotificationHandle, Executive, KernelMode,
FALSE, NULL);
}
NTStatus = ioStatusBlock.Status;
```

```
}
else
{
NTStatus = STATUS_INSUFFICIENT_RESOURCES;
}
if(pFileObject != NULL) ObDereferenceObject(pFileObject);
pFileObject = NULL;
pDeviceObject = NULL;
}
return NTStatus;
}
FORWARD_ACTION FWEngine(VOID **pData, UINT RecvInterfaceIndex,
UINT *pSendInterfaceIndex, UCHAR     *pDestinationType,
VOID *pContext, UINT ContextLength,struct SKRcvBuf **pRcvBuf)
{
PBioIDENT_CONTEXT_T pFWContext = NULL;
FORWARD_ACTION FWAction = FORWARD;
PPacket pPacket;
PSKHeader pSKHeader = NULL;
struct SKRcvBuf* pSKRecvBuffer = NULL;

TDirection Direction;

pPacket = (PPacket)FWAlloc(sizeof(TPacket));

if(pPacket == NULL) return FORWARD;
do
{
pFWContext = (PBioIDENT_CONTEXT_T)pContext;

if(!pFWContext)
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Cathastrophic(1) : BioIDENT context
is NULL.");

FWAction = DROP;

break;
}
if(!pData)
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Cathastrophic(2) : NULL packet
buffer address.");
FWAction = DROP;
break;
}
if(!(*pData))
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Cathastrophic(3) : NULL packet
buffer.");
FWAction = DROP;

break;
```

```
}

pSKRecvBuffer = (struct SKRcvBuf *)*pData;

if(pSKRecvBuffer->SKr_size < sizeof(TSKHeader))
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted BIO PACKET BIR(1)");

FWAction = DROP;

break;
}

pSKHeader = (PSKHeader)pSKRecvBuffer->SKr_buffer;
if(sizeof(TSKHeader) > pSKHeader->SK_hl<<2)
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted BIO PACKET BIR(2)");
FWAction = DROP;
break;
}

if(pSKRecvBuffer->SKr_size < pSKHeader->SK_hl<<2)
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted BIO PACKET BIR(3)");
FWAction = DROP;
break;
}

NbioZeroMemory(pPacket,sizeof(TPacket));
NbioMoveMemory(&pPacket->SKHeader,pSKHeader,sizeof(TSKHeader));
switch(pSKHeader->SK_p)
{
case SKPROTO_TCP:
{
PTCPHeader pTCPHeader = NULL;

if(pSKRecvBuffer->SKr_size == pSKHeader->SK_hl<<2)
{
pSKRecvBuffer = pSKRecvBuffer->SKr_next;

if(pSKRecvBuffer)
if(pSKRecvBuffer->SKr_size >= sizeof(TTCPHeader))
pTCPHeader = (PTCPHeader)pSKRecvBuffer->SKr_buffer;
else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than minimum
TCP header size.");
}else
if(pSKRecvBuffer->SKr_size >= (pSKHeader->SK_hl<<2) + sizeof(TTCPHeader))
{
pTCPHeader = (PTCPHeader)((PCHAR)pSKHeader + (pSKHeader->SK_hl<<2));
}else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than expected
SK + TCP header size.");
```

```
if(pTCPHeader)NbioMoveMemory(&pPacket->TRHeader.TCPHeader,
pTCPHeader,sizeof(TTCPHeader));
else
{
FWAction = DROP;
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted packet(5) : A valid TCP
header does not exist.");
}

break;
}
case SKPROTO_UDP:
{
PUDPHeader pUDPHeader = NULL;

if(pSKRecvBuffer->SKr_size == pSKHeader->SK_hl<<2)
{

pSKRecvBuffer = pSKRecvBuffer->SKr_next;

if(pSKRecvBuffer)
if(pSKRecvBuffer->SKr_size >= sizeof(TUDPHeader))
pUDPHeader = (PUDPHeader)pSKRecvBuffer->SKr_buffer;
else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than minimum
UDP header size.");

}else
if(pSKRecvBuffer->SKr_size >= (pSKHeader->SK_hl<<2) + sizeof(TUDPHeader))
{
pUDPHeader = (PUDPHeader)((PCHAR)pSKHeader + (pSKHeader->SK_hl<<2));

}else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than expected
SK + UDP header size.");

if(pUDPHeader)
NbioMoveMemory(&pPacket->TRHeader.UDPHeader,pUDPHeader,sizeof(TUDPHeader));
else
{
FWAction = DROP;
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted packet(6) : A valid UDP
header does not exist.");
}

break;
}
case SKPROTO_ICMP:
{
PICMPHeader pICMPHeader = NULL;

if(pSKRecvBuffer->SKr_size == pSKHeader->SK_hl<<2)
{
```

```
pSKRecvBuffer = pSKRecvBuffer->SKr_next;

if(pSKRecvBuffer)
if(pSKRecvBuffer->SKr_size >= sizeof(TICMPHeader))
pICMPHeader = (PICMPHeader)pSKRecvBuffer->SKr_buffer;
else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than minimum
ICMP header size.");

}else
if(pSKRecvBuffer->SKr_size >= (pSKHeader->SK_hl<<2) + sizeof(TICMPHeader))
{
pICMPHeader = (PICMPHeader)((PCHAR)pSKHeader + (pSKHeader->SK_hl<<2));

}else
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Buffer size is smaller than expected
SK + ICMP header size.");

if(pICMPHeader)
NbioMoveMemory(&pPacket->TRHeader.ICMPHeader,pICMPHeader,sizeof(TICMPHeader));
else
{
FWAction = DROP;
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Corrupted packet(7) : A valid ICMP
header does not exist.");
}

break;
}
default:
break;
}

Direction = pFWContext->Direction == SK_TRANSMIT ? DIRECTION_OUT:DIRECTION_IN;
if(FWAction == FORWARD)
FWAction = FWRulesFilter(gpFWPasport,Direction,pPacket);

}while(0);

if(pPacket)
FWFree(pPacket);
return FWAction;
}
FORWARD_ACTION FWSKFilter(PFWRulesQueue pFWSKRules, TDirection Direction,
PSKHeader pSKHeader)
{
PFWRule pFWRule = NULL;
FORWARD_ACTION FWAction = FORWARD;
if((pFWSKRules == NULL) || (pSKHeader == NULL))
return FORWARD;
__try
{
```

```
for (pFWRule = pFWSKRules->Head; pFWRule != NULL; pFWRule = pFWRule->Next)
{
if (((pFWRule->Direction == Direction) || (pFWRule->Direction ==
DIRECTION_BOTH)) &&
((pFWRule->FWRule.SK.srcSK & pFWRule->FWRule.SK.srcMask) == (pSKHeader->SK_src
& pFWRule->FWRule.SK.srcMask)) &&
((pFWRule->FWRule.SK.dstSK & pFWRule->FWRule.SK.dstMask) == (pSKHeader->SK_dst
& pFWRule->FWRule.SK.dstMask)) &&
(!pFWRule->FWRule.SK.bProto || pFWRule->FWRule.SK.SKProto == pSKHeader->SK_p))
{
FWAction = pFWRule->Action;
if (pFWRule->bLog)
{
FWPostLogMessage("%s|%d|SK|%s|%d.%d.%d.%d|%d.%d.%d.%d|%d|",
FWAction == FORWARD ? "PASS" : "DROP", pFWRule->RuleID,
Direction == DIRECTION_IN ? "IN" : (Direction == DIRECTION_OUT
?"OUT":"IN/OUT"),
PRINT_SK_ADDR(pSKHeader->SK_src),
PRINT_SK_ADDR(pSKHeader->SK_dst),
pSKHeader->SK_p);
}
break;
}
}
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
FWAction = FORWARD;
}
return FWAction;
}
FORWARD_ACTIONFWTCPFilter(PFWRulesQueue pFWTCPRules, TDirection Direction,
PPacket pPacket)
{
PFWRule pFWRule = NULL;
FORWARD_ACTION FWAction = FORWARD;
if((pFWTCPRules == NULL) || (pPacket == NULL))
return FORWARD;
__try
{
for (pFWRule = pFWTCPRules->Head; pFWRule != NULL; pFWRule = pFWRule->Next)
{
if (((pFWRule->Direction == Direction) || (pFWRule->Direction ==
DIRECTION_BOTH)) &&
((pFWRule->FWRule.TCP.SKRule.srcSK & pFWRule->FWRule.TCP.SKRule.srcMask) ==
(pPacket->SKHeader.SK_src & pFWRule->FWRule.TCP.SKRule.srcMask)) &&
(pFWRule->FWRule.TCP.srcPortFrom == 0 || (pFWRule->FWRule.TCP.srcPortTo == 0 ?
(pPacket->TRHeader.TCPHeader.th_sport == pFWRule->FWRule.TCP.srcPortFrom) :
(ntohs(pPacket->TRHeader.TCPHeader.th_sport) >= ntohs(pFWRule-
>FWRule.TCP.srcPortFrom) && ntohs(pPacket->TRHeader.TCPHeader.th_sport) <=
ntohs(pFWRule->FWRule.TCP.srcPortTo)))) &&

((pFWRule->FWRule.TCP.SKRule.dstSK &
```

```
pFWRule->FWRule.TCP.SKRule.dstMask) == (pPacket->SKHeader.SK_dst & pFWRule-
>FWRule.TCP.SKRule.dstMask))&&
(pFWRule->FWRule.TCP.dstPortFrom == 0||(pFWRule-
>FWRule.TCP.dstPortTo==0?(pPacket->TRHeader.TCPHeader.th_dport == pFWRule-
>FWRule.TCP.dstPortFrom):(ntohs(pPacket-
>TRHeader.TCPHeader.th_dport)>=ntohs(pFWRule-
>FWRule.TCP.dstPortFrom)&&ntohs(pPacket->TRHeader.TCPHeader.th_dport)<=
ntohs(pFWRule->FWRule.TCP.dstPortTo))))&&(pFWRule->FWRule.TCP.FlagsSet == 0 ||
(pPacket->TRHeader.TCPHeader.th_flags & pFWRule->FWRule.TCP.FlagsSet)!=0)&&
(pFWRule->FWRule.TCP.FlagsUnset == 0 || (pPacket->TRHeader.TCPHeader.th_flags
& pFWRule->FWRule.TCP.FlagsUnset) == 0)){
FWAction = pFWRule->Action;
if (pFWRule->bLog)
{
char flags[9];
flags[0]=(pPacket->TRHeader.TCPHeader.th_flags & TH_FIN)?'F':'-';
flags[1]=(pPacket->TRHeader.TCPHeader.th_flags & TH_SYN)?'S':'-';
flags[2]=(pPacket->TRHeader.TCPHeader.th_flags & TH_RST)?'R':'-';
flags[3]=(pPacket->TRHeader.TCPHeader.th_flags&TH_PUSH)?'P':'-';
flags[4]=(pPacket->TRHeader.TCPHeader.th_flags & TH_ACK)?'A':'-';
flags[5]=(pPacket->TRHeader.TCPHeader.th_flags & TH_URG)?'U':'-';
flags[6]=(pPacket->TRHeader.TCPHeader.th_flags & TH_CWR)?'C':'-';
flags[7]=(pPacket->TRHeader.TCPHeader.th_flags & TH_ECE)?'E':'-';
flags[8]='\0';
FWPostLogMessage("%s|%d|TCP|%s|%d.%d.%d.%d:%d|%d.%d.%d.%d:%d|%s",
FWAction==FORWARD ? "PASS" : "DROP", pFWRule->RuleID,
Direction==DIRECTION_IN?"IN":(Direction == DIRECTION_OUT
?"OUT":"IN/OUT"),PRINT_SK_ADDR(pPacket->SKHeader.SK_src),
ntohs(pPacket->TRHeader.TCPHeader.th_sport),
PRINT_SK_ADDR(pPacket->SKHeader.SK_dst),
ntohs(pPacket->TRHeader.TCPHeader.th_dport),
flags);
}
break;
}
}
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
FWAction = FORWARD;
}
return FWAction;
}
FORWARD_ACTION FWUDPFilter(PFWRulesQueue pFWUDPRules, TDirection Direction,
PPacket pPacket)
{
PFWRule pFWRule = NULL;
FORWARD_ACTION FWAction = FORWARD;
if((pFWUDPRules == NULL) || (pPacket == NULL))
return FORWARD;
__try
{
```

```
for (pFWRule = pFWUDPRules->Head; pFWRule != NULL; pFWRule = pFWRule->Next)
{
if (((pFWRule->Direction == Direction) || (pFWRule->Direction ==
DIRECTION_BOTH)) &&
((pFWRule->FWRule.UDP.SKRule.srcSK & pFWRule->FWRule.UDP.SKRule.srcMask) ==
(pPacket->SKHeader.SK_src & pFWRule->FWRule.UDP.SKRule.srcMask)) &&
(pFWRule->FWRule.UDP.srcPortFrom == 0 || (pFWRule->FWRule.UDP.srcPortTo == 0 ?
(pPacket->TRHeader.UDPHeader.uh_sport == pFWRule->FWRule.UDP.srcPortFrom) :
(ntohs(pPacket->TRHeader.UDPHeader.uh_sport) >= ntohs(pFWRule-
>FWRule.UDP.srcPortFrom) && ntohs(pPacket->TRHeader.UDPHeader.uh_sport) <=
ntohs(pFWRule->FWRule.UDP.srcPortTo)))) &&

((pFWRule->FWRule.UDP.SKRule.dstSK & pFWRule->FWRule.UDP.SKRule.dstMask) ==
(pPacket->SKHeader.SK_dst & pFWRule->FWRule.UDP.SKRule.dstMask)) &&
(pFWRule->FWRule.UDP.dstPortFrom == 0 || (pFWRule->FWRule.UDP.dstPortTo == 0 ?
(pPacket->TRHeader.UDPHeader.uh_dport == pFWRule->FWRule.UDP.dstPortFrom) :
(ntohs(pPacket->TRHeader.UDPHeader.uh_dport) >= ntohs(pFWRule-
>FWRule.UDP.dstPortFrom) && ntohs(pPacket->TRHeader.UDPHeader.uh_dport) <=
ntohs(pFWRule->FWRule.UDP.dstPortTo)))))
{

FWAction = pFWRule->Action;

if (pFWRule->bLog)
{
FWPostLogMessage("%s|%d|UDP|%s|%d.%d.%d.%d:%d|%d.%d.%d.%d:%d",
FWAction == FORWARD ? "PASS" : "DROP", pFWRule->RuleID,
Direction == DIRECTION_IN ? "IN" : (Direction == DIRECTION_OUT
?"OUT":"IN/OUT"),
PRINT_SK_ADDR(pPacket->SKHeader.SK_src),
ntohs(pPacket->TRHeader.UDPHeader.uh_sport),
PRINT_SK_ADDR(pPacket->SKHeader.SK_dst),
ntohs(pPacket->TRHeader.UDPHeader.uh_dport));
}
break;
}
}
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
FWAction = FORWARD;
}

return FWAction;
}

FORWARD_ACTION FWICMPFilter(PFWRulesQueue pFWICMPRules, TDirection Direction,
PPacket pPacket)
{
PFWRule pFWRule = NULL;
FORWARD_ACTION FWAction = FORWARD;

if((pFWICMPRules == NULL) || (pPacket == NULL))
```

```
return FORWARD;

__try
{

for (pFWRule = pFWICMPRules->Head; pFWRule != NULL; pFWRule = pFWRule->Next)
{
if (((pFWRule->Direction == Direction) || (pFWRule->Direction ==
DIRECTION_BOTH)) &&
((pFWRule->FWRule.ICMP.SKRule.srcSK & pFWRule->FWRule.ICMP.SKRule.srcMask) ==
(pPacket->SKHeader.SK_src & pFWRule->FWRule.ICMP.SKRule.srcMask)) &&
((pFWRule->FWRule.ICMP.SKRule.dstSK & pFWRule->FWRule.ICMP.SKRule.dstMask) ==
(pPacket->SKHeader.SK_dst & pFWRule->FWRule.ICMP.SKRule.dstMask)) &&
(!pFWRule->FWRule.ICMP.bTypeSet || (pPacket->TRHeader.ICMPHeader.icmp_type ==
pFWRule->FWRule.ICMP.Type)) &&
(!pFWRule->FWRule.ICMP.bCodeSet || (pPacket->TRHeader.ICMPHeader.icmp_code ==
pFWRule->FWRule.ICMP.Code)))
{

FWAction = pFWRule->Action;

if (pFWRule->bLog)
{
FWPostLogMessage("%s|%d|ICMP|%s|%d.%d.%d.%d|%d.%d.%d.%d|%d.%d",
FWAction == FORWARD ? "PASS" : "DROP", pFWRule->RuleID,
Direction == DIRECTION_IN ? "IN" : (Direction == DIRECTION_OUT
?"OUT":"IN/OUT"),
PRINT_SK_ADDR(pPacket->SKHeader.SK_src),
PRINT_SK_ADDR(pPacket->SKHeader.SK_dst),
pPacket->TRHeader.ICMPHeader.icmp_type, pPacket-
>TRHeader.ICMPHeader.icmp_code);

}

break;
}
}
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
FWAction = FORWARD;
}

return FWAction;
}

FORWARD_ACTION  FWRulesFilter(PFWPasport pFWPasport, TDirection Direction,
PPacket pPacket)
{
FORWARD_ACTION FWAction = FORWARD;

PFWRuleSet   pFWRuleSet = NULL;
if(pFWPasport == NULL)
```

```
return FORWARD;
NbioAcquireSpinLock(&pFWPasport->PasportLock);
pFWRuleSet = pFWPasport->Head;


while(pFWRuleSet)
{
FWAction = FWFilter(pFWRuleSet, Direction, pPacket);

if(FWAction != FORWARD)
break;

pFWRuleSet = pFWRuleSet->Next;
}

NbioReleaseSpinLock(&pFWPasport->PasportLock);

return FWAction;
}

FORWARD_ACTION FWFilter(PFWRuleSet pFWRuleSet, TDirection Direction, PPacket
pPacket)
{

FORWARD_ACTION FWAction = FORWARD;

if(pFWRuleSet == NULL)
return FWAction;

if(pPacket)
{
NbioAcquireSpinLock(&pFWRuleSet->RuleSetLock);

FWAction = FWSKFilter(&pFWRuleSet->FWSKRules,Direction,&pPacket->SKHeader);

if(FWAction == FORWARD)
{
switch(pPacket->SKHeader.SK_p)
{
case SKPROTO_TCP:
FWAction = FWTCPFilter(&pFWRuleSet->FWTCPRules,Direction,pPacket);
break;
case SKPROTO_UDP:
FWAction = FWUDPFilter(&pFWRuleSet->FWUDPRules,Direction,pPacket);
break;
case SKPROTO_ICMP:
FWAction = FWICMPFilter(&pFWRuleSet->FWICMPRules,Direction,pPacket);
break;
default:
dprintf("SKFWSERVACTIVEX.sys : FWFilter - Unsupported SK protocol.");
break;
}
}
```

```
NbioReleaseSpinLock(&pFWRuleSet->RuleSetLock);
}

return FWAction;
}

Nbio_STATUS FWCreatePasport(PFWPasport *pFWPasport)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
PFWPasport pTemp = NULL;

do
{
__try
{

if(*pFWPasport != NULL)
{
NbioStatus = Nbio_STATUS_INVALID_DATA;

dprintf("SKFWSERVACTIVEX.sys : FWCreatePasport - A Pasport already exists.");

break;
}

NbioStatus = FWNewPasport(pFWPasport);

if(NbioStatus != Nbio_STATUS_SUCCESS)
{
dprintf("SKFWSERVACTIVEX.sys : FWCreatePasport - Can't create Pasport.");
break;
}

pTemp = *pFWPasport;

NbioAllocateSpinLock(&pTemp->PasportLock);

}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;

dprintf("SKFWSERVACTIVEX.sys : FWCreatePasport - Thrown exception.");
}
}while(0);


return NbioStatus;
}

Nbio_STATUS FWNewPasport(PFWPasport *pFWPasport)
{
```

```
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;

__try
{
*pFWPasport = (PFWPasport)FWAlloc(sizeof(TFWPasport));

if(*pFWPasport)
{
NbioZeroMemory(*pFWPasport, sizeof(TFWPasport));

NbioStatus = Nbio_STATUS_SUCCESS;

}else
{
NbioStatus = Nbio_STATUS_INVALID_DATA;

dprintf("SKFWSERVACTIVEX.sys : FWNewPasport - Can't allocate memory for
Pasport.");
}
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}

Nbio_STATUS FWDestroyPasport(PFWPasport *pFWPasport)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
PFWRuleSet  Prev = NULL;
PFWRuleSet  Next = NULL;
PFWPasport   pTemp = NULL;

if(*pFWPasport == NULL)
{
dprintf("SKFWSERVACTIVEX.sys : FWDestroyPasport - A Pasport does not exist.");

return Nbio_STATUS_INVALID_DATA;
}

__try
{
pTemp = *pFWPasport;

NbioAcquireSpinLock(&pTemp->PasportLock);

Next = pTemp->Head;

while(Next)
{
Prev = Next;
Next = Next->Next;
```

```
FWFreeRuleSet(&Prev);

}

NbioReleaseSpinLock(&pTemp->PasportLock);

NbioFreeSpinLock(&pTemp->PasportLock);

FWFree(*pFWPasport);

*pFWPasport = NULL;
NbioStatus = Nbio_STATUS_SUCCESS;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}

PFWRuleSet FWRegisterRuleSet(PFWPasport pFWPasport, PNbio_STATUS pNbioStatus)
{
PFWRuleSet  Next = NULL;
PFWRuleSet  Temp = NULL;

*pNbioStatus = Nbio_STATUS_SUCCESS;

if(pFWPasport == NULL)
{
*pNbioStatus = Nbio_STATUS_INVALID_DATA;

return NULL;
}

NbioAcquireSpinLock(&pFWPasport->PasportLock);

do
{
__try
{

*pNbioStatus = FWNewRuleSet(&Temp);

if(*pNbioStatus != Nbio_STATUS_SUCCESS)
{
Temp = NULL;
break;
}

NbioAllocateSpinLock(&Temp->RuleSetLock);

pFWPasport->Count += 1;
```

```
Next = pFWPasport->Head;

if(Next == NULL)
{
pFWPasport->Head = Temp;
break;
}
while(Next->Next)
Next = Next->Next;
Next->Next = Temp;
}
__except( EXCEPTION_EXECUTE_HANDLER )
{
*pNbioStatus = Nbio_Status_Failule;
}

}while(0);

NbioReleaseSpinLock(&pFWPasport->PasportLock);
return Temp;

}

Nbio_STATUS FWNewRuleSet(PFWRuleSet *pFWRuleSet)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
__try
{
*pFWRuleSet = (PFWRuleSet)FWAlloc(sizeof(TFWRuleSet));

if(*pFWRuleSet)
{
NbioZeroMemory(*pFWRuleSet, sizeof(TFWRuleSet));

NbioStatus = Nbio_STATUS_SUCCESS;

}else
NbioStatus = Nbio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}

Nbio_STATUS FWFreeRuleSet(PFWRuleSet *pFWRuleSet)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
__try
{
if(pFWRuleSet)
{
```

```
FWFreeRulesQueue(&(*pFWRuleSet)->FWSKRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWTCPRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWUDPRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWICMPRules);
NbioFreeSpinLock(&(*pFWRuleSet)->RuleSetLock);
FWFree((*pFWRuleSet));
(*pFWRuleSet) = NULL;
NbioStatus = Nbio_STATUS_SUCCESS;

}else
NbioStatus = Nbio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}

Nbio_STATUS FWFreeRulesQueue(PFWRulesQueue pFWRulesQueue)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
PFWRule Prev = NULL;
PFWRule Next = NULL;

__try
{
if(pFWRulesQueue)
{
Next = pFWRulesQueue->Head;

while(Next)
{
Prev = Next;
Next = Next->Next;
NbioStatus = FWFreeRule(Prev);

if(NbioStatus != Nbio_STATUS_SUCCESS)
dprintf("SKFWSERVACTIVEX.sys : FWFreeRulesQueue.");
}

pFWRulesQueue->Head = NULL;
NbioStatus = Nbio_STATUS_SUCCESS;
}else
NbioStatus = Nbio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}
```

```
Nbio_STATUS FWNewRule(PFWRule *pFWRule)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;

__try
{

*pFWRule = (PFWRule) FWAlloc(sizeof(TFWRule));
if(pFWRule)
{
NbioZeroMemory(*pFWRule,sizeof(TFWRule));

NbioStatus = Nbio_STATUS_SUCCESS;

}else
NbioStatus = Nbio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{ NbioStatus = Nbio_Status_Failule; }
return NbioStatus;
}
Nbio_STATUS FWFreeRule(PFWRule pFWRule)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
__try
{
if(pFWRule)
{
FWFree(pFWRule);

NbioStatus = Nbio_STATUS_SUCCESS;

}else
NbioStatus = Nbio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_Status_Failule;
}
return NbioStatus;
}
Nbio_STATUS FWInsertRule(PFWRuleSet pFWRuleSet, PFWRule pFWRule, TEFWRule
RuleType)
{
Nbio_STATUS   NbioStatus = Nbio_STATUS_SUCCESS;
PFWRulesQueue pFWRulesQueue = NULL;
PFWRule   Next = NULL;
PFWRule Temp = NULL;

if((pFWRuleSet == NULL)|| (pFWRule == NULL))
{
return Nbio_STATUS_INVALID_DATA;
```

```
}
NbioAcquireSpinLock(&pFWRuleSet->RuleSetLock);

do
{
__try
{
switch(RuleType)
{
case TEFWSK:
pFWRulesQueue = &pFWRuleSet->FWSKRules;
break;
case TEFWTCP:pFWRulesQueue = &pFWRuleSet->FWTCPRules;
break;
case TEFWUDP:pFWRulesQueue = &pFWRuleSet->FWUDPRules;
break;
case TEFWICMP:pFWRulesQueue = &pFWRuleSet->FWICMPRules;
break;
default:pFWRulesQueue = NULL;
break;
}
if(pFWRulesQueue == NULL)
{
NbioStatus = Nbio_STATUS_INVALID_DATA;
break;
}
NbioStatus = FWNewRule(&Temp);
if(NbioStatus != Nbio_STATUS_SUCCESS)
break;

NbioMoveMemory(Temp, pFWRule, sizeof(TFWRule));

pFWRulesQueue->Count += 1;

Next = pFWRulesQueue->Head;

if(Next == NULL)
{
pFWRulesQueue->Head = Temp;
break;
}
while(Next->Next)
Next = Next->Next;

Next->Next = Temp;

}
__except( EXCEPTION_EXECUTE_HANDLER )
{
NbioStatus = Nbio_Status_Failule;
}
}while(0);
```

```
NbioReleaseSpinLock(&pFWRuleSet->RuleSetLock);

return NbioStatus;
}
```

```
#include "precomp.h"

extern PFWLogList gpFWLogList;

NTSTATUS FWInitLoggingSubSystem(PFWLogList *ppFWLogList)
{
UNICODE_STRING    str;
OBJECT_ATTRIBUTES oa;
if(*ppFWLogList != NULL)
{
dprintf("SkFWHook.sys : FWInitLoggingSubSystem - The Subsystem has already
been initialized.");
return STATUS_INVALID_PARAMETER;
}
__try
{
(*ppFWLogList) = (PFWLogList)FWAlloc(sizeof(TFWLogList));

if((*ppFWLogList) == NULL)
{
dprintf("SkFWHook.sys : FWInitLoggingSubSystem - Can't allocate memory.");
return STATUS_BUFFER_TOO_SMALL;
}
NbioZeroMemory((*ppFWLogList),sizeof(TFWLogList));
NbioAllocateSpinLock(&(*ppFWLogList)->Lock);
RtlInitUnicodeString(&str,L"\\BaseNamedObjects\\SkFWHookLogger");

InitializeObjectAttributes(&oa, &str, 0, NULL, NULL);

NtStatus = ZwCreateEvent(&(*ppFWLogList)->hLogEvent, EVENT_ALL_ACCESS,&oa,
SynchronizationEvent, FALSE);
if(NtStatus != STATUS_SUCCESS)
{
dprintf("SkFWHook.sys : FWInitLoggingSubSystem - Catastrophic failure (1):
Can't create event.");

NbioFreeSpinLock(&(*ppFWLogList)->Lock);

FWFree((*ppFWLogList));

(*ppFWLogList) = NULL;

return NtStatus;
}

NtStatus = ObReferenceObjectByHandle((*ppFWLogList)->hLogEvent,
EVENT_ALL_ACCESS, NULL, KernelMode,&(*ppFWLogList)->LogEvent, NULL);
if (NtStatus != STATUS_SUCCESS)
{
dprintf("SkFWHook.sys : FWInitLoggingSubSystem - Catastrophic failure (2):
Can't reference event.");
```

```
NbioFreeSpinLock(&(*ppFWLogList)->Lock);
ZwClose((*ppFWLogList)->hLogEvent);
FWFree((*ppFWLogList));
(*ppFWLogList) = NULL;
return NtStatus;
}

}__except(EXCEPTION_EXECUTE_HANDLER)
{
NtStatus = STATUS_UNSUCCESSFUL;
}
return NtStatus;
}
NTSTATUS FWDeInitLoggingSubSystem(PFWLogList *ppFWLogList)
{
NTSTATUS NtStatus = STATUS_SUCCESS;
UINT     Flushed  = 0;

if((*ppFWLogList) == NULL)
{
dprintf("SkFWHook.sys : FWDeInitLoggingSubSystem - The subsystem has not been
initialized.");
return STATUS_INVALID_PARAMETER;
}
__try
{
Flushed = FWFlushLogList((*ppFWLogList));
dprintf("SkFWHook.sys : FWDeInitLoggingSubSystem - Flushed %d entries.",
Flushed);
if (&(*ppFWLogList)->LogEvent != NULL)
{
ObDereferenceObject(&(*ppFWLogList)->LogEvent);
}

if ((*ppFWLogList)->hLogEvent != NULL)
{
ZwClose((*ppFWLogList)->hLogEvent);
}
NbioFreeSpinLock(&(*ppFWLogList)->Lock);
FWFree((*ppFWLogList));

(*ppFWLogList) = NULL;
NtStatus = STATUS_SUCCESS;
}__except(EXCEPTION_EXECUTE_HANDLER)
{
NtStatus = STATUS_UNSUCCESSFUL;
}
return NtStatus;
}
Nbio_STATUS FWPostLogMessage(const char *Format, ...)
{
va_list Args;
```

```
PFWLog  pFWLog = NULL;

Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
do
{
__try
{
pFWLog = (PFWLog)FWAlloc(sizeof(TFWLog));
if(pFWLog == NULL)
{
NbioStatus = Nbio_STATUS_RESOURCES;
break;
}
va_start( Args, Format );
NbioZeroMemory(pFWLog,sizeof(TFWLog));
if(_vsnprintf( pFWLog->Text,MAX_TEXT, Format, Args ) == -1)
{
dprintf("SkFWHook.sys : FWPostLogMesage - Buffer overflow. Truncating...");
pFWLog->Text[MAX_TEXT-1] = '\0';
}
va_end(Args);
if(gpFWLogList == NULL)
{
NbioStatus = Nbio_STATUS_INVALID_DATA;
break;
}
NbioStatus = FWInsertLog(gpFWLogList,pFWLog);
if( NbioStatus == Nbio_STATUS_BUFFER_OVERFLOW )
{
dprintf("SkFWHook.sys : FWPostLogMesage - Log pool is full. Can't insert log
entry.");
FWFree(pFWLog);
NbioStatus = Nbio_STATUS_SUCCESS;
}
if( NbioStatus != Nbio_STATUS_SUCCESS )
{
dprintf("SkFWHook.sys : FWPostLogMesage - Can't insert log entry.");
break;
}
NbioAcquireSpinLock(&gpFWLogList->Lock);
if (gpFWLogList->LogEvent != NULL)
KeSetEvent(gpFWLogList->LogEvent, IO_NO_INCREMENT, FALSE);
else
dprintf("SkFWHook.sys : FWPostLogMesage - Cathastrophic Failure (3) : Logger
mutex has not been initialized.");
NbioReleaseSpinLock(&gpFWLogList->Lock);
}__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_STATUS_FAILURE;
}
}while(0);

if(NbioStatus != Nbio_STATUS_SUCCESS)
```

```
if(pFWLog)
FWFree(pFWLog);

return NbioStatus;
}
UINT FWFlushLogList(PFWLogList pFWLogList)
{
UINT Flushed = 0;
PFWLog Temp = NULL;
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
if(pFWLogList == NULL)
return Nbio_STATUS_INVALID_DATA;
NbioAcquireSpinLock(&pFWLogList->Lock);
__try
{

while(pFWLogList->Head)
{
Temp = pFWLogList->Head;

pFWLogList->Head = pFWLogList->Head->Next;

FWFree(Temp);

Flushed += 1;
}

pFWLogList->LogCount = 0;

pFWLogList->Head = pFWLogList->Tail = NULL;


}__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_STATUS_FAILURE;
}
NbioReleaseSpinLock(&pFWLogList->Lock);
if(NbioStatus == Nbio_STATUS_SUCCESS)
return Flushed;
return -1;
}

Nbio_STATUS FWGetLog(PFWLogList pFWLogList, PVOID ioBuffer)
{


Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
PFWLog Temp = NULL;
if(pFWLogList == NULL)
return Nbio_STATUS_INVALID_DATA;

NbioAcquireSpinLock(&pFWLogList->Lock);
```

```
__try
{
if(pFWLogList->LogCount > 0)
{
Temp = pFWLogList->Tail;

if(Temp)
{
NbioMoveMemory((PCHAR)ioBuffer, Temp->Text, MAX_TEXT*sizeof(CHAR));

pFWLogList->Tail = pFWLogList->Tail->Prev;

if(pFWLogList->Tail != NULL)
{
pFWLogList->Tail->Next = NULL;
}else
{
pFWLogList->Head = pFWLogList->Tail;
}
pFWLogList->LogCount -= 1;

FWFree(Temp);
}else
{
dprintf("SkFWHook.sys : FWGetLog - Cathastrophic Failure (4) : Memory leak
detected!");
NbioStatus = Nbio_STATUS_RESOURCES;
}
}else
{

NbioStatus = Nbio_STATUS_RESOURCES;
}
}__except(EXCEPTION_EXECUTE_HANDLER)
{

NbioStatus = Nbio_STATUS_FAILURE;
}

NbioReleaseSpinLock(&pFWLogList->Lock);

return NbioStatus;
}
Nbio_STATUS FWInsertLog(PFWLogList pFWLogList, PFWLog pFWLog)
{
Nbio_STATUS NbioStatus = Nbio_STATUS_SUCCESS;
PFWLog      Temp = NULL;
if((pFWLogList == NULL) || (pFWLog == NULL))
return Nbio_STATUS_INVALID_DATA;
NbioAcquireSpinLock(&pFWLogList->Lock);
do
{
__try
```

```
{

if(pFWLogList->LogCount > LOGSIZE)
{

NbioStatus = Nbio_STATUS_BUFFER_OVERFLOW;
break;
}
if(pFWLogList->Head == NULL)
{
pFWLog->Next = pFWLog->Prev = NULL;
pFWLogList->Head = pFWLogList->Tail = pFWLog;

}else
{
pFWLog->Next = pFWLogList->Head;

pFWLog->Prev = NULL;
pFWLogList->Head->Prev = pFWLog;
pFWLogList->Head = pFWLog;}
pFWLogList->LogCount += 1;
NbioStatus = Nbio_STATUS_SUCCESS;
}__except(EXCEPTION_EXECUTE_HANDLER)
{
NbioStatus = Nbio_STATUS_FAILURE;
}
}while(0);
NbioReleaseSpinLock(&pFWLogList->Lock);
return NbioStatus;
}
```

Клієнтська форма запиту

```
<head>
<meta http-equiv=Content-Type content="text/html; charset=windows-1251">
<meta name=Generator content="Rukami">
<link rel=File-List href="filelist.xml">
<link rel=Edit-Time-Data href="editdata.mso">
<link rel=OLE-Object-Data href="oledata.mso">
<!--[if !mso]>
<style>
v\:* {behavior:url(#default#VML);}
o\:* {behavior:url(#default#VML);}
w\:* {behavior:url(#default#VML);}
.shape {behavior:url(#default#VML);}
</style>
<![endif]-->
<title>Увага</title>
<!--[if gte mso 9]><xml>
 <o:DocumentProperties>
  <o:Author>AL</o:Author>
  <o:LastAuthor>AL</o:LastAuthor>
  <o:Revision>5</o:Revision>
  <o:TotalTime>40</o:TotalTime>
  <o:Created>2025-02-02T20:52:00Z</o:Created>
  <o:LastSaved>2025-02-02T21:32:00Z</o:LastSaved>
  <o:Pages>1</o:Pages>
  <o:Words>34</o:Words>
  <o:Characters>197</o:Characters>
  <o:Company>HOME</o:Company>
  <o:Lines>1</o:Lines>
  <o:Paragraphs>1</o:Paragraphs>
  <o:CharactersWithSpaces>230</o:CharactersWithSpaces>
  <o:Version>11.5606</o:Version>
 </o:DocumentProperties>
</xml><![endif]--><!--[if gte mso 9]><xml>
 <w:WordDocument>
  <w:View>Print</w:View>
  <w:Zoom>115</w:Zoom>
  <w:SpellingState>Clean</w:SpellingState>
  <w:GrammarState>Clean</w:GrammarState>
  <w:FormsDesign/>
  <w:PunctuationKerning/>
  <w:ValidateAgainstSchemas/>
  <w:SaveIfXMLInvalid>false</w:SaveIfXMLInvalid>
  <w:IgnoreMixedContent>false</w:IgnoreMixedContent>
  <w:AlwaysShowPlaceholderText>false</w:AlwaysShowPlaceholderText>
  <w:Compatibility>
   <w:BreakWrappedTables/>
   <w:SnapToGridInCell/>
   <w:WrapTextWithPunct/>
   <w:UseAsianBreakRules/>
   <w:DontGrowAutofit/>
  </w:Compatibility>
  <w:BrowserLevel>MicrosoftInternetExplorer4</w:BrowserLevel>
 </w:WordDocument>
</xml><![endif]--><!--[if gte mso 9]><xml>
 <w:LatentStyles DefLockedState="false" LatentStyleCount="156">
 </w:LatentStyles>
</xml><![endif]-->
<style>
<!--
 /* Style Definitions */
 p.MsoNormal, li.MsoNormal, div.MsoNormal
      {mso-style-parent:"";
      margin-bottom:.0001pt;
      mso-pagination:widow-orphan;
      font-size:12.0pt;
      font-family:"Times New Roman";
      mso-fareast-font-family:"Times New Roman"; margin-left:0cm; margin-
right:0cm; margin-top:0cm}
```

```
span.SpellE
      {mso-style-name:"";
      mso-spl-e:yes}
@page Section1
      {size:595.3pt 841.9pt;
      margin:2.0cm 42.5pt 2.0cm 3.0cm;
      mso-header-margin:35.4pt;
      mso-footer-margin:35.4pt;
      mso-paper-source:0;}
div.Section1
      {page:Section1;}
-->
</style>
<!--[if gte mso 10]>
<style>
 /* Style Definitions */
 table.MsoNormalTable
      {mso-style-name:"wrg";
      mso-tstyle-rowband-size:0;
      mso-tstyle-colband-size:0;
      mso-style-noshow:yes;
      mso-style-parent:"";
      mso-padding-alt:0cm 5.4pt 0cm 5.4pt;
      mso-para-margin:0cm;
      mso-para-margin-bottom:.0001pt;
      mso-pagination:widow-orphan;
      font-size:10.0pt;
      font-family:"Times New Roman";
      mso-ansi-language:#0400;
      mso-fareast-language:#0400;
      mso-bidi-language:#0400}
</style>
<![endif]-->
<!--[if gte mso 9]>
<xml><o:shapedefaults v:ext="edit" spidmax="1027"/>
</xml><![endif]-->
</head>

<body lang=RU style='tab-interval:35.4pt'>

<div class=Section1 style="width: 617; height: 327">

<p class=MsoNormal><span lang=UK style='font-size:16.0pt;font-family:Arial;
mso-ansi-language:UK'>Увага! Для доступу до захищеної частини сайту<br>
необхідно пройти <span class=SpellE>біопараметричне</span>
сканування!<o:p></o:p></span></p>


<OBJECT id="bioClient"   codebase="AxClient.cab"
classid="CLSID:7D6416BB-4417-4B92-B564-A39A474DC84E" VIEWASTEXT>
<PARAM NAME="BgColor" VALUE="255"> <PARAM NAME="FgColor" VALUE ="65535">
</OBJECT>

<form>

<p class=MsoNormal><span lang=UK style='font-family:Arial;mso-ansi-language:
UK'><INPUT TYPE="RADIO" CHECKED><span class="SpellE">Сканер відбитків
пальці          
</span></span>
<input type="submit" value="Провести сканування" name="B1" style="width:
200"></p>

<p class=MsoNormal><span lang=UK style='font-family:Arial;mso-ansi-language:
UK'><INPUT TYPE="RADIO" VALUE="віпотцнко"></span><span class="SpellE"><span
lang=UK style='font-family:Arial;mso-ansi-language:
UK'>Сканер
підпису          </span>
</span><input type="button" value="Відміна" name="B2" style="width: 200"></p>
```

```
</form>
<p class=MsoNormal align="center"><span style='font-size:16.0pt;font-
family:Arial'><!--[if gte vml 1]><v:shapetype id="_x0000_t75"
 coordsize="21600,21600" o:spt="75" o:preferrelative="t"
path="m@4@5l@4@11@9@11@9@5xe"
 filled="f" stroked="f">
 <v:stroke joinstyle="miter"/>
 <v:formulas>
  <v:f eqn="if lineDrawn pixelLineWidth 0"/>
  <v:f eqn="sum @0 1 0"/>
  <v:f eqn="sum 0 0 @1"/>
  <v:f eqn="prod @2 1 2"/>
  <v:f eqn="prod @3 21600 pixelWidth"/>
  <v:f eqn="prod @3 21600 pixelHeight"/>
  <v:f eqn="sum @0 0 1"/>
  <v:f eqn="prod @6 1 2"/>
  <v:f eqn="prod @7 21600 pixelWidth"/>
  <v:f eqn="sum @8 21600 0"/>
  <v:f eqn="prod @7 21600 pixelHeight"/>
  <v:f eqn="sum @10 21600 0"/>
 </v:formulas>
 <v:path o:extrusionok="f" gradientshapeok="t" o:connecttype="rect"/>
 <o:lock v:ext="edit" aspectratio="t"/>
</v:shapetype><v:shape id="_x0000_s1026" type="#_x0000_t75" style='width:72.6pt;
 height:117pt' fillcolor="window">
 <v:imagedata src="Увага1.files/image001.png" o:title=""/>
</v:shape><![endif]--><![if !vml]><img border=0 width=121 height=195
src="Увага1.files/image002.jpg" v:shapes="_x0000_s1026"><![endif]> 
</span><span style='font-size:16.0pt;font-family:Arial'><o:p> 
<img border="0" src="Увага.files/Graphic1.jpg" width="251"
height="134"></o:p></span></p>
<p class=MsoNormal align="center"> </p>
<p class=MsoNormal align="center"> </p>
<p class=MsoNormal align="center"> </p>
<p class=MsoNormal align="center"><font face="Arial">Дія проведена. Якість
біопараметричного зразку задовільна.</font></p>
<p class=MsoNormal align="center"><font face="Arial">Чекайте іде формування
моделі сканування ..... </font></p>
<p class=MsoNormal align="right"> </p>
<p class=MsoNormal align="right"> </p>
<p class=MsoNormal align="right"><font face="Tahoma" size="2">Бакалаврська
дипломна робота 2025 р.</font></p>
<p class=MsoNormal align="right"><font face="Tahoma"
size="2">Центральноукраїнській
національний технічний університет</font></p>
<p class=MsoNormal align="right"><span style="font-family: Tahoma">
<font size="2">Розробник Філіпенко Микита Сергійович, КБ-22-МБ</font></span></p>
</div>
</body>
</html>
```