Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ «ПриватБанк»
Назва теми

КвРКБ.170150.17.01.10 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *125 «Кібербезпека»*
Шифр, назва

Освітня програма *«Кібербезпека»*
Назва

Виконав: студент IV курсу, група КБ–17–1 О.В. Пахар
Підпис Ініціали, прізвище

Керівник _____ В.С. Орленко
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.В. Муляр
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки та
комп'ютерних систем і мереж _____ Ю.П. Кльоц
Підпис Ініціали, прізвище

«17» червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ПРОГРАМУВАННЯ ТА КОМП'ЮТЕРНИХ І ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Кафедра КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 125 КІБЕРБЕЗПЕКА

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ БАКАЛАВРІВ

ЗАТВЕРДЖУЮ

Завідувач кафедри ___________

Клебов Ю.П.

5.02 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пахару Олександру Валерійовичу

Прізвище, ім'я, по батькові студента

1 Тема роботи Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ "ПриватБанк"

Керівник роботи ___________

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджено наказом ректора університету від 05022021р. №11 додаток 9

2 Строк подання студентом роботи на кафедру: ___________

3 Вихідні дані до роботи системи біометричної автентифікації, методи покращення вхідних зображень для автентифікації, криптографічні засоби захисту інформації

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити) Аналіз об'єкта захисту, обґрунтування вибору засобів для побудови системи безпеки, проектування системи безпеки, реалізація роботи

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень) «Загальна структура автентифікації», «Алгоритм роботи розроблювальної системи», «Алгоритм запитів на автентифікацію користувача», «Блок-схема генерування хеш коду за допомогою евклідової відстані», «Алгоритм сегментації на основі двовимірного відсікання ч1»,

6 Консультанти розділів курсового проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийн
Нормоконтроль	Муляр І.В., доцент кафедри КбКСМ		
Антиплагіат	Муляр І.В., доцент кафедри КбКСМ		

7 Дата видачі завдання 5 02 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Приме
1	Вибір та затвердження теми кваліфікаційної роботи.	Січень	-
2	Аналіз об'єкта захисту.	Січень–лютий	-
3	Проектування та розробка загальної архітектури і структури системи захисту.	Лютий–березень	-
4	Програмна реалізація запропонованого рішення та тестування системи; аналіз результатів і оцінювання прийнятих рішень.	Квітень	-
5	Написання тексту пояснювальної записки та розробка графічних матеріалів.	Травень	-
6	Остаточне коригування кваліфікаційної роботи з урахуванням зауважень керівника.		-
7	Оформлення кваліфікаційної роботи як документа відповідно до вимог.		-
8	Отримання супровідних документів. Нормоконтроль.		-
9	Підготовка до захисту та захист кваліфікаційної роботи.	Червень	-

Студент _____ О.В. Пахар
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ В.С. Орленко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ «ПриватБанк».

Автор роботи: Пахар Олександр Валерійович.

Керівник роботи: Орленко Вікторія Сергіївна.

Обсяг – 52 с., 22 рис., 2 додатка, 15 джерел.

Графічна частина: 7 презентаційних слайдів, 6 плакати.

СИСТЕМА ЗАХИСТУ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ, КЕРУВАННЯ ПОТОКАМИ ДАНИХ, ЗАХИСТ ПОТОКІВ ДАНИХ

Основною метою даної розроблювана системи є введення превентивних ір по забезпечені захисту від витоків інформації та впровадження іометричної автентифікації працівників в середині відділу.

Додатковою метою створення даної системи євпровадження біометричної втентифікації клієнтів банку без використання стандартних ідентифікаторів.

Системи призначена для запобігання використанню чужих ентифікаційних даних.Сферою застосування є забезпечення захисту від итоків конфіденційної інформації в середині компанії, захист користувачів від роду.

В рамках кваліфікаційної роботи була розроблена система біометричної автентифікації, спроектована із врахуванням побажань співробітників відділу банку.

Підпис студента

Дата 01.06.21

Зона	Позиц	Позначення	Найменування	Кільк.	Прим.
	1		Завдання на дипломний	1	
			проект		
	2		Анотація	1	
	3	КвРКБ.170150.17.01.10 ПЗ	Захист потоків даних в	1	
			інформаційно-комунікаційній філії		
			мережі відділення Хмельницької		
			АТ КБ "ПриватБанк"		
			Пояснювальна записка		
	4	КвРКБ.170150.17.01.10 Е8	Загальна	1	
			структура автентифікації		
			Схема структурна		
	5	КвРКБ.170150.17.01.10 Е8	Алгоритм роботи	1	
			розроблюваної		
			автентифікації		
			Алгоритм роботи		
	6	КвРКБ.170150.17.01.10 Е8	Блок-схема алгоритм	1	
			запитів на автентифікацію		
			користувача		
			Алгоритм роботи		

Змін.	Арк.	№ Докум.	Підп.	Дата	КвРКБ.170150.17.01.10 ВП		
Розробив		Пахар О.В.		04.06	Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ "ПриватБанк" Відомість проекту	Літера	Аркуш 1 / Аркушів 2
Перев.		Орленко В.С.		2006			
Н. контр.		Муляр І.В.				ХНУ, КБ-17-1	
Затв.		Кльоц Ю.П.					

Формат	Зона	Позиц	Позначення	Найменування	Кільк.	Прим.
А2		7	КвРКБ.170150.17.01.10 Е8	Блок-схема генерування	1	
				хеш-коду з використанням		
				евклідової відстані		
				Алгоритм роботи		
А2		8	КвРКБ.170150.17.01.10 Е8	Блок-схема алгоритму	2	
				сегментації на основі		
				двовимірного відсікання		

Зм.	Арк.	№ Докум.	Підп.	Дата	*КвРКБ.170150.17.01.10 ВП*	Арк. 2

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ....4
ВСТУП....5
1 АНАЛІЗ ОБ'ЄКТА ЗАХИСТУ....8
1.1Історія розвитку інформаційного простору і загрози інформаційної безпеки...8
1.2 Технології захисту даних....14
1.2.1 Резервне копіювання та відновлення даних....15
1.2.2 Хмарні технології захисту даних....16
1.2.3 Blockchain....17
1.3Висновки....18
2 ОБГРУТНУВАННЯ ВИБОРУ ЗАСОБІВ ДЛЯ ПОБУДОВИ СИСТЕМИ АВТЕНТИФІКАЦІЇ....19
2.1 Опис мови програмування C++....19
2.2 Опис системи управління базами даних....20
2.3 Середовище розробки....21
3 ПРОЕКТУВАННЯ СИСТЕМИ АВТЕНТИФІКАЦІЇ....22
3.1 Система біметричної автентифікації....22
3.1.1 Типова архітектура....22
3.1.2 Методи підвищення покращення зображення відбитків пальців....23
3.2 Технічне завдання на розроблювану систему....28
3.3Опис функціонування системи....29
3.4 Висновки....31
4 РЕАЛІЗАЦІЯ РОБОТИ....32
4.1 Розробка блок-схем і опис алгоритмів функціонування системи....32

4.2 Тестування системи.....36
4.3 Впровадження системи в промислову експлуатацію.....47
ВИСНОВКИ.....50
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....52
ДОДАТОК А Копія графічної частини…….49
ДОДАТОК Б Програмна реалізація……..56

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

ДТЗ – допоміжні технічні засоби

ЕОМ – електронно-обчислювальна машина;

ІД – інформаційна діяльність;

ІзОД – інформація з обмеженим доступом;

ІТС – інформаційно-телекомунікаційна система;

КЗЗ – комплекс засобів захисту;

КРТ – копіювально-розмножувальна техніка;

КС – комп'ютерна система;

КСЗІ – комплексна система захисту інформації;

НД – нормативний документ;

НД ТЗІ – нормативний документ системи технічного захисту інформації;

НСД – несанкціонований доступ;

ОС – обчислювальна система;

ОТЗ – основні технічні засоби;

ПЗІ – підрозділ захисту інформації;

ПЕМВН – побічні електромагнітні випромінювання і наведення;

ПЗ – програмне забезпечення;

ПЗП – постійний запам'ятовувальний пристрій;

ПРД – правила розмежування доступу;

ВСТУП

Велика частина сучасного життя залежить від комп'ютерів і комп'ютерних мереж. Для багатьох людей найбільш помітним взаємодією з комп'ютером є набір тексту на клавіатурі комп'ютера. Але комп'ютери і мережі мають вирішальне значення для ключових функцій, таких як управління і експлуатація атомних електростанцій, гребель, електромережі, системи управління повітряним рухом і фінансової інфраструктури. Комп'ютери також грають важливу роль в повсякденній діяльності компаній, організацій і уряду. Великі і малі компанії покладаються на комп'ютери для управління платіжною системою, для відстеження запасів і продажів, а також для проведення досліджень і розробок. Розподіл продуктів харчування і енергії від виробника до роздрібного споживача залежить від комп'ютерів і мереж на кожному етапі. Майже кожен в бізнесі або уряді покладається на електронні засоби зв'язку - будь то телефон, факс, електронна пошта або миттєві повідомлення, - які, очевидно, підтримуються комп'ютерами. Багато (можливо, навіть більшість) комп'ютерних систем сьогодні якимось чином об'єднані в мережу, найбільш наочно і зазвичай через велику сукупність глобальних взаємозалежних комп'ютерних мереж, відомих як Інтернет. Пізніша тенденція полягає в тому, щоб вбудовувати обчислювальні можливості в усі види пристроїв і середовищ, а вбудовані мережеві системи - в більші системи. Ці тенденції роблять багато обчислювальні і комунікаційні системи критично важливими інфраструктури самі по собі і компонентами інших видів критичної інфраструктури, від енергетичних до транспортних систем.

Навмисні проблеми - результат свідомого людського вибору. Маючи справу з навмисними проблемами, людина стикається зі злим умислом. Зловмисник може спробувати приховати свої сліди, що ускладнює визначення характеру викликаної проблеми (або навіть визначення того, що

проблема була викликана). В принципі, зловмисний людина може адаптувати дії для отримання бажаного ефекту, крім шкоди, завданої реальної атакований системі - на відміну від випадкової проблеми, наслідки якої визначаються випадковим чином. Експерти з безпеки часто називають дії зловмисників «атаками». Центральна проблема при реагуванні на атаку інформаційної системи - це визначити, хто є зловмисником, і визначити, чи є його мотив пустощі, тероризмом або нападом на країну. Пов'язана проблема полягає в тому, щоб визначити, чи пов'язані події, далекі в часі або просторі, - частини даної атаки.

Існує багато способів навмисно викликати проблеми. Один із способів, який отримує велику увагу у ЗМІ, - це атака що надходить "через дроти". Інтернет дозволяє здійснити таку атаку віддалено, анонімно та у великих масштабах. Одним із прикладів таких «лише кібернетичні-атак» є комп’ютерні віруси, які заражають комп’ютер користувача, здійснюють певні руйнівні дії, такі як видалення файлів у мережі або локальному жорсткому диску, та розповсюджуються далі, наприклад, надсилаючи свої копії по електронній пошті. Другий приклад - розподілена атака відмови в обслуговуванні.

Шкода, яку завдає кібернетичні атака, може виявитися не відразу (або коли-небудь) очевидною. Успішна атака може закласти основу для пізніших атак, може спричинити шкоду і після первинного проникнення, або забезпечити нелегальну передачу конфіденційної інформації, що зберігається в атакованій системі. Наприклад, низка останніх інцидентів поставила під загрозу комп’ютери нічого не підозрюючи користувачів домашні комп’ютерів, імплантуючи несанкціонований код; згодом ці комп’ютери використовувались як пункти запуску в скоординованій та розподіленій атаці відмови в обслуговуванні.

Той факт, що Інтернет з'єднує багато комп'ютерів світу, означає, що кібернетичні атака може бути здійснена з різних місць світу, прокладена

через інші країни (можливо, таємно і невідомо кому в цих інших країнах) і спрямована проти будь-якого комп'ютера в Інтернет. Наявність безлічі пунктів запуску та маршрутів для кібернетичні атаки значно ускладнює можливість зупинити атаку до того, як вона досягне захисних бар'єрів відповідних комп'ютерів , а також визначити її джерело. Різні навмисні або ненавмисні посилання на інші комунікаційні мережі роблять їх потенційно вразливими до світових атак.

Об’єкт кваліфікаційної роботи – система захисту потоків даних.

Предмет кваліфікаційної роботи – застосування протоколів захисту та інструменти моніторингу потоків даних.

Мета і завдання кваліфікаційної роботи. Метою є вивчення та аналіз типових проблем в інформаційно-комунікаційній мережі, побудова системи захисту потоків даних, що унеможливлять або кардинально утруднять реалізації загроз або витоків для даних як в середині компанії так і зовні.

1 АНАЛІЗ ОБ'ЄКТА ЗАХИСТУ

1.1 Історія розвитку інформаційного простору і загрози інформаційної безпеки

У наші дні інформація обтікає всі комп'ютерні системи, як риба пливе по воді. Це представляє безліч можливостей для крадіжки даних; саме тому інформаційна безпека є необхідністю.

Якщо вдаватись до історії то інформаційна безпека з'явилась ще за довго до того, як почали використовувати різні канали зв'язку. Головною задачею ІБ до 1820 року був захист різного робу інформації, яка мала для суб'єкта особливе значення [1].

З'явився радіозв'язку та виникла необхідність забезпечити захист радіозв'язку від перешкод за допомогою кодування і декодування сигналу. В 1935 році з'явилися гідроакустичні та радіолокаційні засоби, безпека яких забезпечувалася через підвищення захищеності радіолокаційних засобів від впливу радіоелектронних перешкод.

Починаючи з 1946 року почали практичну діяльність електронно-обчислювальні-машини(ЕОМ). В основному захист цього обладнання здійснювався за допомогою обмеження фізичного доступу.

З 1960 років почали використовувати локальні мережі захист яких очолювався шляхом адміністрування та керуванням доступу мережевих потоків. Організації починають захищати свої комп'ютери. Найбільші проблеми безпеки в цей проміжок часу були у точках доступу. Той, хто має достатньо знань про те, як працювати з комп'ютером, може проникнути в об'єкт і отримати доступ до конфіденційних даних[2]. Для захисту терміналів та пристроїв були додані паролі та кілька рівнів захисту.

З 70-х років починаються перші хакерські атаки. На цей момент в історії інформаційної безпеки обчислювальна мережа знаходилася в зародковому

стані. Однак, хоча не існувало масивної глобальної мережі, що з'єднувала б усі пристрої, які бажали бути підключеними, великі організації, особливо уряди, починали зв'язувати комп'ютери за допомогою телефонних ліній. Визнавши це, люди почали шукати шляхи проникнення в телефонні лінії (Рисунок 1.1), підключені до комп’ютерів, щоб вони могли красти дані. Ці люди стали першими групами хакерів[3].

Рисунок 1.1 Схема підключення спеціальних посилювачів до телефонної лінії через адаптер

Уряди стали активнішими у боротьбі з кібернетичнізлочинністю. До 1980-х хакерство вже загрожувало проблемою міжнародної злочинності. Обмежені системи захисту інформації не могли встигнути за постійним шквалом розумних підходів хакерів, які використовувались для проникнення в комп’ютерні системи. Цей факт став надзвичайно відомим, коли невелика група підлітків з Мілуокі, відома як "414-ці", увірвалася у понад 60 військових та корпоративних комп'ютерних систем і викрала понад 70 мільйонів доларів у американських банках. У відповідь на цю кризу інформаційної безпеки уряди почали активно переслідувати хакерів, зокрема 414-х. На цей момент покарання були надзвичайно легкими - від суворих попереджень до умовного терміну [4].

В 90-х році всесвітня мережа стала доступною, люди почали розміщувати свою особисту інформацію в мережі; суб'єкти організованої злочинності розглядали це як потенційне джерело доходу і почали красти дані у людей та урядів через Інтернет. Захистити від цього допомогли брандмауери та антивірусні програми, але мережа була в основному незахищеною та швидко зростаючою мережею.

Кібернетичнізлочинність як злочин почали розцінювати у 2000-х. У той час як уряди переслідували кібернетичнізлочинців протягом десятиліть, більшість покарань були легкими, часто обмежуючись конфіскацією комп'ютерного обладнання та забороною користування комп'ютером протягом певного періоду. Це змінилося у 2000-х роках, коли уряди почали усвідомлювати небезпеку хакерства [5]. Хакери були ув'язнені роками в якості покарання за кібернетичнізлочинну діяльність. Наприклад, Джинсон Джеймс Анчета, який за допомогою хакерства вкрав менше мільйонної частки відсотків суми, яку вкрали "414-і", був засуджений до п'яти років ув'язнення. До 2010 року гучні хакери отримували десятиліття у в'язниці за кібернетичнізлочини.

У 2010 році ІБ набуває серйозності. Незважаючи на те, що кримінальне переслідування, брандмауери та антивірусне програмне забезпечення служили стримуючим фактором для кібернетичнізлочинців, вони не зупиняли хакерів, які були достатньо кваліфікованими та сміливими для проникнення в комп'ютерні мережі. На цьому етапі історії інформаційної безпеки експерти з безпеки почали розуміти, що найкращим способом захисту даних є зробити їх по-справжньому недоступними для хакерів. З цією метою ширше шифрування даних, яке кодує дані, щоб зробити їх нечитабельними для неавторизованих користувачів, набуло більш широкого поширення. У багатьох випадках шифрування відбувається на різних рівнях, включаючи цифрові файли, мережі та під час передачі даних. Зараз організації також впроваджують вичерпну політику інформаційної безпеки,

яка заважає їхнім працівникам робити помилки, які роблять дані доступними для зловмисників.

Під загрозою кібернетичні безпеки розуміється будь-яка можлива зловмисна атака, яка прагне незаконно отримати доступ до даних, порушити цифрові операції або пошкодити інформацію. Кібернетичнізагрози можуть походити від різних суб'єктів, включаючи корпоративних шпигунів, хактивістів, терористичні групи, ворожі національні держави, злочинні організації, одиноких хакерів та незадоволених працівників. В останні роки численні гучні кібернетичніатаки призвели до викриття конфіденційних даних. Наприклад, порушення 2017 року Equifax порушило особисті дані приблизно 143 мільйонів споживачів, включаючи дати народження, адреси та номери соціального страхування. У 2018 році Marriott International повідомила, що хакери отримали доступ до його серверів і викрали дані приблизно 500 мільйонів клієнтів. В обох випадках загроза кібернетичнібезпеці була увімкнена тим, що організація не впровадила, протестувала та перевірила технічні засоби захисту, такі як шифрування, автентифікація та брандмауери. Кібернетичні-зловмисники можуть використовувати конфіденційні дані особи чи компанії для викрадення інформації або отримання доступу до їх фінансових рахунків, серед інших потенційно шкідливих дій, саме тому фахівці з кібернетичні безпеки мають важливе значення для захисту приватних даних.

Фахівці з кібернетичні безпеки повинні глибоко розуміти наступні типи загроз кібернетичні безпеки.

Шкідливе програмне забезпечення - це зловмисне програмне забезпечення, таке як шпигунське програмне забезпечення, викупники, віруси та хробаки. Шкідливе програмне забезпечення активується, коли користувач натискає на шкідливе посилання або вкладення, що призводить до встановлення небезпечного програмного забезпечення. CISCO повідомляє, що після активації зловмисне програмне забезпечення може:

- Блокує доступ до ключових мережевих компонентів (програм-вимог)
- Встановлює додаткове шкідливе програмне забезпечення
- Таємно отримувати інформацію, передаючи дані з жорсткого диска (шпигунське програмне забезпечення)
- Порушує роботу окремих частин, роблячи систему непрацездатною

Агентство з питань кібернетичні безпеки та безпеки інфраструктури (CISA) описує EMOTE як «вдосконалений модульний банківський троянський контент, який в першу чергу функціонує як завантажувач або крапельниця інших банківських троянських програм. EMOTE продовжує залишатися одним із найдорожчих та руйнівних шкідливих програм».

Відмова в обслуговуванні (DOS) - це тип кібернетичні атаки, який заповнює комп'ютер або мережу, тому він не може відповісти на запити. Розподілений DOS (DDOS) робить те саме, але атака відбувається з комп'ютерної мережі. Кібернетичнінетичні-зловмисники часто використовують атаку повінь, щоб порушити процес "рукостискання" та виконати DOS. Можуть бути використані кілька інших методів, а деякі кібернетичні-зловмисники використовують час відключення мережі для запуску інших атак. БОТНЕТ - це тип DDOS, в якому мільйони систем можуть бути заражені шкідливим програмним забезпеченням і контролюватися хакером, за словами ДЖЕФФА МЕЛЬГИКА з NETWIX, компанії, що займається розробкою програмного забезпечення для захисту інформаційних технологій [15]. БОТНЕТ, які іноді називають системами зомбі, націлюють і перевершують можливості обробки цілі. БОТНЕТ знаходяться в різних географічних місцях і важко простежити.

Атака "посеред посередника" (MITM) відбувається, коли хакери вставляють себе в двосторонню транзакцію. Після переривання трафіку вони можуть фільтрувати та красти дані, повідомляє CISCO. Атаки MITM часто

трапляються, коли відвідувач використовує незахищену загальнодоступну мережу WI-FI. Зловмисники вставляються між відвідувачем та мережею, а потім використовують шкідливе програмне забезпечення для встановлення програмного забезпечення та зловмисного використання даних.

ФІШИНІНГ-атаки використовують підроблене спілкування, наприклад електронну пошту, щоб обдурити приймача, щоб він відкрив його і виконав інструкції всередині, наприклад, вказав номер кредитної картки. "Мета полягає у викраденні конфіденційних даних, таких як кредитна картка та інформація для входу, або встановлення шкідливого програмного забезпечення на машині жертви" [6].

Ін’єкція мови структурованих запитів (SQL) - це тип кібернетичні атаки, що виникає в результаті вставки шкідливого коду на сервер, який використовує SQL. При зараженні сервер видає інформацію. Надсилання зловмисного коду може бути простим, як введення його у вікно пошуку вразливого веб-сайту.

Маючи правильний пароль, кібернетичні-зловмисник має доступ до великої кількості інформації [14]. Соціальна інженерія - це тип атаки паролем, який Data Insider визначає як "стратегію, яку використовують кібернетичні-зловмисники, що значною мірою покладається на взаємодію людей і часто передбачає обман людей, які порушують стандартні практики безпеки". Інші типи атак паролем включають доступ до бази даних паролів або відгадування.

Порушення кібернетичної безпеки продовжують зростати як по частоті, так і по вишуканості для всіх галузей, а фінансовий сектор особливо вразливий. Фірми фінансових послуг стають жертвами атак кібернетичні безпеки набагато частіше, ніж підприємства інших галузей. Порушення безпеки призводять до втрати доходів банківських установ, перебоїв в роботі та втрати як репутації, так і клієнтів [13].

Клієнти банківських установ відмовляються від використання готівки та чек і більше покладаються на електронний БАНКІНГ для завершення операцій. У відповідь на цей зсув фінансові організації продовжують розробляти більше веб-порталів та мобільних додатків. Хоча ці програми та портали спрямовані на збільшення зручності та покращення взаємодії з клієнтами, вони становлять унікальний ризик з точки зору кібернетичні безпеки.

Дослідження ACCENTURE 2018 року розглянуло 30 основних банківських додатків і виявило, що всі 30 мали вразливі місця - від небезпечного зберігання даних до небезпечної автентифікації та фальсифікації коду. Більше того, подібне дослідження показало, що 85% перевірених веб-програм мали недоліки, які дозволяли б здійснювати кібернетичні атаки на користувачів[12].

Починаючи з відсутності безпечного зберігання даних і закінчуючи неефективною криптографією, існує низка причин, чому портали та програми для банківських послуг в Інтернеті становлять особливу загрозу:

- Відсутність безпеки сервера
- Небезпечне або неефективне зберігання даних
- Дані не захищені на транспортному рівні від сервера до клієнта та / або від клієнта до сервера
- Витік даних на стороні користувача
- Неадекватна автентифікація та авторизація під час входу користувача
- Неадекватне або неефективне шифрування
- Клієнтська ін’єкція (наприклад, ін’єкція або виконання шкідливого коду на мобільному пристрої через мобільний додаток)

1.2 Технології захисту даних

1.2.1 Резервне копіювання та відновлення даних

Протягом багатьох років було зроблено кілька вдосконалена у методах резервного копіювання даних - швидша швидкість копіювання, ціль точки відновлення та ціль часу відновлення [7]. Деякі новіші технології резервного копіювання даних включають наступне:

- Хмарне резервне копіювання: також відоме як резервне копіювання даних в Інтернеті, воно зберігає копію даних на віддалених або зовнішніх серверах. Залежно від того, де ви зберігаєте дані, скопіюйте свої локальні дані в хмару або виберіть резервне копіювання між хмарами. Ця технологія набагато дешевша, ніж багато варіантів локального зберігання. Він пропонує можливості відновлення після аварій, що робить його ідеальним для малого бізнесу. Хмарне програмне забезпечення для резервного копіювання також дозволяє резервне копіювання даних з мобільних пристроїв.
- Програмне забезпечення для зберігання: Програмне забезпечення управляє ресурсами та функціональними можливостями для зберігання даних, незалежно від основного обладнання фізичного сховища. Програмні засоби зберігання можуть розміщуватися на фізичних серверах або віртуальних машинах, звідки користувачі можуть контролювати кілька ресурсів зберігання. Він також надає функції для управління політикою зберігання даних, такі як де дуплікація, реплікація, знімки та резервне копіювання.
- Тверді диски та твердо тільні накопичувачі: дисководи - це швидкий та надійний засіб локального зберігання. Накопичувачі на жорстких дисках широко використовуються, оскільки вони зберігають терабайт даних і коштують менше, ніж хмарне сховище. Однак вони не мають простих варіантів відновлення, і диски можуть бути втрачені або знищені. З іншого боку, твердо тільні накопичувачі є більш досконалими (і дорогими),

підтримують швидше копіювання даних, мають більший термін служби та нижчий рівень відмов.

Особливості технологій резервного копіювання даних роблять їх надзвичайно корисними бо автоматичне резервне копіювання даних гарантує, що зміни, внесені у ваші файли, одночасно копіюються в місце зберігання [8]. Це дозволяє відновити навіть найновіші зміни у разі втрати даних, тим самим знижуючи ціль точки відновлення. Додаткове резервне копіювання. Це тип резервного копіювання, в якому копіюються лише зміни, а не повний файл. Це зменшує час, необхідний для копіювання даних, і не сповільнює вашу роботу. Миттєве відновлення дозволяє тимчасово запускати знімок резервної копії на вторинному сховищі, щоб зменшити час простою програми.

1.2.2 Хмарні технології захисту даних

Хмарні технології захисту даних гарантують, що дані, що зберігаються або передаються в хмару, не можуть бути зламу , підроблені або вкрадені. Ці інструменти також допомагають дотримуватися нормативних актів, таких як GDPR та CCPA , які вимагають захистити дані та забезпечити конфіденційність [9].

Ключові особливості хмарних технологій захисту даних:

- Шифрування : Всебічне шифрування файлів є основою всіх заходів безпеки в хмарі. Навіть якщо ваш CSP шифрує збережені дані, додайте додаткове шифрування, щоб приховати дані, що завантажуються в хмару. Переконайтесь, що дані, що перебувають у стані спокою, і дані, що передаються, шифруються, щоб мінімізувати порушення / втрату даних.
- ТОКЕНІЗАЦІЯ: Цей процес замінює чутливий елемент даних нечутливим еквівалентом, який називається маркером. ТОКЕН діє як заповнювач, тоді як конфіденційні дані зберігаються в іншому місці. На

відміну від шифрування, ТОКЕНІЗАЦІЯ не може бути скасована. Він в основному використовується для зберігання конфіденційних фінансових даних. Регулюючі органи, такі як Рада з питань безпеки PCI, розглядають це як більш безпечну форму захисту даних.

- Безпека кінцевої точки : Ви повинні захистити пристрої кінцевих користувачів, які отримують доступ до хмарних ресурсів, щоб гарантувати, що вони не націлені хакерами. Брандмауери та антивірусні рішення можуть допомогти захистити ваші кінцевих користувачів, які підписані на інструменти IAAS, SAAS або PAAS.
- Контроль автентифікації та доступу : Ці інструменти гарантують, що лише авторизовані люди мають права доступу та модифікації ваших хмарних даних. Це підтримує цілісність даних і зменшує крадіжку даних. Засоби управління привілейованим доступом перевіряють особу користувачів і дозволяють лише уповноваженим особам переглядати або змінювати дані.

1.2.3 BLOCKCHAIN

BLOCKCHAIN - це розподілена книга, яка додає нові записи з позначками часу та посиланнями на попередні записи. Всі записи в БЛОКЧЕЙНІ захищені за допомогою криптографії, кожен учасник мережі перевіряє правдивість даних за допомогою приватних, відкритих та приймаючих ключів. Після додавання даних до БЛОКЧЕЙНУ їх неможливо змінити або змінити [10].

Наступні особливості БЛОКЧЕЙНУ роблять його корисним для безпеки даних:

- Децентралізація: замість того, щоб завантажувати дані на хмарний сервер або зберігати їх в одному місці, БЛОКЧЕЙН розбиває їх на фрагменти та розподіляє по мережі комп’ютерів. Кожен комп'ютер або вузол у мережі

матиме копію даних. Це робить маніпулювання даними на БЛОКЧЕЙНІ майже неможливим.

- Шифрування: Дані, додані або збережені в мережі БЛОКЧЕЙНІВ, зашифровані. Технологія використовує методи криптографії, такі як відкриті ключі, для захисту ідентифікації користувачів та забезпечення безпечних транзакцій.
- Перевірка: Модель БЛОКЧЕЙНУ дозволяє легко перевірити збережені дані через їх розподілений характер. Ви можете перевірити підписи файлів між книгами на всіх вузлах мережі та переконатися, що дані не були змінені.

1.3 Висновки

Збільшення використання інформаційних технологій та Інтернету гарантує, що захист даних залишається одним із найважливіших та найактуальніших законів, яких зобов’язані дотримуватися онлайн-компанії. Інтернет - це все про передачу інформації. Інтернет використовується не лише для розповсюдження інформації, а й для її збору. Організації повинні зараз поглянути на те, як вони збирають, зберігають та використовують персональні дані, і запитати себе, чи відповідають вони Закону. Це може передбачати внесення змін до практики зайнятості та маркетингу на додаток до внутрішнього навчання

2 ОБҐРУТНУВАННЯ ВИБОРУ ЗАСОБІВ ДЛЯ ПОБУДОВИ СИСТЕМИ АВТЕНТИФІКАЦІЇ

2.1 Опис мови програмування С++

Для розробки проекту я використовував мову С++, тому що це низько рівняно, складна мова програмування, яка використовується для перетворення машинного коду після компіляції. Також має класи та об’єкти що є великим плюсом в нашій системі [11]. Мова С ++ добре підходить для проектів, де потрібно більше контролю. Це кращий вибір для розробників, які створюють частини веб-додатків, що вимагають великої швидкості. С ++ також забезпечує швидший код виконання. Більша швидкість перетворюється на кращу продуктивність веб-сайту для користувачів загалом. Незважаючи на це, його складність потрібно враховувати щодо використання для веб-розробки. Мова має більше апаратних засобів управління на ПК або сервері. С++ я обрав тому що:

- Підтримує концепції ООРS;
- Низький рівень абстракції;
- Управління пам'яттю здійснюється вручну;
- Продуктивність надзвичайно висока;
- Незалежна від платформи і можна писати програми для будь-якої ОС (платформи);
- Гнучкий код, ви можете кодувати що завгодно, компілятор не генерує попереджень, якщо синтаксис неправильний;
- Підходить для високопродуктивних додатків, таких як ігри, драйвери пристроїв та додатки на стороні сервера;
- Реалізує загальні засоби за допомогою шаблонів
- Підтримує множинне успадкування

Також разом з C++ будемо використовувати MATLAB. У MATLAB ви можете розширити свій код C та C ++ за допомогою функції MEX і викликати його як будь-яку вбудовану функцію MATLAB. Це означає, що ви можете використовувати існуючий код C і C ++, не переписуючи свої алгоритми в MATLAB. Функції MEX дозволяють коду C і C ++ створювати та змінювати масиви MATLAB у робочій області MATLAB. За допомогою функцій C ++ MEX ви можете безпосередньо отримувати доступ до даних MATLAB, використовуючи стандартну бібліотеку C ++ без зайвих копій даних.

2.2 Опис системи управління базами даних

Для зберігання даних у нашій системі було обрано СУБД MS SQL Server. Завдяки вбудованим прозорим функціям стиснення та шифрування даних, сервер SQL пропонує підвищену продуктивність. Щоб захистити та зашифрувати дані, користувачам не потрібно змінювати програми. SQL Server надає ефективні засоби управління дозволами із засобами контролю доступу, призначеними для захисту користувачів конфіденційної ділової інформації.

База даних SQL Server є надзвичайно безпечною і використовує складні алгоритми шифрування, що робить практично неможливим порушення рівня безпеки. SQL Server - це комерційна реляційна база даних з додатковими функціями безпеки для зменшення ризику атак.

SQL Server складається з декількох складних функцій, які допомагають відновити та відновити втрачені або пошкоджені дані. За допомогою вдосконалених інструментів відновлення можна відновити повну базу даних. Основний компонент SQL Server, DATABASE ENGINE, контролює зберігання даних і допомагає виконувати запити та запити користувачів,

включаючи транзакції, файли та індекси. Великі організації зазвичай використовують ці засоби SQL Server.

Ефективний інтелектуальний аналіз даних, розділення дисків та інструменти управління даними SQL-сервера допомагають підтримувати важливі дані та забезпечують доступ до місця для зберігання високочутливої інформації.

Також можна використовувати додаткові інструменти по типу:

- Аналізатор продуктивності бази даних SOLARWINDS (DPA)
- SQL SENTRY
- Програмне забезпечення для моніторингу PAESSLER SQL
- DBWATCH Керування базою даних
- Моніторинг SQL Server SPICEWORKS

2.3 Середовище розробки

Для розробки програми було обрано програму VISUAL STUDIO ENTERPRISE. Тут можна виконувати архітектурну перевірку програми за допомогою діаграм архітектурного шару . Наприклад, ви можете інтегрувати зазначену перевірку у свій процес побудови. Якби клас посилався на простір імен у якомусь шарі, до якого він не повинен був звертатися, збірка не вдалася.

В останній версії цю функцію також вдосконалено, щоб запропонувати перевірку залежносте в режимі реального часу. Замість того, щоб чекати, поки збірка перерветься, VISUAL STUDIO, використовуючи потужність аналізаторів ROSLYN, надаватиме вам зворотний зв’язок у реальному часі кожного разу, коли ви збираєтесь ввести недійсну залежність. Ще має багато функцій для тестів коду.

Саме ці причини подвигали мною обрати збірку VISUAL STUDIO для нашої системи автентифікації.

3 ПРОЕКТУВАННЯ СИСТЕМИ АВТЕНТИФІКАЦІЇ

3.1 Система біометричної автентифікації

3.1.1 Типова архітектура

Основною метою розроблювальної системи, є те щоб уникнути використанню стандартний ідентифікаторів логіну та паролю. Загальна структура автентифікації представлена на рисунку Рисунок 3.1

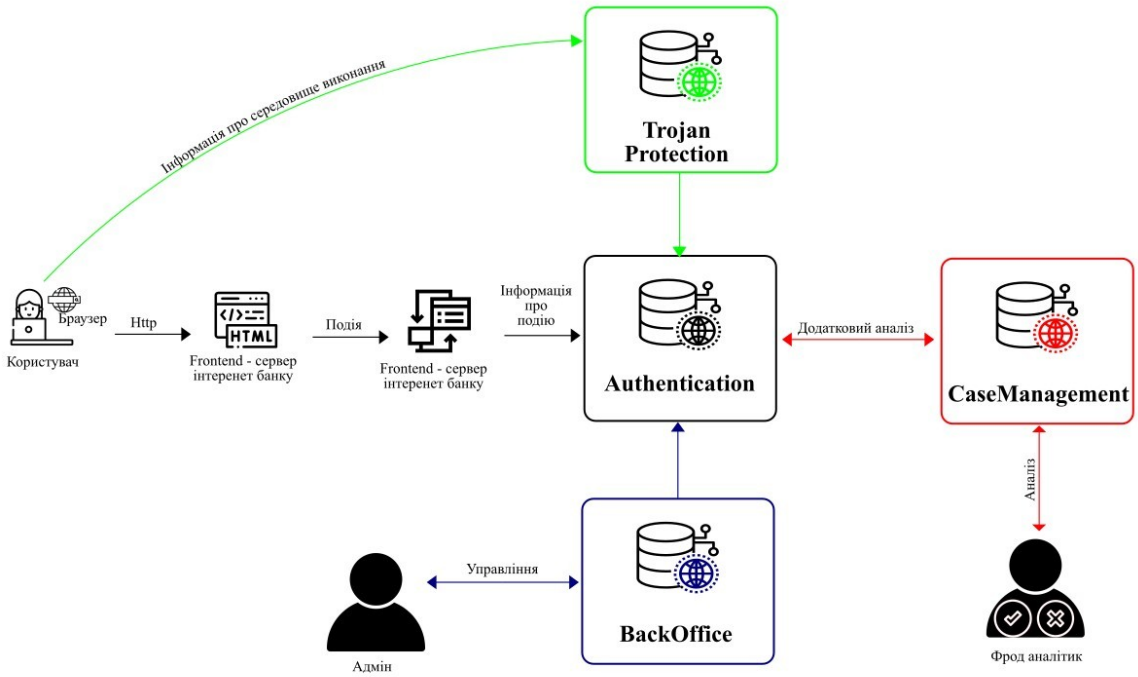


Рисунок 3.1 Загальна структура автентифікації

Для покращення системи автентифікації, було запропоновано другий варіант де на стороні сервера буде обраховуватись хеш код та не будуть зберігатись біометричні дані, а будуть зберігатись тільки шаблони. Шаблон не буде нести ніякої користі для зловмисника, тому що без оригіналу відбитка пальця, увійти в систему буде не можливо. Також як варіант була додана додаткова автентифікація, якщо система запідозрить якісь підозрілі

дії. Під додатковою автентифікація мається сканування лиця або сітківки ока. На сервері будуть працювати алгоритми покращення зображення щоб мінімізувати кількість невдалих спроб автентифікації. Для відбитків пальців будуть працювати певні фільтри які якісно будуть сегментувати палець, знаходити хребти, іншими словами покращувати пізнавальність відбитка. На основі цих даних які будуть зчитуватись буде генеруватися хеш-код. І порівнювати від сканований хеш, та хеш при першій реєстрації. Для лиця теж є певний алгоритм впізнавання.

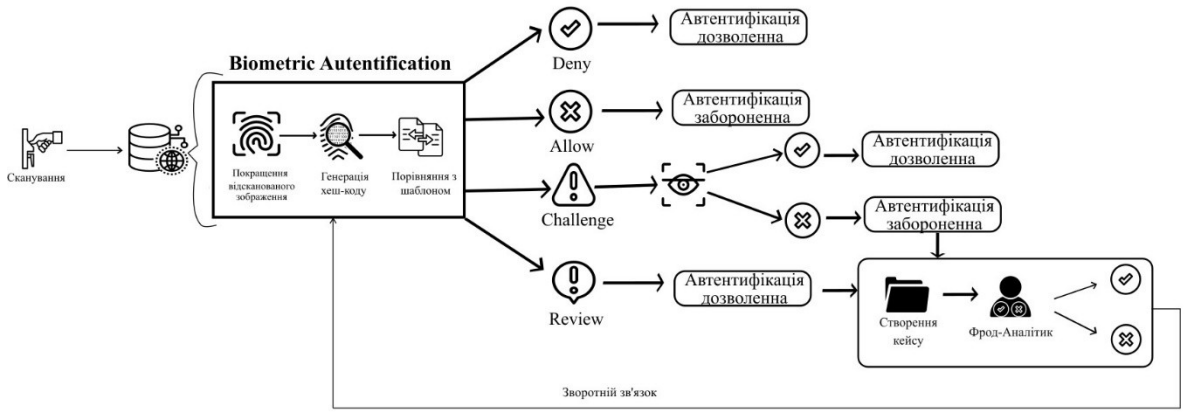


Рисунок 3.2 Алгоритм роботи розроблювальної автентифікації

3.1.2 Методи підвищення покращення зображення відбитків пальців

Методи покращення зображення відбитків пальців в основному зосереджені на зменшенні шумів можливо в навчальних наборах даних зображення або вибраного зображення і отримати високі бали в процесі узгодження. Точність зображення тестових наборів даних безпосередньо залежить від якості зображення в навчальному наборі даних.

Мною було запропоновано декілька методів покращення зображення.

Перший це гістограма модель зображення відбитка пальця відображає, як часто або скільки разів рівень сірого виникає в групі загальних рівнів сірого зображення. Вихідне зображення, отримане гістограмою, містить шум

з точки зору рівнів інтенсивності. Вирівнювання гістограми - це спеціальний прийом для регулювання інтенсивності або яскравості та підвищення контрастності зображення. За допомогою вирівнювання гістограм ми можемо отримати рівномірну або ідентичну гістограму для вихідного зображення. Ця техніка в основному використовується для покращення контрасту зображення шляхом регулювання індивідуального рівня сірого зображення. Результати гістограми наведені на рисунку 3.3

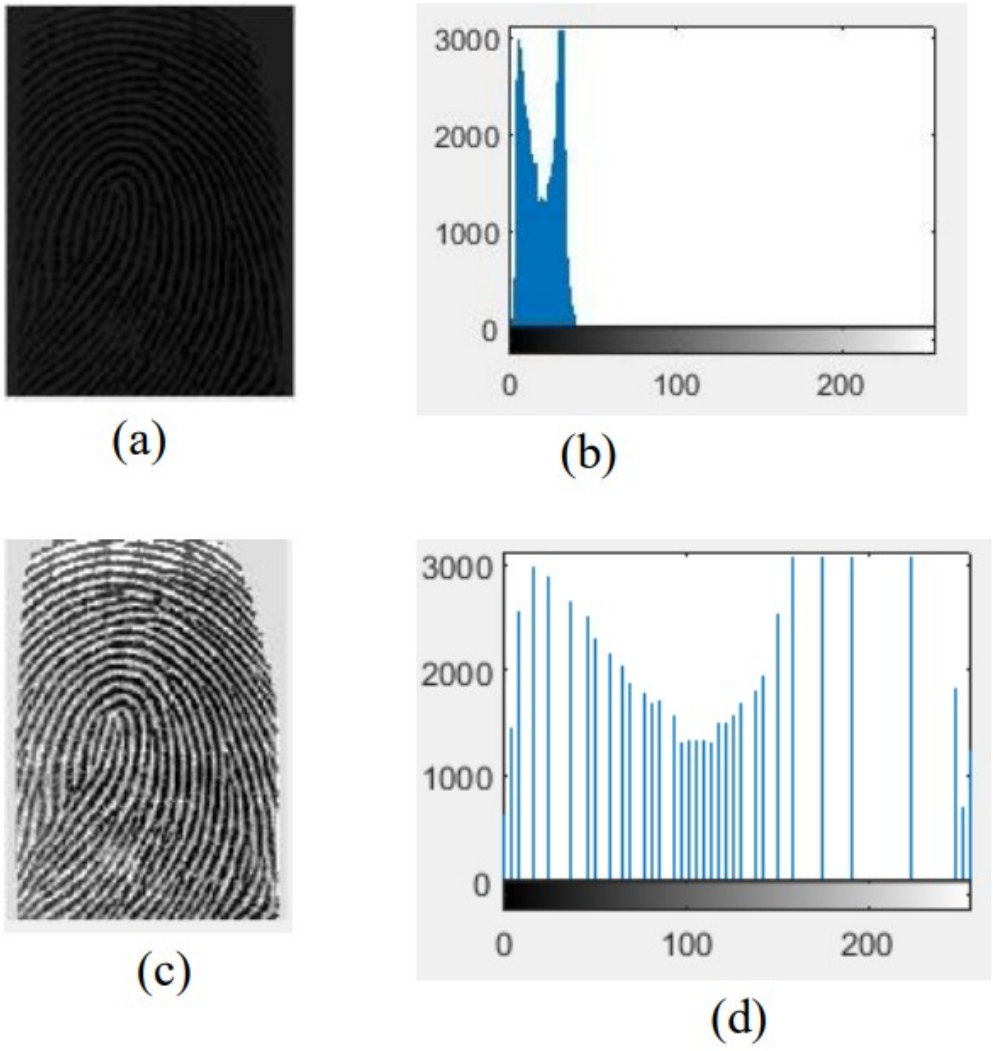


Рисунок 3.3 - (a) Оригінальне зображення (b) Гістограма оригінального зображення (c) Гістограма вирівняного зображення (d) Гістограма вирівняного зображення.

Далі йдуть методи фільтрування. Кінцева мета цих методів - усунути всі типи помилок, що виникають у вхідному чи початковому зображенні, та покращити здатність розпізнавання зображень у всіх аспектах. Нижче розглянуто декілька методів.

Медіанна фільтрація враховує статистичну медіану пікселів, що містяться навколо пікселя, і цей нелінійний процес фільтрації використовується для видалення імпульсивних шумів та підвищення якості зображення відбитків пальців. Медіанна фільтрація показана на рисунку 3.4.

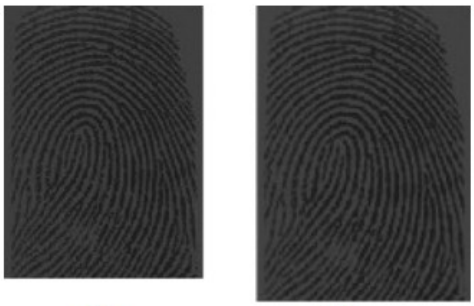


Рисунок 3.4 (a) Зображення до медіанної фільтрації (b) Зображення після застосування медіанної фільтрації

Медіана обчислюється наступним чином V (m, n) = медіана {y (m-k, n-1), (k, 1) є W} . Де W - вибране Вікно. Як і статистичні показники, для медіанної фільтрації потрібні піксельні значення вікна, розташовані по порядку.

Наступна фільтрація - фільтрація високих частот. В основному використовується для вилучення країв зображень. Фільтр високих частот допомагає поліпшити якість зображення за рахунок загострення країв зображення відбитків пальців. З цієї причини завжди є гарною практикою робити фільтр високих частот для вихідного зображення. Високочастотна фільтрація просто видаляє або розмиває зображення з вихідного зображення. Високочастотна фільтрація показана на рисунку 3.5.

Рисунок 3.5 (a) Зображення до фільтрації високих частот (b) Зображення після застосування фільтра високих частот

Фільтрація Вайнера дає зображення високої якості, видаляючи додаткові шуми, навіть коли зображення має розмитість або низьку інтенсивність. Це мінімізує максимальну похибку в процесі шуму згладжування, для поліпшення якості зображення. На рисунку 3.6 показано фільтрування Вайнера.

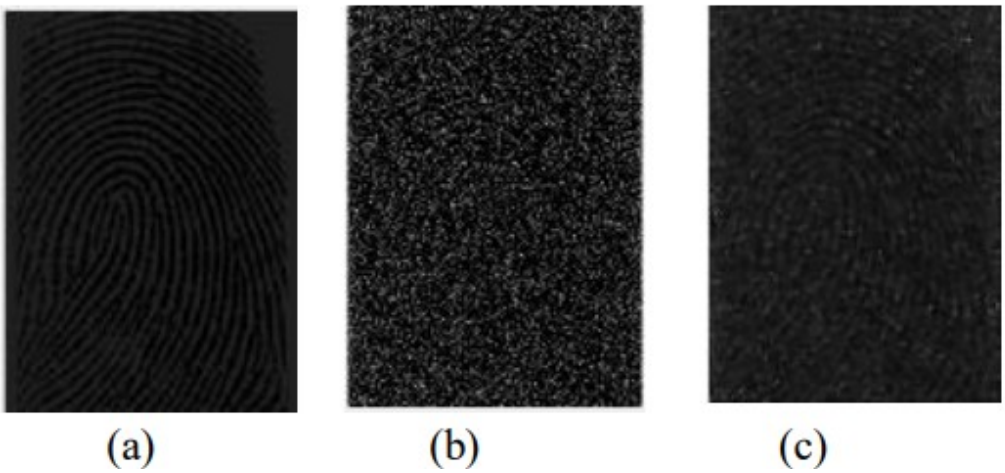


Рисунок 3.6 (a) Оригінальне зображення (b) Зображення після застосування шуму (c) Зображення після застосування фільтра Вайнера

Та останій метод бінаризація та проріджування. Бінаризація є одним із етапів попередньої обробки в автоматичних системах розпізнавання відбитків пальців. Якщо вхідним зображенням є кольорове зображення,

спочатку воно повинно бути перетворене в зображення масштабу сірого. Зі шкали сірого зображення отримують двійкове зображення, розглядаючи лише два стани як нуль для хребтів, які представлені чорним кольором, і один для долини, який представлений білим кольором. У бінаризації ми задали деякий поріг для пікселів, а піксель, який нижчий і вищий, ніж цей поріг, представлений білим і чорним кольором відповідно. Проріджування - це спеціальний процес, який послідовно стирає пікселі переднього плану і, нарешті, створює лінії, що мають ширину майже в один піксель. Першою і найголовнішою умовою для стоншення є вхідне зображення, яке повинно бути двійковим зображенням і створюватиме вихід у вигляді двійкового зображення. Витончення - це останній попередній крок до вилучення дрібниць в системі автоматичного розпізнавання відбитків пальців. Витончення не досягається за один крок, а досягається за допомогою ітераційного процесу. Зв'язок хребтів і роздвоєння можна відтворити за допомогою витончення, тобто воно зберігає основну структуру зображення, не впливаючи на його початкову структуру. На рисунку 3.7 показано процес бінаризації та витончення вихідного зображення відбитків пальців.

Рисунок 3.7 (a) Оригінальне зображення (b) Зображення після бінаризації (c) Розріджувальне зображення.

Даний метод найкраще підходить до нашої системи. Тому в наступний алгоритмах ми будемо використовувати саме його.

3.2 Технічне завдання на розроблювану систему

Назва розроблюваної системи: «Система захисту потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ «ПриватБанк».

Основною метою даної розроблювана системи є введення превентивних мір по забезпечені захисту від витоків інформації та впровадження біометричної автентифікації працівників в середині відділу.

Додатковою метою створення даної системи є впровадження біометричної автентифікації клієнтів банку без використання стандартних ідентифікаторів.

Системи призначена для запобігання використанню чужих ідентифікаційних даних.

Сферою застосування є забезпечення захисту від витоків конфіденційної інформації в середині компанії, захист користувачів від ФРОДУ.

В проекті потрібно зробити сервер з адаптивною автентифікацією, серед функцій якого буде:

- Реєструвати дії.
- Надсилати журнали на інший комп’ютер.
- Фізичне розділення даних облікового запису користувача на окремих кластерах баз даних.
- Користувацький, груповий та рольовий контроль доступу до певних сторінок, звітів та елементів звіту.
- Обмеження рівня видимості даних на рівні користувача (наприклад, обмеження для певного підрозділу чи офісу).
- Ізоляція баз даних на рівні мережі від серверів, що обробляють Інтернет-трафік.
- Покращення зображення відбитків пальців.

- Збереження шаблону відбитку.
- Відслідковування підключення зовнішніх пристроїв.
- Створення хеш функцій для автентифікації.
- Перевірка на справжність біологічного матеріалу.

3.3 Опис функціонування системи

В розроблювальній системі для автентифікації працівника спочатку треба зареєструвати цього працівника в базі. Реєстрація працівника буде відбуватись таким чином. Працівник від сканує відбиток пальця та біометричне сканування захопить зображення пальця. Для додаткової автентифікації працівник за допомогою біометричного терміналу від сканує силует долоні, сітківку ока, райдужку ока, обличчя. Додаткова біометрія буде спотворюватись як на рисунку 3.8.

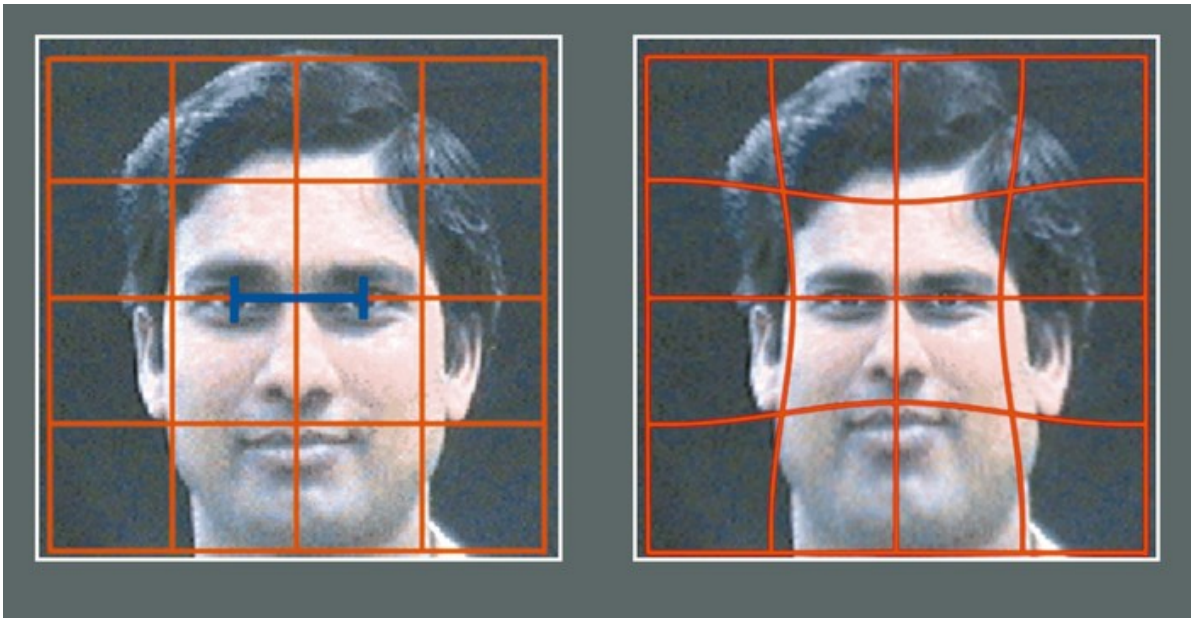


Рисунок 3.8 Спотворення зображення обличчя

На рисунку показана оригінальна фотографія з накладеною сіткою, де вирівнювання відбувається по особових ознаках. Поруч з нею - фотографія зі зміненою сіткою і результат спотворення обличчя. У шаблон будуть записуватись унікальні риси по яким потім можна буде порівняти від скановане обличчя.

Далі алгоритм буде покращувати зображення, та витягувати унікальні риси з відбитку пальця або додаткових автентифікації. Наступний крок це генерування відкритого та закритого ключа. Закритий ключ буде хешувати. Формується шаблон на основі хеш-коду. Далі зберігається відкритий ключ, закритий ключ стирається з системи. Ніяких біометричних даних не зберігається ні в одному випадку.

Для автентифікації працівник сканує палець. Алгоритм витягує теж самі унікальні риси що й при реєстрації. Публічний код повідомляє системі де знаходиться шаблон з приватним кодом. Створюється один і той самий закритий ключ, для автентифікації видаються одні ж і ті ж хеш та криптографічні ключі. У разі декількох не спів падінь, система запросить додаткову автентифікацію.

В запропонованій системі(рисунок 3.2) є можливість модифікації, яка посилить надійність системи від несанкціонованого доступу. Біометричний термінал який запропонований в розділі 4.3 ZKTECO SpeedFace-V5L має функцію вимірювання температури.

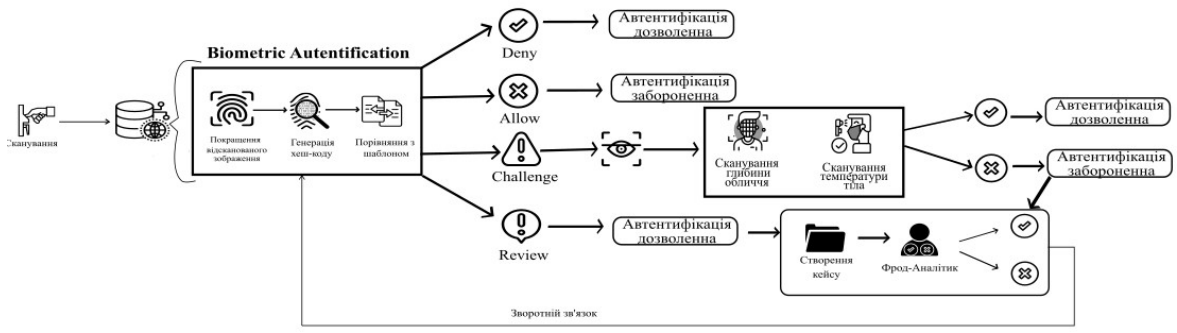


Рисунок 3.9 Модифікована версія розроблювальної системи

Сканування глибини обличчя та сканування температури тіла дозволить системі впевнитись що це не фотографія, муляж і так далі.

3.4 Висновки

В даному розділі ми спроектували концепт методів вдосконалення зображень відбитків пальців. Наведені тут схеми отримані шляхом написання коду на C++ та запуску в MATLAB. На систему розпізнавання відбитків пальців сильно впливає якість вхідного зображення, і якщо ми застосуємо методи вдосконалення зображення до вхідного зображення, рівень шуму може бути зменшений, а також покращена робота системи розпізнавання. Хотілося б, щоб ця робота могла відігравати активну роль у процесі покращення зображення в системі автоматичного розпізнавання відбитків пальців.

4 РЕАЛІЗАЦІЯ РОБОТИ

4.1 Розробка блок-схем і опис алгоритмів функціонування системи

Для реалізації даної автентифікації потрібно хешувати дані отримані з від сканованого пальця. Тому була розроблена наступна блок схема (Рисунок 4.1). Зображення базового набору даних є БІНАРИЗОВАНИМ, і для кожного пікселя обчислюється евклідова відстань зображення. Розглядаються різні значення значень матриць відстані Евкліда та генерується хеш-код довжиною 32 біти. Виразне підсумовування значень відстані Евкліда, середнє значення та значення стандартного відхилення враховуються для формування хеш-коду.

З метою зробити хеш-код MD5 більш надійним, сума, середнє та стандартне відхилення поєднуються та передаються в алгоритм MD5.

- Перетворення вхідного зображення у зображення у відтінках сірого 256 × 256.
- Перетворення у двійкове зображення. Знаходження евклідової відстані.
- Пошук чіткого значення евклідової відстані.
- Знаходження суми різної евклідової відстані.
- Знаходження середнього значення чіткої евклідової відстані.
- Знаходження середньоквадратичного відхилення чіткої евклідової відстані.
- Генерування хеш-коду із використанням комбінованої суми, середнього та стандартного відхилення різного значення Евклідової відстані

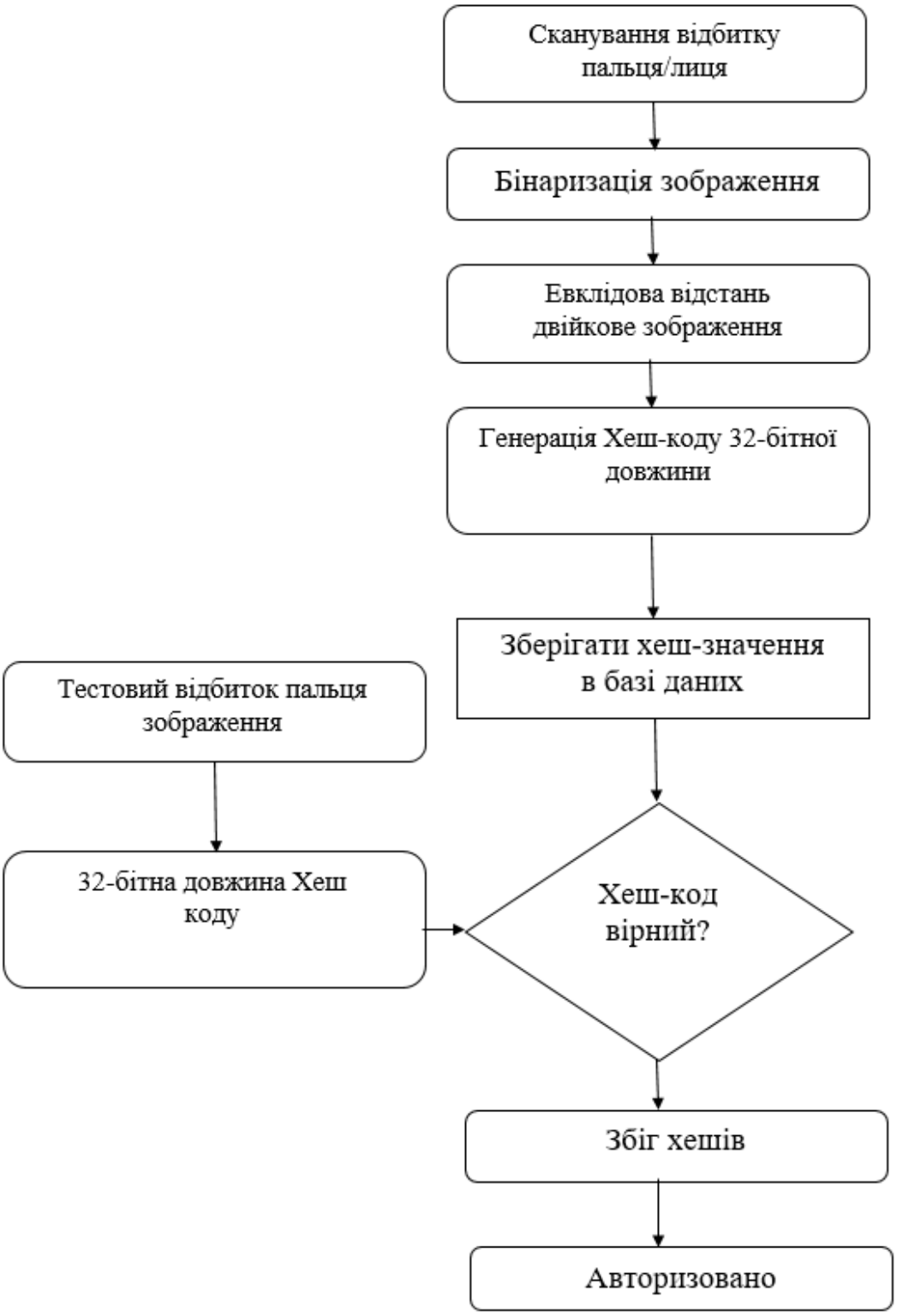


Рисунок 4.1 - Блок-схема алгоритм запитів на автентифікацію користувача

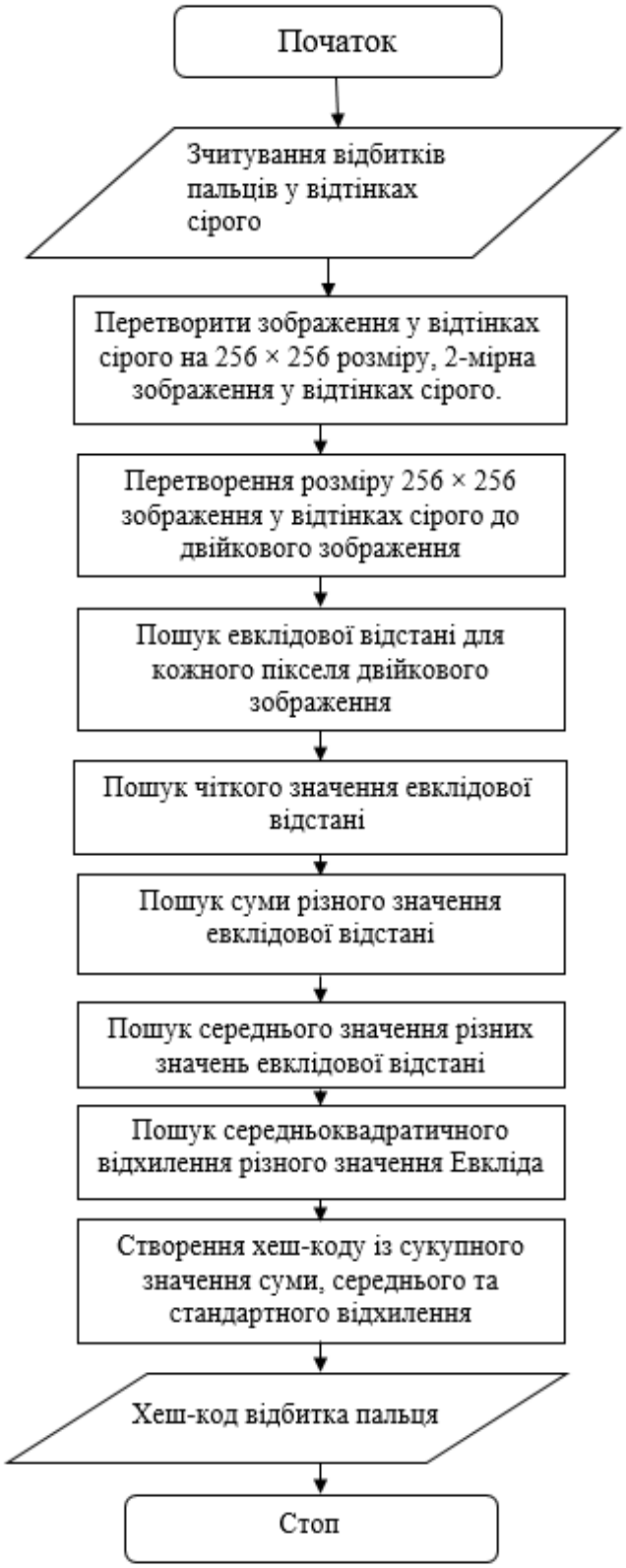


Рисунок 4.2 Блок-схема генерування хеш-коду з використанням евклідової відстані

4.2 Тестування системи

Для тестування системи будемо використовувати MATLAB 2015a. Розглянемо алгоритм сегментації на основі двовимірного відсікання.

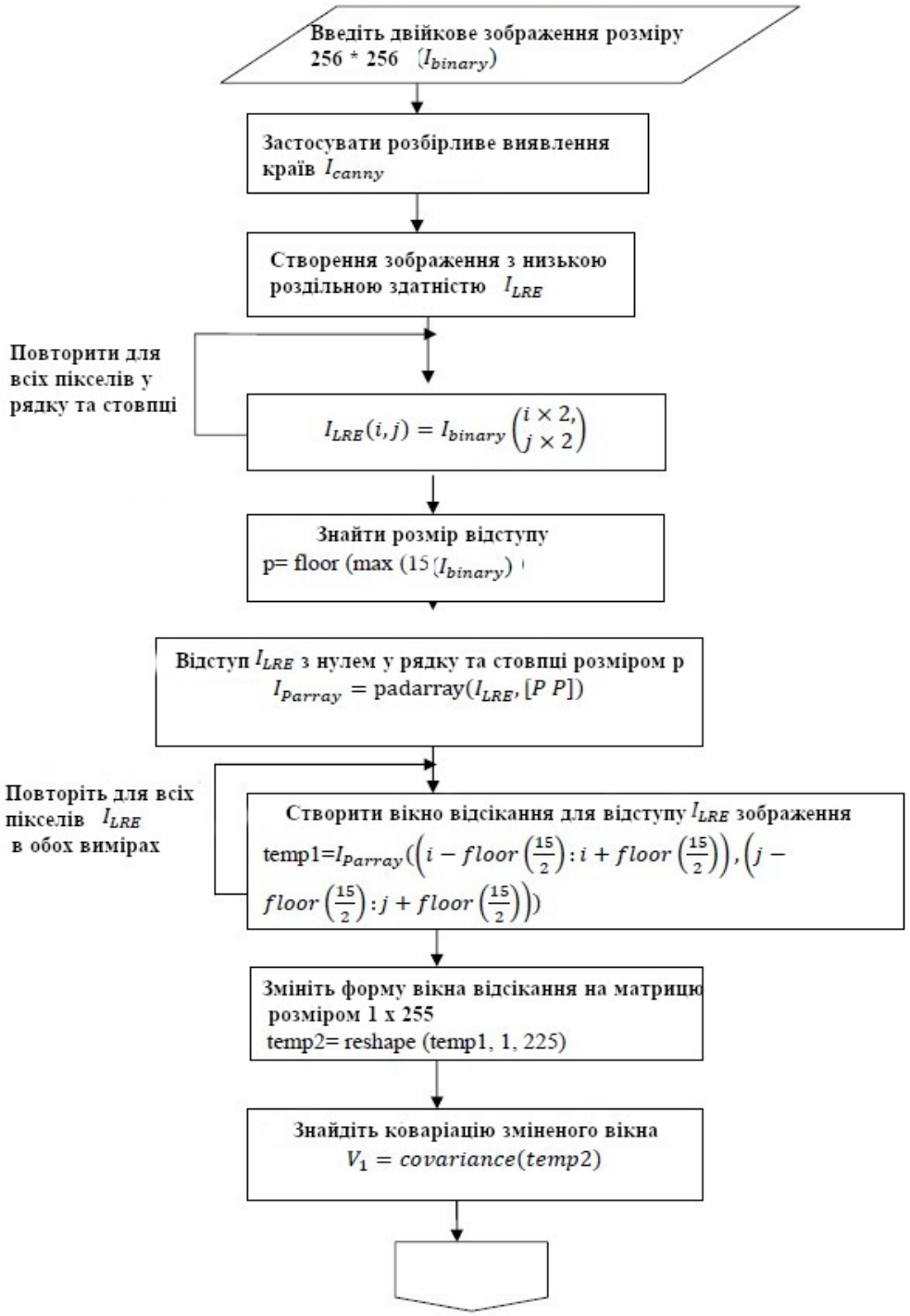
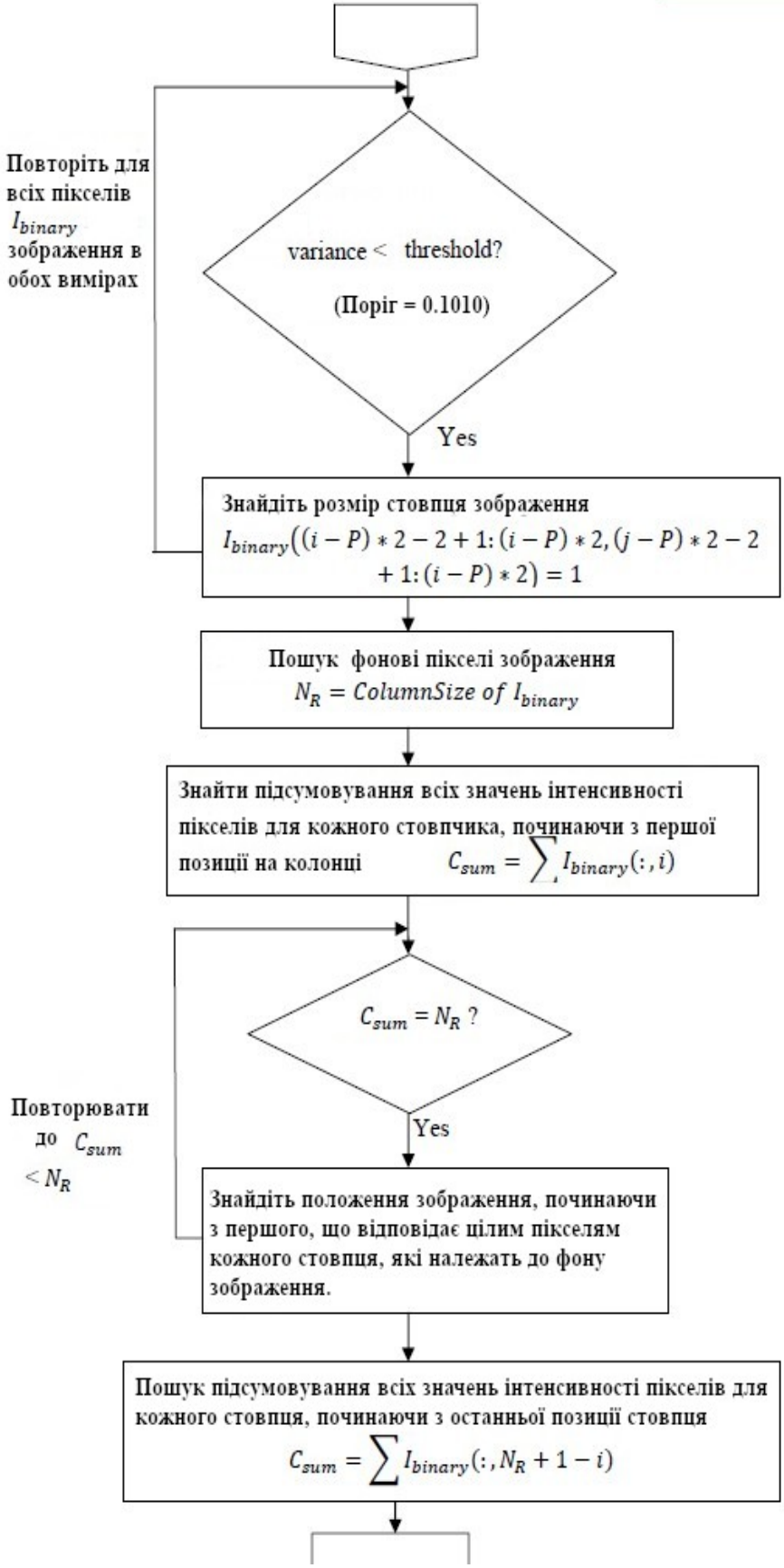
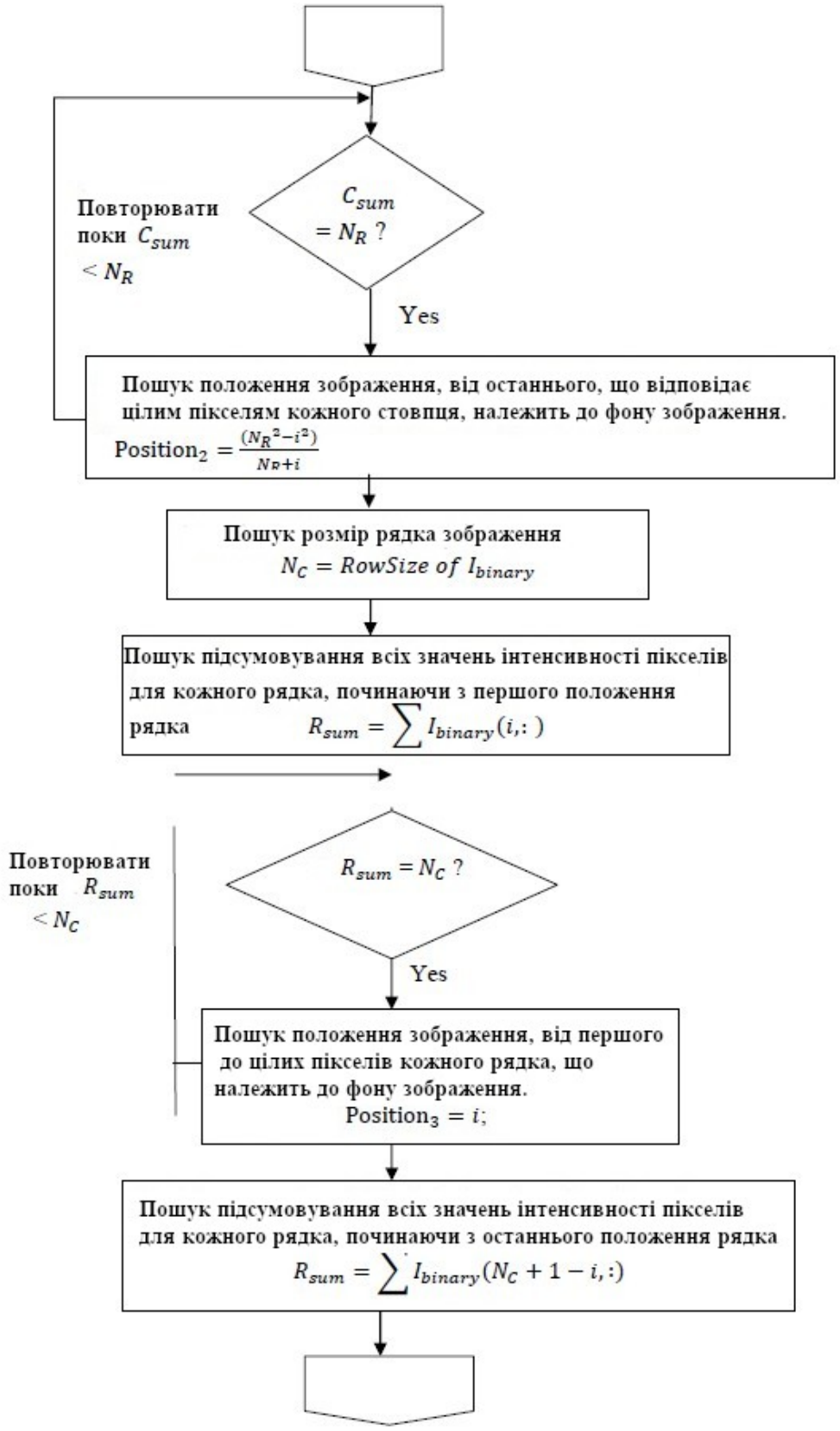


Рисунок 4.3 Блок-схема алгоритму сегментації на основі двовимірного відсікання

Продовження рисунка 4.3



Продовження рисунка 4.3



Кінець рисунка 4.3

Цей запропонований алгоритм сегментації пояснюється за допомогою блок-схеми. Вхідні дані для цього алгоритму

- це двійкове зображення розміром 256 × 256, яке представляється як I_{binary}. Кінцевий результат сегментується зображення, позначене як, $I_{segment}$. Різні робочі процеси запропонованого алгоритму наведені нижче.

- Знайдіть край зображення, використовуючи хибний метод виявлення країв

- Створіть зображення з низькою роздільною здатністю із зображення у градаціях сірого 256 × 256

- Знайдіть розмір відступу.

- Покладіть зображення з низькою роздільною здатністю нулем по напрямках рядків і стовпців.

- Створіть вікно відсікання розміром 15 × 15 для заповненого зображення з низькою роздільною здатністю.

- Змінити форму відсічної вдови як вікно розміром 1 × 256

- Знайдіть ко варіацію матриці зображення для кожного елемента вхідного зображення через заповнене зображення.

- Якщо ко варіація менша за порогову, розглядайте її як фонове зображення, інакше розглядайте його як зображення переднього плану.

- Відкиньте кожен стовпець зображення, якщо він повністю містить значення інтенсивності 1 як з лівого, так і з правого напрямку зображення.

- Відкиньте кожен рядок зображення, якщо він повністю містить значення інтенсивності 1 як з верхнього, так і з нижнього напрямку зображення.

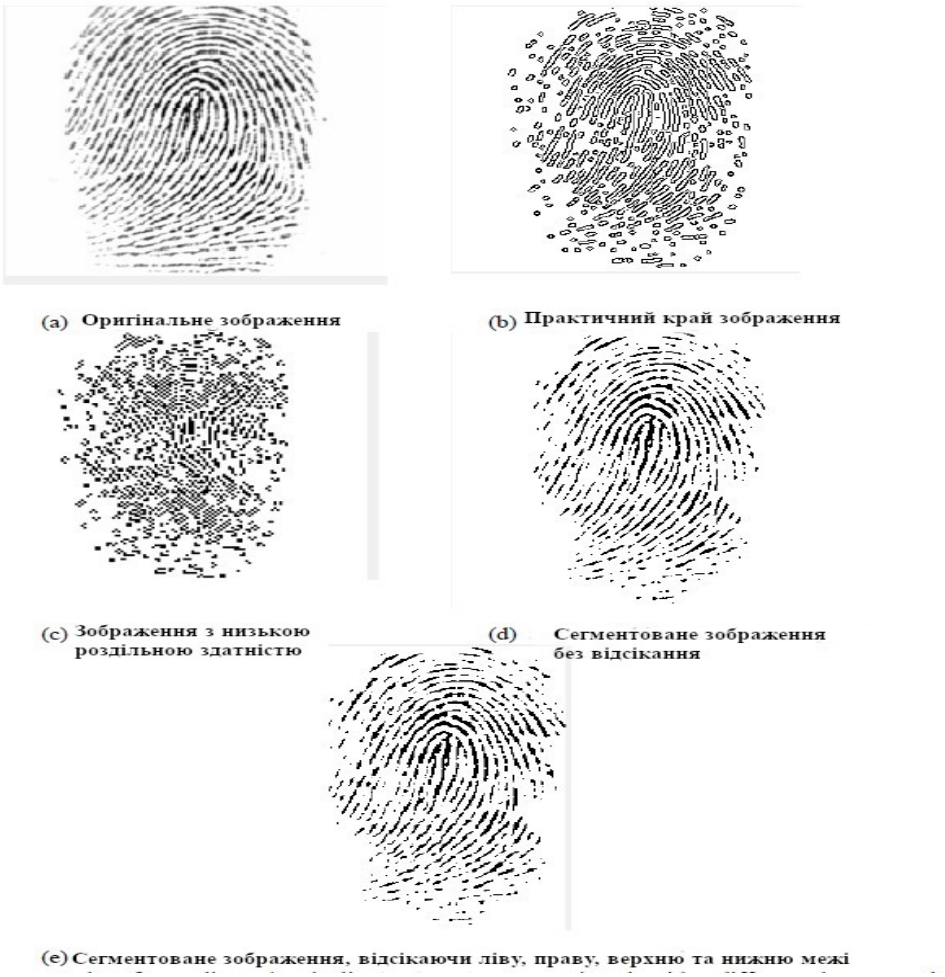


Рисунок 4.4 Приклади алгоритму сегментації на основі двовимірного відсікання, результати різних фаз

У таблиці 1 наведено ребра, отримані за допомогою методу виявлення ребра. Загальна кількість ребра розглядається як загальна кількість пікселів.

Таблиця 3.1 Загальна кількість країв, виявлених за допомогою хитрого виявлення країв

Номер	Назва зображення	Загальна кількість країв, виявлених за допомогою хитрого виявлення краю
1	1_1.tif	8764
2	1_2.tif	8309
3	1_3.tif	10123
4	1_4.tif	10503
5	1_5.tif	7465
6	1_6.tif	11023
7	1_7.tif	10141

На рисунку (4.5) показано вхідне зображення та результати процесу сегментації за допомогою двовимірного відсікання

заснована сегментація для різних зразків зображень поточних наборів даних DB1_B FVC. Переглядаючи два зображення, ми не знаходимо жодних відмінностей. Але якщо ми уважно спостерігаємо після сегментації, відсікається ліва та права частина відфільтрованого зображення, що відповідає фоновим пікселям.

(a) Вхідне зображення (b) Сегментоване зображення

Рисунок 4.5 Приклади введеного та сегментованого зображення із використанням запропонованого методу

У таблиці 4.2 наведено загальну кількість пікселів до сегментації та після сегментації. Якщо кількість пікселів зменшується, це покращує продуктивність виконання або швидкість на наступних етапах автоматичних систем ідентифікації відбитків пальців, таких як формування дрібниць, вилучення особливостей та узгодження.

Таблиця 4.2 Порівняння загальної кількості пікселів до та після сегментації

№	Назва зображення	Розмір зображення до фрагментації	Розмір зображення після фрагментації	Загальна кількість пікселів до сегментації	Загальна кількість пікселів після сегментації
1	1_1.tif	256 × 256	230 × 148	65536	34048
2	1_2.tif	256 × 256	251 × 149	65536	37417
3	1_3.tif	256 × 256	228 × 183	65536	41802

Продовження таблиці 4.2

4	1_4.tif	256 × 256	238 × 156	65536	42875
5	1_5.tif	256 × 256	213 × 179	65536	36109
6	1_6.tif	256 × 256	244 × 138	65536	38336
7	1_7.tif	256 × 256	250 × 151	65536	39189

У таблиці 4.3 наведено час виконання алгоритму сегментації на основі двовимірного відсікання.

Час виконання обчислюється на основі тривалості або часу між входом і виходом. Час виконання розраховується на двох ноутбуках різної конфігурації. Один з них називається LAPTOP-I, а інший - LAPTOP-II. Конфігурація LAPTOP-I та LAPTOP-II наведена в таблиці 4.4.

Таблиця 4.3 Час виконання для різних зразків зображень із використанням двовимірної відсікання на основі сегментації

№	Назва зображення	Час виконання в секундах на пристрої LAPTOP-I	Середній час виконання, використовуючи LAPTOP-I	Час виконання в секундах на пристрої LAPTOP-II	Середній час виконання, використовуючи LAPTOP-II
1	1_1.tif	4.019673	3,990686	1.725552	1,769019
2	1_2.tif	4.012645		1.812424	
3	1_3.tif	3.931072		1.712452	
4	1_4.tif	3.956202		1.699745	
5	1_5.tif	3.888345		1.765524	
6	1_6.tif	4.098124		1.684772	
7	1_7.tif	4.028741		1.982664	

Таблиця 4.4 Конфігурація LAPTOP-I та LAPTOP-II, що використовуються для пошуку часу виконання

№	Параметри	LAPTOP-I	LAPTOP-II
1	Модель	ACER ASPIRE 5253	ASUS X550V
2	Процесор	AMD E350 1.60 GHZ	I7-7700HQ 3.8 GHZ
3	Оперативка	3 GB (2 GB USABLE)	16GB
4	Тип системи	32-bit OPERATING SYSTEM	64-bit операційна система
5	Операційна система	Windows 7 STARTER	Windows 10 HOME
6	Програма	MATLAB 2015a 32-bit	MATLAB 2015a 32-bit

Важливим кроком для отримання високої якості та продуктивності для всіх типів зображень є точна сегментація. Сегментація відбитків пальців - це один із основних процесів попередньої обробки відбитків пальців, і він стосується процесу поділу або поділу зображення на дві непересічні області,

як передній план та фон. Алгоритм сегментації на основі двовимірного відсікання ефективно відсікає фонову область

відбитка пальця у всіх чотирьох допустимих межах, лівий край, правий край, верхній край і нижній край. Використання методу виявлення крайок ефективно покращує ідентифікацію країв.

Запропонований алгоритм має хороший час виконання у високо конфігурований системах. Запропонований алгоритм має наступні характеристики.

- Використання хитрих методів виявлення країв ефективно знаходить усі краї зображення.
- Наявність можливості генерувати зображення з низькою роздільною здатністю із зображення у градаціях сірого 256 × 256
- Накладає зображення з низькою роздільною здатністю на нуль уздовж напрямків рядків і стовпців.
- Створіть вікно відсікання розміром 15 × 15 для заповненого зображення з низькою роздільною здатністю.
- Змінити форму відсічної вдови як вікно розміром 1 × 256
- Знайдіть ко варіацію матриці зображення для кожного елемента вхідного зображення через заповнене зображення
- Якщо ко варіація менша за порогове значення, це розглядається як фонове зображення, інакше розглядається як зображення переднього плану.
- Відкидає кожен стовпець зображення, якщо він повністю, містить значення інтенсивності 1 з лівого та правого напрямків зображення.
- Відкидає кожен рядок зображення, якщо він повністю, містить значення інтенсивності 1 як з верхнього, так і з нижнього напрямку зображення.

Тепер перейдемо до розробленої хеш функції для даної системи. Якщо розглядати таблицю 4.5 та графік до цієї таблиці то можна зрозуміти що дана хеш функція працює швидше на 3-4% ніж інші.

Таблиця 4.5 – Тестування хеш функцій на час виконання

Назва методу	Назва зображення	Час виконання (в секундах)	Середнє значення
Розроблений хеш	1_1.tif	0,135625	0,196741
	1_2.tif	0,262152	
	1_3.tif	0,241242	
	1_4.tif	0,172497	
	1_5.tif	0,044251	
	1_6.tif	0,135772	
	1_7.tif	0,385648	
MD5	1_1.tif	0,147182	0,207474

Продовження таблиці 4.5

	1_2.tif	0,273536	
	1_3.tif	0,252735	
	1_4.tif	0,183266	
	1_5.tif	0,060636	
	1_6.tif	0,142636	
	1_7.tif	0,392327	
SHA 1	1_1.tif	0,162482	0,226321
	1_2.tif	0,299472	
	1_3.tif	0,270427	
	1_4.tif	0,204284	
	1_5.tif	0,090426	
	1_6.tif	0,152427	
	1_7.tif	0,404729	

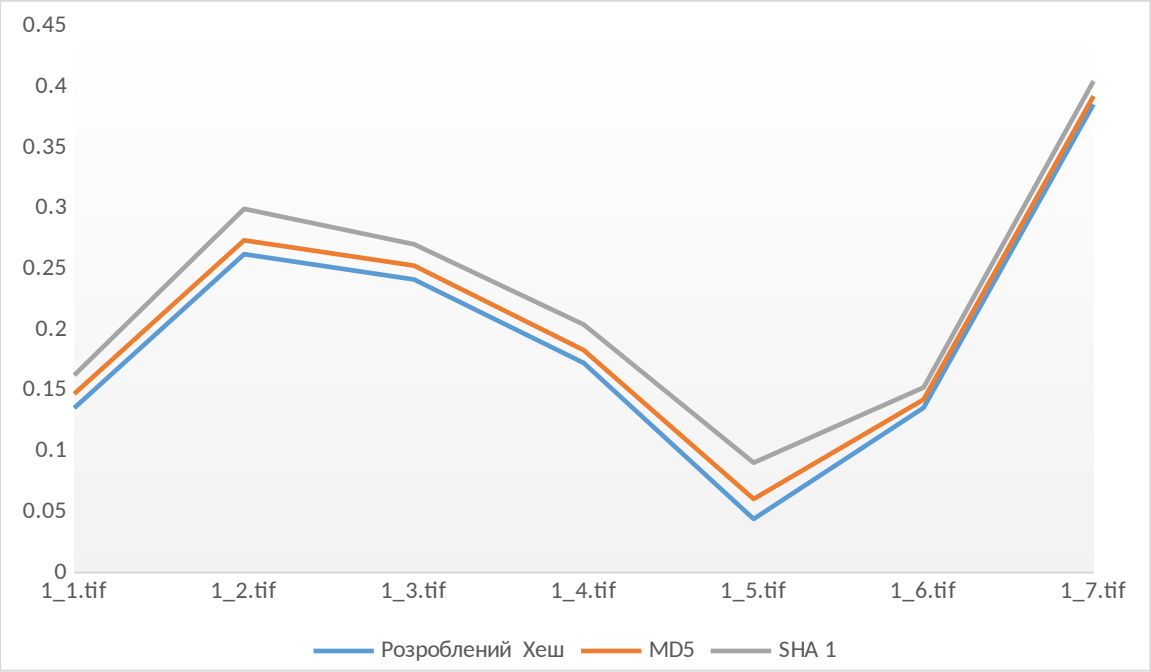


Рисунок 4.6 Графік швидкості хеш функцій

За результатами графіку можна зробити висновок що розроблений хеш працює швидше на 3-6%. За рахунок розробленого алгоритму роботи сегментації зображення відбитку пальця.

4.3 Впровадження системи в промислову експлуатацію

Для впровадження системи з біометричною автентифікацією потрібно виділити потужний сервер де будуть проводитись обрахунки, зберігатись шаблони, та аналізувати авторизації. Також на кожне місце треба закупити біометричні зчитувачі.

Найкращий варіант це біометричний термінал ZKTECO SpeedFace-V5L(рисунок 4.6) був би ефективним рішенням в цій системі, тому що у нього є такі функції: вимірювання температури, розпізнання обличчя, сканування відбитка пальця та долоні. Ці функції могли б розширити аналіз, і зменшити кількість спроб зловмисного автентифікуваня. Але з ергономічної

точки він не зручний так як його потрібно вмонтовувати в стіну, або кріпити до стіни. Коштує від 21000 грн.

Рисунок 4.6 біометричний термінал ZKTeco SpeedFace-V5L

Тому був запропонований варіант середній варіант сканер відбитків пальців ZKTECO FPV10R (рисунок 4.7). Він достатньо ергономічний щоб розмістити на робочому місці. Має інтерфейс USB 1.1/2.0. Має розмір зображені 256 * 360 пікселів/240 * 320 пікселів, що цілком підходить до нашої системи. Коштує 5850 грн та для розпізнання обличчя камера LOGITECH C920 HD PRO коштує 4 999грн. Разом вийде біля 10 000грн за 1 місце.

Рисунок 4.7 Сканер відбитків пальців ZKTECO FPV10R

Якщо розглядати бюджетний варіант то було запропоновано сканер відбитків ZKTECO ZK9500(рисунок 4.8). Він найдешевший та най урізаній по функціоналу, але теж може конкурувати з вищеперерахованими варіантами. Тому що на сервері буде працювати розроблений алгоритм покращення вхідного зображення, для подальшого хешування. Коштує 1800грн, а разом із бюджетно камерою LOGITECH C270 HD буде 3000 грн за 1 робоче місце.

Рисунок 4.8 Сканер відбитків ZKTECO ZK9500

4.4 Висновок

У даному розділі ми протестували хеш функцію яку створили для системи та порівняли вже з існуючими та визначили, що хеш який створили працює на 4-6% швидше ніж MD5 та SHA 1. Протестували 7 відбитків на алгоритмі який створили. Запропонували 3 варіанти сканерів відбитків пальців та оцінили їх вартість за 1 робоче місце.

ВИСНОВКИ

Дана кваліфікаційна робота вказує на актуальність захисту біометрії на в інформаційно-комунікаційній системі на сьогоднішній день. Ці дані треба захищати в першу чергу тому що це ваш ідентифікатор в любій майбутній системі. Біометричні дані забезпечують безпечний спосіб передачі даних, а також ідентифікують порти даних та пристрої, а також забезпечують їх збереження. Біометрія є оптимальним заходом безпеки, і їх подальший розвиток буде ключовим компонентом для створення важко порушених протоколів безпеки. Оскільки характеристики, визначені біометричними сканерами, не змінюються і є унікальними для кожної людини, вони роблять дуже безпечний засіб передачі даних та створення ідентифікаторів для обміну захищеними даними.

Більш сильні, ніж шифрування або захист паролем - обидва з яких можна порушити практикою та терпінням - біометричні характеристики надзвичайно важко продублювати або підробити. Ця лише функція робить їх гідними розгляду в будь-якій системі безпеки.

Зростання біометричної галузі продовжуватиметься швидкими темпами, а біометрія продовжуватиме проникати на всі рівні технологій. З необхідністю захищати дані та простотою їх інтеграції в різноманітні системи, технологія буде продовжувати розширюватися і допомагатиме розвивати безперебійну передачу даних, якомога безпечнішу. Встановлюючи зв’язки, важливі для IOT, та рекомендації, на які покладаються кінцеві користувачі, роль біометрії буде рости та розвиватися в міру того, як IOT продовжить рости та розвиватися. Унікальний характер біометрії та безліч способів їх використання можуть, мабуть, зіграти ключову роль у зростанні IOT.

Під час тестування ми побачили як алгоритми кардинально впливають на пізнавальність біометричного зображення. Та скільки потрібно процедур щоб якісно захистити від несанкціонованої ідентифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Крішна Прасад, К. та Ейтал, П. С. (2017). Концептуальне дослідження з покращення іміджу Техніка для відбитків пальців. Міжнародний журнал прикладної інженерії та управління Листи [Електронный ресурс] (IJAEML), 1 (1), 63-72. DOI: http://dx.doi.org/10.5281/zenodo.831678 (дата обращения: 10.05.2021). – Назва з екрану.
2) Zhang, J., Lai, R., & Kuo, C. C. J. (2012). Латентне виявлення та сегментація відбитків пальців за допомогою моделі спрямованої загальної варіації [Електронный ресурс]. In Proceedings - Міжнародна конференція з обробки зображень, 1145–1148. DOI: https://doi.org/10.1109/ICIP.2012.6467067 (дата обращения: 12.05.2021). – Назва з екрану.
3) Адамс, Р. та Бішоф, Л. (1994). Насіннєвий регіон зростає. Транзакції IEEE з аналізу шаблонів та машинного інтелекту, 16 (6), 641-647.
4) Chakraborty, A., Staib, L. H., & Duncan, J. S. (1996). Деформоване знаходження межі в медицині зображення шляхом інтеграції інформації про градієнт та область. Транзакції IEEE щодо медичної візуалізації, 15 (6), 859-870.
5) Mehtre, B. M., Murthy, N. N., Kapoor, S., & Chatterjee, B. (1987). Сегментація відбитків пальців зображення із використанням спрямованого зображення. Розпізнавання зразків, 20 (4), 429-435.
6) Mehtre, B. M., & Chatterjee, B. (1989). Сегментація зображень відбитків пальців – композиція метод. Розпізнавання зразків, 22 (4), 381-385.
7) Рата, Н. К., Чен, С. та Джейн, А. К. (1995). Адаптивна витяжка функцій на основі орієнтації потоку на відбитках пальців. Розпізнавання зразків, 28 (11), 1657-1672.

8) Vielhauer, C., Dittmann, J., Drygajlo, A., Juul, N. C., & Fairhurst, M. (Eds.). (2011). Біометрія та управління ідентифікатором: Європейський семінар COST 2101, BioID 2011, Бранденбург (Гавел), 8 березня 10, 2011, Proceedings (Vol. 6583). Springer Science & Business Media.

9) Jain, A. K., & Dubes, R. C. (1988). Алгоритми кластеризації даних. Prentice-Hall, Inc.

10) Канні, Дж. (1986). Обчислювальний підхід до виявлення краю. Транзакції IEEE з аналізу шаблонів та машинного інтелекту, (6), 679-698.

11) Де Марсіко, М., Гальді, К., Наппі, М., та Річчо, Д. (2014). ФІРМА: обличчя та райдужка визнання за мобільну взаємодію. Обчислення зображення та зору, 32 (12), 1161-1172.

12) Кумар Д. та Рю, Ю. (2009). Короткий вступ до біометрії та відбитків пальців платіжна технологія. Міжнародний журнал передових наук і технологій, 4, 25-38.

13) Туляков С., Фарук Ф., Мансухані П. та Говіндараю В. (2007). Симетричні хеш-функції для безпечних біометричних систем відбитків пальців. Листи про розпізнавання зразків, 28 (16), 2427-2436.

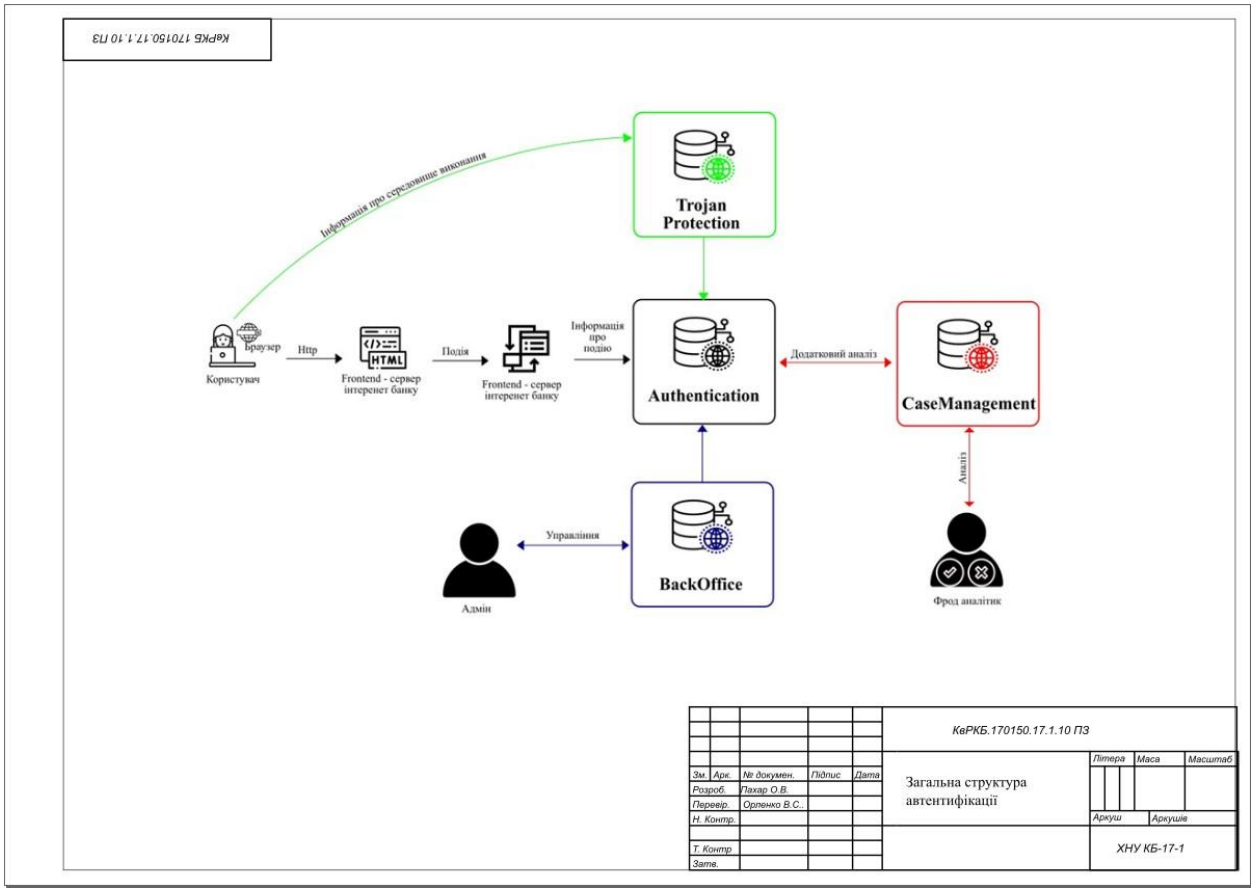
14) Das, P. P., Chakrabarti, P. P., & Chatterji, B. N. (1987). Функції відстані в цифровій геометрії. Інформаційні науки, 42 (2), 113-136.

15) Aithal, P. S., Shailashree, V. T., Suresh Kumar, P. M. (2015). Нова техніка ABCD для аналізу бізнес-моделей та концепцій, Міжнародний журнал з управління, інформаційних технологій та інженерії [Електронный ресурс] (IJMIE), 5 (4), 409-423. DOI: http://doi.org/10.5281/zenodo.61652. (дата обращения: 28.05.2021). – Назва з екрану.

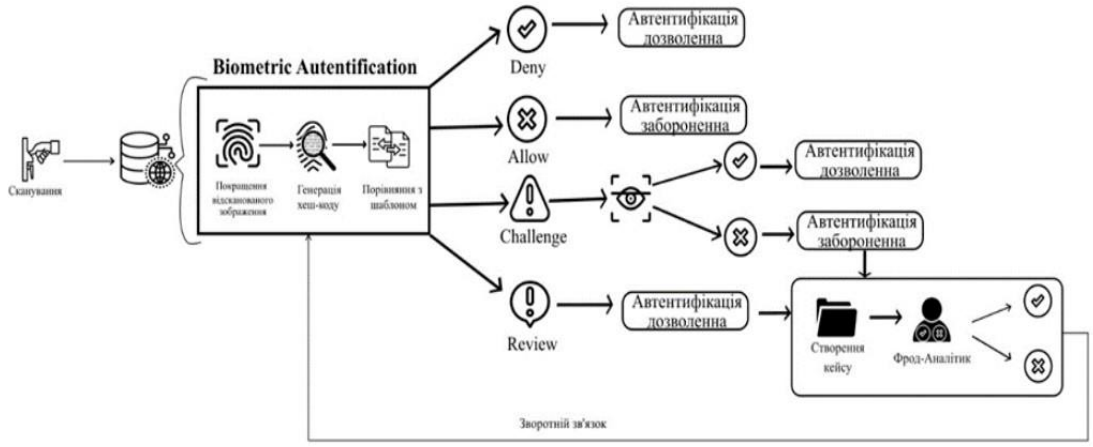
ДОДАТОК А

(Обов’язковий)

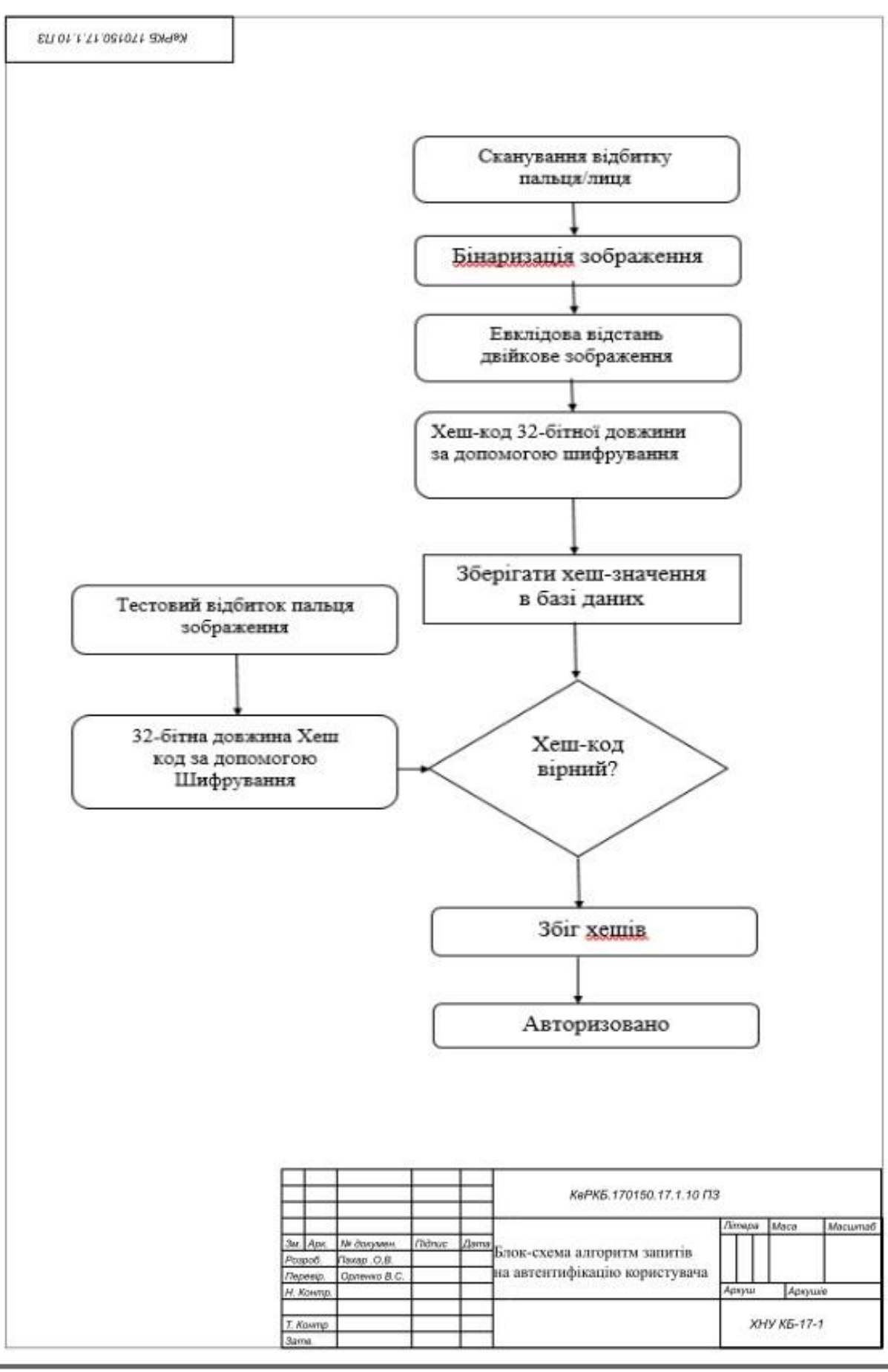
Копія графічної частини



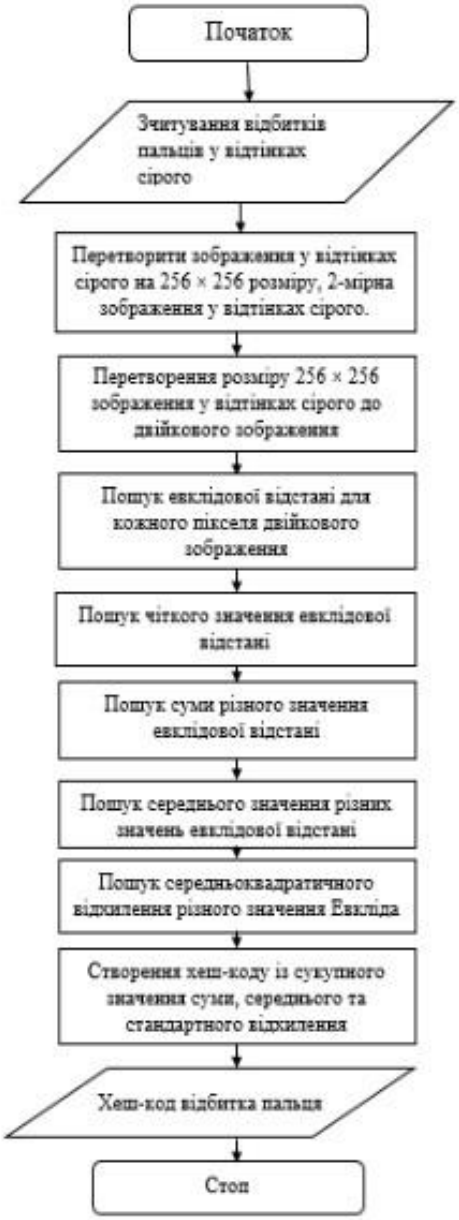
КвРКБ 170150.17.1.10 ПЗ



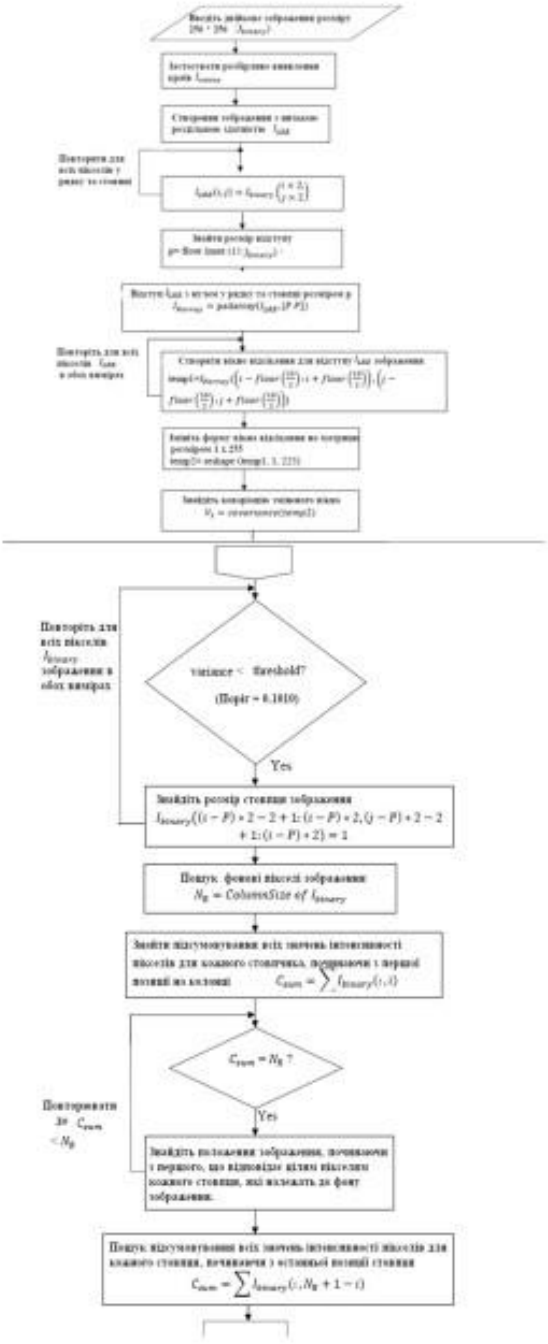
					КвРКБ.170150.17.1.10 ПЗ	
						Літера / Маса / Масштаб
Зм.	Арк.	№ докумен.	Підпис	Дата	Алгоритм роботи розроблювальної автентифікації	
Розроб.		Пахар О.В.				Аркуш / Аркушів
Перевір.		Орленко В.С.				
Н. Контр.						ХНУ КБ-17-1
Т. Контр						
Затв.						


КвРКБ.170150.17.1.10 ПЗ
Сканування відбитку пальця/лиця
Бінаризація зображення
Евклідова відстань двійкове зображення
Хеш-код 32-бітної довжини за допомогою шифрування
Зберігати хеш-значення в базі даних
Тестовий відбиток пальця зображення
32-бітна довжина Хеш код за допомогою Шифрування
Хеш-код вірний?
Збіг хешів
Авторизовано
КвРКБ.170150.17.1.10 ПЗ
Зм. Арк. № докумен. Підпис Дата
Розроб. Пахар .О.В.
Перевір. Орленко В.С.
Н. Контр.
Т. Контр.
Затв.
Блок-схема алгоритм запитів на автентифікацію користувача
Літера Маса Масштаб
Аркуш Аркушів
ХНУ КБ-17-1

КвРКБ 170150.17.1.10 ПЗ

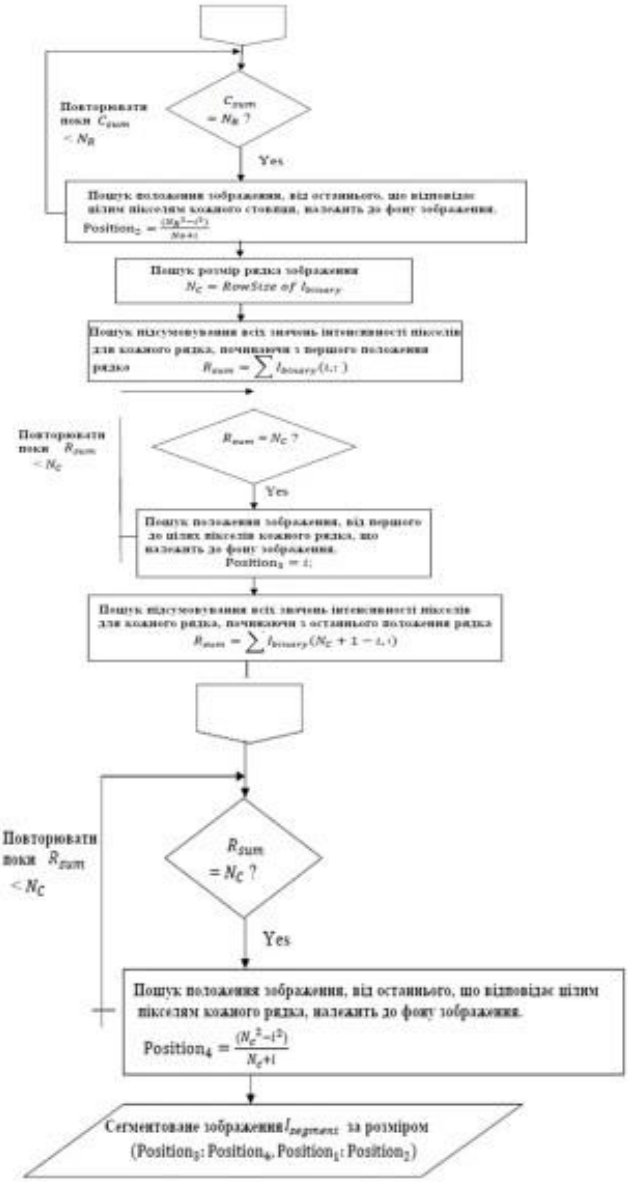


					КвРКБ.170150.17.1.10 ПЗ			
						Літера	Маса	Масштаб
Зм.	Арк.	№ докумен.	Підпис	Дата	Блок-схема генерування хеш-коду з використанням евклідової відстані			
Розроб.		Пахар О.В.						
Перевір.		Орленко В.С.				Аркуш	Аркушів	
Н. Контр.						ХНУ КБ-17-1		
Т. Контр.								
Затв.								



					КвРКБ.170150.17.1.10 ПЗ	
Зм.	Арк.	№ докумен.	Підпис	Дата	Блок-схема алгоритму сегментації на основі двовимірного відсікання 1 ч	Літера / Маса / Масштаб
Розроб.		Пахар .О.В.				
Перевір.		Оленко В.С.				Аркуш / Аркушів
Н. Контр.						
Т. Контр.						ХНУ КБ-17-1
Затв.						

КвРКБ 170150.17.1.10 ПЗ



					КвРКБ.170150.17.1.10 ПЗ			
Зм.	Арк.	№ докумен.	Підпис	Дата	Блок-схема алгоритму сегментації на основі двовимірного відсікання 2 ч	Літера	Маса	Масштаб
Розроб.		Пахар О.В.						
Перевір.		Сотенко В.С.				Аркуш	Аркушів	
Н. Контр.								
Т. Контр.						ХНУ КБ-17-1		
Затв.								

ДОДАТОК Б

(Обов’язковий)

Програмна реалізація

```
ifndef INCLUDE_FROM_MEMORYFINGERPRINT_H
static_assert(0, "DO NOT include this file directly!");
#endif

namespace alexp {

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::updateExpr(const ref<Expr> &expr) {
  llvm::raw_ostream &os = getDerived().updateOstream();
  std::unique_ptr<ExprPPrinter> p(ExprPPrinter::create(os));
  p->scan(expr);
  p->print(expr);
  os.flush();

  bufferContainsSymbolic = true;
  for (auto v : p->getUsedArrays()) {
    bufferSymbolicReferences[v] += 1;
  }
}
```

```
template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::updateConstantExpr(const ConstantExpr
&expr) {
  if (expr.getWidth() <= 64) {
    std::uint64_t constantValue = expr.getZExtValue(64);
    getDerived().updateUint64(constantValue);
  } else {
    const llvm::APInt &value = expr.getAPValue();
    for (std::size_t i = 0; i != value.getNumWords(); i++) {
      std::uint64_t word = value.getRawData()[i];
      getDerived().updateUint64(word);
    }
  }
}

template <typename D, std::size_t S, typename V>
std::string MemoryFingerprintT<D, S, V>::toString(const
MemoryFingerprintDelta &delta) {
  return std::move(toString(delta.fingerprintValue));
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::addToFingerprint() {
  getDerived().generateHash();
  executeAdd(fingerprintValue, buffer);
  getDerived().clearHash();
```

```
  if (bufferContainsSymbolic) {
    for (auto [array, count] : bufferSymbolicReferences) {
      auto [it, _] = symbolicReferences.try_emplace(array, 0);
      it->second += count;
      if (it->second == 0) {
        symbolicReferences.erase(it);
      }
    }
    resetBufferRefCount();
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::removeFromFingerprint() {
  getDerived().generateHash();
  executeRemove(fingerprintValue, buffer);
  getDerived().clearHash();

  if (bufferContainsSymbolic) {
    for (auto [array, count] : bufferSymbolicReferences) {
      auto [it, _] = symbolicReferences.try_emplace(array, 0);
      it->second -= count;
      if (it->second == 0) {
        symbolicReferences.erase(it);
```

```
      }
    }
    resetBufferRefCount();
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S,
V>::addToFingerprintAndDelta(MemoryFingerprintDelta &delta) {
  getDerived().generateHash();
  executeAdd(delta.fingerprintValue, buffer);
  executeAdd(fingerprintValue, buffer);
  getDerived().clearHash();

  if (bufferContainsSymbolic) {
    for (auto [array, count] : bufferSymbolicReferences) {
      {
        auto [it, _] = symbolicReferences.try_emplace(array, 0);
        it->second += count;
        if (it->second == 0) {
          symbolicReferences.erase(it);
        }
      }
      {
        auto [it, _] = delta.symbolicReferences.try_emplace(array, 0);
        it->second += count;
```

```
      if (it->second == 0) {
        delta.symbolicReferences.erase(it);
      }
    }
  }
  resetBufferRefCount();
 }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S,
V>::removeFromFingerprintAndDelta(MemoryFingerprintDelta &delta) {
 getDerived().generateHash();
 executeRemove(delta.fingerprintValue, buffer);
 executeRemove(fingerprintValue, buffer);
 getDerived().clearHash();

 if (bufferContainsSymbolic) {
  for (auto [array, count] : bufferSymbolicReferences) {
   {
    auto [it, _] = symbolicReferences.try_emplace(array, 0);
    it->second -= count;
    if (it->second == 0) {
     symbolicReferences.erase(it);
    }
   }
```

```
      {
        auto [it, _] = delta.symbolicReferences.try_emplace(array, 0);
        it->second -= count;
        if (it->second == 0) {
          delta.symbolicReferences.erase(it);
        }
      }
    }
    resetBufferRefCount();
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::addToDeltaOnly(MemoryFingerprintDelta
&delta) {
  getDerived().generateHash();
  executeAdd(delta.fingerprintValue, buffer);
  getDerived().clearHash();

  if (bufferContainsSymbolic) {
    for (auto [array, count] : bufferSymbolicReferences) {
      auto [it, _] = delta.symbolicReferences.try_emplace(array, 0);
      it->second += count;
      if (it->second == 0) {
        delta.symbolicReferences.erase(it);
      }
```

```
    }
    resetBufferRefCount();
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S,
V>::removeFromDeltaOnly(MemoryFingerprintDelta &delta) {
  getDerived().generateHash();
  executeRemove(delta.fingerprintValue, buffer);
  getDerived().clearHash();

  if (bufferContainsSymbolic) {
    for (auto [array, count] : bufferSymbolicReferences) {
      auto [it, _] = delta.symbolicReferences.try_emplace(array, 0);
      it->second -= count;
      if (it->second == 0) {
        delta.symbolicReferences.erase(it);
      }
    }
    resetBufferRefCount();
  }
}

template <typename D, std::size_t S, typename V>
```

```
void MemoryFingerprintT<D, S, V>::addDelta(const MemoryFingerprintDelta
&delta) {
  executeAdd(fingerprintValue, delta.fingerprintValue);

  for (auto [array, count] : delta.symbolicReferences) {
    auto [it, _] = symbolicReferences.try_emplace(array, 0);
    it->second += count;
    if (it->second == 0) {
      symbolicReferences.erase(it);
    }
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::removeDelta(const MemoryFingerprintDelta
&delta) {
  executeRemove(fingerprintValue, delta.fingerprintValue);

  for (auto [array, count] : delta.symbolicReferences) {
    auto [it, _] = symbolicReferences.try_emplace(array, 0);
    it->second -= count;
    if (it->second == 0) {
      symbolicReferences.erase(it);
    }
  }
}
```

```
template <typename D, std::size_t S, typename V>
MemoryFingerprintDelta MemoryFingerprintT<D, S, V>::getFingerprintAsDelta()
{
  MemoryFingerprintDelta delta;
  delta.fingerprintValue = fingerprintValue;
  delta.symbolicReferences = symbolicReferences;
  return delta;
}

template <typename D, std::size_t S, typename valueT>
valueT MemoryFingerprintT<D, S,
valueT>::getFingerprint(std::vector<ref<Expr>> &expressions) {
  std::set<const Array *> arraysReferenced;
  for (auto s : symbolicReferences) {
    assert(s.second > 0);
    arraysReferenced.insert(s.first);
  }

  if (arraysReferenced.empty())
    return fingerprintValue;

  auto exprSort = [](const ref<Expr> &a, const ref<Expr> &b) {
            auto aHash = a->hash();
            auto bHash = b->hash();
            if (aHash != bHash) {
```

```
      return aHash < bHash;
    } else {
      return a < b;
    }
  };

std::sort(expressions.begin(), expressions.end(), exprSort);


std::string tmpString;
llvm::raw_string_ostream tmpOS(tmpString);

std::unordered_map<const Array *, ExprHashSet> constraintsMap;
ExprHashMap<std::set<const Array *>> exprToArray;
for (auto &expr : expressions) {
  std::unique_ptr<ExprPPrinter> p(ExprPPrinter::create(tmpOS));
  p->scan(expr);

  for (auto s : p->getUsedArrays()) {
    constraintsMap[s].insert(expr);
    exprToArray[expr].insert(s);
  }
}
```

```
std::set<const Array *> newReferences = arraysReferenced;

do {
  std::set<const Array *> tmp;
  using std::swap;
  swap(newReferences, tmp);
  assert(newReferences.empty());
  for (auto *a : tmp) {
    auto it = constraintsMap.find(a);
    if (it == constraintsMap.end()) {
      continue;
    }
    for (const ref<Expr> &c : it->second) {
      for (auto *b : exprToArray[c]) {
        if (!arraysReferenced.count(b)) {
          newReferences.insert(b);
          arraysReferenced.insert(b);
        }
      }
    }
  }
} while (!newReferences.empty());

MemoryFingerprintDelta temp;
if (!arraysReferenced.empty()) {
```

```
    getDerived().updateUint8(10);
    std::vector<const Array*> arrSet(arraysReferenced.begin(),
arraysReferenced.end());
    std::sort(arrSet.begin(), arrSet.end(), [](const Array *a, const Array *b) {
      return a->getName() < b->getName();
    });
    for (auto v : arrSet) {
      auto it = constraintsMap.find(v);
      if (it == constraintsMap.end()) {
        continue;
      }
      std::vector<ref<Expr>> set(it->second.begin(), it->second.end());
      std::sort(set.begin(), set.end(), exprSort);
      for (const ref<Expr> &expr : set) {
        llvm::raw_ostream &os = getDerived().updateOstream();
        ExprPPrinter::printSingleExpr(os, expr);
        os.flush();
      }
    }
    addToDeltaOnly(temp);
  }

  addDelta(temp);
  auto result = fingerprintValue;
  removeDelta(temp);
```

```
  return result;
}

template <typename D, std::size_t S, typename valueT>
valueT MemoryFingerprintT<D, S,
valueT>::getFingerprintWithDelta(std::vector<ref<Expr>> &expressions,
                                 const MemoryFingerprintDelta &delta) {
  addDelta(delta);
  auto result = getFingerprint(expressions);
  removeDelta(delta);
  return result;
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateWriteFragment(std::uint64_t address,
                                      ref<Expr> value) {
  if (ConstantExpr *constant = dyn_cast<ConstantExpr>(value)) {
    getDerived().updateUint8(1);
    getDerived().updateUint64(address);
    std::uint8_t byte = constant->getZExtValue(8);
    getDerived().updateUint8(byte);
    return false;
  } else {
    getDerived().updateUint8(2);
    getDerived().updateUint64(address);
    getDerived().updateExpr(value);
```

```
    return true;
  }
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::updateThreadId(const ThreadId& tid) {
  getDerived().updateUint64(tid.size());
  for (std::size_t i = 0; i < tid.size(); i++) {
    const std::uint16_t v = tid.ids()[i];
    getDerived().updateUint16(v);
  }
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateLocalFragment(const ThreadId
&threadID,
                                                      std::uint64_t stackFrameIndex,
                                                      const llvm::Instruction *inst,
                                                      ref<Expr> value) {
  if (ConstantExpr *constant = dyn_cast<ConstantExpr>(value)) {
    // concrete value
    getDerived().updateUint8(3);
    updateThreadId(threadID);
    getDerived().updateUint64(stackFrameIndex);
    getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(inst));
    getDerived().updateConstantExpr(*constant);
```

```
    return false;
  } else {
    // symbolic value
    getDerived().updateUint8(4);
    updateThreadId(threadID);
    getDerived().updateUint64(stackFrameIndex);
    getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(inst));
    getDerived().updateExpr(value);
    return true;
  }
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateArgumentFragment(const ThreadId
&threadID,
                                                         std::uint64_t sfIndex,
                                                         const KFunction *kf,
                                                         std::uint64_t argumentIndex,
                                                         ref<Expr> value) {
  if (ConstantExpr *constant = dyn_cast<ConstantExpr>(value)) {
    // concrete value
    getDerived().updateUint8(5);
    updateThreadId(threadID);
    getDerived().updateUint64(sfIndex);
    getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(kf));
    getDerived().updateUint64(argumentIndex);
```

```
    getDerived().updateConstantExpr(*constant);
    return false;
  } else {
    // symbolic value
    getDerived().updateUint8(6);
    updateThreadId(threadID);
    getDerived().updateUint64(sfIndex);
    getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(kf));
    getDerived().updateUint64(argumentIndex);
    getDerived().updateExpr(value);
    return true;
  }
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateProgramCounterFragment(const
ThreadId &threadID,
                                                               std::uint64_t sfIndex,
                                                               const llvm::Instruction *i,
                                                               std::uint64_t step) {
  getDerived().updateUint8(7);
  updateThreadId(threadID);
  getDerived().updateUint64(sfIndex);
  getDerived().updateUint64(step);
  getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(i));
  return false;
```

```
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateFunctionFragment(const ThreadId&
threadID,
                                                         std::uint64_t sfIndex,
                                                         const KFunction *callee,
                                                         const KInstruction *caller) {
  getDerived().updateUint8(8);
  updateThreadId(threadID);
  getDerived().updateUint64(sfIndex);
  getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(caller));
  getDerived().updateUint64(reinterpret_cast<std::uintptr_t>(callee));
  return false;
}

template <typename D, std::size_t S, typename V>
bool MemoryFingerprintT<D, S, V>::updateExternalCallFragment(std::uint64_t
externalCallCounter) {
  getDerived().updateUint8(9);
  getDerived().updateUint64(externalCallCounter);
  return false;
}

template <typename D, std::size_t S, typename V>
```

```
bool MemoryFingerprintT<D, S, V>::updateAcquiredLockFragment(std::uint64_t
lockId,
                                                const ThreadId &holdingThread) {
  getDerived().updateUint8(11);
  getDerived().updateUint64(lockId);
  updateThreadId(holdingThread);
  return false;
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::updateThreadStateFragment(const ThreadId
&threadId, std::uint8_t state) {
  getDerived().updateUint8(12);
  updateThreadId(threadId);
  getDerived().updateUint8(state);
}

template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S,
V>::updateThreadWaitingOnLockFragment(const ThreadId &threadId,
std::uint64_t lockId) {
  getDerived().updateUint8(13);
  updateThreadId(threadId);
  getDerived().updateUint64(lockId);
}
template <typename D, std::size_t S, typename V>
```

```
void MemoryFingerprintT<D, S,
V>::updateThreadWaitingOnCV_1Fragment(const ThreadId &threadId,
                                      std::uint64_t condId,
                                      std::uint64_t lockId) {
  getDerived().updateUint8(14);
  updateThreadId(threadId);
  getDerived().updateUint64(condId);
  getDerived().updateUint64(lockId);
}
template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S,
V>::updateThreadWaitingOnCV_2Fragment(const ThreadId &threadId,
                                      std::uint64_t condId,
                                      std::uint64_t lockId) {
  getDerived().updateUint8(15);
  updateThreadId(threadId);
  getDerived().updateUint64(condId);
  getDerived().updateUint64(lockId);
}
template <typename D, std::size_t S, typename V>
void MemoryFingerprintT<D, S, V>::updateThreadWaitingOnJoinFragment(const
ThreadId &threadId,
                                      const ThreadId &joinedId) {
  getDerived().updateUint8(16);
  updateThreadId(threadId);
  updateThreadId(joinedId);
```

```
}


} // name space alexp
```

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ "ПриватБанк".

Автор: Пахар Олександр Валерійович

Спеціальність: 125 – Кібербезпека

Освітня програма:Кібербезпека

Керівник:Орленко ВікторіяСергіївна, к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформлені посилання;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає1.34% що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В.С. Орленко

Завідувач кафедри КбКСМ, гарантОП _____ Ю.П. Кльоц

Дата: 17.06.2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Пахар Олександр Валерійович

Тема Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ «ПриватБанк»

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «магістр»:

кількість листів креслень 5 ; кількість сторінок записки 52

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі розроблено систему авторизації на основі біометричних даних .

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи .

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подана загальна характеристика поставленої задачі, чітко визначено об’єкт, предмет та методи дослідження, сформульована актуальність. Визначені задачі, які необхідно вирішити для досягнення поставленої мети, практична цінність отриманих результатів. У першому розділі проведено огляд використовуваних в комп’ютерних системах методів біомитеричної авторизації та захисту біометричних даних користувачів системи, виконане обґрунтування актуальності теми дослідження і виконана постановка задачі. В другому розділі наведені засоби та технології використані для побудови системи авторизації. В третьому розділі визначено основні положення системи та розроблено алгоритми її роботи. Четвертий розділ було присвячено апробації системи біометричної авторизації та алгоритмів її реалізації

4. Позитивні сторони роботи Кваліфікаційна робота має комплексну практичну цінність. Практична цінність полягає у розробці біометричної автентифікації за допомогою якої надається можливість не використовувати логін та пароль. На основі цього в подальшому здійснюється керуючий вплив на витік конфіденційних даних. Практична цінність результатів кваліфікаційної роботи полягає у створені системи біометричної автентифікації, що може бути підлаштована для відділень банку та використанні у клієнтській базі .

5. Негативні сторони роботи Технічні засоби для реалізації розроблювальної системи коштують дорого, є шанс що система може відмовити в доступі якщо пошкоджений відбиток

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження Окремі описи в пояснювальній записці подано занадто деталізовано, що ускладнює сприйняття матеріалу фахівцями в обраній предметній галузі.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Говорущенко Т.О., зав. каф. КІСП, д.т.н., професор

«16» 06 2021. (підпис)

UNICHECK

User name:
Кафедра кибербезпеки

Check ID:
1008318956

Check date:
17.06.2021 11:31:37 EEST

Check type:
Doc vs Internet

Report date:
17.06.2021 11:37:08 EEST

User ID:
100005590

File name: **Кваліфікаційна робота Пахар 1 на юнічек**

Page count: **50** Word count: **7402** Character count: **59250** File size: **3.07 MB** File ID: **1008385875**

1.34% Matches

Highest match: **1.09%** with Internet source (**https://web.posibnyky.vntu.edu.ua/fmib/41yaremchuk_kompleksni_systemy_zahystu**

1.34% Internet sources	93

Page 52

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters	2

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 8%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 94461 Название: Захист потоків даних в інформаційно-комунікаційній мережі відділення Хмельницької філії АТ КБ "ПриватБанк" Добавлено в БД: 2021-06-17 Авторы: Пахар О.В. Руководители: Орленко В.С. Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	46668	432	648 (1%)	13 (3%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы