Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-20МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Кузнецова Т.Ю.
«____» _____ 2021 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Наталія ЯКИМЕНКО
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Центр *Заочної та дистанційної освіти*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"Інформаційні технології"*_____

Спеціальність *122 "*Комп'ютерні науки*"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Кузнецовій Тетяні Юріївні
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 40-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі*

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

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Кузнецова Т.Ю. Дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи біометричного доступу до даних у хмарному сервісі.

Метою розробки є дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

Об'єктом дослідження є процес біометричного доступу до даних у хмарному сервісі.

Предметом дослідження є методи біометричного доступу до даних у хмарному сервісі.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

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 архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Visual C++, HTML, ASP, Jscript.

Ключові слова: Комп'ютерні науки, біометричний доступ, хмарний сервіс

ABSTRACT

Kuznetsova T.Yu. Research and software implementation of biometric data access system in the cloud service. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work for the second (master's) level of higher education, software has been developed that is designed for a system of biometric access to data in the cloud service.

The purpose of development is research and software implementation of the system of biometric access to data in the cloud service.

The object of research is the process of biometric access to data in the cloud service.

The subject of research is the methods of biometric access to data in the cloud service.

Research methods are based on methods of information security theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of a system of biometric access to data in the cloud service.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in Visual C ++, HTML, ASP, Jscript.

Keywords: Computer science, biometric access, cloud service

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. Ідентифікацію й автентифікацію можна вважати основою програмно-технічних засобів безпеки, оскільки інші сервіси розраховані на обслуговування іменованих суб'єктів. Ідентифікація й автентифікація – це перша лінія оборони, "прохідна" інформаційного простору організації [1].

У відкритому мережному середовищі між сторонами ідентифікації/автентифікації не існує довіреного маршруту; це значить, що в загальному випадку дані, передані суб'єктом, можуть не збігатися з даними, отриманої й використаними для перевірки дійсності. Необхідно забезпечити захист від пасивного й активного прослуховування мережі, тобто від перехоплення, зміни й/або відтворення даних. Передача паролів у відкритому виді, мабуть, незадовільна; не рятує положення й шифрування паролів, тому що воно не захищає від відтворення. Потрібні більш складні протоколи автентифікації [1-6].

Сучасні засоби ідентифікації/автентифікаціи повинні підтримувати концепцію єдиного входу в мережу. Єдиний вхід у мережу – це, у першу чергу, вимогп зручності для користувачів.

Традиційно в Інтернет автентификація користувачів виробляється за допомогою Імені (Login) і Пароля (Password). Використання цього підходу, як і все у світі, має свої переваги й свої недоліки. Переваги очевидні – простота реалізації, відсутність необхідності здобувати додаткові пристрої, за винятком клавіатури, зрозуміло, звичка, нарешті.... Але є й недоліки, в основному пов'язані з «людським фактором», а саме те, що людині важко запам'ятовувати довгі й складні паролі й отут починаються проблеми – користувачі забувають паролі, передають свої паролі третім особам, або пишуть свої паролі на папірцях (і навіть прикріплюють їх на монітор), придумують собі прості паролі, або просто використовують той самий пароль «на всі випадки життя» [2-3]. Все це рано або

пізно приводить до втрати пароля в найкращому разі або до його «крадіжки» і використанню без відома користувача.

Наявність проблеми підстьобнуло розробку альтернативних шляхів автентифікації й ідентифікації. Була розроблена безліч рішень, що дозволяють так чи інакше замінити спосіб їхнього зберігання й введення або замінити саму схему Ім'я/Пароль. До першої групи рішень відносять різні пристрої, такі як смарт-карти й електронні таблетки й ключі, у яких зберігається Ім'я й Пароль або інформація, що їх заміняє, до останньої групи рішень відносять різні біопараметричні способи ідентифікації й автентифікації по персональних особливостях користувача, таким як відбитки пальців, сітківка ока, форми особи й рук і інші. Смарт-карти, електронні таблетки й ключі (далі – електронні ключі) безумовно, усувають деякі з недолік ів схеми Ім'я/Пароль: користувачеві не потрібно запам'ятовувати свій пароль, не потрібно його вводити, а також можна створювати й зберігати в електронних ключах паролі будь -якої складності й довжини. Однак, у цієї схеми є кілька очевидних недоліків – ключ із паролем як і раніше можуть украсти, його можна втратити або забути, можна передати добровільно (або не зовсім добровільно) третім особам, ну й нарешті необхідний додатковий прилад – зчитувач електронних ключів [1, 8].

Біопараметричні способи автентифікації й ідентифікації усувають багато проблем, тому що як ключ використовується невід'ємна частина людського організму [4-6]. Такий ключ не можна втратити або забути, передати іншій особі, та й пароль як такий у принципі не використовується в цій схемі. Недоліків два – додаткові прилади для зчитування біопараметричних особливостей і необхідність у надійному й швидкому алгоритмі розпізнавання.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

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

– Огляд існуючих систем біометричного доступу до даних у хмарному сервісі.

– Дослідження системи біометричного доступу до даних у хмарному сервісі.

– Програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

Об'єктом дослідження є процес біометричного доступу до даних у хмарному сервісі.

Предметом дослідження є методи біометричного доступу до даних у хмарному сервісі.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод біометричного доступу до даних у хмарному сервісі.

– Розроблено вітчизняний продукт біометричного доступу до даних у хмарному сервісі, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі біометричного доступу до даних у хмарному сервісі.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У звичайному житті ми узнаємо один одного в обличчя. Якщо знайомі. Якщо не знайомі – по паспорту або аналогічному документу з фотографією. «Узнати» же людину, що сидить за комп'ютером по ту сторону Мережі, трохи складніше – це вимагає досить специфічних методів.

Ідентифікація й автентифікація

Перш ніж перевіряти істинність користувача, його потрібно ідентифікувати, тобто з багатьох користувачів, зареєстрованих у системі, вибрати по якомусь унікальному ідентифікаторі одного. Його система й буде перевіряти. Ідентифікатор – це ім'я, під яким зареєстрований користувач у комп'ютерній системі, що його перевіряє. Інакше кажучи, ідентифікація користувача – це одержання від нього відповіді на питання: «Хто ти?». А автентифікація – це вимога доказу що це саме ти, і наступна перевірка доказів. Тобто перевірка, чи дійсно користувач є тим, за кого він себе видає.

Автентифікація користувачів звичайно виконується якимось програмним модулем, що перебуває безпосередньо на комп'ютері, на якому користувач намагається одержати прямій або вилучений доступ. Всю роботу даного модуля можна умовно розділити на два етапи.

Попередній, на якому модуль формує «еталонний зразок», наприклад, запитує пароль користувача (це саме тоді пароль запитується двічі, щоб виключити помилку його введення) – по ньому користувач буде орієнтуватися згодом. Пароль може й призначатися користувачеві – так буває, наприклад, у різних системах доступу в Інтернет. Звичайно модуль автентифікації зберігає еталонні зразки в таблиці відповідностей «користувач – еталон».

І завершальний етап, коли користувач проходить автентифікацію й у нього запитується автентифікаційна інформація, що рівняється з еталоном. На підставі цього порівняння він вважається пізнаним чи ні.

Насправді в реальних системах еталонний пароль може зберігатися в таблиці в зашифрованому виді або замість пароля зберігається його хеш. Це не дозволить зловмисникові, що одержав доступ до сховища еталонів, ознайомитися з паролями всіх користувачів системи.

У більш складних випадках (насамперед при віддаленій автентифікації) пропонована користувачем автентифікаційна інформація і її еталонний зразок можуть доповнювати один одного, беручи участь у яких-небудь криптографічних перетвореннях. Для цього використовуються різні протоколи мережної автентифікації.

Інформація, по якій зорієнтується користувач, буває трьох видів:

- Користувач знає щось унікальне й демонструє комп'ютеру це знання. Такою інформацією може бути, наприклад, пароль.

- Користувач має предмет з унікальним умістом або з унікальними характеристиками.

- Автентифікаційна інформація є невід'ємною частиною користувача. По цьому принципі будуються системи біопараметричної автентифікації, що використовують у якості інформації, наприклад, відбиток пальця.

Як видно, такі методи прийшли з реального життя: паролями люди користувалися з незапам'ятних часів, та й зараз вони зустрічаються в різних областях, у тому числі не пов'язаних з комп'ютерною технікою, – скажемо, різні номерні комбінації кодових замків. Ще частіше застосовуються унікальні предмети: це ключі від будь-яких замків, електронні таблетки для швидкого відкриття домофонів, проїзні квитки у вигляді карток з магнітною смугою. Поширені в повсякденному житті й методи біопараметричної автентифікації: паспорт і права водія з еталонною фотографією, відбитки пальців, використовувані в криміналістиці й т.д.

Кожний з перерахованих вище методів має свої переваги й недоліки. Для усунення останніх у комп'ютерних системах часто використовують комбінацію різних методів автентифікації, наприклад смарт-карту у вигляді предмета з унікальним умістом (скажемо, криптографічним ключем), для доступу до якого необхідно ввести PIN-код. Тобто користувач для входу в систему не тільки повинен мати при собі смарт-карту, але й знати унікальну послідовність – PIN-код. Така автентифікація називається двухфакторною – по числу параметрів, що перевіряються. Приклад двухфакторной автентифікації в житті знайти складніше, навскидку пригадується тільки той же паспорт, що являє собою предмет з унікальним умістом, серед якого є фотографія особи його власника – біопараметричної характеристики, що є невід'ємною його частиною.

1.2 Область застосування

Поговоримо про переваги й недоліки згаданих вище методів.

Пароль

По парольному принципі будуються найпростіші системи автентифікації, у яких користувачеві досить ввести правильний пароль для одержання доступу до потрібного йому ресурсу. Парольна автентифікація є найпоширенішою: по-перше, це найпростіший з розглянутих нами методів автентифікації (що є єдиним його достоїнством), по-друге, він з'явився набагато раніше інших, тому до теперішнього часу реалізований у величезній кількості різних комп'ютерних програм.

Недоліків же в парольної автентифікації не порахувати.

По-перше, дуже часто недосвідчені користувачі вибирають прості або паролі, що вгадуються легко:

- яку-небудь похідну від ідентифікатора користувача (у тому числі нерідко безпосередньо сам ідентифікатор);

- слово якої-небудь мови (у тому числі власне ім'я – кличку собаки, назва улюбленої футбольної команди й т.д.) або загальновживану фразу. Такі паролі підбираються зловмисником шляхом *словникової атаки* – перебором слів і найбільше часто вживаних фраз по певному словнику. Причому відомо, що переважна більшість таких паролів укладається у відносно невеликий «список найбільш частих паролів»;

- нерідко користувачі застосовують короткі паролі, які легко підбираються перебором всіх можливих варіантів.

По-друге, пароль можуть підглянути або перехопити при введенні.

По-третє, пароль може бути отриманий шляхом застосування насильства до його власника.

Нарешті, існують і застосовуються зловмисниками діючі методи *соціальної інженерії*, за допомогою яких можна одержати пароль користувача шляхом обману – недосвідчений користувач сам назве його зловмиснику, якщо той зможе спритно прикинутися адміністратором системи. Останнім досягненням зловмисників на даному поприщі є *фішинг*: користувач заманюється на фальшиву веб-сторінку, що імітує, скажемо, потрібну сторінку сайту його банку, – там він уводить параметри своєї кредитної карти, з якої потім зніма ють гроші злочинці. До речі, у таких випадках допомагають методи *взаємної автентифікації*, коли не тільки сервер перевіряє користувача, але й користувач переконується, що це саме сервер його банку. Не можна сказати, що технічний прогрес у відношенні парольної автентифікації стоїть на місці. Не припиняються спроби побудувати сильну автентифікацію в сполученні зі зручністю й простотою застосування паролів. Розроблено безліч програмних і апаратних *генераторів паролів*, які виробляють довгі й сильні випадкові паролі, невразливі для словникових атак і інших варіантів підбора. Зворотна сторона медалі: користувачі змушені запам'ятовувати довгі й складні паролі, результат – спокуса записати пароль на папірець і повісити її на монітор. Існують системи, що включають *пароль під примусом*: користувач застосовує один пароль для нормального входу в систему,

а інший, також визначений пароль, сигналізує модулю автентифікації про те, що користувача примушують до входу в систему. Програми прозорого шифрування, наприклад, надають при введенні пароля під примусом дезінформацію, попередньо підібрану користувачем, якого примушують.

Однак видно, що подібні вдосконалення ускладнюють парольну автентифікацію, основне достоїнство якої – простота.

Унікальний предмет

Для автентифікації користувачів найбільше часто застосовуються наступні предмети: смарт-карти, карти з магнітною смугою, електронні таблетки iButton, USB-токени. Унікальність кожного з перерахованих предметів визначається інформацією, що він містить. У найпростішому випадку ця інформація являє собою ідентифікатор і пароль користувача, які просто зчитуються з носія й передаються модулю автентифікації. Більше складний випадок – носій містить криптографічний ключ, що використовується в якому-небудь із протоколів вилученої автентифікації. До речі кажучи, мікропроцесорні смарт-карти й USB-токени можуть самі виконувати криптографічні перетворення й відігравати активну роль в автентифікації. Унікальний предмет використовується сам по собі досить рідко: найчастіше це один з елементів двухфакторной автентифікації, приклад якої був наведений вище.

Недоліків в «предметної» автентифікації трохи менше, ніж у парольної:

– предмет може бути викрадений або віднятий у його власника;
– потрібне спеціальне встаткування для роботи із предметами;
– іноді можливе виготовлення копії або емулятора предмета.

Незважаючи на ці недоліки, носії автентифікаційної інформації зараз досить популярні. Особливо це стосується USB-токенів, для використання яких не потрібно ніякого встаткування – досить не дуже древнього комп'ютера. Крім того, для автентифікації за допомогою USB-токенів в операційних системах Windows 2000 і вище із програмного забезпечення необхідний тільки драйвер

спеціального формату для використовуваного USB-токена – тут уже включена підтримка такий автентифікації.

Біопараметрична автентифікація

Ще десяток років тому біопараметрична автентифікація зустрічалася в основному у фантастичних добутках. Зараз біопараметричні технології переживають період бурхливого росту, що різко підсилився після подій 11 вересня 2001 р. Тоді експерти порахували, що біопараметричні технології, особливе розпізнавання людей по рисах особи, допоможуть у пошуку терористів і інших зловмисників. На жаль, не можна сказати, що очікування експертів повністю виправдалися, однак біопараметричні технології міцно зайняли своє місце на ринку засобів ідентифікації й автентифікації користувачів.

У якості автентифікаційної інформації в цьому випадку беруться в увагу оригінальні й невід'ємні характеристики людини. Найбільше часто використовуються наступні з них:

1. Відбитки пальців. Відомо, що вони унікальні для кожної людини, причому не міняються протягом життя. Для сканування відбитків пальців застосовується найдешевше встаткування (у порівнянні з іншими методами біопараметричної автентифікації), крім того, даний метод звичний для користувачів і не викликає яких-небудь побоювань. Однак вважається, що недорогі сканери відбитків пальців можна обдурити спеціально виготовленим штучним пальцем.

2. Рисунок райдужної оболонки ока. Це на сьогодні найбільш точний метод біопараметричної автентифікації. Але багато користувачів бояться процесу сканування райдужної оболонки, та й устаткування для сканування є дорогим. До того ж даний спосіб викликає дорікання з боку правозахисників. Вони говорять, що око людини несе багато інформації про стан його здоров'я, про зловживання спиртними напоями, наркотиками й т.д. Є побоювання, що цю інформацію про користувачів (побічну для процесу автентифікації) налаштована відповідним чином система може зберігати, після чого можливо її використання їм на шкоду.

3. Риси особи. Дана технологія розпізнавання вважається дуже перспективної, оскільки саме по рисах особи довідаються один одного люди. На жаль, системи, що реалізують даний метод, поки не блищать точністю.

Як унікальні ознаки людини використовуються також характеристики його голосу, зразок рукописного підпису, «клавіатурний почерк» (інтервали часу між натисканнями клавіш, що становлять кодове слово, і інтенсивність натискань), геометрія руки й ін. Однак ці технології значно менше поширені.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Огляд засобів і принципів роботи біопараметричних систем

Незважаючи на наявність на ринку альтернативних засобів автентифікації, може трапитися так, що без імен користувачів і паролів обійтися буде неможливо. От кілька рекомендацій для таких випадків.

– Потребуючі введення імен користувачів і паролів Web-сайти відвідуйте лише в тому випадку, якщо в них реалізована технологія Secure Sockets Layer (SSL).

– Вибирайте надійні паролі довжиною не менш восьми символів. Вони повинні складатися із символів верхнього й нижнього регістрів, чисел і знаків пунктуації.

– Для кожної системи використовуйте особливий пароль (і, якщо можливо, особливе ім'я користувача). Якщо ваші облікові дані для однієї системи стануть надбанням зловмисника, він не зможе використовувати їх на інших системах.

– Якщо для кожного облікового запису ви будете застосовувати окреме ім'я користувача й пароль, у вас нагромадиться великий обсяг різних облікових даних. У цьому випадку рекомендується придбати недорогий засіб автентифікації за допомогою біометричних даних, наприклад пристрій зчитування відбитків пальців, що дозволить зберігати кожний набір імен користувача й паролів і автоматично реєструватися на Web-сайтах, пред'явивши біометричні дані.

– Не зберігаєте на Web-сайтах відомості про свою кредитну карту або інші що дозволяє ідентифікувати вас інформацію.

Різні системи контрольованого забезпечення доступу можна розділити на три класи відповідно до того, що людина повинна пред'являти системі:

– те, що він знає;

– те, чим він володіє;

– те, що є частиною його самого.

Перший клас використовує різного роду шифри, що набираються людиною (наприклад, PIN-коди, криптографічні коди й т.п.).

Другий клас використовує шифри, передані за допомогою фізичних носіїв інформації (пластикові карти з магнітною смугою, електронні таблетки "touch memory", електронні token-пристрої, proximity-карти й т.д.).

Третій – біопараметричний клас принципово відрізняється тим, що ідентифікації піддається властиво особистість людини – його індивідуальні характеристики (рисунок папіллярного візерунка, райдужна оболонка ока й т.д.).

Біопараметричні системи доступу є досить зручними й дружелюбними для користувачів. У відмінності від паролів і носіїв інформації для систем контролю доступу (ключів, карт, токенів), які можуть бути загублені, украдені, скопійовані, вони засновані на біопараметричних параметрах (відбитки пальців, форма руки, особи, рисунок райдужної оболонки ока,...), які завжди з нами й проблема їхньої схоронності, як правило, вирішується автоматично. Втратити їх набагато складніше. У ряді систем, що ще не одержали масового поширення, досягнута висока якість "біопараметричних ключів/замків". Треба думати, що через якийсь час вони проникнуть у сферу побуту.

Біопараметричні технології дуже швидко розвиваються останнім часом, і ринок цих пристроїв уже зараз становить біля одного мільярда доларів. За прогнозами International Biometric Group через чотири роки цей ринок складе більше 4 млрд. доларів. Дотепер всі прогнози цієї організації трохи перевиконувалися реальним розвитком ринку. У світі працює більше 400 компаній, які пов'язані з розробками й реалізацією біопараметричної продукції.

Застосування біопараметричних систем

Раніше відбитки пальців виготовлялися шляхом прикладання пофарбованих пальців до паперу. Робота з такою інформацією була вкрай трудомісткою. На цей момент, практично всі бази переведені на електронні носії. У США одній з найбільш відомих у цій області компаній є " SAGEM-MORPHO".

У цивільному й міліцейському застосуванні біопараметрики (у частині дактилоскопії) є певні розходження. Так як криміналістам треба ідентифікувати особистість часто по невеликих частинах відбитків пальців, вони повинні зберігати у своїх базах повні відбитки всіх пальців. Очевидно, що джерела інформації й "споживачі" тут не прагнуть пред'являти свої відбитки пальців.

У випадку ж застосування біопараметрики в системах контролю доступу, навпроти, кожна людина намагається якнайкраще й вірніше пред'явити свій відбиток пальця, щоб його швидше й краще розпізнали. У силу цього дуже часто обмежуються зіставленням щодо невеликої центральної області відбитка пальця, розміром іноді до 10х10 мм. У даний момент у рамках роботи комітету SC37 JTC1 ISO-IEC саме обговорюється питання про стандартизацію його розміру. При прийнятті цього рішення варто пам'ятати, що чим менше ділянка пальця, по якому виробляється ідентифікація, тим менш точна ця ідентифікація. При дуже маленьких ділянках, скажемо 6х6 мм, імовірність помилки велика й досить швидко убуває в міру збільшення розміру скануємої ділянки відбитка пальців. Після досягнення розмірів 12х18 мм із подальшим збільшенням розмірів імовірність помилки зменшується вже значно слабкіше, а після того, як починають використовувати зону сканування 20х25 мм, їхня зміна не істотна.

Ще одна важлива відмінність міліцейських систем від систем контролю доступу полягає у важливості такого параметра, як швидкодія алгоритму розпізнавання. У криміналістиці треба пізнати пропонований відбиток, зрівнявши його з відбитками всієї або обґрунтовано обмеженої бази даних. Тут швидкодія дуже важлива. У системах контролю доступу можна надходити по-іншому, порівнюючи відбиток пропонованого пальця з відбитком пальця, що витягнутий з

бази, виходячи з якої-небудь додаткової ознаки людини, наприклад, названого ім'я. Реально, однак, порівнюють звичайно не відбитки пальців, а моделі, деякий набір параметрів особливих точок відбитків, тобто тих точок, де візерунок має особливості, наприклад, де лінія візерунка припиняється або розгалужується. Як параметри звичайно використовуються відносні координати особливих точок, кути нахилу дотичних до ліній візерунка.

Потреба в надійній ідентифікації й досягнутий рівень біопараметрики в наш час приводять до створення біопараметричних паспортів і введенню біопараметричного контролю при перетинанні границь.

Всі розвинені країни готуються до впровадження біопараметричних паспортів, і Україна не є виключенням. Активно йде робота з вироблення біопараметричних стандартів для забезпечення реальної роботи з біопараметричними паспортами на всій земній кулі.

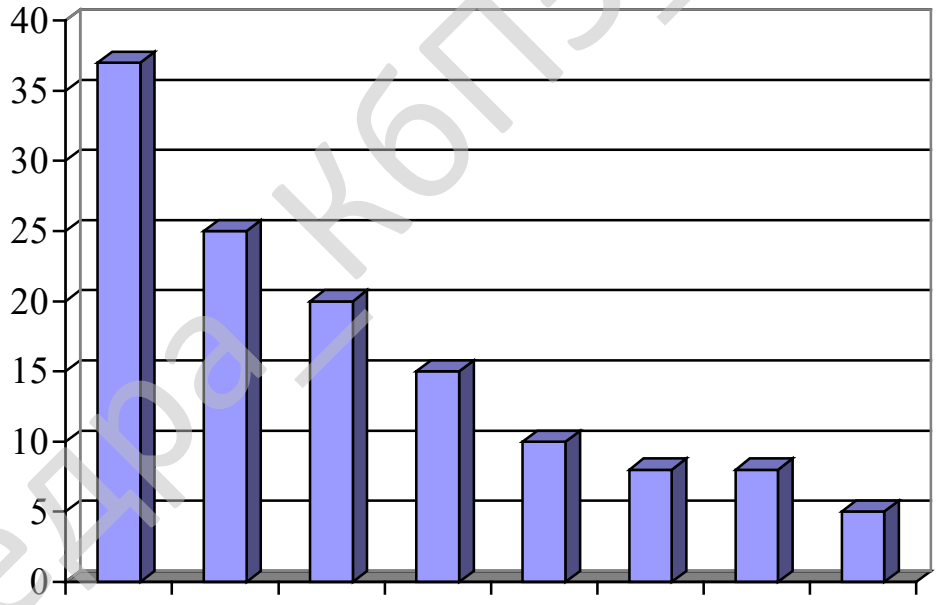


Рисунок 2.1 – Області використання біопараметричних систем

Практично все для технічного рішення цього питання вже є. Більше складним є рішення соціологічних, психологічних і юридичних питань. Із цих питань створена спеціальна група вже згаданого міжнародного комітету зі стандартизації в області біопараметрики. Найімовірніше біопараметричні паспорти будуть містити в собі інформацію про відбитки пальців, візерунку райдужної оболонки ока, формі особи. На минулій в 2007 році виставці CEBIT, наприклад, Німеччина демонструвала зразок такого паспорта, реалізованого у вигляді спеціального чипа, що вбудовується у звичайний паспорт. На чипі зберігається звичайна текстова інформація, як і в паспорті, високоякісне фото, дані про райдужну оболонку й відбитки пальців.

Огляд біопараметричних параметрів

Що варто розуміти під біопараметричними параметрами. У цей час біопараметричним глоссарієм у рамках проекту стандарту ISO/IEC займається одна з робочих груп спеціального біопараметричного комітету SC37, що входить у найбільший міжнародний комітет зі стандартизації в області інформаційних технологій – JTC1. Відповідно до його біопараметричним параметром/характеристикою є вимірювана фізична характеристика або поведінкова риса, використовувана для розпізнавання людини, його ідентифікації, або перевірки чи є він тим, ким себе заявляє.

До теперішнього часу розроблена досить велика кількість технологій, що використовують для ідентифікації біопараметричні параметри, які добре й регулярно відтворюються й мало або зовсім не змінюються.

До основних біопараметричних параметрів відносяться:

А) фізіологічна група

– форма руки й розташування в ній кровоносних судин;
– папиллярні візерунки відбитків пальців;
– рисунок райдужної оболонки ока;
– розташування кровоносних судин в оці – їхній візерунок;
– голос;

– форма особи, термограмма особи;

– ДНК.

Б) поведінкова група

– вид і характер написання підпису або кодового слова;

– характер роботи із клавіатурою.

Слід зазначити, що параметр "Голос" носить почасти поведінковий характер. Існують, звичайно, і інші параметри, але наведені використовуються в найбільшій мірі.

Пристрої для зчитування біопараметричних даних

Найвідомішими з експлуатованих нині пристроїв цієї категорії є пристрої зчитування відбитків пальців. Все частіше ноутбуки поставляються з убудованими дактилоскопічними зчитувачами, і ви можете легко придбати пристрій зчитування відбитків пальців, що підключається через порт USB. Рідше зустрічаються пристрої зчитування відбитків долонь, а також зчитувачі, що фіксують відстань між пальцями користувача, що розчепленні, й тиск, надаваний ними на опуклі датчики на поверхні пристрою. Системи розпізнавання райдужної оболонки ока, а також голосу поки не знаходять широкого застосування, за винятком організацій, що діють в умовах високої захищеності.

Принцип роботи пристроїв для зчитування біопараметричних даних

Прилад зчитує біопараметричні дані, проводить необхідні виміри й зводить їхні результати до унікального значення (це хеш). Приміром, пристрій зчитування відбитків пальців може розрахувати відношення між гребінцем і западиною папіллярного візерунка на пальці й перетворити його в значення хеш-функції. Втім, тут варто мати на увазі одну обставину. Відносно дешеві пристрої для зчитування біопараметричних даних можуть частіше робити помилки типу I і II, чим більш дорогі пристрої. Помилка типу I – це помилка виключення (скажемо, відбиток пальця користувача або інші його біопараметричні дані помилково зізнаються недійсними). Помилка типу II, більш небезпечна, – це помилкове розпізнавання (ситуація, коли відбиток пальця або інші біопараметричні дані іншої

людини помилково зізнаються даними користувача). Внаслідок недостатньої надійності біопараметричної технології виробники багатьох пристроїв зчитування біопараметричних даних не рекомендують застосовувати ці пристрої для керування доступом до корпоративних мереж або важливої фінансовій інформації. Засіб для зчитування біопараметричних даних по своїй природі однофакторний пристрій ідентифікації; він базується на тому, що біопараметричні дані кожної людини унікальні. Деякі системи біопараметричної автентифікації вимагають введення користувачем персонального ідентифікаційного номера з метою скорочення числа помилок типу II.

Якщо ви вирішите скористатися пристроями зчитування біопараметричних даних, вам потрібно буде поряд з ними встановити програмні засоби, використовувані для автентифікації користувачів. Щоб мати можливість працювати із пристроями зчитування біопараметричних даних, необхідно записати біопараметричну інформацію про особи, ідентичність яких буде встановлюватися за допомогою зчитувачей. Часто кожний біопараметричний параметр зчитується неодноразово, а іноді скануванню піддається кілька ділянок тіла (наприклад, вказівні пальці лівої й правої руки). Потім ці записані біопараметричні дані асоціюються з обліковим записом користувача. Щораз по пред'явленні біопараметричних даних пристрій зчитує їх і зіставляє результуюче унікальне значення зі значеннями, асоційованими з обліковими записами користувачів. У такий спосіб виявляється користувач, що представив свої біопараметричні дані.

По великому рахунку призначення пристроїв зчитування біопараметричних даних полягає в тому, щоб звести до мінімуму обсяг облікових даних, які повинні тримати в пам'яті користувачі, а не в тому, щоб взагалі усунути необхідність у використанні облікових даних.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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 багатооконного 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, чим 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 багатооконний 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 багатооконні 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-орієнтованного 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 м істить 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.

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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи біометричного доступу до даних у хмарному сервісі.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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 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 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

Автентифікація на основі біопараметричних параметрів

Розглянемо коротко використання біопараметричних параметрів у процесі автентифікаціи.

Автентифікація за формою руки

Спосіб заснований на аналізі зображення кисті. Цей спосіб досить широко розповсюджений, але не дуже надійний. Це зв'язано, у першу чергу, з великою варіабельністю форми кисті, як протягом життя людини, так і у відносно короткий термін. Деяким плюсом може служити малий розмір математичного "опису" кисті. В HandKey-системах довжина "опису" становить порядку десятків байт, що служить додатковим підтвердженням малої надійності методу, тому що інформації явно недостатньо для охвату всіх можливих "варіантів" форми кисті, якими природа наділили людей.

Автентифікація за формою особи

Спосіб заснований на аналізі великої кількості параметрів, таких як колір, форма, контраст, риси й т.д. Системи подібного роду в цей час мають недостатню надійність розпізнавання через велику чутливість до освітленості й ракурсу особи під час уведення параметрів ідентифікації.

Автентифікація по термограммі особи

Спосіб заснований на результатах досліджень, які показали, що термограмма, тобто схема розташування кровоносних посудин особи (вени й артерії), є унікальною для кожної людини. Для введення характеристики використовується спеціально розроблена інфрачервона камера. Система дозволяє провести ідентифікацію навіть у випадку, коли людина перебуває на іншому кінці неосвітленої кімнати. Температура тіла, охолодження шкіри особи в

морозну погоду, природне старіння організму людини, використання спеціальних масок, проведення пластичних операцій не впливають на точність термограмми. Якість розпізнавання невисока, поширення метод одержав невелике.

Автентифікація по голосу

Надійна ідентифікація людини по голосу поки залишається нерозв'язною проблемою. Основна проблема полягає у великій розмаїтості проявів голосу однієї людини: голос може мінятися залежно від настрою, стану здоров'я, віку й т.д. Ця розмаїтість визначає серйозні труднощі при виділенні індивідуальних властивостей голосу людини. Крім того, облік шумового компонента є ще однією серйозною й не до кінця вирішеною проблемою в практичному застосуванні ідентифікації по голосу. Надійність розпізнавання не висока – помилка пропустити чужого змінюється в межах відсотків – часток відсотків.

Автентифікація по підпису людини

Ідентифікація людини по його підпису – середньо надійний метод біопараметричної ідентифікації особистості. Широкого поширення поки не одержав, хоча в банківських системах використовується часто.

Автентифікація по райдужній оболонці ока

Цей метод має високий ступінь надійності, однак, відповідні пристрої введення зображення райдужної оболонки ока мають поки досить високу вартість. Вдобавок – існують проблеми, пов'язані із процедурою сканування. Пристрій сканування, фактично, є високоякісною телекамерою, що визначає деякі незручності, пов'язані з камерою й властиво процедурою сканування.

Автентифікація по рисунку папіллярного візерунка пальця

По надійності відповідних процедур ідентифікації метод зіставимо, а в деяких випадках забезпечує якість вище попереднього. Відповідні пристрої введення мають більше низьку вартість. Папіллярный візерунок пальця людини є унікальною біопараметричною характеристикою: в усьому світі немає двох різних пальців з тим самим візерунком.

Автентифікація по фрагментах генетичного коду

Жодна з перерахованих вище персональних характеристик людини не може зрівнятися по надійності розпізнавання з генетичним кодом людини. Однак практичні способи ідентифікації, що використовують унікальні особливості фрагментів генетичного коду, у цей час застосовуються рідко через їхню складність і високу вартість.

Перспективи використання

У цей час все більша увага приділяється розвитку й удосконалюванню технологій, що використовують як біопараметричний параметр відбитки пальців і рисунок райдужної оболонки ока, як найбільш перспективні в змісті мінімізації помилок розпізнавання, добре пророблені. Разом з тим, зовсім ясно, що голос і особа людини також будуть використовуватися для ідентифікації.

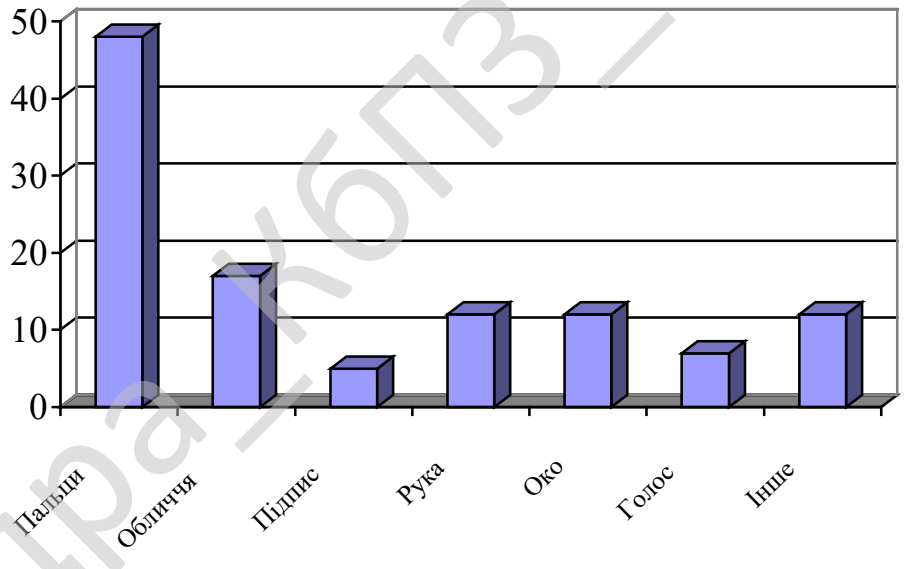


Рисунок 3.1 – Статистика застосування біопараметричних систем

Опис біопараметричної системи

Біопараметрична система – це автоматична система, здатна:

– одержати біопараметричний зразок (наприклад, відбиток пальця) від людини;

– витягти з нього біопараметричні дані (наприклад, особливі точки і їхні параметри);

– зрівняти ці дані з одним або декількома даними такого ж сорту, що зберігаються в базі даних;

– визначити, наскільки добре збігаються пред'явлені дані з якимись даними з бази;

– зробити висновок про те, чи вдалося ідентифікувати людину за пред'явленим даними, або підтвердити (перевірити), що цей саме той, за кого він себе видає.

Оцінка якості роботи біопараметричних систем

Параметри якості

Робота біопараметричної системи описується рядом технічних і цінових параметрів. Важливими технічними параметрами, які позначаються на цінах і можливості використання для того або іншого застосування є наступні.

FAR (False Acceptance Rate) – помилка (імовірність) прийняти "чужого" за "свого"/помилка пропустити "чужого". У системах, що присутні на ринку, ця помилка в основному змінюється в межах від 10^{-3} до 10^{-6}, хоча є рішення й з FAR= 10^{-9}.

FRR (False Rejection Rate) – помилка (імовірність) прийняти "свого" за "чужого"/відмовити в доступі "своєму". Звичайно в комерційних системах ця помилка вибирається рівної приблизно 0,01. У ряді випадків є спеціальні вимоги (при великому потоці, щоб не створювати черг) поліпшення FRR до 0, 001-0,0001. У системах, що є присутні на ринку, FRR звичайно перебуває в діапазоні 0, 025-0,01.

Варто помітити, що у всіх алгоритмах FAR і FRR зв'язані: чим краще один параметр, тим гірше інший. Тому системи з малим FAR гарні ще й тим, що, погіршуючи цей параметр, але залишаючи його ще досить гарним, ми можемо поліпшити FRR.

Третій досить важливий параметр – розмір моделі відбитка пальця. Його величина визначає швидкість порівнянь і обсяг пам'яті, необхідної для зберіганні бази відбитків (обсяг інформації про відбиток пальця може бути на 3-4 порядки більше, ніж у моделі). У більшості систем зберігаються саме моделі й тільки в міліцейських системах зберігаються зображення відбитків пальців.

Наступний параметр – швидкість порівняння інформації, що зберігається в базі, із пред'явленим для контролю відбитком пальця. Це параметр, обумовлений алгоритмом, розміром моделі й використовуваних обчислювальних потужностей. Велике значення має при значних обсягах баз.

Варто враховувати також швидкість зчитування відбитка пальця. Вона може становити від 0,1 до 2 секунд і визначається в основному типом сканера, засобами боротьби з муляжами, а також конструкцією сканера.

Ще один важливий параметр – можливість розпізнати муляж (копію відбитка пальця). Деякі групи сканерів завдяки використовуваній технології відразу відкидають деякі види муляжів, наприклад, ємнісні сканери відбитків пальців або сканери із сенсорами тиску відразу відкидають фотомуляж, тоді як самі примітивні оптичні сканери його пропускають. Проте розроблено й використовується для різних типів сканерів досить багато систем, що розпізнають муляжі.

Також важливо, які інтерфейси використовуються при передачі зображення від сканера в обчислювальну систему (це визначає загальну швидкодію системи), які засоби захисту використовуються при передачі біопараметричної інформації до обчислювального пристрою, що приймає рішення про дозвіл доступу.

Можуть бути названі й інші параметри – розміри, вага сканерів, їхня тривалість життя, припустимі умови експлуатації й т.п.

Набір параметрів досить великий. Можливості тестування системи в споживача обмежені. Це означає, що для ухвалення рішення про використання

біопараметричних систем доступу доцільне залучення професіоналів для консультацій. І це стає особливо актуальним через швидкий ріст ринку.

Оцінка реальної якості біопараметричних систем

Коли ми читаємо щось на тему біопараметрики в засобах масової інформації й навіть у проспектах біопараметричних фірм, те найчастіше зустрічаємося із двома крайніми точками зору. Одна з них – рівень розвитку біопараметрики дуже високий, вирішені всі проблеми, друга – усе перебуває ще в зародковому стані, працює погано, придумані муляжі, які обманюють всі пристрої. Істина, як завжди, перебуває посередині, але тенденції розвитку позитивні. Повідомлення про те, що деякі системи працюють не так добре, як хотілося б і забезпечують не дуже високий рівень надійності, або пропускають деякі види муляжів по конкретних фактах не позбавлені підстави. При ближчому розгляді найчастіше виявляється, що була обрана більш дешева система, що вміє менше, ніж більш дорога. У принципі кращі алгоритми вже забезпечують рівні помилок:

– "пропустити чужого" (FAR) – не більш ніж 1 на мільярд;

– "не пропустити свого" (FRR) – не менш чим 1 на 1000.

Проблеми біопараметричних систем

Проблема №1

Стійкість значень деяких біопараметричних характеристик часто перебільшується. У реальній роботі система (наприклад, класу "Handkey") досить часто "не довідається" зареєстрованого користувача.

Крім помилок FAR і FRR існує помилка третього роду, коли приймається рішення "чужий" через неможливість одержати спостереження обраної біопараметричної характеристики. Наприклад, дактилоскопічний сканер не може зняти зображення через які-небудь недоліки шкіри, дуже малої кількості особливих точок або відсутності папіллярного візерунка.

На простому персональному комп'ютері сучасні програмні засоби біопараметричних систем дозволяють робити порівняння більше 10 000 відбитків

у секунду, є технічні рішення, які виявляють будь -які придумані на сьогодні муляжі біопараметричних параметрів. Однак, якщо ми хочемо мати все, то коштує це поки що не дуже дешево.

Проблема №2

Для оцінювання співвідношень FAR/FRR проводять велику кількість випробувань системи з використанням спеціальних баз даних біопараметричних характеристик, знятих у різний час. На жаль, це робиться не всіма виробниками біопараметричних систем. Через цього ефективність біопараметричних систем, що вже існують на ринку, часто реально менш висока, чим це анонсується.

Для тестування дактилоскопічних систем є стандартні бази відбитків пальців NIST-9 і NIST-14, є також бази даних відбитків, знятих за допомогою різних ємнісних і оптичних сенсорів.

Незважаючи на "хвороби росту", уже зараз можна сказати, що, якщо в людини існує досить гарний біопараметричний параметр, тобто системи, які можуть забезпечити FAR на рівні 10^{-9}, FRR – $10^{-2(-3)}$ і не пропустити жоден з наявних муляжів. Це не дуже дешево, але є. На жаль, для біопараметрики біля відсотка людей не мають деякі біопараметричні характеристики. У людей з більмом немає рисунка райдужної оболонки, у деяких людей відбитки пальці майже не мають характерних рис, що стримує поширення цих технологій.

Проблема №3

Існує ряд проблем, пов'язаних з невірним розумінням технічних параметрів пристроїв. Можна побачити твердження, що розмір сенсора 12х12мм цілком достатній для коректного зняття відбитка пальця, а деякі фірми рекламують сенсори розміром 9х9мм. Однак, більша частина напівпровідникових сенсорів насправді має недостатню площу сканування, що погіршує якість біопараметричної системи. Часто виготовлювачі сенсорів говорять про дозвіл, маючи на увазі насправді кількість чутливих елементів матриці.

Методи розпізнавання відбитків пальців

Метод виявлення ключових точок

Кожний відбиток пальця складається з певної кількості борозен і смужок. Смуги – це підняті частини шкірного покриву, борозни – впалі частини. Смуги становлять так звані ключові точки: **край смуг** (там, де смуги закінчуються) і **роздвоєння** – там, де вони розгалужуються.

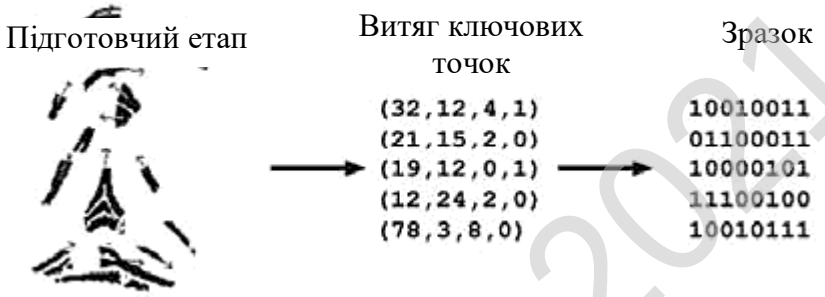


Рисунок 3.2 – Реєстрація відбитка по набору ключових точок

Під час реєстрації ключові точки розташовуються в певному місці (рис. 3.2), а їхнє розташування відносно один одного і їхній напрямок реєструються. На основі цих даних створюється **зразок** (шаблон) – інформація, що згодом буде використана для посвідчення особи користувача.

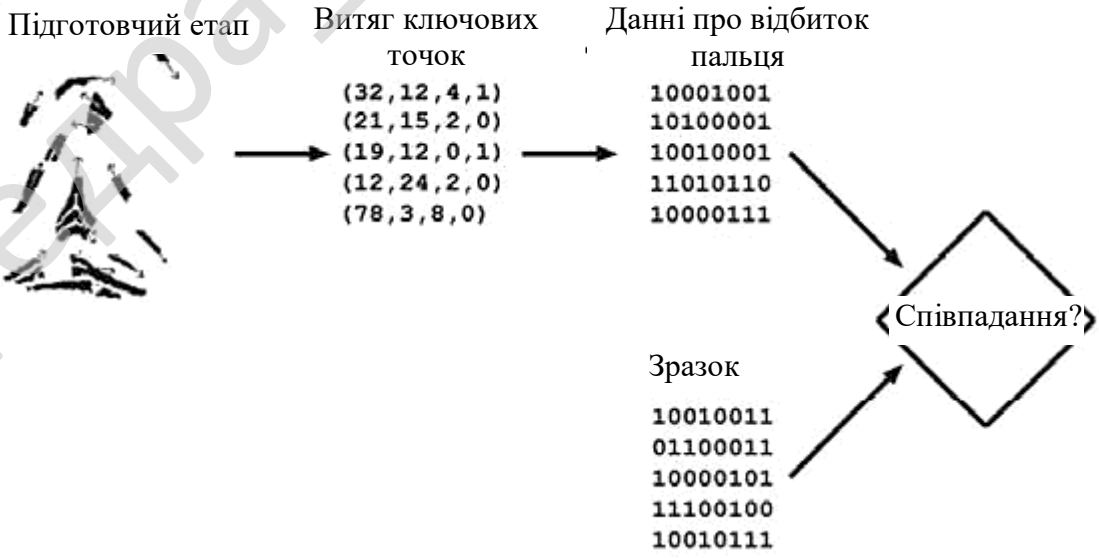


Рисунок 3.3 – Перевірка з використанням ключових точок

На етапі зіставлення (рис. 3.3), облічене зображення відбитка пальця піддається попередній обробці, у ході якої витягають ключові точки. Вони зіставляються із зареєстрованим зразком, намагаючись розташувати в певному місці як можна більша кількість схожих то чок у межах заданих границь. Результатом зіставлення, як правило, є набір ключових точок. Потім використовується поріг, що визначає, наскільки більшим повинне бути це число, щоб було можливо зіставити відбиток пальця зі зразком.

Переваги:

– Використовується в додатках AFIS;

– Широко відомий, добре досліджений метод;

– Алгоритм підходить для множинного зіставлення.

Недоліки:

- В зв’язку з тим, що метод висуває більші вимоги до дозволу й розмірів чутливого датчика, він може бути використаний не у всіх технологіях, що зчитують відбитки пальців. При використанні сканерів, менш зроблених, чим апаратура AFIS-класу, дає низькі результати;

- Люди, що не мають зовсім або які мають невелику кількість ключових точок (особливий стан шкірного покриву) не можуть користуватися даною системою. Кількість ключових точок може бути обмежуючим фактором для безпеки алгоритму;

- Можливі збої в системі через помилкові ключові точки (ділянку, що містить помилку, що виникла через низьку якість реєстрації, відтворення зображення або нечіткого відбитка смуг).

Метод зіставлення візерунків

Важливою властивістю алгоритму зіставлення візерунків є те, що в увагу приймаються не тільки окремо взяті точки, але й більші характеристики відбитка пальця. Ці характеристики можуть також включати певний відсоток додаткових даних, включаючи товщину смуг, їхню кривизну або щільність. У зв'язку із цією кількістю даних, що збільшилася, алгоритм, заснований на зіставленні візерунків,

менш залежить від величини сканера й абсолютно не залежить від кількості ключових точок у відбитку пальця. Заснований на зіставленні візерунків алгоритм, на відміну від методу виділення ключових точок, не зустрічає складностей при розпізнаванні пальця з відбитком гіршої якості.

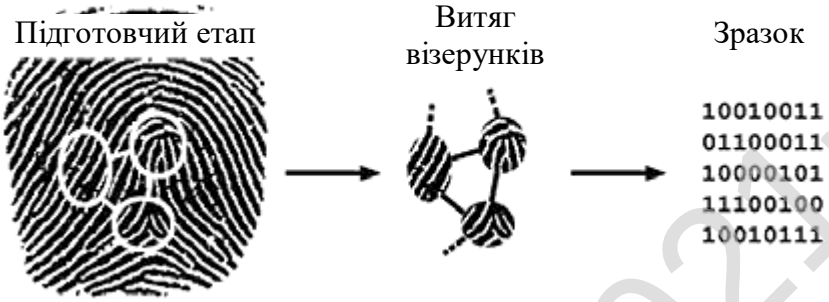


Рисунок 3.4 – Реєстрація відбитка за допомогою алгоритму зіставлення візерунків

Запатентований алгоритм зіставлення візерунків Precise Biometrics під час реєстрації відбитка визначає наявність вищезгаданих додаткових характеристик. Невеликі ділянки відбитка й відстань між ними витягають із загальної картини (рисунок 3.4) з метою максимально збільшити кількість унікальної інформації. Найбільш значимі ділянки навколо ключових точок і ділянки з невеликим радіусом вигину. Основна структура й унікальні комбінації смуг також є коштовними даними.

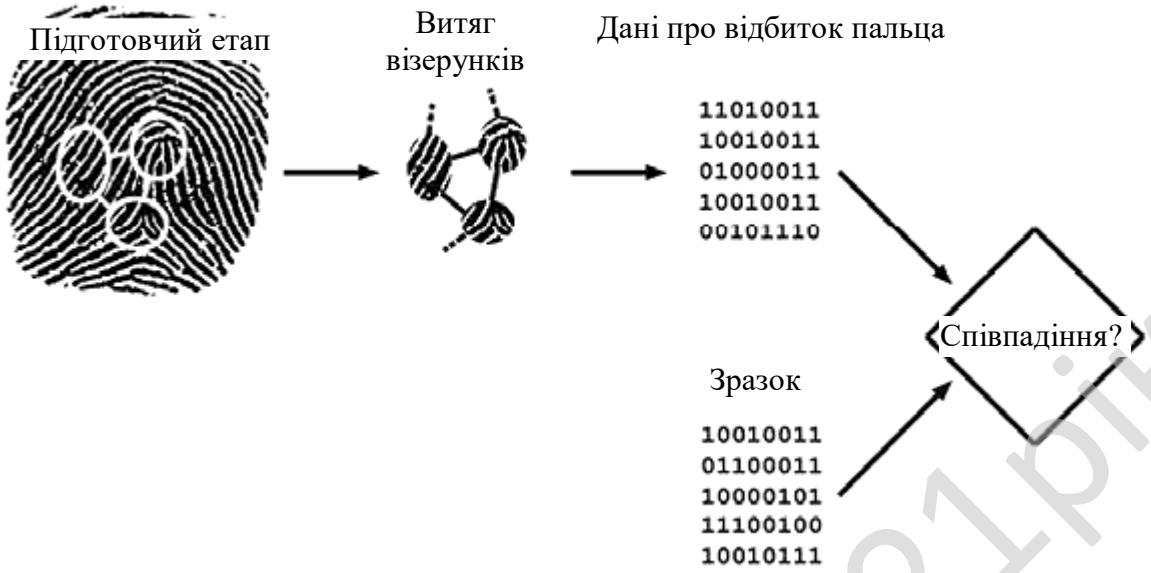


Рисунок 3.5 – Перевірка з використанням алгоритму зіставлення візерунків

Процес підтвердження (рисунок 3.5) починається з попередньої обробки ліченого зображення відбитка. Зареєстровані візерунки, що зберігаються в зразку, зіставляються з перевіряється изображением, що, відбитка, щоб визначити, наскільки зразок збігається із зображенням. Поріг, що описує найменше припустиме відхилення згодом використовується при визначенні ступеня відповідності відбитка наявному зразку.

Переваги:

– Прекрасно працює з усіма відомими типами сканерів відбитків пальців;

– Будь-який відбиток, якому можна записати, може бути зареєстрований, навіть якщо він не має або має невелику кількість ключових точок;

– Прекрасно підходить для здійснення роботи з недостатньою кількістю обчислювальних ресурсів, наприклад, смарт-картою.

Недоліки:

– Не може використовувати базу даних AFIS (однак, може використовувати неопрацьовані зображення);

– Не оптимізований для ідентифікації (тобто для встановлення конкретної особистості, для схеми «один до багатьох»).

3.2 Розробка структурної схеми

Для перевірки коректності реалізації даної задачі було виконано багато розрахунків та експериментальних матеріалів. Цьому питанню приділялась особлива увага тому, що помилка при розрахунку привела б до ряду негативних наслідків. Відлагодження та перевірка, що підтверджує вірність програмних рішень відбувалась за декількаами етапами:

– математична перевірка окремих модулів;

– математична перевірка всієї системи (з допомогою математичної логіки будується логічна схема всієї системи);

– практична перевірка підпрограм (перевіряється процедурна частина кожної підпрограми окремо);

– практична перевірка всієї системи у дії (перевіряється ситема в цілому за допомогою вводу різних даних у програму, потім на виході з програми перевіряємо отриману інформацію з очікуваною).

Для підтвердження правильності розрахунку програми були використані експериментальні дані різних форматів, були проведені консультації з даного питання зі спеціалістами.

Простота мови проектування та маніпулювання даними, зручність спілкування користувача з системою до мінімума вивчення цієї програми. Користувач програми – це людина, яка повинна володіти азами програмування. При написанні програми я намагалася, щоб програма відповідала наступним параметрам:

– Швидкодія. Програма працює постійно з великою кількістю кінцевих абонентів (селективний зв’язок).

– Захист. Забезпечити надійний захишенний канал зв`язку.

– Відсутність проблеми дорожнечі сучасних персональних комп’ютерів. Система, що написана може встановлюватись на будь-якому персональному комп’ютері – використовувати відносно швидкі алгоритми захисту зв'язку.

– Можливість зручно і швидко формувати приклади і теорію для користувача.

– Можливість звертання до системних ресурсів. Користувача системи цікавить її інформаційний та сенсовий зміст. Подробиці організації фізичного зберігання даних його не цікавлять.

Перш за все перед розробкою системи слід одержати уявлення на слідуючи моменти:

– на які частини можна розбити систему;

– одержати уявлення про кожну частину (фрагмент);

– яка інформація і з якою де тальністю необхідна користувачу кожного фрагменту;

– які процеси передачі і обробки даних знаходяться в кожному фрагменті;

– технологія накопичування і обробки аудіо інформації системи;

– на якому обладнанні планується реалізувати систему;

– технологія функціонуваня системи;

– чи необхідна адаптація і настройка системи при змінах деяких умов.

Для розробки програми були попередньо розроблені структурна схема роботи системи, структурна схема формату передачі даних, функціональна схема роботи системи, діаграма процесів, а також блок-схеми алгоритму програми, розглянемо їх детально.

Створена автором система біометричного доступу до даних у хмарному сервісі складається із двох частин – клієнтської частини на стороні користувача й серверної частини що перебуває віддалено в Інтернеті. Створена система універсальна. Застосовуючи стандартизовані бібліотеки Bioparametrs Aplication Programming Interface і використовуючи стандартизований формат передачі

повідомлень (розділ 3.8) користувач може використовувати будь-які біопараметричні сканери. При створенні й тестуванні системи автор використовував два біопареметричні сканери, а саме – сканер відбитків пальців MorphoSmart 1300 Sagem для розпізнання папілярного малюнка пальця й планшетний сканер SignatureGem 1x5 розпізнавання підпису людини.

Клієнтська частина виконана у вигляді Active X додатка з застосуванням стандартизованої бібліотеки Bioparametrs Aplication Programming Interface 1.1 і застосовна з Internet Explorer версії 4.0 і вище, а також у будь-яких інших програмах, що підтримують роботу з Active X компонентами.

Робота програмного забезпечення із застосуванням стандартизованих бібліотек Bioparametrs Aplication Programming Interface показана на рисунку 3.6.

Серверна частина виконана у вигляді модуля COM і повністю сумісна з IIS сервером із застосуванням стандартної бази даних від корпорації Майкрософт – Microsoft Access з інтерфейсом ODBC.

Для подальшого розуміння роботи розробленої системи введемо наступну термінологію.

Навчання – процедура зняття серії відбитків того самого пальця або інших біопараметричних характеристик (підпис людини, форма руки, і.т.буд.) з метою відбору деякої їхньої кількості з найбільше яскраво вираженими особливостями, чим ці особливості більше виділені тим краще **якість зразка** поняття прямо пов'язане із застосовуваними апаратними засобами зняття біопараметричних характеристик. У деяких випадках зразок непридатний для розпізнавання через різні причини (наприклад у випадки зняття відбитків пальців забруднення).

Паспорт – еталонний зразок (перевірений 100% зразок) для ідентифікації особи.

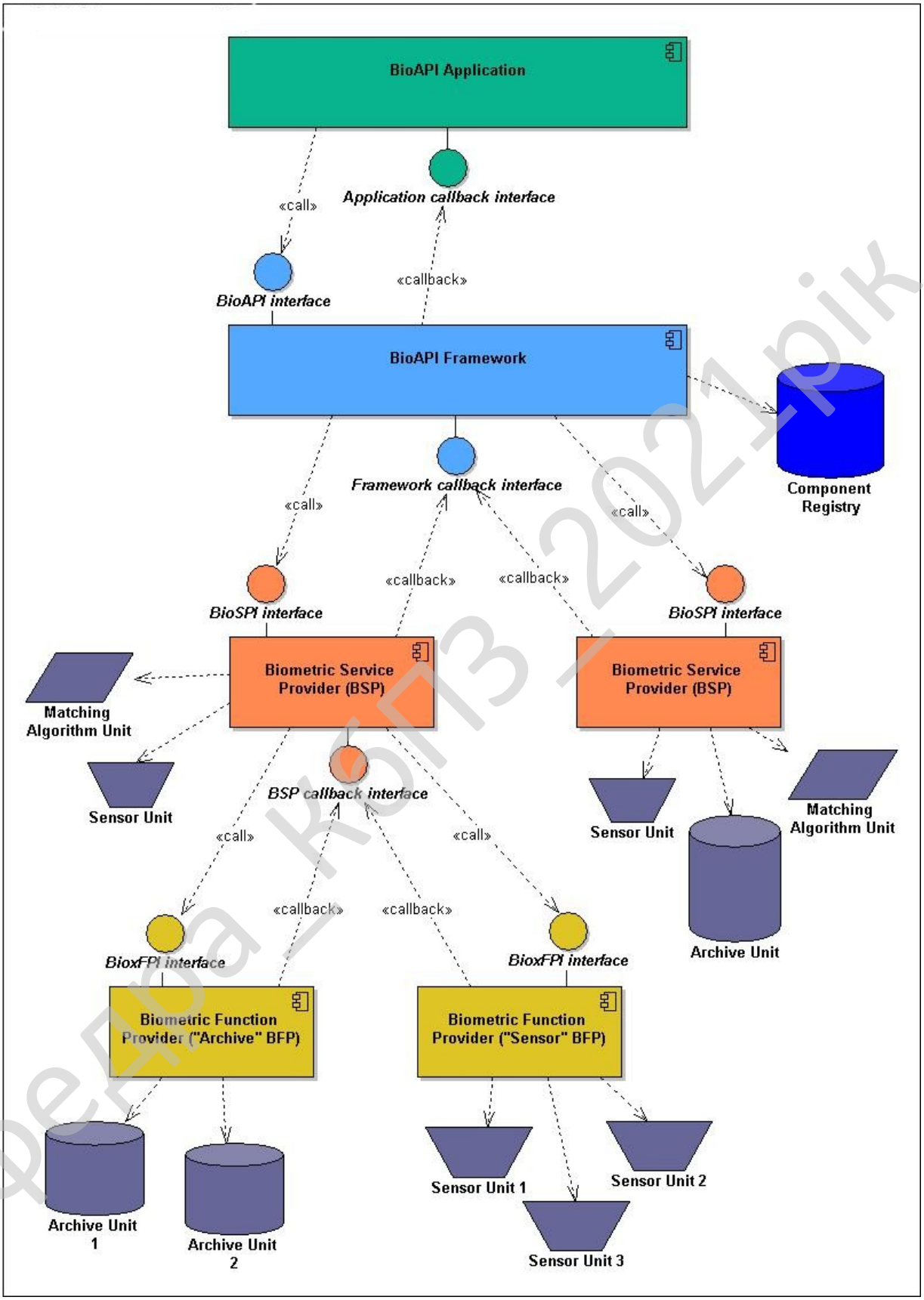


Рисунок 3.6 – Робота та взаємодія ПЗ з Bioparametrs Aplication Programming Interface

Модель – математичний образ (пакет даних) відбитка пальця, одержуваного з біопараметричного сканера, а також **біометричний ідентифікаційний запис** і **криптографічне додавання**. Модель необхідний запис для **розпізнавання** користувача й надання йому прав.

Біометричний ідентифікаційний запис (BIR, BSP) – група даних з **моделі** відповідальних за біопараметричні характеристики (використовуваний сканер, якість зразка, опис тип даних і т.д.).

Криптографічне додавання – група даних з **моделі** відповідальних за автентифікацію.

Розпізнавання – процедура порівняння Моделі й Паспорти, результатом якої є вивід про їхню ідентичність і видачі коду перевірки.

Розглянемо докладніше особливості реалізації й взаємодії клієнтської частини із серверної зображеної на рисунку 3.7.

Розглянемо схему знизу нагору, при одержанні з біопараметричного сканера даних на стороні клієнта розроблений Active X компонент використовуючи бібліотеку BIOAPI перевіряє якість отриманого зразка й формує біометричний ідентифікаційний запис (розділ 3.8) складається з 32 кілобайт інформації. Далі відбувається додавання криптографічного додавання (40 Кб) підтверджувального, що цей запис є записом користувача, формат запису стандартизований на території України. Після формування біометричного ідентифікаційного запису й криптографічного додавання відбувається остаточне формування моделі для перевірки користувача.

Зі сторінки доступу модель передається через Інтернет на серверну частину системи де обробляється скриптом доступу й надходить в DCOM модуль автентифікації де відбувається перевірка на відповідність. При позитивному проходженні перевірки й добуванні біометричного ідентифікаційного запису. DCOM модуль витягає з Microsoft Access паспорт користувача. Відбувається біопараметричне зіставлення даних і повернення клієнтові коду результату.

Розглянемо необхідність застосування технологій компонента Active X і COM модулів при розробці клієнтської й серверної частини системи.

Як відоме написання Active компонент і COM на основі моделі компонентних об'єктів дозволяє вбудовувати розроблені програми в різні джерела, такі як HTML сторінки використовуючи JavaScrpt, PHP скріпти, Pearl скріпти й т.ін., що приводить до універсальності. Виходячи з поставленого завдання застосування даної технології доцільно. Знаючи всього CLSID Active X компонента, ми одержуємо волю вибору в застосуванні в різних джерелах.

CLSID розробленого Active X об'єкта – "**CLSID:7D6416 BB-4417-4B 92-B564-A39A474DC84E**" (він унікальний)

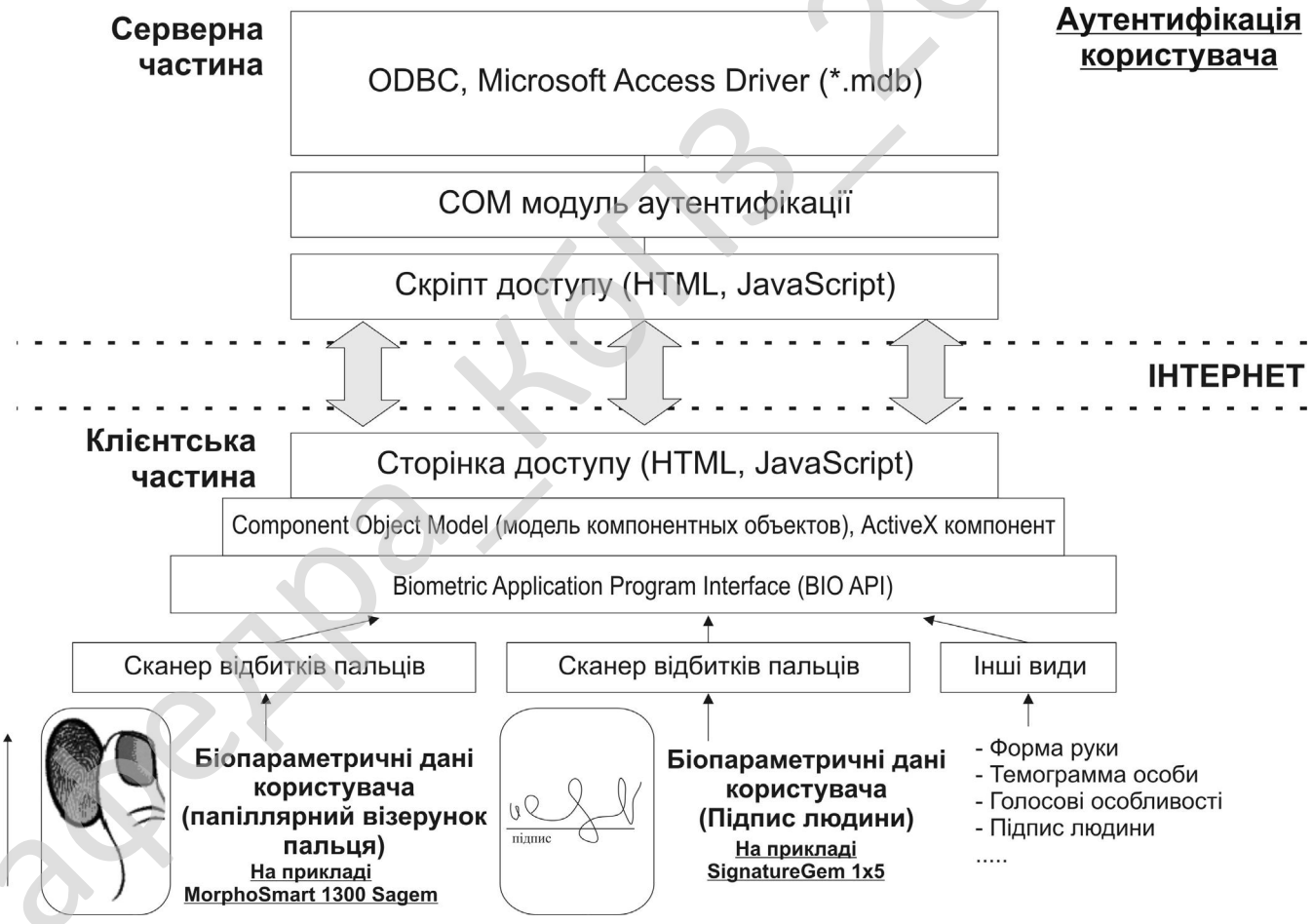


Рисунок 3.7 – Структурна схема системи

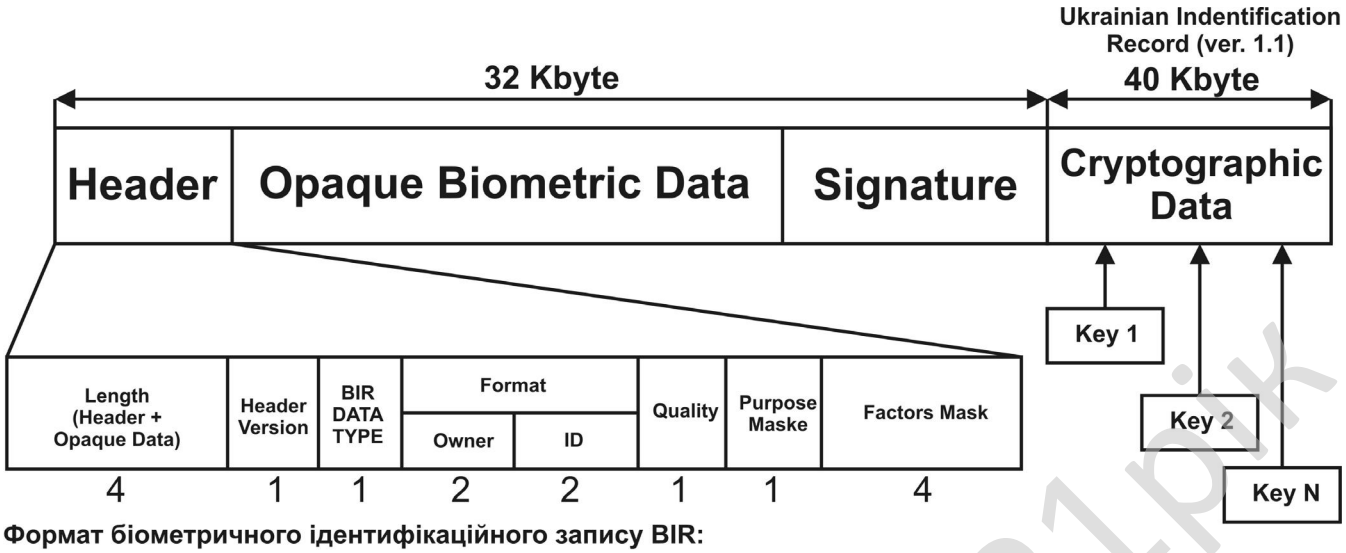
Для функціонування серверної частини системи, побудованої із застосуванням COM модуля, не буде потрібно здобувати ніяких додаткових апаратних засобів. Це дуже зручно для проектів з невеликою клієнтською базою. У випадку більших вимог до швидкості й обсягу розпізнавання в передбачені технології нарощування продуктивності із застосуванням програмних засобів від Майкрософт. Безсумнівний плюс такого підходу – можливість поступового нарощування продуктивності системи в міру росту клієнтської бази, чим забезпечується оптимізація й мінімізація витрат на експлуатацію системи.

Серверна частина через вибір COM модуля підтримує всі існуючі на даний момент технології масштабування й вирівнювання навантаження додатків від Майкрософт: Network Load Balancing, Component Load Balancing, Application Center, Object Pooling, JIT activation.

Розглянемо формат передачі даних, які складаються з біометричного ідентифікаційного запису й українського криптографічного додавання, зображених на рисунку 3.8.

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, Росії, 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 3.8 – 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:

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

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

3. реєстрація;

4. реєстрація тільки для верифікації;

5. реєстрація тільки для ідентифікації;

6. 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:

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

2. **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;

3. **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.

3.3 Розробка функціональної схеми

Розглянемо поетапно функціональну схему (рисунок 3.9) як і вищеописану структурну схему. Функціональна схема – це схема, що описує саму суть роботи пристрою, програми, взаємодії й має більш узагальнений рівень, ніж структурна.

Рисунок 3.9 – Функціональна схема роботи системи

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

Як було розглянуто на структурній схемі, 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.

3.4 Розробка діаграми процесів

Одним з важливих критеріїв при розробці будь-якого програмного забезпечення це грамотна розробка структури роботи системи потоків і процесів.

Розглянемо діаграми процесів, рисунок 3.10. На діаграмі процесів можна точно зрозуміти, як працює ПЗ у цілому.

Починається й закінчується програма в першому блоці, які є основною точкою звіту в діаграмі, при переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їхнього входження друг у друга.

Чітко простежується те, що в серверній частині без обробки ідентифікаційного запису не буде задіяний внутрішній буфер порівняння завантаженої моделі, як і інші інші блоки обробки, що перебувають нижче блоку, ідентифікаційного запису. Також видно як у клієнтській частині впливає блок обробки біопараметричних запитів.

Рисунок 3.10 – Діаграма процесів

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

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. 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’єкно-орієнтована, проект, що розробляється, вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулям: обробки біопараметричних запитів; обробки і формуванню ідентифікаційного додатку; обробки помилок; формування даних перевіряемої моделі; перевірки якості біопараметричних записів.

Створення повноцінного коду програмного продукту з розробленою ціновою політикою (розділ 5) і системою знижок покупцям програмної продукції.

На рисунку 4.1 зображена серверна частина програми. При детальному розгляді програма розбита на дещо важливих блоків:

Розглянемо складові елементи блок-схеми та їхній опис**:**

Блок схема серверної частини ПЗ.

Початок (термінатор)

Початковий блок, вхід із зовнішнього середовища операційної системи. Розглянемо місце встановлення і особливості застосування серверної програми. Програма встановлена на серверній частині розробленої системи. Програма написана COM модулем за технологією моделі компонентних об'єктів і дозволяє вбудовувати розроблену серверну програму в різні джерела. У нашому випадку

програма взаємодіє з WEB сервером фірми Microsoft – IIS сервер (Інформаційний Інтернет-Сервер).

Виділення пам`яті (символ процесу)

Первісний блок у будь-якій блок-схемі, що вказує на початкову стадію запуску програми – виділення пам'яті програмі операційною системою.

Ініціалізація початкових змінних (символ процесу)

Ініціалізація початкових значень використовуваємих змінних, масивів, динамічних структур, динамічно створюваних класів, записів і.т.і.

Підключення файлу налагодження ПЗ (символ процесу)

Підключення файлу настроювання програмного забезпечення, де перебуває критично важлива інформація про параметри запуску програми, початкового положення вікна програми, можливі записи здійснення необхідних дій, ліцензійні угоди, стан попереднього вимикання програми, статистика роботи й багато інших даних, що формуються при роботі програмного забезпечення.

Перевірка доступу до БД (символ процесу)

При використанні у програмі бази даних критично важливо знати при початковій взаємодії з нею, її поточний стан (включена, виключена, перебуває у невизначеному стані і.т.і.). Можливі не діагностовані й непередбачені наслідки при доступі до бази даних без первісної перевірки (залежно від використовуваної БД – mySQL, Access, Interbase і.т.буд.)

Доступ до БД є? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки доступу до БД.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід по мітці М3 до завершення роботи ПЗ.

Виведення повідомлення, перехід у режим обробки моделей (введення/виведення даних).

Після успішної перевірки доступу до БД відбувається перехід у режим обробки моделей (робочий режим).

Є запит обробки біопараметричної моделі клієнта? (функція перемикаючого типу)

Входи.

Вхід – Очікування дій користувача чі системного сповіщення модулю COM з біопараметричною моделлю від перевіряємого користувача.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід у кінець блок-схеми та повернення за міткою М2 до початку цього блока (цикл очікування дій від користувача).

Завантаження біометричної моделі (символ процесу)

Розбір моделі що надійшла, виділення блоків ідентифікаційного додатку та біопараметричного запису.

Перевірка ідентифікаційного додатку (символ процесу)

Перевірка ідентифікаційного додатку на відповідність.

Перевірка пройдена? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки ідентифікаційного додатку на відповідність.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід у кінець блок-схеми та повернення за міткою М2 до початку цього блока (цикл очікування дій від користувача).

Запит відповідного паспорта з БД та порівняння

З'єднання з БД. Завантаження паспорта. Порівняння завантаженого еталонного паспорта із присланою перевіряємою моделлю.

Модель відповідна? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки відповідності моделі.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід у кінець блок-схеми та повернення за міткою М2 до початку цього блока (цикл очікування дій від користувача).

Занесення даних у БД, повернення коду біопараметричної перевірки (символ процесу)

При успішній перевірці паспорта з моделлю у БД заноситься запис про вхід користувача, також користувачеві вертається код доступу до об'єкта, у нашому випадку це доступ на сайт.

Виведення системних повідомлень (введення/виведення даних)

Виведення повідомлень про модель, що надійшла, від користувача й результатів біопараметричного порівняння моделі та паспорта.

Вихід з ПЗ? (функція перемикаючого типу)

Входи.

Вхід – Ця функція перемикаючого типу забезпечує цикл роботи ПЗ з відповідністю до об’єктно-орієнтованного програмування в Windows.

Так як програмне забезпечення э об’єктно-орієнтованним та орієнтоване під операційну систему Windows. Для завершення програми необхідно подати системне сповіщення яке може бути подано користувачем натискаючи комбінацію клавіш Alt+F4, нажимаючи мишкою на хрестик та запропонованими другими засобами чі програмістом.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід за міткою М2 до блоку запиту обробки біопараметричної моделі клієнта.

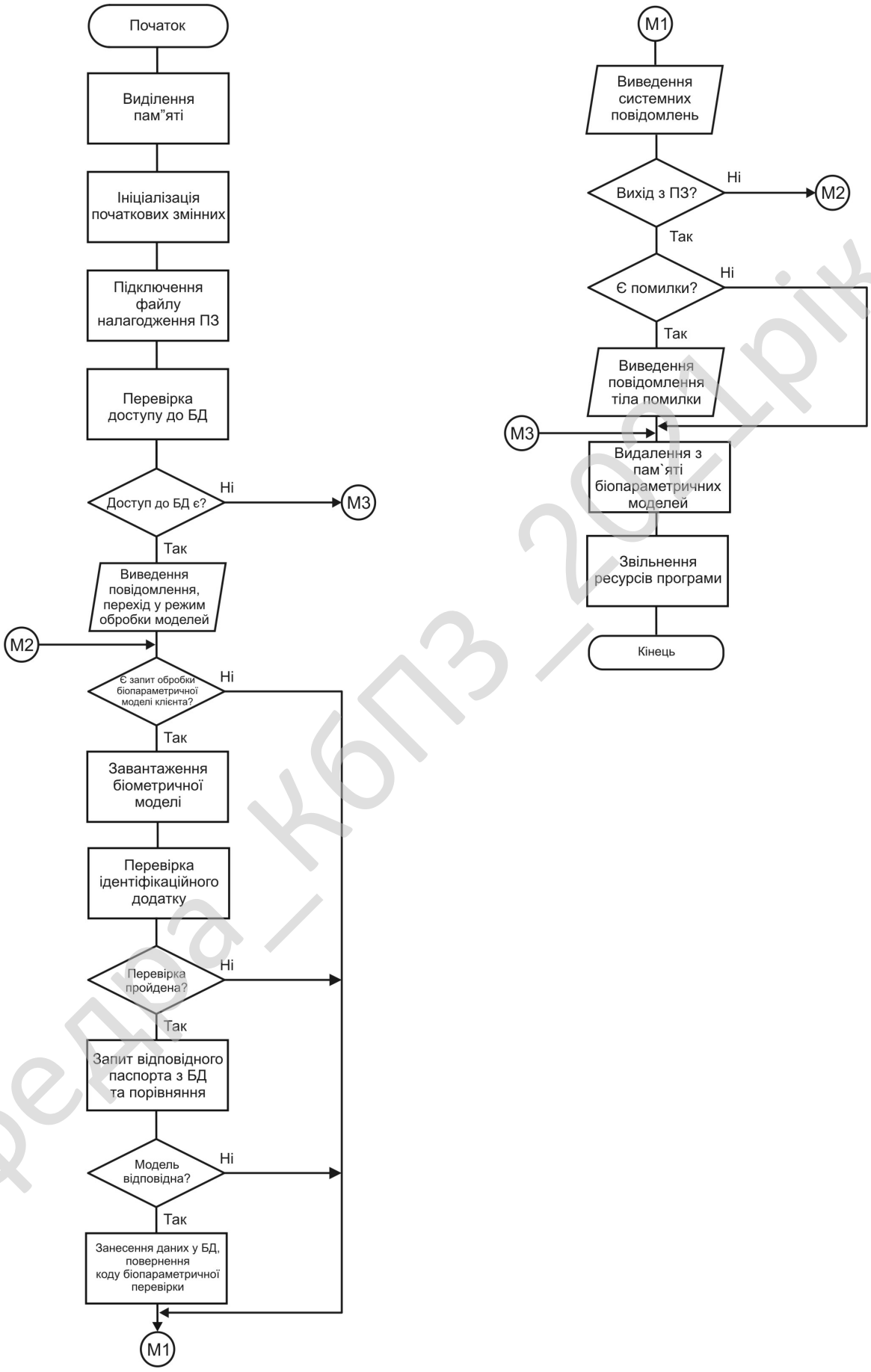


Рисунок 4.1 – Блок схема серверної частини ПЗ

Є помилки? (функція перемикаючого типу)

Входи.

Вхід – Перевірка на виникнення помилок при роботі ПЗ накопичувального типу які виникають при завершенні роботи ПЗ .

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід у блок видалення з пам`яті біопараметричних моделей.

Виведення повідомлення тіла помилки (введення/виведення даних)

Виведення повідомлення з помилкою накопичувального типу.

Видалення з пам`яті біопараметричних моделей (символ процесу)

Видалення з пам`яті моделей які оброблялись.

Звільнення ресурсів програми (символ процесу)

Кінцевий блок у будь-якій блок-схемі, що вказує на к інцеву стадію завершення програми – звільнення ресурсів програми програмі операційною системою.

Кінець (термінатор)

Кінцевий блок. Завершення роботи серверної частини ПЗ.

Блок схема клієнтської частини ПЗ.

Початок (термінатор)

Початковий блок, вхід із зовнішнього середовища операційної системи. Розглянемо місце встановлення і особливості застосування клієнтської програми. Програма встановлена на клієнтській частині у користувачів. Програма написана ActiveX компонентом за технологією моделі компонентних об'єктів. Взаємодіє з сервером через HTML форму за допомогою Javascript. І взаємодіє з стандартизованими бібліотеками Biometric Application Program Interface (BIO API) та формує біопараметричний запис.

Виділення пам`яті (символ процесу)

Первісний блок у будь-якій блок-схемі, що вказує на початкову стадію запуску програми – виділення пам'яті програмі операційною системою.

Ініціалізація початкових змінних (символ процесу)

Ініціалізація початкових значень використовуваємих змінних, масивів, динамічних структур, динамічно створюваних класів, записів і.т.і.

Ініціалізація додаткових бібліотек BioApi (символ процесу)

Підключення та ініціалізація бібліотек Biometric Application Program Interface.

Завантаження початкових значень форми біометричного ідентифікаційного запису (символ процесу)

Підключення файлу настроювання програмного забезпечення, де перебуває критично важлива інформація про біометричний ідентифікаційний запис.

Перевірка наявності Інтернету та апаратного забезпечення (символ процесу)

При використанні у програмі передачі даних за допомогою Інтернет важливо знати при початковій взаємодії, поточний стан.

Перевірка успішна? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки доступу до Інтернет.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід по мітці М3 до завершення роботи ПЗ.

Очікування запиту з апаратного забезпечення (символ процесу)

Очікування дій від апаратного забезпечення – MorphoSmart 1300 Sagem чі SignatureGem 1x5.

Отримані біопараметричні дані? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки очікування дій від апаратного забезпечення.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід на блок очікування запиту з апаратного забезпечення (цикл очікування дій від апаратного забезпечення).

Обробка біопараметричних даних, перевірка якості зразку (символ процесу)

Обробка даних що поступають. Перевірка структури даних на якість.

Якість зразка задовільна? (функція перемикаючого типу)

Входи.

Вхід – Результати перевірки обробки даних.

Виходи.

Вихід так – перехід у наступний блок.

Вихід ні – перехід на блок очікування запиту з апаратного забезпечення (цикл очікування дій від апаратного забезпечення).

Формування біометричного ідентифікаційного запису (символ процесу)

Виділення пам`яті для моделі, заповнення полів BIR (Biometric Identification Record).

Додавання ідентифікаційного додатку, кінцеве формування моделі (символ процесу)

Заповнення полів ідентифікації запису на стороні сервера.

Відправлення моделі (символ процесу)

Формування кінцевих значень та відправлення моделі до серверної частини програмного забезпечення.

Очікування коду повернення (символ процесу)

Очікування дій з сторони сервера (присвоєння коду доступу чи відхилення)

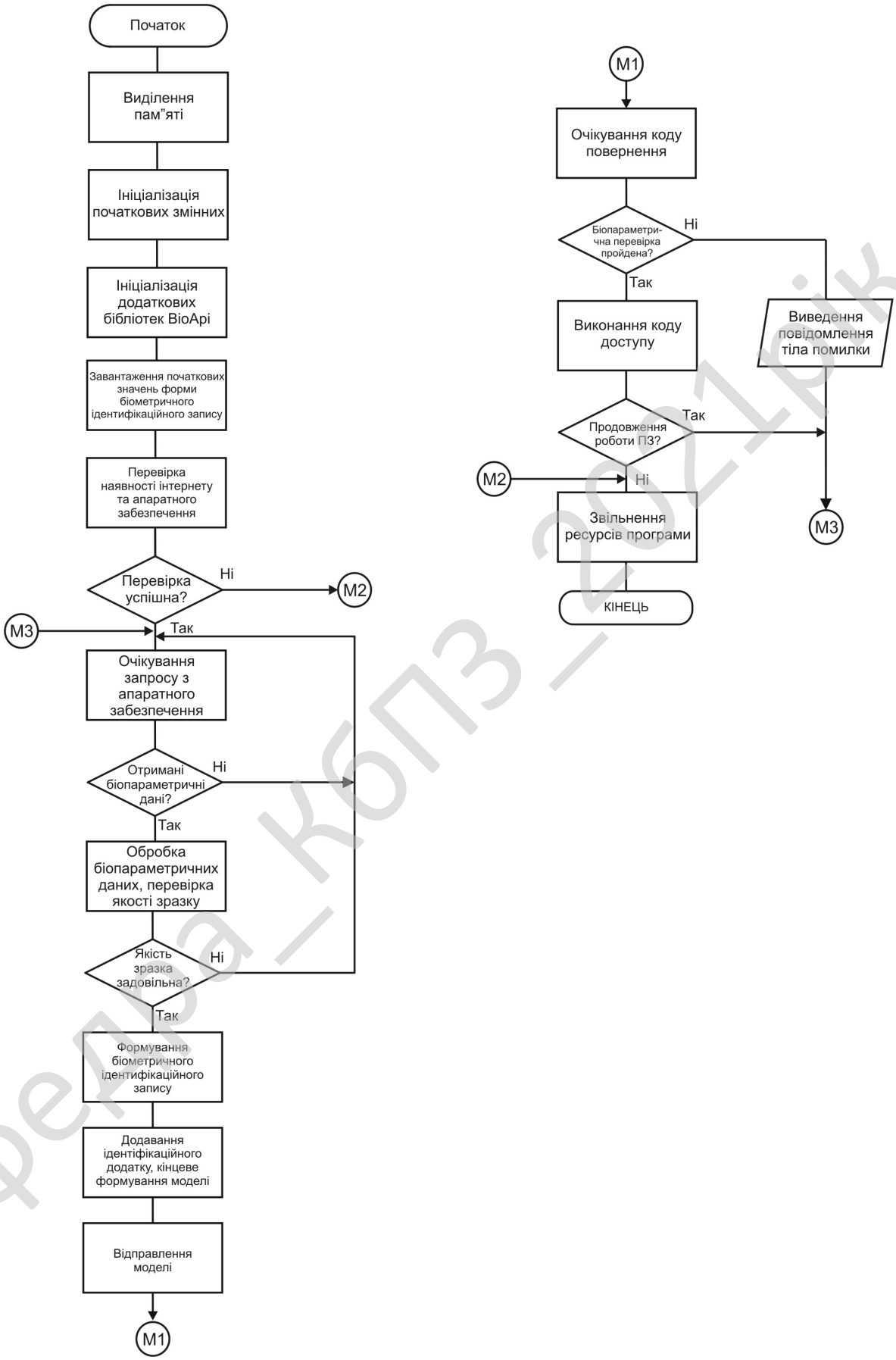


Рисунок 4.2 – Блок схема клієнтської частини ПЗ

Біопараметрична перевірка пройдена? (функція перемикаючого типу)

Входи

Вхід –Результати перевірки. При успішній перевірці паспорта з моделлю на сервері повертається код доступу до сайту.

Виходи

Вихід так – перехід у блок виконання коду доступу.

Вихід ні – перехід на блок виведення повідомлення про помилку.

Виконання коду доступу (символ процесу)

Виконання у ActiveX компоненті дій аутентифікованного користувача.

Продовження роботи ПЗ?

Звільнення ресурсів програми (символ процесу)

Кінцевий блок у будь-якій блок-схемі, що вказує на к інцеву стадію завершення програми – звільнення ресурсів програми програмі операційною системою.

Кінець (термінатор)

Кінцевий блок. Завершення роботи клієнтської частини ПЗ.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Madryga. Алгоритм Madryga складається із двох вкладених циклів. Зовнішній цикл повторюється вісім разів (для гарантії надійності число циклів можна збільшити) і полягає в застосуванні внутрішнього циклу до відкритого тексту. Внутрішній цикл перетворює відкритий текст у шифртекст і виконується однократно над кожним 8-бітовим блоком (байтом) відкритого тексту. Таким чином, весь відкритий текст послідовно вісім разів обробляється алгоритмом.

Ітерація внутрішнього циклу оперує з 3-байтовим вікном даних, називаним робочим кадром (рисунок 4.3). Це вікно зрушується на 1 байт за ітерацію. (При

роботі з останніми 2 байтами дані покладаються циклічно замкнутими). Перші два байти робочого кадру циклічно зрушуються на змінне число позицій, а для останнього байта виконується операція XOR з декількома бітами ключа. У міру переміщення робочого кадру всі байти послідовно циклічно зрушуються й піддаються операції XOR із частинами ключа. Послідовні циклічні зрушення перемішують результати попередніх операцій XOR і циклічного зрушення, причому на циклічне зрушення впливають результати XOR. Завдяки цьому процес у цілому оборотний.

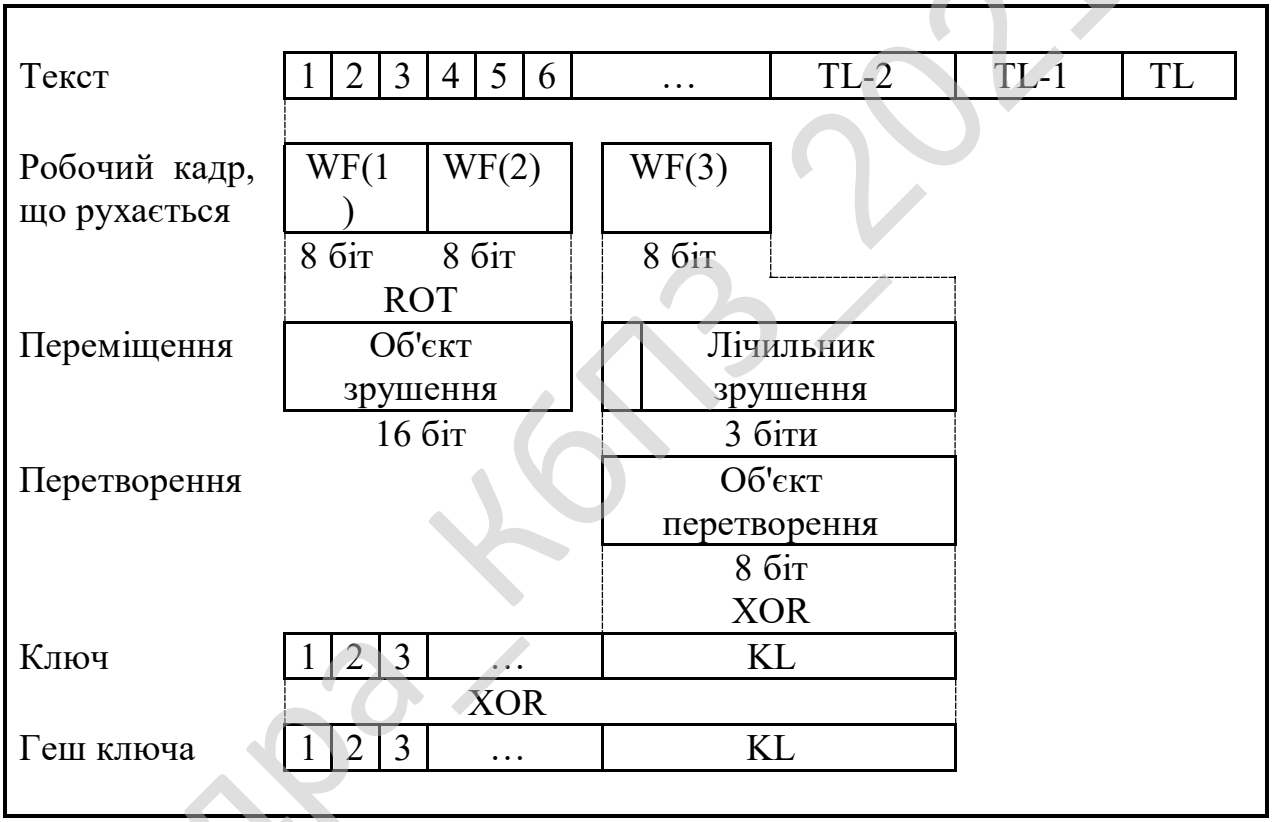


Рисунок 4.3 – Одна ітерація алгоритму Madryga

Оскільки кожний байт даних впливає на два байти ліворуч і на один байт праворуч від себе, після восьми проходів кожний байт шифртексту залежить від 16 байтів ліворуч і 8 байтів праворуч.

При шифруванні кожна ітерація внутрішнього циклу встановлює робочий кадр на передостанній байт відкритого тексту й циклічно переміщає його до

третього з кінця байту відкритого тексту. Спочатку весь ключ піддається операції XOR з випадковою константою й потім циклічно зрушується вправо на 3 біти (ключ і дані рухаються в різних напрямках, щоб мінімізувати надлишкові операції з бітами ключа). Молодші три біти молодшого байта робочого кадру зберігаються, вони визначають циклічне зрушення інших двох байтів. Далі конкатенація двох старших байтом циклічно зрушується вліво на змінне число біт (від 0 до 7). Потім над молодшим байтом робочого кадру виконується операція XOR з молодшим байтом ключа. Нарешті робочий кадр зміщається вправо на один байт і весь процес повторюється.

Випадкова константа призначена для перетворення ключа в псевдовипадкову послідовність. Довжина константи повинна бути рівній довжині ключа. При обміні даними абоненти повинні користуватися однієї й тією же константою. Для 64-бітового ключа Madryga рекомендує константу 0x0fle2d3c4b5a6978.

При розшифруванні процес повторюється у зворотному порядку. У кожній ітерації внутрішнього циклу робочий кадр установлюється на байт, третій ліворуч від останнього байта шифртексту, і циклічно зрушується у зворотному напрямку до байта, розташованого на 2 байти уліво відносно останнього байта шифртексту. 2 байти шифртексту в процесі циклічно зрушуються вправо, а ключ – уліво. Після циклічних зрушень виконується операція XOR.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть цю технологію та програму, оскільки її інтерфейс повністю розроблений під дане операційне середовище.

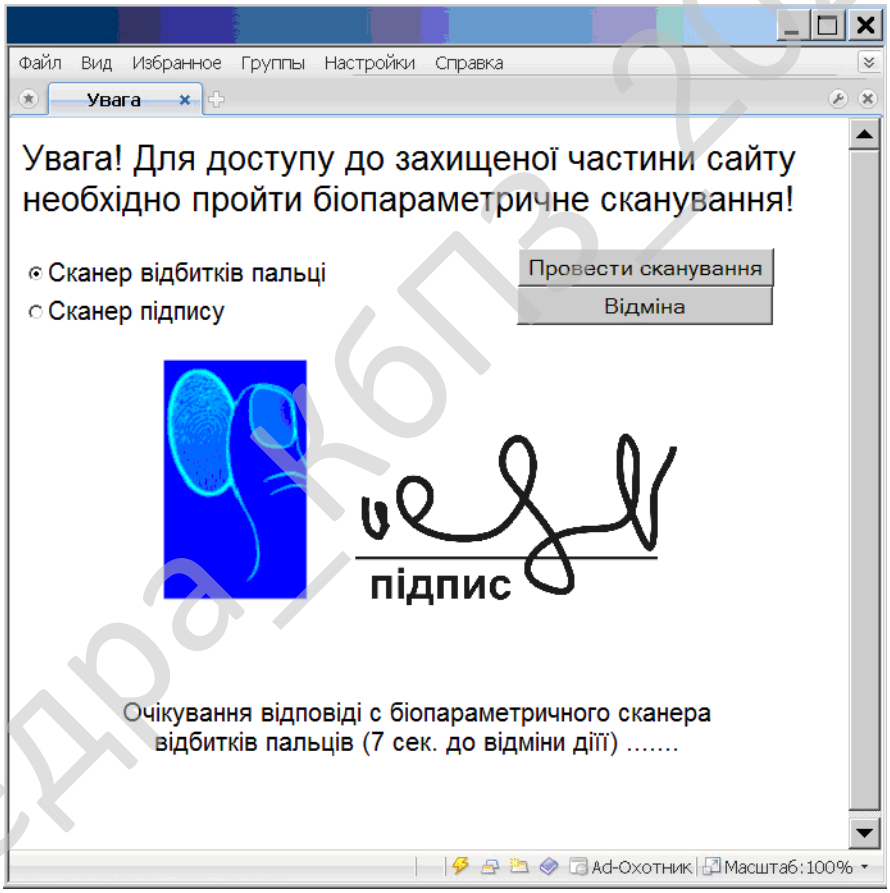


Рисунок 5.1 – Вікно сканування біопараметричних даних

Розглянемо роботу розробленої системи. Якщо користувачеві необхідно зайти на захищену сторінку сайту перед ним з'являється сторінка біометричної

перевірки особистості, що взаємодіє з розробленим Active компонентом. Розроблений Active компонент взаємодіючи з BioApi перевіряє наявність апаратного забезпечення для проведення перевірки. При наявності видається повідомлення зображене на рисунку 5.1 якщо апаратне забезпечення не виявлене видається повідомлення (рисунок 5.2).

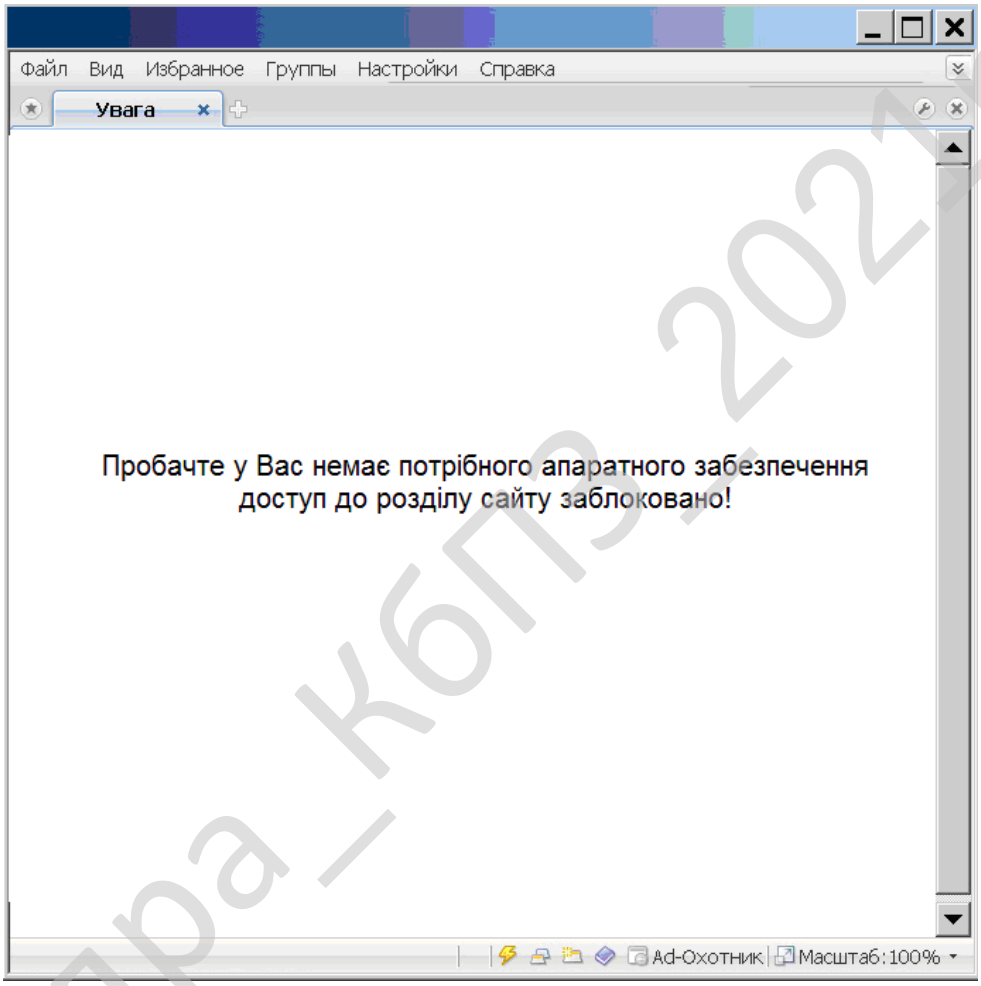


Рисунок 5.2 – Вікно перевірки наявності апаратного забезпечення

До складу розробленої системи входить докладна документація користувача із прикладами, покрокова інструкція з переходу від аутентифікації по паролю за допомогою WEB форм до аутентифікації по біометричним особливостям відбитка пальця й підпису. У розроблену систему імпортовані бібліотеки Bioparametrs Aplication Programming Interface 1.1, за допомогою яких

застосовується технологія біометричної ідентифікації й аутентифікації на основі відбитків пальців і підпису. При проведені дій біопараметричного сканування необхідно постаратися надати зразок який найблизче підходить к эталонному та не є змазанним, на рисунку 5.3 зображена перевірки якості зразку та формування біопараметричного запису (моделі).

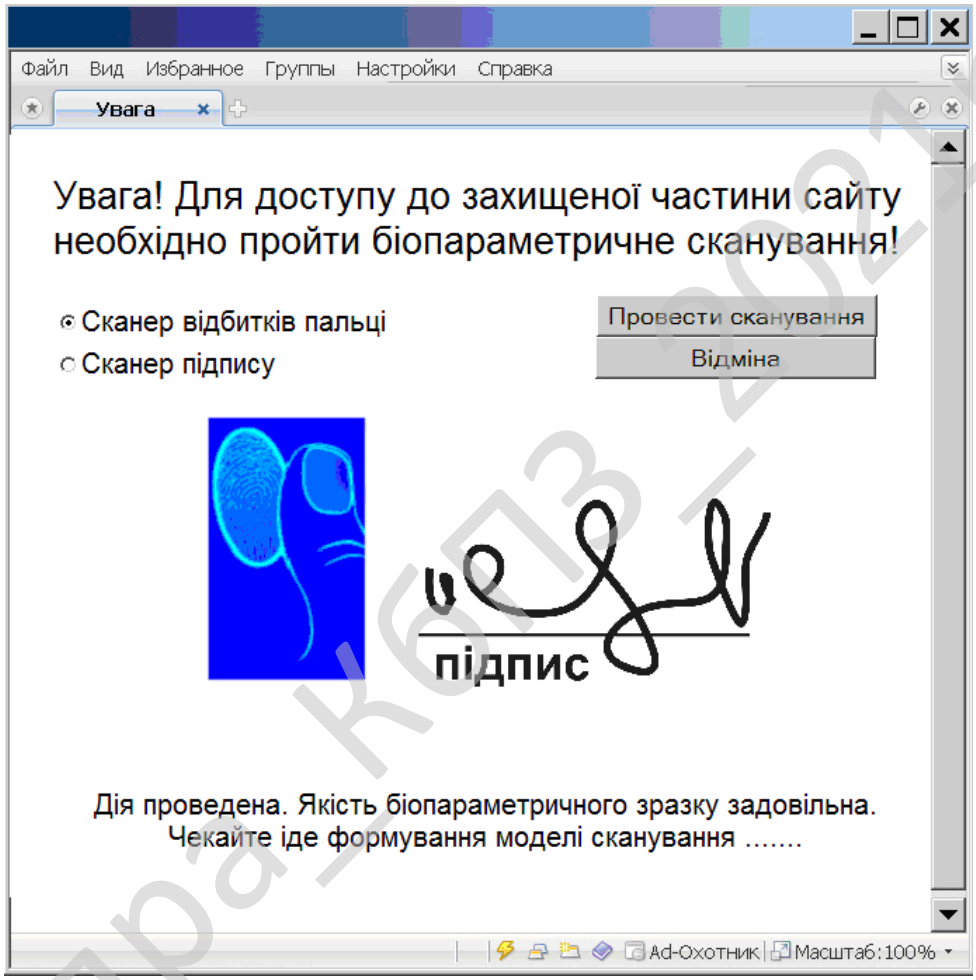


Рисунок 5.3 – Вікно перевірки якості зразку та формування моделі перевірки

Всі тонкіші формування залишаються за кадром, користувачеві необхідно лише надати зразок далі всі дії йдуть у автоматичному режимі. Якщо зразок вдалий програма відсилає сформовану модель до серверу.

Де проходить перевірку і аналіз з еталонним зразком. Якщо перевірка вдала користувачеві буде видано сповіщення о вдалої перевірки, якщо перевірка не пройдена доступ до контенту буде заблокований.

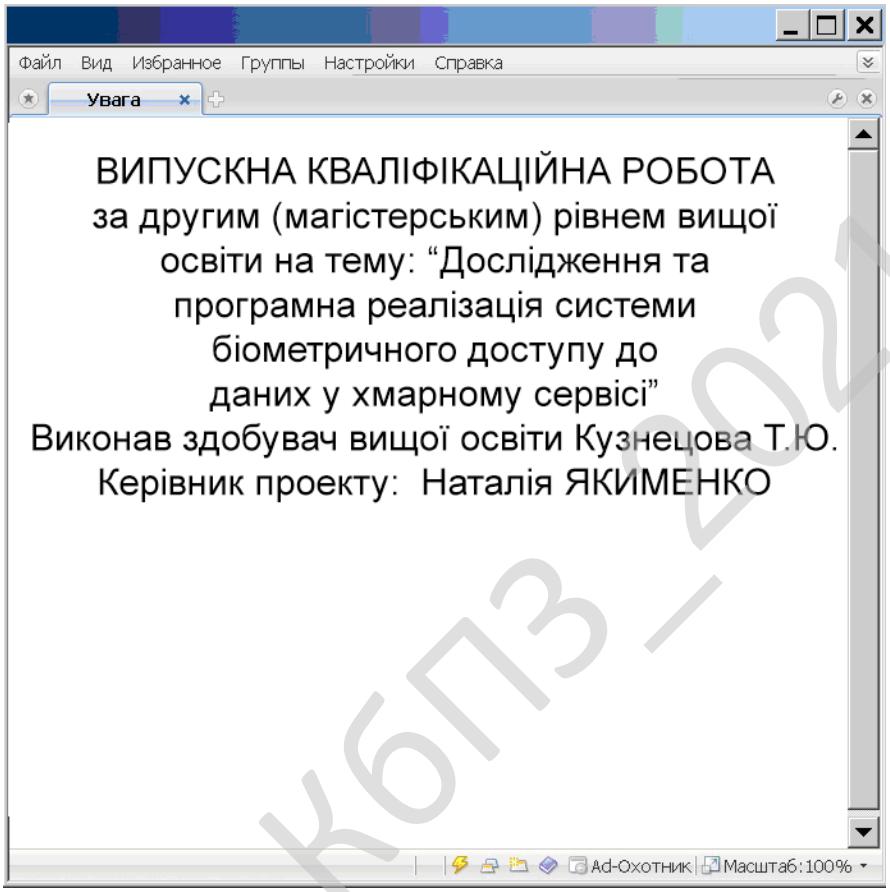


Рисунок 5.4 – Вікно авторства

Розглянемо етапи впровадження системи у експлуатацію. Це є найголовніший пункт так як це є система захисту.

Розроблене програмне забезпечення, рекомендується для впровадження в фірмах, підприємствах, установах, організаціях, де необхідний надійний захист завдяки біопараметричній ідентифікації.

При впровадженні системи необхідно вирішити ряд проблем, які дозволять розпочати промислову експлуатацію розробленої системи.

Це, перш за все, розробка організаційного та методичного забезпечення.

Методичне забезпечення впровадження системи в промислову експлуатацію включає слідуючи документи:

– інструкції по використанню програмного забезпечення;

– інструкції користувача, який буде працювати з програмним забезпеченням;

– правила по забезпеченню техніки безпеки при експлуатації програмного забезпечення;

– документи з описом методики вивчення методу послідовного аналізу;

– таблиці нормативної інформації;

Організаційне забезпечення впровадження системи включає в себе наступні документи:

– накази та розпорядження по регламентації взаємодії підрозділів, які приймають участь в експлуатації створеної програмного забезпечення;

– накази та розпорядження, які визначають функції підрозділів по експлуатації та обслуговуванню програмного забезпечення та технічного обладнання;

– затверджений перелік матеріально-технічного забезпечення підсистеми;

– графік регламентних робіт;

– журнали для регламентації роботи користувачів, обліку машинного часу, обліку проведених регламентних робіт та інші;

– штатний розклад обслуговуючого персоналу;

– інструкції по супроводженню промислової експлуатації програмного забезпечення;

Організаційна система впровадження також передбачає розробку плану впровадження і організацію підготовки та навчання персоналу, налагодження системи на підприємстві, приймальню здавальні іспити, промислову експлуатацію.

План впровадження системи встановлює:

– обсяг фінансування проектних та налагоджувальних робіт;

– терміни розробки проектної документації по елементах системи, терміни постачання технічних заходів.

План впровадження також передбачає організацію підготовки кадрів, експлуатаційного персоналу і аналіз функціонування програмного забезпечення системи після виконання всіх робіт з її впровадження.

Крім конкретних термінів виконання робіт, у плані впровадження зазначені підрозділи, які відповідають за виконання робіт в повному обсязі із забезпеченням якісних показників.

Необхідні системні вимоги

Серверна частина:

- Microsoft Windows Server 2019;
- Microsoft Internet Information Server;
- Microsoft Application Center (опція, може використатися для масштабування й балансування навантаження).
- Клієнт:
- Microsoft Internet Explorer 4.0 або вище;
- Сканер біопараметричного сканування (оптичний сканер відбитків пальців, планшетний сканер перевірки підпису). Система тестувалася на малогабаритному сканері відбитків пальців MorphoSmart 1300 французької компанії Sagem та планшетний сканер SignatureGem 1x5.
- Вільний USB порт.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи біометричного доступу до даних у хмарному сервісі.

Метою розробки є дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

Об'єктом дослідження є процес біометричного доступу до даних у хмарному сервісі.

Предметом дослідження є методи біометричного доступу до даних у хмарному сервісі.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод біометричного доступу до даних у хмарному сервісі.

– Розроблено вітчизняний продукт біометричного доступу до даних у хмарному сервісі, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові данні

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	96 (2 ост. цифри № зал*10^1)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	96000 (2 ост. цифри № зал*10^4)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	37
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B\ , \qquad (7.1)$$

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$

$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot83=168 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3{,}8 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=567,5/(60·8)=1,2 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів – електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6000	21600
Інженер-системотехнік	0,25	6000	4500
Адміністратор мережі	0,5	6000	9000
Системний програміст	0,25	6000	4500
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6000	4500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$=8,25	-	$Ф_{роб}$=198300

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 400 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sy – питома площа на одне робоче місце, $м^2$; $Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми hotline за 19.11.20 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Socket FM2+ AMD Athlon X4 845 3,5-3,8 GHz 4C 2 MB 65 Вт BOX	1350
Системна плата	ASRock FM2+ FM2A68M-DG3+ A68H/DDR3 1600/DVI+D-Sub+PCI-E16x/4 SATA/2xUSB 3.0/5.1/1G	1200
Відеокарта	GIGABYTE GV-N710D3-1GL GeForce GT710 1 GB DDR3 954/1800 MHz 64-bit D-Sub+HDMI+DVI	1150
Жорсткий диск	HDD Seagate Barracuda 750 Gb 7200 32Mb SATAII ST3750528AS (ST3750528AS)	1200
Оперативна пам’ять	DIMM 1024Mb DDR3 PC3-10600 CL9 Transcend JetRam 1333Mhz, 128M x 64, non-Reg., no-ECC , CL 9 (2 модулі)	900
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	416
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 2.0, Mic+Audio, silver/black	911
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	50000	10	5000
Разом	$К_р$= 1606075		$А_р$= 141140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 400 \cdot 209/50 = 1672 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 1672 \cdot 10 \cdot 0{,}01 = 167{,}2 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=37% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 37(1672+167{,}2) = 681 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1672 \cdot 15 \cdot 0{,}01 = 251 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм $n_{міс}$ приймаємо 0,33 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=105 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 105 \cdot 3 \cdot 0{,}33 = 105 \text{ грн}.$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_{д}., \quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 8 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 12 грн/шт.

$$З_{М2} = 50 \cdot 8 + 12 = 412 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (105 + 412 + 1702)/50=44 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців

$$О_{п} = З_{о} \cdot Н_{п} \cdot 0{,}01, \quad (7.19)$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_{п} = 1672 \cdot 15 \cdot 0{,}01=251 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 50 прим.)

$$А_{м} = \frac{А_{р} \cdot N_{міс}}{N_{е} \cdot 12}, \quad (7.20)$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_{м} = 141140 \cdot 3/(50 \cdot 12)=706 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 1672+167+681+251+44+251+706 = 3772 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 3772 = 2075 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	1672
2. Додаткова зарплата виконавців	$З_д$	167
3. Відрахування на соціальні потреби	$С_{оц}$	681
4. Загальногосподарські витрати	$Г_{осп}$	251
5. Витрати на матеріали	$З_М$	44
6. Освоєння нових операційних систем, мов програмування	$О_п$	251
7. Амортизація основних фондів	$А_м$	706
8. Повна собівартість програмного забезпечення	$С_п$	3772
9. Плановий прибуток	$П_р$	2075
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5847
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1169,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	7922

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	7922
Всього капітальних витрат	–	7922

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	27914	5234
2. Витрати на електроенергію	$З_{ел}$	2565	1282
Витрати на амортизацію	$З_{ам}$	0	1981
Всього витрат за рік	$І$	30479	8497

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин обслуговування системи зменшилась з 400 годин до 75 годин на рік, тому витрати на обслуговування склали:

$$З_{р\,баз} = 400 \cdot 52 \cdot 1{,}1 \cdot 1{,}22 = 27914 \; грн.$$

до

$$З_{р\,нов} = 75 \cdot 52 \cdot 1{,}1 \cdot 1{,}22 = 5234 \; грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності (Пел) в кіловатах, ча су експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз=·0,475·2455·2,2 = 2565 грн

Зел нов=·0,475·1227·2,2 = 1282 грн

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	7922	–	1980,5
Всього відрахувань	-	–	7922	–	1980,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (5847 - 3772) \cdot 50 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 50000) \cdot 3/12 = 68465 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{198075}{(5847 - 3772) \cdot 50 \cdot 12 / 3} = 0,5 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (30479 - 8497) - 0,25 \cdot 7922 = 20002 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \qquad (7.28)$$

$$T_{сп} = \frac{7922}{30479 - 8497} = 0,4 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	50
2. Повна собівартість розробленої програми	Грн.	3772
3. Ціна розробленої програми	Грн.	5847
4. Плановий прибуток від реалізації розробленої програми	Грн.	2075
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1606075
7. Загальний прибуток від реалізації програмної продукції	Грн.	103750
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	68465
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	7922
11.Величина економічного ефекту у користувача програмної продукції	Грн.	20002
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,4

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Шкідливі і небезпечні фактори при роботі з комп'ютером

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя.

Законом України "Про охорону праці" регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'яз и рук) при роботі з клавіатурою мають теж іс тотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	6,4
Довжина	8,4
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,7
Обсяг, V	$м^3$	не менше 20.0	20,1

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працює 8 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-

обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-58	0,11
Тепла	23-25	50-70	0,1	24-25	56-69	0,12

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер *Epson L3151*, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V -го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Розрахункова частина

Початкові данні для розрахунку захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Заземленню підлягає виробниче обладнання організації. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – садова земля, нижнього шару грунта – супісок. Умовна товщина верхнього шару грунта: *H=1,1 м*. Для захисного заземлення: застосовуються вертикальні електроди – пруток довжиною *L=2 м.* Відстань між вертикальними заземлювачами (електродами) *A=2 м*. Діаметр ве ртикального електрода (прутка) *D=40 мм*, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: 80 х 6 мм. (*b=80 мм*.). Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Глибина закладення контура заземлення *t= 0,7 м*.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 Метододичних вказівок до виконання розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. Машинобуд.; URL : http://dspace.kntu. kr.ua/jspui/handle/123456789/4358, або будь якої відповідної іншої, наприклад, програми «Електрик» (автор Сумеркін Є.О.). Додаткову перевірку отриманих результатів можна робити за допомогою он-лайн сервісу URL: https://kalk.pro/electricity/earthing/.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+2/2=1,7 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв}= \psi \rho_1 \rho_2 L /[\rho_1 \psi(L-H+t)+ \rho_2 \psi(H-t)]=153,54 \text{ Ом}$$

де

ψ = 1,1 табличне значення коефіцієнта сезонності у багатошаровому грунті;

ρ_1 = 150 Ом*м. – табличне значення питомого опору верхнього шару грунта;

ρ_2= 100 Ом*м. – табличне значення питомого опору нижнього шару грунта.

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O= [\rho_1/(2\pi*L)]*[(ln)2L/D)+0,5ln[(4T+L)/(4T-L)]=59,9 \text{ Ом.}$$

Опір розтіканню електричного струму заземлювача, який нормується, якщо $\rho_{екв}$ >100 (у нас $\rho_{екв}$ = 153,54 >100):

$$R= R_{ЗН} *\rho_{екв}/100=4*194,5/100=6,1 \text{ Ом.}$$

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_{П} =0,366(\rho_{екв}\ \psi / L_{П} \cdot\eta_{П})\lg(2/ L_{П} * L_{П}/bt)=8,35 \text{ Ом.}$$

де

$\eta_{П}$ = 0,27 – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L;

довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру: $L_{П}$=A*n =2*15=30 м.;

де n – ітераційна кількість вертикальних заземлювачів.

Загальниц опір штучного заземлюючого пристрою розтіканню електричного струму на землю:

$$R_{B} = R_{П} R / (R_{П} -R) =4,0 \text{ Ом.}$$

Кількість вертикальних заземлювачів:

$$n = R_{O} / R_{B} * \eta_{B} =15 \text{ шт.}$$

де η_{B} = 0,47 – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L.

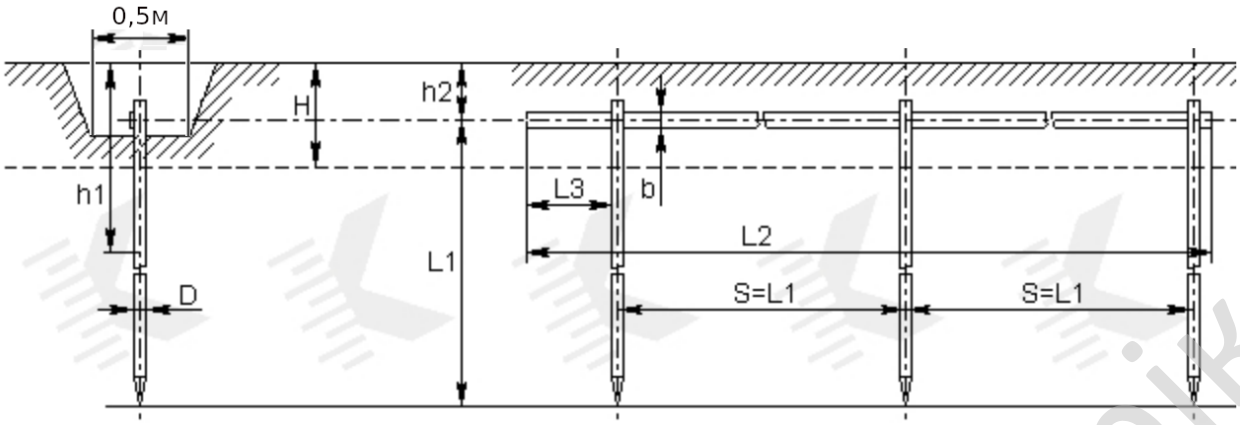


Рисунок 8.1 – Штучний заземлювач

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи біометричного доступу до даних у хмарному сервісі.

В межах Ук раїни в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів біометричного доступу до даних у хмарному сервісі.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем біометричного доступу до даних у хмарному сервісі.

– Досліджена система біометричного доступу до даних у хмарному сервісі.

– На основі отриманих результатів досліджень створена програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання біометричного доступу до даних у хмарному сервісі.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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. 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 й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Madryga.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 20002 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,4 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузнецова Т.Ю. Дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.
2. Соколов А.В., Шаньгин В.Ф. Защита информации в распределенных корпоративных сетях и системах. М.: ДМК Пресс, 2002.
3. Леонтьев Б. Хакинг без секретов. М.: Познавательная книга плюс, 2000.
4. Медведовский И.Д., Семьянов П.В., Леонов Д.Г. Атака на Internet. М.: ДМК Пресс, 2000.
5. Евангели А. PC Week/RE 2003, № 7, с. 24—25. Технологии биоидентификации и біопараметрический рынок.
6. Матвеев И.А., Ганькин К.А. Распознавание человека по радужке // Системы безопасности, 2004, № 5, с. 72.
7. «Электронная цифровая подпись», «Мир ПК», № 3/02.
8. «Конкурсы AES и NESSIE», «Мир ПК», № 12/04.
9. «Виртуальные частные сети и другие способы защиты информации», «Мир ПК», № 4/02
10. ANSI B10.8. Finger Minutiae Extraction and Format Standard for One-to-One Matching
11. NISTIR 6529, ISO/IEC 19785. Common Biometric Exchange File Format (CBEFF)
12. ANSI X9.84. Biometric Information Management and Security
13. ISO/IEC 7816-11. Personal Verification Through Biometric Methods
14. ISO/IEC 19784. BioAPI 2.0
15. ANSI/INCITS 358. BioAPI 1.1
16. ISO/IEC 19794-2. Finger Minutiae Data

17. ISO/IEC 19794-4. Finger Image Data

18. Смірнова Т.В., Дрєєв О.М., «Інформаційна технологія оптимізації технологічного процесу відновлення та зміцнювання поверхонь валів зі сталі як хмарний сервіс» у *Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка*. – Х. : Вид. Рожко С.Г. 2019. С. 92-107.

19. Т.В.Смірнова, Л.І. Поліщук, «Дослідження хмарних технологій як сервісів для системи інженерних розрахунків» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. С. 106-121.

20. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», *Системи управління, навігації та зв'язку*, № 2 (54). с. 149-154, 2019.

21. Т.В. Смірнова, Є.К. Солових, О.А. Смірнов, О.М. Дрєєв, «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей», *Центральноукраїнський науковий вісник. Технічні науки.* № 1(32). с. 184-194, 2019.

22. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

23. Т. В. Смирнова, А. А. Смирнов, А. Н. Дреев, А. В. Дудан, «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса», Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. **Республика Беларусь** - 2020. - № 3. - С. 50-61.Режим доступу: http://elib.psu.by:8080/handle/123456789/24988

24. Т.В.Смірнова, Л.І. Поліщук, О.А.Смірнов, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020. (Категорія Б)

25. Т.В.Смірнова, «Формалізація та реалізація структури технологічного процесу електродугового напилення для оптимізаційної експертної системи», *Технічні науки та технології*. № 1(19). С. 104-113. 2020. (Категорія Б)

26. Т.В. Смірнова «Формування евристичних правил, бази знань та формалізація структури й правил технологічного процесу для оптимізаційної хмарної інформаційної системи», *Системи управління, навігації та зв'язку*, № 2 (60). с. 101-104, 2020. (Категорія Б)

27. Т.В. Смірнова, О.А. Смірнов, О.М. Дрєєв, С.А. Смірнов, «Використання хмарних експертних систем в сфері інформаційного забезпечення обробки поверхні деталей», *Комп'ютерна інженерія і кібербезпека: досягнення та інновації*, м. Кропивницький. 27-29 листопада, 2018, с. 111-113

28. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Захист даних у інформаційній технології відновлення поверхонь деталей у вигляді хмарної платформи як послуги», *Проблеми кібербезпеки інформаційно-телекомунікаційних систем*, м. Київ, 11-12 квітня, 2019, с. 221-224.

29. Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Хмарний сервіс інформаційної технології оптимізації технологічного процесу відновлення та зміцнювання поверхонь зі сталі», *Інформаційна безпека та інформаційні технології, Information Security and Information Technologies*, м. Харків, 24-25 квітня, 2019, с. 36

30. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Побудова хмарної експертної системи оптимізації технологічного процесу електродугового напилення сталевих покриттів», *Міжнародний форум з інформаційних систем і технологій INFOS-2019,* м. Харків, 24-27 квітня, 2019, с. 95-98.

31. Т. В. Смирнова, А. А. Смирнов, Е. К. Соловых « Разработка математической модели и программного обеспечения для оптимизации режима

электроконтактной обработки газотермических покрытий», *Комплексне забезпечення якості технологічних процесів та систем,* том 2, м. Чернігів, 14 - 16 травня, 2019, с. 78-79.

32. Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, О.А. Смірнов, «Експертна система інформаційної технології оптимізації абстрактного технологічного процесу як хмарного сервісу», *Інформаційні технології: Наука, техніка, технологія, освіта, здоров'я. MicroCAD-2019,* м. Харків, 15-17 травня, 2019, с. 195.

33. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Формування абстрактних експертних систем на основі досліджень відомих експертних систем», *XXI Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування»*, м. Кропивницький, 17-18 травня, 2019, с. 143-147.

34. Т.В. Смірнова, О.А. Смірнов, «Захист даних у інформаційній технології відновлення поверхонь деталей у вигляді хмарної платформи як послуги», *Інформація, комунікація, суспільство – 2019*, см.т. Чинадієво, 16-18 травня, 2019, с. 25-26

35. Т.В. Смирнова, А.Н. Дреев, А.А. Смирнов, «Информационная технология оптимизации технологического процесса восстановления и упрочнения поверхностей в виде облачного сервиса», *Modern Information, Measurement And Control Systems: Problems And Perspectives (MIMCS 2019)*, Baku, Azerbaijan, 01-02 July, 2019, p. 282.

36. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Огляд відомих експертних систем оптимізації технологічних процесів», *Стратегія якості в промисловості і освіті,*м. Варна, Болгарія, 3-6 червня, 2019, с.442-444

37. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Формалізація та узагальнення інформаційної моделі технологічних операцій зміцнення та відновлення сталевих поверхонь», *Інтелектуальні системи та інформаційні технології*, м.Одеса, 19-24 серпня, 2019, с. 211-213.

38. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Формування інформаційної моделі технологічного процесу». *V Всеукраїнська науково-практична Інтернет-конференція «Електронні та мехатронні системи: теорія, інновації, практика»*, м. Полтава, 8 листопада, 2019, с. 87-91.

39. T.V. Smirnova, M.S. Chernovol, Ageev M. «Study of the spraying process and the influence of its factors on the properties of electric arc spraying coatings». *Materials of the 20th international scientific and technical seminar «Modern questions of production and repair in industry and in transport»*, *Tbilisi, Georgia*, 23-28 *March* 2020, с. 201-205.

40. Т.В.Смірнова, Є.К. Солових, О.А.Смірнов, «Дослідження стандартів забезпечення кібербезпеки хмарних технологій як сервісів», *X міжнародна науково-технічна конференція «ITSEC»*, Єгипет, м. Шарм -ель-Шейх, 19 -24 березня, 2020. с. 36.

41. Т.В.Смірнова, Л.І. Поліщук, О.А.Смірнов, «Аналіз хмарних технологій як сервісів», *XIII всеукраїнська науково-практична конференція «Комп'ютерні інтелектуальні системи та мережі (КІСМ-2020)»*. м. Кривий Ріг. 24-26 березня, 2020, С. 210-212.

42. Т.В. Смірнова, Л.І. Поліщук, О.М. Дрєєв, «Застосування сервісу CAEaaS як системи інженерних розрахунків з використанням хмарних технологій», *II Міжнародна науково-практична конференція «Інформаційна безпека та інформаційні технології, Information Security and Information Technologies»*, м. Кропивницький, 2-3 квітня, 2020, с. 52.

43. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, Є.К. Солових, «Інформаційна структура технологічного процесу електродугового напилення». *Всеукраїнська науково-технічна конференція «Стан, досягнення та перспективи інформаційних систем і технологій»*, м. Одеса, 21-22 квітня, 2020, с. 184-186.

44. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, Є.К. Солових, «Реалізація інформаційної структури технологічного процесу електродугового напилення»,

XXII Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування», м. Кропивницький, 15-16 травня, 2020, с. 158-162.

45. Т.В. Смірнова, А.В. Щербань, Ю.Ю. Моторін, О.А. Смірнов, «Перспективні напрямки забезпечення кібербезпеки сервісу CAEaaS», *VI Всеукраїнська науково-практична конференція «Перспективні напрями захисту інформації»*, м. Одеса, 2-6 вересня 2020, с. 58-59

46. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

47. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

48. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

49. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

50. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов //

Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

51. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

52. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

53. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

54. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

55. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

56. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-122.21.0096.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Кузнецова Т.Ю.			Дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Якименко Н.М.					
Н. Контр.		Гермак В.С.				ЦНТУ КН-20МЗ	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи біометричного доступу до даних у хмарному сервісі.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 40-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи біометричного доступу до даних у хмарному сервісі.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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 з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи біометричного доступу до даних у хмарному сервісі;

– 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 архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

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 XP/Vista/7/8/10.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 108 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 23.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Якименко Н.М.

Дослідження та програмна реалізація

системи біометричного доступу до даних у хмарному сервісі

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2021 року

Для правильної роботи системи необхідна наявність наступних бібліотек Windows:
- btWebRecognitionServer.dll - COM-модуль, що реалізує інтерфейс до математичних бібліотек
- CLOUD_BIOHFW32.DLL, BTBSPCln.DLL, BTBSPCTL.DLL, BTBSPSRV.DLL, DSPSTM32.DLL бібліотеки математичної підтримки, що входять до складу Cloud_bioWebSDK.
Ім'я серверної частини (Com модуля)- **cloud_bioSERVERAutoidentificathion**
Ім'я клієнтської частини (ActiveX)- **cloud_bioClient**

Серверна частина файл Cloud_bioIDENT.cpp

```
// Copyright (C)  2021  Kuznetsova T.Yu. All Rights Reserved.
//
// Name: Cloud_bioIDENT.cpp
//
// Description:
//    Magister work. Cloud_bioparameters Identificathion

#include "precomp.h"
extern BOOLEAN   gbInitialized;
extern PFWPasport gpFWPasport;
NTSTATUS SERVACTIVEXSKFilter(SKPacketCloud_bioIDENTPtr pFilterFunction,
BOOLEAN Add)
{
PDEVICE_OBJECT pDeviceObject = NULL;
PFILE_OBJECT   pFileObject   = NULL;
SK_SET_Cloud_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.Cloud_bioIDENTPtr = pFilterFunction;
SERVACTIVEXInfo.Priority = 2;
SERVACTIVEXInfo.Add = Add;
KeInitializeEvent(&NotificationHandle, NotificationEvent, FALSE);
Irp=IoBuildDeviceIoControlRequest(IOCTL_SK_SET_Cloud_bioIDENT_SERVACTIVEX,pDev
iceObject,(PVOID) &SERVACTIVEXInfo,
sizeof(SK_SET_Cloud_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)
{
PCloud_bioIDENT_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 = (PCloud_bioIDENT_CONTEXT_T)pContext;

if(!pFWContext)
{
dprintf("SKFWSERVACTIVEX.sys : FWEngine - Cathastrophic(1) : Cloud_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 CLOUD_BIO PACKET BIR(1)");

FWAction = DROP;

break;
}

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

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

Ncloud_bioZeroMemory(pPacket,sizeof(TPacket));
Ncloud_bioMoveMemory(&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)Ncloud_bioMoveMemory(&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)
Ncloud_bioMoveMemory(&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)
Ncloud_bioMoveMemory(&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;
Ncloud_bioAcquireSpinLock(&pFWPasport->PasportLock);
pFWRuleSet = pFWPasport->Head;


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

if(FWAction != FORWARD)
break;

pFWRuleSet = pFWRuleSet->Next;
}

Ncloud_bioReleaseSpinLock(&pFWPasport->PasportLock);

return FWAction;
}

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

FORWARD_ACTION FWAction = FORWARD;

if(pFWRuleSet == NULL)
return FWAction;

if(pPacket)
{
Ncloud_bioAcquireSpinLock(&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;
}
}

Ncloud_bioReleaseSpinLock(&pFWRuleSet->RuleSetLock);
}

return FWAction;
}

Ncloud_bio_STATUS FWCreatePasport(PFWPasport *pFWPasport)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWPasport pTemp = NULL;

do
{
__try
{

if(*pFWPasport != NULL)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;

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

break;
}

Ncloud_bioStatus = FWNewPasport(pFWPasport);

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

pTemp = *pFWPasport;

Ncloud_bioAllocateSpinLock(&pTemp->PasportLock);

}
__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;

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


return Ncloud_bioStatus;
```

```
}

Ncloud_bio_STATUS FWNewPasport(PFWPasport *pFWPasport)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

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

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

Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

}else
{
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;

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

Ncloud_bio_STATUS FWDestroyPasport(PFWPasport *pFWPasport)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWRuleSet  Prev = NULL;
PFWRuleSet  Next = NULL;
PFWPasport   pTemp = NULL;

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

return Ncloud_bio_STATUS_INVALID_DATA;
}

__try
{
pTemp = *pFWPasport;

Ncloud_bioAcquireSpinLock(&pTemp->PasportLock);

Next = pTemp->Head;
```

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

}

Ncloud_bioReleaseSpinLock(&pTemp->PasportLock);

Ncloud_bioFreeSpinLock(&pTemp->PasportLock);

FWFree(*pFWPasport);

*pFWPasport = NULL;
Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
return Ncloud_bioStatus;
}

PFWRuleSet FWRegisterRuleSet(PFWPasport pFWPasport, PNcloud_bio_STATUS
pNcloud_bioStatus)
{
PFWRuleSet  Next = NULL;
PFWRuleSet  Temp = NULL;

*pNcloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

if(pFWPasport == NULL)
{
*pNcloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;

return NULL;
}

Ncloud_bioAcquireSpinLock(&pFWPasport->PasportLock);

do
{
__try
{

*pNcloud_bioStatus = FWNewRuleSet(&Temp);

if(*pNcloud_bioStatus != Ncloud_bio_STATUS_SUCCESS)
{
Temp = NULL;
break;
}
```

```
Ncloud_bioAllocateSpinLock(&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 )
{
*pNcloud_bioStatus = Ncloud_bio_Status_Failule;
}

}while(0);

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

}

Ncloud_bio_STATUS FWNewRuleSet(PFWRuleSet *pFWRuleSet)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
__try
{
*pFWRuleSet = (PFWRuleSet)FWAlloc(sizeof(TFWRuleSet));

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

Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

}else
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
return Ncloud_bioStatus;
}

Ncloud_bio_STATUS FWFreeRuleSet(PFWRuleSet *pFWRuleSet)
{
```

```
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
__try
{
if(pFWRuleSet)
{

FWFreeRulesQueue(&(*pFWRuleSet)->FWSKRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWTCPRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWUDPRules);
FWFreeRulesQueue(&(*pFWRuleSet)->FWICMPRules);
Ncloud_bioFreeSpinLock(&(*pFWRuleSet)->RuleSetLock);
FWFree((*pFWRuleSet));
(*pFWRuleSet) = NULL;
Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

}else
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
return Ncloud_bioStatus;
}

Ncloud_bio_STATUS FWFreeRulesQueue(PFWRulesQueue pFWRulesQueue)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWRule Prev = NULL;
PFWRule Next = NULL;

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

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

if(Ncloud_bioStatus != Ncloud_bio_STATUS_SUCCESS)
dprintf("SKFWSERVACTIVEX.sys : FWFreeRulesQueue.");
}

pFWRulesQueue->Head = NULL;
Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
}else
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
```

```
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
return Ncloud_bioStatus;
}

Ncloud_bio_STATUS FWNewRule(PFWRule *pFWRule)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

__try
{

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

Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

}else
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{ Ncloud_bioStatus = Ncloud_bio_Status_Failule; }
return Ncloud_bioStatus;
}
Ncloud_bio_STATUS FWFreeRule(PFWRule pFWRule)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
__try
{
if(pFWRule)
{
FWFree(pFWRule);

Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;

}else
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
}
__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
return Ncloud_bioStatus;
}
Ncloud_bio_STATUS FWInsertRule(PFWRuleSet pFWRuleSet, PFWRule pFWRule,
TEFWRule RuleType)
{
Ncloud_bio_STATUS   Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWRulesQueue pFWRulesQueue = NULL;
PFWRule   Next = NULL;
```

```
PFWRule Temp = NULL;

if((pFWRuleSet == NULL)|| (pFWRule == NULL))
{
return Ncloud_bio_STATUS_INVALID_DATA;
}
Ncloud_bioAcquireSpinLock(&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)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
break;
}
Ncloud_bioStatus = FWNewRule(&Temp);
if(Ncloud_bioStatus != Ncloud_bio_STATUS_SUCCESS)
break;

Ncloud_bioMoveMemory(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 )
```

```
{
Ncloud_bioStatus = Ncloud_bio_Status_Failule;
}
}while(0);

Ncloud_bioReleaseSpinLock(&pFWRuleSet->RuleSetLock);

return Ncloud_bioStatus;
}
```

Формування звітів

```
#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;
}
Ncloud_bioZeroMemory((*ppFWLogList),sizeof(TFWLogList));
Ncloud_bioAllocateSpinLock(&(*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.");

Ncloud_bioFreeSpinLock(&(*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.");
```

```
Ncloud_bioFreeSpinLock(&(*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);
}
Ncloud_bioFreeSpinLock(&(*ppFWLogList)->Lock);
FWFree((*ppFWLogList));

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

```
PFWLog  pFWLog = NULL;

Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
do
{
__try
{
pFWLog = (PFWLog)FWAlloc(sizeof(TFWLog));
if(pFWLog == NULL)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_RESOURCES;
break;
}
va_start( Args, Format );
Ncloud_bioZeroMemory(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)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_INVALID_DATA;
break;
}
Ncloud_bioStatus = FWInsertLog(gpFWLogList,pFWLog);
if( Ncloud_bioStatus == Ncloud_bio_STATUS_BUFFER_OVERFLOW )
{
dprintf("SkFWHook.sys : FWPostLogMesage - Log pool is full. Can't insert log
entry.");
FWFree(pFWLog);
Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
}
if( Ncloud_bioStatus != Ncloud_bio_STATUS_SUCCESS )
{
dprintf("SkFWHook.sys : FWPostLogMesage - Can't insert log entry.");
break;
}
Ncloud_bioAcquireSpinLock(&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.");
Ncloud_bioReleaseSpinLock(&gpFWLogList->Lock);
}__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_FAILURE;
}
}while(0);

if(Ncloud_bioStatus != Ncloud_bio_STATUS_SUCCESS)
```

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

return Ncloud_bioStatus;
}
UINT FWFlushLogList(PFWLogList pFWLogList)
{
UINT Flushed = 0;
PFWLog Temp = NULL;
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
if(pFWLogList == NULL)
return Ncloud_bio_STATUS_INVALID_DATA;
Ncloud_bioAcquireSpinLock(&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)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_FAILURE;
}
Ncloud_bioReleaseSpinLock(&pFWLogList->Lock);
if(Ncloud_bioStatus == Ncloud_bio_STATUS_SUCCESS)
return Flushed;
return -1;
}

Ncloud_bio_STATUS FWGetLog(PFWLogList pFWLogList, PVOID ioBuffer)
{


Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWLog Temp = NULL;
if(pFWLogList == NULL)
return Ncloud_bio_STATUS_INVALID_DATA;

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

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

if(Temp)
{
Ncloud_bioMoveMemory((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!");
Ncloud_bioStatus = Ncloud_bio_STATUS_RESOURCES;
}
}else
{

Ncloud_bioStatus = Ncloud_bio_STATUS_RESOURCES;
}
}__except(EXCEPTION_EXECUTE_HANDLER)
{

Ncloud_bioStatus = Ncloud_bio_STATUS_FAILURE;
}

Ncloud_bioReleaseSpinLock(&pFWLogList->Lock);

return Ncloud_bioStatus;
}
Ncloud_bio_STATUS FWInsertLog(PFWLogList pFWLogList, PFWLog pFWLog)
{
Ncloud_bio_STATUS Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
PFWLog      Temp = NULL;
if((pFWLogList == NULL) || (pFWLog == NULL))
return Ncloud_bio_STATUS_INVALID_DATA;
Ncloud_bioAcquireSpinLock(&pFWLogList->Lock);
do
{
__try
```

```
{

if(pFWLogList->LogCount > LOGSIZE)
{

Ncloud_bioStatus = Ncloud_bio_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;
Ncloud_bioStatus = Ncloud_bio_STATUS_SUCCESS;
}__except(EXCEPTION_EXECUTE_HANDLER)
{
Ncloud_bioStatus = Ncloud_bio_STATUS_FAILURE;
}
}while(0);
Ncloud_bioReleaseSpinLock(&pFWLogList->Lock);
return Ncloud_bioStatus;
}
```

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>2021-02-02T20:52:00Z</o:Created>
  <o:LastSaved>2021-02-02T21:32:00Z</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="cloud_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">Магістерська
робота 2021 р.</font></p>
<p class=MsoNormal align="right"><font face="Tahoma"
size="2">Центральноукраїнський
національний технічний університет</font></p>
<p class=MsoNormal align="right"><span style="font-family: Tahoma">
<font size="2">Розробник Кузнецова Т.Ю.</font></span></p>
</div>
</body>
</html>
```