МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

ПРОДАН Тарас Іванович

Алгоритм біометричної аутентифікації для систем захисту інформації / Biometric authentication algorithm for information security systems

спеціальність: 125 – Кібербезпека

освітньо-професійна програма –Кібербезпека

Кваліфікаційна робота

Виконав студент групи КБм -21
Т.І. Продан

Науковий керівник
к.т.н., доцент С.В.Івасьєв

Кваліфікаційну роботу допущено
до захисту:

«____» _____ ___2022 р.

Завідувач кафедри
_____ В.В.Яцків

ТЕРНОПІЛЬ - 2022

Факультет комп'ютерних інформаційних технологій

Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 - Кібербезпека
освітньо-професійна програма –Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
____” _____ 2021 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ПРОДАН Тарас Іванович
(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи:
Алгоритм біометричної аутентифікації для систем захисту інформації / Biometric authentication algorithm for information security systems
керівник роботи к.т.н., доцент Н.Г. Яцків

затверджені наказом по університету від 31 грудня 2021 року № 606

2. Строк подання студентом закінченої випускної кваліфікаційної роботи 16 листопада 2022 року.

3. Вихідні дані до кваліфікаційної роботи: завдання на випускну кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- дослідити процес аутентифікації;
- дослідити протоколи аутентифікації із симетричними та асиметричними алгоритмами шифрування і хеш функціями;
- проаналізувати математичні основи алгоритмів біометричної авутентифікації;
- виконати формалізацію процесу авутентифікації на основі біометричних даних;
- дослідити алгоритми розпізнавання осіб: в бібліотеці OpenCV, гістограм локальних бінарних шаблонів та метод розпізнавання Віоли-Джонса.
- реалізувати систему авутентифікації за допомогою алгоритмів розпізнавання осіб .

5. Перелік графічного матеріалу у роботі.
- Схема взаємної автентифікація на основі сертифікатів.

- Схема взаємної аутентифікації на основі пароля та імені користувача.
- Схема алгоритму системи аутентифікації.
- Результати тестування розробленої системи аутентифікації.

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11 жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.2021 р. – 03.2022 р.	
2	Формалізація систем аутентифікації	03.2022 р. – 05.2022 р.	
3	Реалізація системи аутентифікації	05.2022 р. – 11.2022 р.	

Студент _____ Т.І. Продан
(підпис)

Керівник роботи _____ к.т.н., доцент С.В. Івасьєв
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритм біометричної аутентифікації для систем захисту інформації» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека» освітньо-професійної програми «Кібербезпека» написана обсягом 71 сторінка і містить 24 ілюстрацій, 2 додатки та 20 джерел за переліком посилань.

Метою кваліфікаційної роботи є розробка системи аутентифікації користувачів на основі біометричних даних.

Проведено дослідження процесів аутентифікації та авторизації та досліджено протоколи аутентифікації із симетричними та асиметричними алгоритмами шифрування і хеш функціями.

Виконано формалізацію процесу авторизації на основі біометричних даних для побудови алгоритму біометричної аутентифікації. Досліджено алгоритми розпізнавання осіб: в бібліотеці OpenCV, гістограм локальних бінарних шаблонів та метод розпізнавання Віоли-Джонса. Детально розглянуто алгоритми виявлення та розпізнавання обличчя людини у бібліотеці OpenCV. У числі розглянутих алгоритмів такі, як: виявлення об'єктів за допомогою класифікатора Віоли-Джонса, розпізнавання осіб із використанням алгоритмів Eigenfaces, Fisherfaces та LBPH.

Формалізовано та побудовано обчислювальні вирази побудови моделі розпізнавання осіб для систем аутентифікації.

Реалізувано систему аутентифікації за допомогою систем розпізнавання осіб на мові програмування python та проведено тестування розроблених програмних модулів.

Ключові слова: АЛГОРИТМ ВІОЛИ-ДЖОНСА, БІОМЕТРИЧНА АУТЕНТИФІКАЦІЯ, OPENCV.

ABSTRACT

The qualification work on the topic "Biometric authentication algorithm for information protection systems" for obtaining the Master's degree in the specialty 125 "Cybersecurity" of the educational and professional program "Cybersecurity" is written in the volume of 71 pages and contains 16 illustrations, 2 appendices and 20 sources according to the list of references .

The purpose of the qualification work is to develop a user authorization system based on biometric data.

A study of authentication and authorization processes was conducted, and authentication protocols with symmetric and asymmetric encryption algorithms and hash functions were investigated.

The formalization of the authorization process based on biometric data for the construction of the biometric authentication algorithm was carried out. Face recognition algorithms were studied: in the OpenCV library, a histogram of local binary patterns, and the Viola-Jones recognition method. Algorithms for detecting and recognizing a person's face in the OpenCV library are considered in detail. The considered algorithms include: object detection using the Viola-Jones classifier, face recognition using the Eigenfaces, Fisherfaces, and LBPH algorithms.

Computational expressions for building a face recognition model for authentication systems were formalized and constructed.

An authentication system was implemented with the help of face recognition systems in the python programming language, and the developed software modules were tested.

Keywords: VIOLA-JONES ALGORITHM, BIOMETRIC AUTHENTICATION, OPENCV.

ЗМІСТ

ВСТУП

Актуальність роботи. Системи біометричної аутентифікації є більш безпечними, оскільки біометричні дані користувача унікальні. Зловмиснику дуже складно злочинним чином обійти сканування відбитка пальца або обличчя людини, якщо воно виконується надійними рішеннями з сильним виявленням оригінальності / підробки, і все ж для аутентифікації відповідного користувача потрібно не великий детермінований час. Через це, біометрія вважається більш зручною та безпечнішою, ніж паролі.

Мета роботи полягає в розробці системи авторизації користувачів на основі біометричних даних:

Для досягнення даної мети ставились наступні завдання:

- дослідити процес аутентифікації;
- дослідити протоколи аутентифікації із симетричними та асиметричними алгоритмами шифрування і хеш функціями;
- проаналізувати математичні основи алгоритмів біометричної авутентифікації;
- виконати формалізацію процесу авутентифікації на основі біометричних даних;
- дослідити алгоритми розпізнавання осіб: в бібліотеці OpenCV, гістограм локальних бінарних шаблонів та метод розпізнавання Віоли-Джонса.
- реалізувати систему авутентифікації за допомогою алгоритмів розпізнавання осіб .

Об’єкт дослідження – процеси біометричної авторизації та розпізнавання користувача.

Предмет досліджень – алгоритми біометричної авторизації.

Методи дослідження базуються на методах розпізнавання образів та алгоритмах аутентифікації.

Наукова новизна одержаних результатів визначається наступним чином:

– Формалізовано та побудовано обчислювальні вирази побудови моделі розпізнавання осіб для систем аутентифікації;

Практична цінність одержаних результатів полягає в тому, що:

– реалізовано програмне забезпечення для розпізнавання обличчя на мові Python для системи біометричної аутентифікації користувачів.

Публікації та апробація до магістерської роботи.

1. Стефурак Н.А., Продан Т.І., Дослідження створення цифрового підпису засобами C#. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ - 2021), Тернопіль, 2021. 143 -144 с.

2. Продан Т.І., Івасьєв С.В., Сучасні методи біометричної ідентифікації. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2022), Тернопіль, 2022. 62-65 с.

3. Кузик В.М., Продан Т.І., Івасьєв С.В., Слєпцова О.Я. Біометрична система аутентифікації з використанням голосових даних. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та компютерно-інтегровані технології»(КБКІТ-2020), Тернопіль, 2020. – С.56-59.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття автентифікації

Ідентифікація, автентифікація та авторизація. Більшість стикається з цими трьома концепціями щодня, але не всі знають різницю.

Аутентифікація – це процес визначення того, чи дійсно хтось чи щось є тим, за кого він себе оголошує. Технологія автентифікації забезпечує контроль доступу для систем, перевіряючи, чи облікові дані користувача відповідають обліковим даним у базі даних авторизованих користувачів або на сервері автентифікації даних [7, 38].

Зазвичай процес ідентифікації користувачів використовує ідентифікатор користувача, а автентифікація виконується, коли користувач надає облікові дані, наприклад, пароль, що відповідає цьому ідентифікатору користувача. Більшість користувачів найбільш знайомі з використанням пароля, який як частина інформації, яка має бути відома тільки користувачеві, називається фактором автентифікації знань. Інші фактори автентифікації та способи їх використання для двофакторної або багатофакторної автентифікації (MFA) описані нижче.

Аутентифікація важлива, оскільки вона дозволяє організаціям захищати свої мережі, дозволяючи доступ лише для автентифікованих користувачів лише до довірених їм ресурсів (або процесів), які можуть включати КС, мережі, бази даних, веб-сайти та інші додатки до мережі або служби.

Після автентифікації користувач або процес зазвичай піддаються процесу авторизації, щоб визначити, чи автентифікований суб'єкт повинен мати доступ до захищеного ресурсу або системи. Користувач може бути автентифікований, але може не отримати доступ до ресурсу, якщо користувач не мав дозволу на його доступ.

Терміни автентифікації та авторизації часто використовуються як взаємозамінні; хоча часто можуть бути реалізовані разом, дві функції різні. Аутентифікація – це процес автентифікації зареєстрованого користувача, що

відбувається перед дозволом доступу до захищеного ресурсу, авторизація – це процес перевірки того, що автентифікований користувач отримав дозвіл на доступ до запрошених ресурсів. Процес, з якого доступ до цих ресурсів дозволено чи обмежений певному числу користувачів, називається контролем доступу.

Процес аутентифікації завжди постає перед процесом авторизації. Проведемо дослідження використання аутентифікації. Аутентифікація користувача відбувається в більшості взаємодій людини з КС за межами гостьових облікових записів, які автоматично реєструються. Як правило, користувач повинен вибрати ім'я користувача або ідентифікатор користувача та надати дійсний пароль для початку використання системи. Аутентифікація користувача дозволяє взаємодію між машинами, операційними системами та додатками, а також дротовими та бездротовими мережами для забезпечення доступу до мережевих та інтернет-підключених систем, додатків та ресурсів.

Багато компаній використовують автентифікацію для перевірки користувачів, які реєструються на їхніх сайтах. Без правильних заходів безпеки дані користувача, такі як номери кредитних і дебетових карт, а також номери соціального страхування, можуть потрапити в руки кіберзлочинців.

Організації також використовують автентифікацію для контролю доступу користувачів до корпоративних мереж та ресурсів, до комп'ютерів та серверів, до своїх додатків та мереж.

Для підприємств та інших великих організацій автентифікація може бути виконана за допомогою системи єдиного входу (SSO), яка надає доступ до кількох систем з одним набором облікових даних для входу.

1.2 Процес аутентифікації

Під час автентифікації облікові дані, надані користувачем, порівнюються з даними у файлі в базі даних користувача авторизованих користувачів або в

локальній операційній системі, або через сервер аутентифікації. Якщо облікові дані збігаються, і автентифікований суб'єкт має право використовувати ресурс, процес завершується, і користувачеві надається доступ. Надані дозволи та папки визначають як середовище, яке бачить користувач, так і спосіб взаємодії з нею, включаючи годинник доступу та інші права, такі як обсяг простору ресурсів.

Традиційно автентифікація здійснювалася системами чи ресурсами, яких здійснюється доступ; наприклад, сервер аутентифікації користувачів, використовуючи свою власну систему паролів, реалізовану локально, використовуючи ідентифікатори входу (імена користувачів) та паролі.

Передбачається, що знання облікових даних для входу гарантує дійсність користувача. Кожен користувач спочатку реєструється (або зареєстрований кимось іншим, наприклад, системним адміністратором), використовуючи призначений або самостійно оголошений пароль. При кожному наступному використанні користувач повинен знати та використовувати раніше оголошений пароль.

Тим не менш, протоколи додатків в Інтернеті, HTTP і HTTPS, не мають стану, що означає, що при суворій автентичності кінцеві користувачі перевіряються щоразу, коли вони звертаються до ресурсу за допомогою HTTPS. Замість того, щоб обтяжувати кінцевих користувачів цим процесом для кожної взаємодії через Інтернет, захищені системи часто покладаються на автентифікацію на токенах, в якій автентифікація виконується один раз на початку сеансу. Система автентифікації видає підписаний токен автентифікації в додаток кінцевого користувача, і токен додається до кожного запиту від клієнта.

Аутентифікація для систем і процесів може виконуватися з використанням облікових даних машини, які працюють як ідентифікатор користувача та пароль, за винятком того, що облікові дані автоматично надсилаються цим пристроєм. Вони можуть також використовувати цифрові сертифікати, які були випущені та перевірені центром сертифікації в рамках

інфраструктури відкритого ключа для автентифікації особи під час обміну інформацією через Інтернет.

Аутентифікація користувача з ідентифікатором користувача та паролем зазвичай вважається основним типом аутентифікації, і це залежить від користувача, що знає дві частини інформації: ідентифікатор користувача або ім'я користувача та пароль. Оскільки цей тип аутентифікації ґрунтується лише на одному факторі аутентифікації, це тип однофакторної аутентифікації.

Фактор автентифікації є частиною даних або атрибута, які можуть використовуватися для автентифікації користувача, що просить доступ до системи. Раніше виділялися такі фактори аутентифікації, як: «те, що ви знаєте, те, що у вас є або хто ви такий». Ці три фактори відповідають фактору знання, фактору володіння та фактору сутності. В останні роки було запропоновано та введено в дію додаткові фактори: місце розташування та час.

Фактори автентифікації, що використовуються в даний час, включають ряд чинників, наприклад чинник знання: «Те, що ви знаєте». Фактором знань можуть бути будь-які облікові дані аутентифікації, які складаються з інформації, яку має користувач, включаючи персональний ідентифікаційний номер (ПІН), ім'я користувача, пароль або відповідь на секретне запитання.

Ще одним чинником є володіння: «Те, що у вас є». Фактор володіння може являти собою будь-які облікові дані, засновані на елементах, якими користувач може володіти та носити з собою, включаючи апаратні пристрої, такі як токен безпеки або мобільний телефон, що використовується для прийому текстового повідомлення або запуску програми автентифікації, яка може генерувати одноразовий пароль або PIN-код.

Часто використовують чинник сутності: «Те, що ви є». Фактор сутності зазвичай ґрунтується на певній формі біометричної ідентифікації, включаючи відбитки пальців або пальців, розпізнавання обличчя, сканування сітківки або будь-яку іншу форму біометричних даних.

Фактор розташування передбачає відповідь на запитання: "Де ви знаходитесь". Хоча він може бути менш конкретним, фактор розташування іноді використовується як доповнення до інших факторів. Розташування може

бути визначене з достатньою точністю пристроями, оснащеними GPS, або з меншою точністю шляхом перевірки мережевих маршрутів. Фактор місцезнаходження зазвичай не може використовуватися для аутентифікації сам по собі, але він може доповнювати інші фактори, надаючи засоби для усунення деяких запитів. Наприклад, це може перешкодити зловмиснику, який знаходиться у віддаленій географічній області, створювати себе як користувач, який зазвичай входить до системи лише з дому чи офісу організації в рідній країні.

Чинник часу відповідає на питання: «Коли Ви аутентифікуєтеся». Як і фактор місцезнаходження, фактор часу сам по собі недостатній, але він може бути додатковим механізмом для відсіювання зловмисників, які намагаються отримати доступ до ресурсу, коли цей ресурс недоступний для авторизованого користувача. Він також може використовуватися разом із розташуванням. Наприклад, якщо користувач був останній раз автентифікований опівдні в США, спроба пройти аутентифікацію з Азії через годину буде відхилена на основі комбінації часу та розташування. Розташування користувача і поточний час самі по собі недостатні для автентифікації користувача без хоч одного з перших трьох факторів. Однак розповсюдженість смартфонів допомагає полегшити навантаження багатофакторної автентифікації для багатьох користувачів. Більшість смартфонів оснащені GPS, що забезпечує впевненість у підтвердженні розташування входу; MAC-адрес смартфона також можуть використовувати для автентифікації віддаленого користувача, незважаючи на те, що MAC-адреси відносно легко підмінити.

1.3 Аутентифікація та авторизація

Авторизація включає процес, за допомогою якого адміністратор надає права на автентифікованих користувачів, а також процес перевірки дозволів облікового запису користувача, щоб переконатися, що користувачеві надано

доступ до цих ресурсів. Права та привілеї, надані для авторизованого облікового запису, залежать від дозволів користувача, які зберігаються локально, або на сервері аутентифікації.

Параметри, визначені всім цих змінних середовища, встановлюються адміністратором.

Системі та процесам також може знадобитися авторизація їх автоматизованих дій у мережі. Онлайн-служби резервного копіювання, системи виправлення та оновлення та системи віддаленого моніторингу, такі як ті, що використовуються в технологіях телемедицини та смарт-сітки, усі мають надійно пройти автентифікацію.

Традиційна автентифікація залежить від використання файлу паролів, в якому ідентифікатори користувачів зберігаються разом з хешами паролів, пов'язаних з кожним користувачем. При вході в систему пароль, надісланий користувачем, хешується та порівнюється зі значенням у файлі пароля. Якщо збігаються два хеші, користувач автентифікується.

Такий підхід до аутентифікації має кілька недоліків, особливо це стосується ресурсів, розгорнутих у різних системах. По-перше, зловмисники, які можуть отримати доступ до файлу паролів для системи, можуть використовувати атаки грубої сили проти хешованих паролів для підбору паролів. Іншим недоліком є те, що цей підхід вимагатиме декількох аутентифікації для сучасних додатків, які отримують доступ до ресурсів у кількох системах. Недоліки автентифікації на основі паролів можуть бути певною мірою усунуті за допомогою більш інтелектуальних імен користувачів та правил пароля, таких як мінімальна довжина та умови складності, наприклад, включаючи цифри та символи. Однак автентифікація на основі пароля та автентифікація на основі знань більш вразливі, ніж системи, що потребують кількох незалежних методів.

Інші методи автентифікації включають можна поділити на декілька факторів. Сувора автентифікація – це термін, який формально не визначений, але зазвичай використовується для позначення того, що тип автентифікації, що використовується, є більш надійним і стійким до атаки. До строгої

аутентифікації в даний час відносять багатофакторну аутентифікацію та аутентифікацію на основі запит-відповідь з використанням криптографічних методів.

Двофакторна та багатофакторна аутентифікація Додавання декількох факторів аутентифікації до процесу аутентифікації зазвичай підвищує безпеку. Сувора автентифікація зазвичай відноситься до аутентифікації, яка використовує щонайменше два фактори різних типів. Відмінність важлива; оскільки ім'я користувача та пароль можна розглядати як типи факторів знання, можна сказати, що базова автентифікація побудована на використанні імені користувача та паролю використовує два фактори знання для аутентифікації – однак це не буде розглядатися як форма двофакторної аутентифікації (2FA).

Двофакторна автентифікація зазвичай залежить від фактора знання у поєднанні з біометричним фактором або фактором володіння, таким як токен безпеки. Системи 2FA часто вимагають, щоб користувач вводив код підтвердження, отриманий текстовим повідомленням на попередньо зареєстрованому мобільному телефоні, або код, створений додатком аутентифікації.

Багатофакторна аутентифікація вимагає від користувачів аутентифікації з кількома факторами аутентифікації, включаючи біометричний фактор, такий як відбиток пальця або розпізнавання особи, фактор володіння, такий як фейс-ключ безпеки або токен, створений програмою-автентифікатором. Багатофакторна аутентифікація може включати аутентифікацію будь-якого типу, яка залежить від двох або більше факторів, але процес аутентифікації за паролем плюс два різних типи біометричних даних не буде вважатися трифакторною аутентифікацією, хоча, якщо для цього процесу необхідний фактор знання, фактора володіння та фактора сутності, тоді вона буде трифакторною.

Трифакторна аутентифікація (3FA) – це тип MFA, який використовує три фактори аутентифікації різних типів, як правило, коефіцієнт знання (пароль) у поєднанні з фактором володіння (маркер безпеки) та коефіцієнтом спадкування (біометричний).

Прикладами чотирифакторної аутентифікації є системи, які потребують цих трьох факторів, а також географічний чи тимчасовий фактор. Одноразовий пароль. Одноразовий пароль є автоматично створюваним числовим або буквено-цифровим рядком символів, який автентифікує користувача. Цей пароль дійсний лише для одного сеансу входу або транзакції і зазвичай використовується для нових користувачів або для користувачів, що втратили свої паролі, і їм надається одноразовий пароль для входу в систему та переходу на новий пароль.

Хоча деякі системи аутентифікації можуть залежати виключно від біометричної ідентифікації, біометричні дані зазвичай використовуються як другий або третій фактор аутентифікації. Найбільш поширені типи біометричної автентифікації включають сканування відбитків пальців, сканування обличчя чи сітківки та розпізнавання голосу.

Мобільна автентифікація – це процес перевірки користувача через пристрої або перевірка самих пристроїв. Це дозволяє користувачам входити в безпечні місця та ресурси з будь-якого місця. Процес мобільної аутентифікації включає багатофакторну аутентифікацію, яка може включати одноразові паролі, біометричну аутентифікацію або перевірку QR-коду. При безперервній автентифікації замість того, щоб користувач увійшов до системи або вийшов з неї, додаток компанії постійно обчислює «оцінку справжності», яка вимірює, наскільки впевнений, що власник облікового запису є фізичною особою, яка використовує пристрій.

API (application programming interface) автентифікація має наступні принципи. Стандартними методами керування API автентифікацією є: базова автентифікація HTTP; ключі API та OAuth. У базовій аутентифікації HTTP сервер запитує інформацію аутентифікації, тобто ім'я користувача та пароль від клієнта. Потім клієнт передає інформацію автентифікації серверу в заголовку авторизації. У методі аутентифікації ключами API першому користувачеві надається унікальне згенероване значення, яке вказує, що відомий користувач. Потім кожен раз, коли користувач намагається знову увійти в систему, його

унікальний ключ використовується для перевірки того, що він той самий користувач, який раніше увійшов до системи.

Open Authorization (OAuth) є відкритим стандартом для аутентифікації та авторизації на токенах в Інтернеті. OAuth дозволяє використовувати інформацію облікового запису користувача сторонніми службами, такими як Facebook, без розкриття пароля користувача. OAuth виступає посередником від імені користувача, надаючи службі токен доступу, який дозволяє спільне використання певної інформації облікового запису.

Аутентифікація користувача та аутентифікація комп'ютера Ідентифікація машини може виконуватися з обліковими даними машини, як і ідентифікатор користувача та пароль, лише представлені цим пристроєм. Вони також можуть використовувати цифрові сертифікати, випущені та перевірені центром сертифікації, як частину інфраструктури відкритого ключа для підтвердження ідентифікації під час обміну інформацією через Інтернет, наприклад тип цифрового пароля. Зі збільшенням числа пристроїв з підтримкою Інтернету надійна автентифікація машини має вирішальне значення для забезпечення безпечного зв'язку для автоматизованої системи «Розумний дім» та іншими програмами Інтернету речей, де практично будь-який суб'єкт чи об'єкт можуть бути адресатами та здатні обмінюватись даними по мережі. Важливо розуміти, кожна точка доступу є потенційною точкою вторгнення. Кожному мережному пристрою потрібна надійна автентифікація машини, а також, незважаючи на їхню обмежену активність, ці пристрої також повинні бути налаштовані для обмеженого доступу до дозволу, щоб обмежити дії порушників.

Сувора автентифікація – це будь-який метод автентифікації користувача або пристрою, який за своєю суттю є досить суворим, щоб забезпечити безпеку системи, яку вона захищає, витримавши будь-які атаки, з якими вона може зіткнутися. Сувора автентифікація – це зазвичай використовуваний термін, який має єдиного стандартизованого визначення. На думку Європейського центрального банку (і багатьох організацій, які дотримуються своїх рекомендацій), сувора автентифікація поєднує як мінімум два взаємозалежні фактори, тому компроміс одного методу не повинен призводити до компромісу

другого. Крім того, метод автентифікації повинен включати один елемент, який не підлягає повторному використанню, який не можна легко відтворити або вкрасти з Інтернету. Тобто. у цьому випадку термін строга аутентифікація є синонімом двофакторної аутентифікації або багатофакторної аутентифікації. Однак таке трактування вводить в оману, оскільки деякі типи надійної аутентифікації засновані на одному факторі аутентифікації. У деяких випадках, наприклад, Recommendation X.509 вважаються строгою автентифікацією системи, що використовують множинні криптографічно захищені відповіді на запит/відповідь. Однак такі системи ґрунтуються на множинних екземплярах фактора знання, а не на кількох незалежних факторах і такий підхід є однофакторною автентифікацією (SFA).

Аналогічно, деякі типи біометричної аутентифікації досить надійні при їх використанні. У національному стандарті ДСТУ 9594-8-98, аналогічно міжнародному стандарту Recommendation X.509 (11/88) викоистовуються криптографічні методи [7]. У цьому ж стандарті виділяють і описують кроки, три процедури суворої аутентифікації: односпрямована, двоспрямована і триспрямована аутентифікація. У суворій автентифікації сторона, що перевіряється (доводить), доводить свою справжність стороні, що перевіряє, демонструючи знання будь-якого секрету за допомогою послідовності запитів і відповідей з використанням криптографічних методів і засобів [37]. Секрет, як правило, це секретний ключ, може бути попередньо розподілений безпечним способом і в ході автентифікаційного обміну не пересилається відкритими каналами.

В односторонній автентифікації справжність доводить лише одна сторона. Двостороння автентифікація в порівнянні з односторонньою містить аналогічну відповідь сторони, що перевіряє, що доводить стороні, що підтверджує її справжність. Тристороння автентифікація містить додаткову передачу даних від сторони перевіряючої, що дозволяє відмовитися від використання міток часу при проведенні аутентифікації [38]. Протоколи суворої автентифікації можуть бути засновані на використанні таких криптографічних методів: - симетричних методів шифрування;

односпрямованих ключових хеш-функцій; асиметричних методів шифрування; цифровий підпис.

Двостороння та тристороння процедури належать до взаємної аутентифікації. Взаємна автентифікація або двостороння автентифікація відноситься до двох сторін, які одночасно автентифікують одна одну, будучи режимом автентифікації за замовчуванням в деяких протоколах (IKE, SSH) і необов'язковим в інших (TLS). За замовчуванням протокол TLS підтверджує лише автентифікацію сервера для клієнта за допомогою сертифіката X.509, а автентифікація клієнта для сервера залишається на рівні додатків. TLS також пропонує клієнт-серверну аутентифікацію за допомогою аутентифікації X.509 на стороні клієнта. Оскільки це вимагає надання сертифікатів клієнтам і вимагає меншої зручності для користувача, він рідко використовується в програмах кінцевих користувачів.

Взаємна автентифікація TLS (mTLS) набагато ширше поширена в додатках для бізнесу (B2B), де обмежена кількість програмних та однорідних клієнтів підключаються до певних веб-служб, експлуатаційне навантаження обмежене, а вимоги до безпеки зазвичай набагато вищі, ніж у порівнянні зі споживчим середовищем .

Взаємна автентифікація між установами та клієнтами завадила б зловмисникам успішно видавати себе за фінансові установи для крадіжки облікових даних клієнтів; і найкраща автентифікація між клієнтами та установами завадила б зловмисникам успішно видавати себе за клієнтів за фінансові установи для шахрайства.

У більшості випадків взаємна автентифікація виконується за принципом «машина-машина», надаючи можливість користувачам стежити за випадками, коли віддалена автентифікація завершиться невдачею (наприклад, червоний замок в адресному рядку браузера або неправильне ім'я домену). Також існує для пом'якшення цієї проблеми нетехнічна взаємна автентифікація, що вимагає від користувача виконання завдання, ефективного примусу його до повідомлення та блокування автентифікації з використанням хибної кінцевої точки.

Взаємна автентифікація буває двох типів [46]:

- на основі сертифікату;

- на основі імені та пароля користувача.

При взаємній аутентифікації сервер та клієнт аутентифікують один одного. У разі використання взаємної автентифікації на основі сертифікатів виконуються такі дії.

1. Клієнт вимагає доступу до захищеного ресурсу.

2. Веб-сервер представляє сертифікат клієнту.

3. Клієнт перевіряє сертифікат сервера.

4. У разі успіху клієнт надсилає свій сертифікат на сервер.

5. Сервер перевіряє облікові дані клієнта.

6. У разі успіху сервер надає доступ до захищеного ресурсу, запрошеного клієнтом.

На рисунку 1.1 показано схему кроків взаємної аутентифікації на основі сертифікатів. Під час взаємної автентифікації на основі імені користувача та пароля виконуються такі дії.

1. Клієнт вимагає доступу до захищеного ресурсу.

2. Веб-сервер представляє сертифікат клієнту.

3. Клієнт перевіряє сертифікат сервера.

4. У разі успіху клієнт надсилає своє ім'я користувача та пароль на сервер, який перевіряє облікові дані клієнта.

5. Якщо перевірка пройшла успішно, сервер надає доступ до захищеного ресурсу, запрошеного клієнтом.

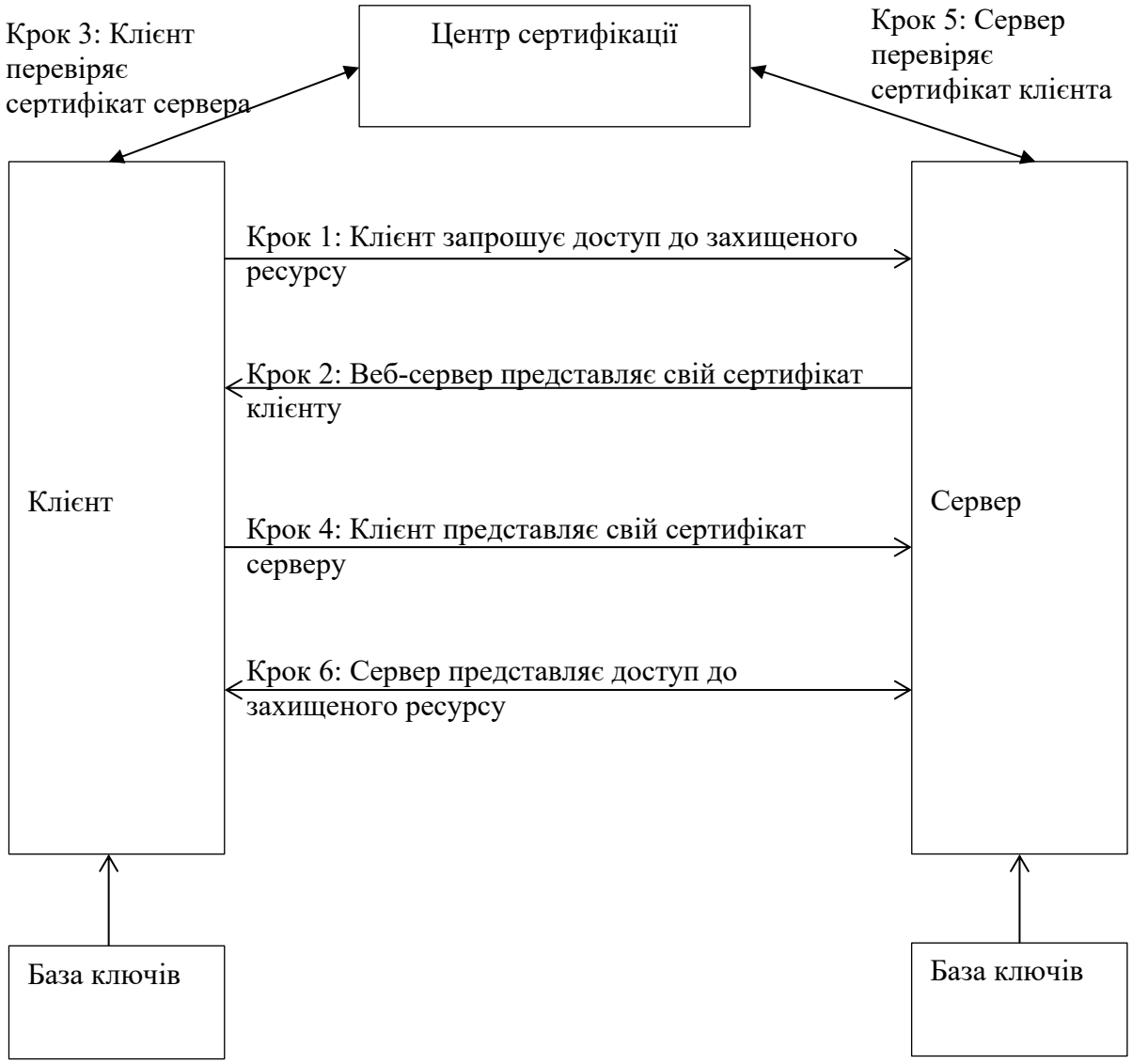

Рисунок 1.1 Взаємна автентифікація на основі сертифікатів

На рисунку 1.2 показано схему взаємної автентифікації на основі імені користувача та пароля. Цей вид автентифікації передбачає виконання кількох кроків. Першим кроком є запит до захищеного ресурсу, далі відбувається обмін сертифікатами та їхня перевірка. Часто для цього виду авторизації використовують центри сертифікації (АЦСК), які є провайдерами кореневих сертифікатів та служать серверами часу.

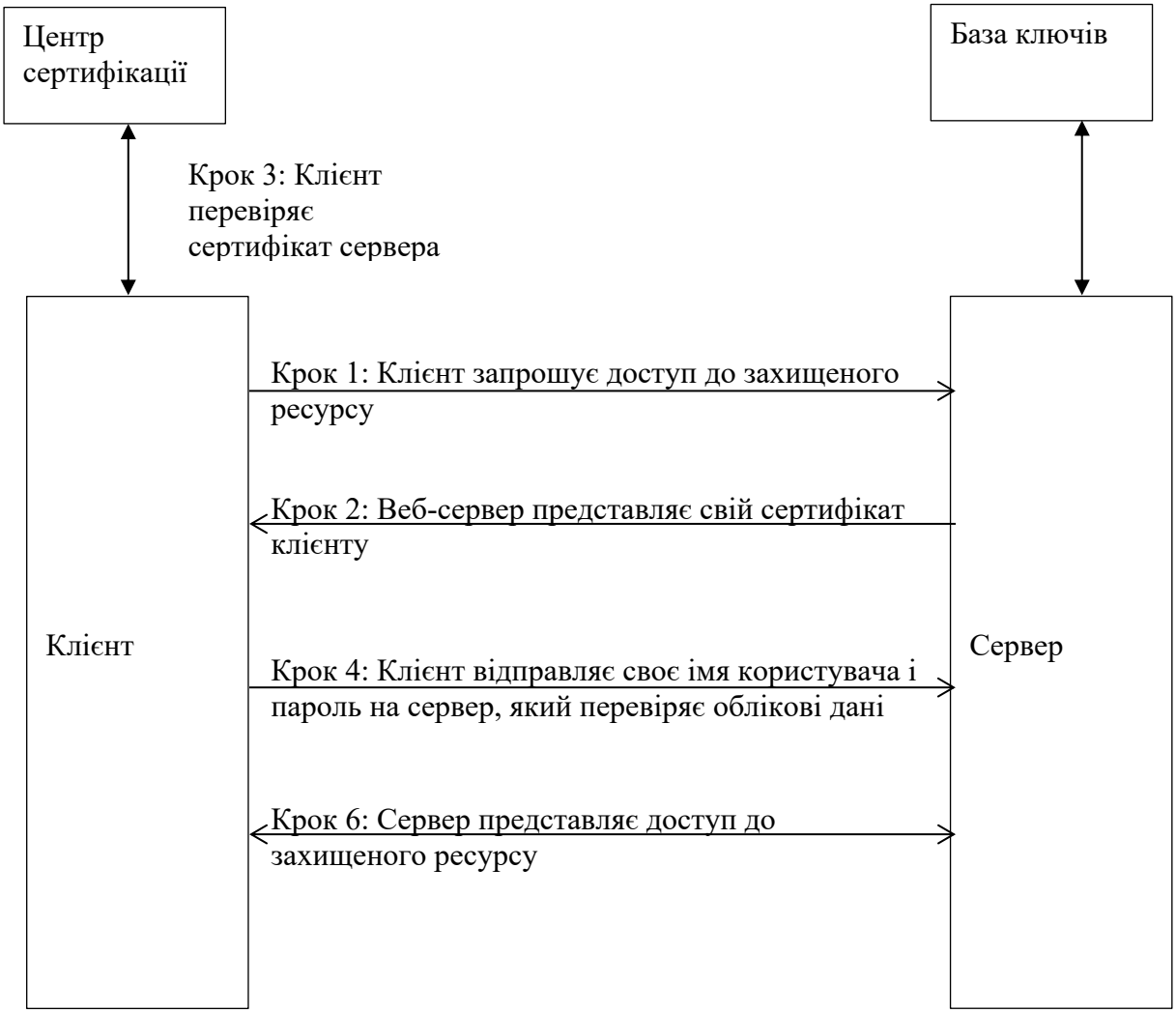

Рисунок 1.2 – Взаємна автентифікація на основі пароля та імені користувача

Схема з одностороннім звязком з центром сертифікації, якому довіряє клієнт приведена на рисунку 1.2.

1.4 Протоколи аутентифікації із симетричними алгоритмами шифрування

Представниками централізованих протоколів аутентифікації є протоколи розподілу секретних ключів: Нідхема і Шредера та Kerberos. Далі будуть розглянуті такі варіанти автентифікації:

- одностороння автентифікація з використанням міток часу або використанням випадкових чисел;

- двостороння автентифікація.

Знання секретного ключа доводиться шляхом розшифрування запитів з його допомогою.

Використовують позначення:

r_A – довільне число, згенероване користувачем A;

r_B - випадкове число, згенероване користувачем B;

t_A - мітка часу, згенерована користувачем A;

E_K – симетричне шифрування на ключі K (ключ До повинен бути попередньо розподілений між A і B).

$A \to B : M$ – Користувач A надсилає користувачеві B повідомлення M.

1. Одностороння автентифікація, заснована на мітках часу:

$$A \to B : E_K(t_A, B).$$

Користувач B перевіряє правильність мітки часу t_A та ідентифікатора B які не можна змінити без знання секретного ключа. Це захищає від повторної передачі повідомлення.

2. Одностороння автентифікація, заснована на використанні випадкових чисел:

$$B \to A : r_B,$$
$$A \to B : E_K(r_B, B).$$

Користувач A шифрує повідомлення, що складається з отриманого випадкового числа r_B та ідентифікатора B секретним ключем K, і відправляє його користувачу B. Тільки B, який володіє секретним ключем, зможе розшифрувати його і перевірити правильність r_B.

3. Двостороння автентифікація, яка використовує випадкові значення:

$$B \to A: r_B,$$

$$A \to B: E_K(r_A, r_B, B),$$

$$B \to A: E_K(r_A, r_B).$$

Аналогічно односторонньої аутентифікації обидва користувачі підтверджують свою справжність шляхом шифрування їх загальним секретним ключем K випадкових исел.

1.5 Протоколи, засновані на використанні односпрямованих ключових хеш-функцій

При аутентифікації за допомогою односторонньої хеш-функції всі користувачі сеансу використовують одну й ту саму процедуру одностороннього шифрування. Виділяють два способи.

Перший метод. Одностороння хеш-функція h_K з параметром-ключом K, застосована до повідомлення M довільного розміру, повертає хеш-значення m (дайджест) фіксованого розміру:

$$m=h_K(\mathrm{M}).$$

1. Користувач A обчислює $m=h_K(\mathrm{M})$ та надсилає повідомлення:

$$A \to B: m, M.$$

2. Користувач B, знаючи h_K і ключ K обчислює свій дайджест $m'=h_K(\mathrm{M})$ і порівнює з отриманим дайджестом: $m'=m?$, при збігу робить висновок, що повідомлення не змінено.

Знання дайджесту не дозволяє відновити вихідне повідомлення, але дозволяє перевірити цілісність даних. Другий спосіб. Одностороння хеш-

функція *h*(.) без параметра-ключа, застосовується до повідомлення *M*, доповненого секретним ключем *K*.

1. Користувач *A* обчислює *m=h* (M,K) та надсилає повідомлення:

$$A \to B: m, M .$$

2. Користувач *B*, знаючи *h* та ключ *K* обчислює свій дайджест *m'=h* (M,K) і порівнює з отриманим дайджестом: *m'=m?*, при збігу робить висновок, повідомлення не змінено.

1.6 Аутентифікація з використанням асиметричних алгоритмів

Протокол аутентифікації з використанням асиметричних алгоритмів. Позначення:

r – випадкове число;

$E_{K_A}(*)$– алгоритм асиметричного шифрування, відкритим ключем користувача *A*;

K_A– відкритий ключ користувача *A*;

h (*) – хеш-функція.

Схема протоколу аутентифікації:

$$B \to A: h(r), B, E_{K_A}(r, B),$$

$$A \to B: r .$$

Користувач *B* демонструє знання *r* без розкриття самого значення *r*. Користувач *A* перевіряє коректність *r* , *B* і *h(r)* і відправляє назад *r*.

Користувач *B* перевіряє коректність отриманого відправленим. , порівнюючи з *r*.

Схема протоколу аутентифікації – модифікований протокол Нідхема та Шредера:

$$A \to B: E_{K_B}(r_1, A),$$

$$B \to A: E_{K_A}(r_1, r_2),$$

$$A \to B: r_2.$$

1.7 Аутентифікація, заснована на використанні цифрового підпису

Розглядаються протоколи автентифікації з використанням цифрової підписи, міток часу та випадкових чисел представлені в рекомендаціях X.509 та відповідному національному стандарті [7].

Вводяться позначення: t_A, r_A, r_B – тимчасова мітка та випадкові числа відповідно;

S_A – підпис, згенерований користувачем A;

S_B - підпис, згенерований користувачем B;

$cert_A$ – сертифікат відкритого ключа користувача A;

$cert_B$-сертифікат відкритого ключа користувача B.

Тут сертифікати можна використовувати для підтвердження справжності відкритих ключів.

1. Одностороння автентифікація із застосуванням міток часу:

$$A \to B: cert_A, t_A, B, S_A(t_A, B).$$

Користувач A перевіряє коректність t_A , B і, використовуючи відкритий ключ із сертифіката $cert_A$, коректність $S_A(t_A, B)$.

2. Одностороння автентифікація з використанням випадкових чисел:

$$B \to A : r_B,$$

$$A \to B : cert_A, r_A, B, S_A(r_A, r_B, B).$$

Користувач *B* перевіряє коректність і, використовуючи відкритий ключ із сертифіката $cert_A$, коректність $S_A(r_A, r_B, B)$. Підписане випадкове число r_A використовується для запобігання атакам із вибіркою відкритого тексту.

3. Двостороння автентифікація з використанням випадкових чисел:

$$B \to A : r_B,$$

$$A \to B : cert_A, r_A, B, S_A(r_A, r_B, B),$$

$$B \to A : cert_B, A, S_B(r_A, r_B, A).$$

У цьому протоколі обробка першого та другого повідомлень виконується так само, як і в попередньому протоколі, а третє повідомлення обробляється аналогічно до другого повідомлення.

1.8 Реалізація протоколів аутентифікації

Коли користувач входить у систему, система повинна автентифікувати його (а іноді користувачеві потрібна автентифікація системи). Існує багато протоколів автентифікації. Нижче описано деякі з найбільш поширених:

1) PAP: протокол аутентифікації пароля - це найпростіша форма аутентифікації та найменш безпечна. Імена користувачів та паролі надсилаються у незашифрованому вигляді у вигляді простого тексту. Це, мабуть, дуже старий метод, який більше не використовується.

2) SPAP: Shiva Password Authentication Protocol протокол аутентифікації Shiva – це розширення для PAP, яке шифрує ім'я користувача та пароль, які надсилаються через Інтернет.

3) CHAP: Challenge Handshake Authentication Protocol: Протокол автентифікації під час виклику рукостискання обчислює хеш після входу користувача в систему. Потім він поділяє цей хеш із клієнтською системою. Періодично сервер запитуватиме у клієнта цей хеш. (Це частина завдання.) Якщо клієнт не підвіщує хеш, тоді зрозуміло, що повідомлення були скомпрометовані.

4) MS-CHAP є розширенням Microsoft для CHAP. Кроки переважно такі:

1 Після завершення фази рукостискання аутентифікатор (часто сервер) надсилає однорангове повідомлення «дзвінок».

2 Аналогова відповідь відповідає значенням, розрахованим із використанням функції одностороннього хеша.

3 Аутентифікатор перевіряє відповідь із власним розрахунком очікуваного значення хеш-функції. Якщо значення збігаються, автентифікація підтверджується; в іншому випадку з'єднання має бути припинено.

4 У випадкові інтервали аутентифікатор відправляє новий виклик партнеру і повторює кроки з 1 по 3. Ціла мета CHAP полягає не тільки в аутентифікації, а й у періодичній повторній аутентифікації, що запобігає атакам захоплення сеансу.

5) EAP: структура, що часто використовується в бездротових мережах і з'єднання точка-точка. Він був спочатку визначений в RFC 3748, але відтоді оновлений. Він обробляє передачу ключів та пов'язаних з ними параметрів. Існує декілька версій EAP. Він має безліч варіацій, у тому числі:

6) Протокол LEAP: полегшений протокол автентифікації, що розширюється, був розроблений Cisco і широко використовується в бездротовому зв'язку. LEAP підтримується багатьма операційними системами Microsoft, включаючи Windows 7 та новіші версії. LEAP використовує модифіковану версію MS-CHAP.

7) Протокол автентифікації, що розширюється - Transport Layer Security безпечний транспортний рівень TLS для забезпечення аутентифікації. Більшість реалізацій EAP-TLS використовують цифрові сертифікати X.509 для автентифікації користувачів.

8) Protected Extensible Authentication Protocol (PEAP): шифрує процес аутентифікації за допомогою аутентифікованого тунелю TLS. PEAP був розроблений консорціумом, включаючи Cisco, Microsoft та RSA Security. Він був уперше включений до Microsoft Windows XP.

9) Kerberos: Kerberos заснований на протоколі Ніхейма-Шредера, що широко використовується, особливо в операційних системах Microsoft. Він був винайдений у Массачусетському технологічному інституті і отримав свою назву від міфічного триголового собаки, який, як вважали, охороняв ворота Пекла. Система трохи складна, але основний процес виглядає так: коли користувач входить до системи, сервер автентифікації перевіряє ідентифікатор користувача, а потім зв'язується із сервером надання мандатів. (Вони часто знаходяться на одній машині.) Сервер надання мандатів відправляє зашифрований мандат на машину користувача. Цей мандат ідентифікує користувача під час входу до системи. Пізніше, коли користувачеві необхідно отримати доступ до деякого ресурсу в мережі, машина користувача використовує цей мандат для надання мандатів для отримання доступу до цільової машини. Оскільки Kerberos настільки широко використовується, він розглядається докладніше, ніж інші методи автентифікації. Хоча існують варіанти виконання, основний процес показаний рисунку 1.3.

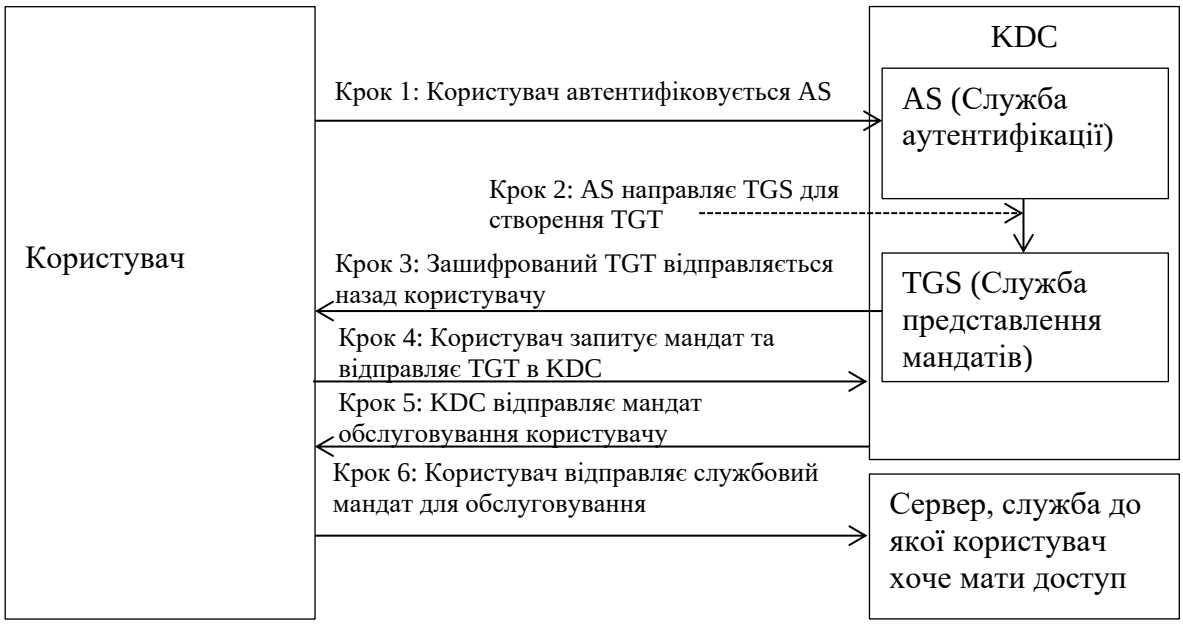

Рисунок 1.3 – Схема авторизації Kerberos

Для мандатів є багато перевірок, і ці мандати закінчуються відносно короткі терміни. Принципал: сервер чи клієнт, яким Kerberos може призначати мандати. Сервер аутентифікації (AS): сервер, який авторизує принципала та підключає його до сервера для подання заявок.

Kerberos включає такі елементи, як сервер видачі мандатів (TGS): надає мандати. Центр розподілу ключів (KDC): сервер, який надає початковий мандат та обробляє запити TGS. Часто він запускає служби AS та TGS. Слід зазначити, що Kerberos є одним із найбільш широко використовуваних протоколів автентифікації. Європа часто використовує альтернативну SESAME Secure European System для програм у середовищі з кількома продуктами. Додатковий опис Kerberos можна переглянути в [40]. Таким чином, аутентифікація приходить у різних видах та способах, і професіонали в галузі безпеки повинні знати якнайбільше про аутентифікацію.

2 ФОРМАЛІЗАЦІЯ СИСТЕМ АУТЕНТИФІКАЦІЇ

2.1 Біометрична система аутентифікації з використанням голосових даних

Ідентифікація людини за голосом є одним із біометричних систем аутентифікації. Біометрична система аутентифікації – система перевірки особистості (аутентифікації) людини за її біометричними показниками Біометричний параметр – параметр, що є частиною самої людини.

Біометричні системи аутентифікації досить зручні для користувачів. На відміну від парольних або ключових систем автентифікації, біометричні використовують параметри, які неможливо забути або втратити. Проблеми збереження автентифікаційних даних не виникає. Основними перевагами використовуваного підходу є простота реалізації та висока надійність.

Біометричний метод аутентифікації за голосом є простим у застосуванні. Він не вимагає спеціальної апаратури, достатньо лише мікрофона та звукової плати. В даний час технологія розвивається, так як цей метод широко використовується у сучасних бізнес-центрах.

Основним недоліком методу голосової автентифікації є його низька точність. Голос людини може змінюватись залежно від стану здоров'я, настрою, віку тощо. Крім того, не варто забувати про сторонні шуми. Через високу ймовірність помилок другого роду його застосовують переважно у приміщеннях середнього рівня безпеки [1].

Один із способів аутентифікації людини за голосом – це завдання системі безлічі зразків голоси однієї і тієї ж людини для порівняння з автентифікаційним ключем, що отримується в майбутньому. Далі існує безліч способів порівняння [1, 2].

Система, яку передбачається розробити, має бути незалежною від конкретної фрази. Через це може виникнути проблема з підміною голосу за допомогою звукозаписних пристроїв, наприклад, диктофонів. Однак цю проблему можна обійти за допомогою генерації простих фраз, які людина має вимовити. Не можна заздалегідь передбачити, яка фраза буде затребувана

системою, що розпізнає, отже зловмисник не зможе записати ключову фразу. Такий підхід додає завдання перекладу мови до тексту, проте вона не є основною і може бути вирішена за допомогою сторонніх бібліотек [3].

На вхід системи ідентифікації надходить запис ключового повідомлення користувача. Одним із простих форматів для обробки є WAV.

WaveformAudioFileFormat – формат файлу-контейнера для зберігання записи оцифрованого аудіопотоку, підвид RIFF. Цей контейнер, як правило, використовується для зберігання несжатого звуку в імпульсно-кодовий модуляції.

Отримані значення амплітуд можуть не збігатися для двох однакових записів через зовнішній шум, різні гучності вхідного сигналу і так далі. Одним із найбільш ефективних способів попередньої підготовки звуку є нормалізація [4].

Нормалізація звуку – процес вирівнювання частотних характеристик студійного звукозапису на магнітний носій. Корекція необхідна, оскільки процес намагнічування покриття плівки відбувається нерівномірно стосовно спектру аудіочастот. Якщо не проводити корекцію, навіть перше відтворення запису звучатиме несхоже на оригінал.

Існують два способи нормалізації [4]:

- пікова нормалізація – це спосіб нормалізації, за якого рівень звукового сигналу піднімається до максимально можливого значення цифрового звуку без появи спотворень. Орієнтиром служить найвищий пік амплітуд. Цей спосіб повністю виключає обмеження амплітуди сигналу (кліпінг), проте, за наявності у файлі сильно виділяється піку, то нормалізація за його рівнем може призвести до тому, що звуковий сигнал залишиться досить тихим, незважаючи на досить високу гучність оригіналу. Розмір звуку при пікової нормалізації вимірюється у відсотках;

- RMS-нормалізація - нормалізація за середньоквадратичним значенням рівня звуку у файлі. Є повною протилежністю до пікової нормалізації. При цьому способі величина звуку вимірюється у децибелах. Цей спосіб найбільш підходить для людського вуха, однак за високої гучності можливий кліпінг.

Оскільки передбачається, що людина вимовлятиме ключове вираз спокійний, то очевидна перевага у пікової нормалізації внаслідок того, що в необробленому звуку не повинно бути надто великих перепадів амплітуд.

Так як розмір унікальних характеристик навіть для секундного зразка звуку величезний, то робити складні операції над такими обсягами даних неможливо. Крім того, не зовсім зрозуміло, як порівнювати об'єкти з різною кількістю унікальних характеристик.

Обчислювальну складність завдання можна зменшити, розбивши її на менш складні підзавдання. Це дозволить за допомогою встановлення фіксованого розміру підзадачі та усереднення результатів обчислень за всім завданням одержати наперед задану кількість ознак для класифікації. Як розбиття мається на увазі використання поділу звукового сигналу на звані кадри певної довжини. Кадри повинні перекривати один одного, тому що у випадку, якщо вони будуть поруч один з одним, то звук спотворюватиметься.

Для усунення небажаних ефектів при обробці кадрів кожен елемент кадру множиться на вікно. Вікно – вагова функція, яка використовується для керування ефектами, зумовленими наявністю бічних пелюсток у спектральних оцінках (розтікання спектра).

У більшості завдань цифрової обробки немає можливості досліджувати сигнал на безкінечному інтервалі. Немає можливості дізнатися, який був сигнал до увімкнення пристрою і який він буде в майбутньому. Також обмеження інтервалу дослідження може бути обумовлено нестаціонарністю досліджуваного сигналу.

Обмеження інтервалу аналізу рівносильне добутку вихідного сигналу на віконну функцію. Таким чином, результатом віконного перетворення Фур'є не спектр вихідного сигналу, а спектр твори сигналу та віконної функції. Спектр, отриманий за допомогою віконного перетворення Фур'є [5] є оцінкою спектра вихідного сигналу і принципово допускає спотворення.

Спотворення, що вносяться застосуванням вікон, визначаються розміром вікна та його формою. Виділяють дві основні властивості частотних характеристик вікон: ширина головної пелюстки та максимальний рівень

бічних пелюсток. Застосування вікон, відмінних від прямокутних, обумовлене бажанням зменшити вплив бічних пелюсток за рахунок збільшення ширини головного.

Типи віконних функцій:

Прямокутне вікно:

$$w(n) = \begin{cases} 1, n \in [0, N-1] \\ 0, n \notin [0, N-1] \end{cases}. \quad (2.1)$$

Вікно Ханна:

$$w(n) = 0.5(1 - \cos(\frac{2\pi n}{N-1})). \quad (2.2)$$

Вікно Хемінга:

$$w(n) = 0.53836 - 0.46164(\cos(\frac{2\pi n}{N-1})). \quad (2.3)$$

Вікно Блекмена:

$$w(n) = a_0 - a_1 \cos(\frac{2\pi n}{N-1}) + a_2 \cos(\frac{4\pi n}{N-1}). \quad (2.4)$$

Вікно Кайзера:

$$w(n) = \frac{\left| l_0(\beta \sqrt{1 - \left(\frac{2n - N + 1}{N-1} \right)^2}) \right|}{|l_0(\beta)|}. \quad (2.5)$$

Найбільш простий і підходящою для вирішення задачі є функція вікна Хеммінгу.

Далі потрібно отримати короткочасну спектрограму кожного кадру окремо. Для цього використовується перетворення Фур'є.

Перетворення Фур'є (Fouriertransform) - це розкладання функцій на синусоїди (далі косинусні функції теж називаємо синусоїдами, оскільки вони відрізняються від «справжніх» синусоїд тільки фазою). Існує кілька видів перетворення Фур'є [5].

1. Неперіодичний безперервний сигнал можна розкласти в інтеграл Фур'є.

2. Періодичний безперервний сигнал можна розкласти у нескінченний ряд Фур'є.

3. Неперіодичний дискретний сигнал можна розкласти на інтеграл Фур'є.

4. Періодичний дискретний сигнал можна розкласти в кінцевий ряд Фур'є.

Комп'ютер здатний працювати лише з обмеженим обсягом даних, отже, реально він здатний обчислювати лише останній вид перетворення Фур'є. Отже, використовуватиметься дискретне перетворення.

На сьогоднішній день найбільш успішними є системи розпізнавання голосу, які використовують знання про слуховий апарат. Велике поширення при розпізнаванні людського мовлення набула mel-шкала, лінійна при частотах нижче 1кГц і логарифмічна при частотах вище 1кГц. Mel-шкала була отримана в результаті експериментів із зразковими тонами (синусоїдами) в яких з випробовуваних вимагалося розділити дані діапазони частот на 4 рівні інтервалу або налаштувати частоту необхідного тону так, щоб він був в половину частоти вихідного. 1 mel визначається як 1 тисячна рівня тону на 1 кГц. як і в будь-яких інших спробах створити подібні шкали, розраховується, що шкала mel більш точно моделює чутливість людського вуха.

Перехід до нової шкали описується нескладною залежністю:

$$m = 1127\ln(1 + \frac{f}{700}), \tag{2.6}$$

де m - Частота в крейдах; f – частота у герцах. Вектор ознак складатиметься з крейдяних коефіцієнтів, що розраховуються за формулою:

$$c_n = \sum_{k=1}^{K} (\log S_k) \left[n(k - \frac{1}{2}) \frac{\pi}{K} \right], \quad (2.7)$$

де c_n - Крейда-кепстральний коефіцієнт під номером n; S_k - Амплітуда kго значення в кадрі в крейдах; K – наперед задану кількість дрібнепстральних коефіцієнтів.

Останньою стадією є класифікація того, хто говорить. Класифікація проводиться обчисленням міри схожості пробних даних та вже відомих. Міра схожості виражається відстанню від вектора ознак пробного сигналу до ознак уже класифікованого вектора.

Вектор ознак представляється як середнє арифметичне векторів, що характеризують окремі кадри мови. Для підвищення точності розпізнавання просто необхідно усереднювати результати не тільки між кадрами, а й враховувати показники кількох мовних зразків. Маючи кілька записів голосу, розумно не усереднювати показники одного вектора, а провести кластеризацію з допомогою нейронних мереж.

2.2 Алгоритми розізнання в бібліотеці OPENCV

Виявлення та розпізнавання обличчя людини часто застосовується у системах ідентифікації особистості. Проведемо дослідження алгоритмів, що використовуються для цих цілей у бібліотеці OpenCV. Зокрема, описано алгоритми розпізнавання обличчя: Eigenfaces, Fisherfaces та LBPH.

Процедура розпізнавання має декілька етапів. Основна мета розпізнавання – зіставити особу на зображенні з інформацією про особистість людини. Ця процедура зазвичай використовується після виявлення та працює із зображеннями осіб, які вже виявлені, витягнуті, обрізані, вирівняні та змінені у розмірі, тому алгоритм може зосередитися на пошуку важливих характеристик

на зображенні особи, які найкраще описують особистість людини та дозволяють розпізнати його максимально можливої кількості зображень.

Розпізнавання завжди складається з двох основних частин, незалежно від використовуваного методу або алгоритму, навчання розпізнавача та самого розпізнавання [3]. У першій частині розпізнавач витягує корисні характеристики з набору даних зображень, визначеного на навчання.

Цей набір даних дуже важливий для точності розпізнавання, на кожному зображенні має бути лише одна особа, щоб згодом зіставити особу з особистістю. Зазвичай набір даних містить підмножину зображень, на яких зображена одна і та сама людина, але з різних ракурсів і з різною ступенем освітлення. Точна реалізація залежить від використовуваного інструменту чи бібліотеки. Незважаючи на те, що розпізнавання може бути на відеозаписі (навіть у реальному часі), розпізнавання осіб виконується на зображеннях, відеозаписах вибираються певні кадри (частота кадрів залежить від реалізації) і лише на цих кадрах виконується розпізнавання.

Бібліотека містить в собі методи для розпізнавання зображень. Безпосередньо у бібліотеці OpenCV реалізовано три основні алгоритми розпізнавання осіб, які можуть використовуватися для ідентифікації особи за відомими та навченими наборами даних, алгоритми сильно різняться за точністю, умовами введення та швидкості виконання.

Для розпізнавання осіб розпізнавач повинен мати у своєму розпорядженні набір підготовлених відповідним чином навчальних зображень. Кожне зображення в наборі має мати ідентифікатор, який може бути номером чи ім'ям у вигляді рядка. Зображення однієї та тієї ж людини повинні мати однаковий ідентифікатор, щоб алгоритм міг поєднати кілька уявлень. Вже обрізані та вирівняні зображення містяться у функції як тренувальні.

Розпізнавачі в OpenCV це окремий набір функцій. Розпізнавачі осіб OpenCV мають загальний абстрактний базовий клас cv::face::FaceRecognizer, від якого походять класи cv::face::BasicFaceRecognizer, що забезпечують як алгоритм Eigenfaces (Створений за допомогою createEigenFaceRecognizer()), так і Fisherfaces (створений за допомогою createFisherFaceRecognizer()) та

cv::face::LBPHFaceRecognizer, що відповідає за розпізнавання осіб LBPH з конструктором createLBPHFaceRecognizer (), [6]. Усі три методи мають загальний інтерфейс вивчення осіб train () і розпізнавання їх predic (). Виявлення осіб добре підходить для фронтального виявлення, змішування фронтальних зображень із зображеннями профілю ускладнює ідентифікацію та знижує загальну точність.

Алгоритм Eigenfaces для розпізнавання облич одним із перших алгоритмів, описаних для розпізнавання об'єктів, і, як Випливає з назви, заснованої на Eigenfaces. Eigenfaces - складові риси особи, і комбінуючи певну кількість рис обличчя з вагою, можна реконструювати зовнішній вигляд людини Власні риси створюються з навчального зображення за допомогою аналізу головних компонентів (PCA), або точніше, на коваріаційній матриці навчальних даних за допомогою цього методу виходить велика кількість пар власних векторів та власних значень, а власний вектор із найбільшими власними значеннями називається власними рисами. Цей алгоритм пропонує відмінний стиск даних, що зберігаються, отже навчальна бібліотека менша, ніж в інших алгоритмах, тому що він представляє кожну особу як лінійну комбінацію зважених власних векторів, які потребують невеликого обсягу пам'яті.

Однак великим недоліком є висока сприйнятливість до різних умов освітлення або зміни фону.

На етапі навчання зображення з набору навчальних даних перетворюється на плоский вектор - з матриці зображення з розміром $[mxn]$ в вектор зображення з розміром $[(mxn)\ x1]$. Послідовно це означає μ всіх векторів (зображень з набору навчальних образів) та $xi \in Rd$, який є вектором зображення із розмірністю.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} xi$$

Матриця підступів вимірює відхилення значень у всіх вимірах, в яких обчислюється. Корисно визначити, які виміри різняться найбільше, що означає, що вони містять багато даних для розпізнавання з вимірів, які дуже відрізняються й у розпізнаванні не важливі [8].

Щоб краще зрозуміти обчислення, розберемо їх крок за кроком. Спочатку віднімаючи середній вектор μ з кожного вектора, отримуємо різницю i між конкретним зображенням i та середнім зображенням. Складаючи всі матриці різниць разом, отримуємо матрицю $A = \{1,. . . , Ph\ in\} \in RDxM$ та коваріаційна матриця просто обчислюється з цієї матриці А як $C = A * transpose\ (A)$. Весь процес за один крок можна подати наступним чином:

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T.$$

З коваріаційної матриці C обчислюються власні значення λ_i та власні вектори $Sv_i = \lambda_i v_i\ i = 1, 2,. . . , n$. На наступному етапі власні вектори v_i упорядковуються відповідно до їх власного значення λi в порядку зменшення. Береться k кількість власних векторів із найвищими власними значеннями, кількість власних векторів k потім вирішується, наскільки точним може бути алгоритм, тому що кожен власний вектор v_i представляє певну межу або особливість. Заключний крок навчання полягає в тому, щоб з'ясувати, наскільки кожна з k рис покрита для кожного навчального зображення, яке пізніше буде використано для розпізнавання.

На етапі розпізнавання перші кроки аналогічні до етапу навчання: перетворення зображення на плоский вектор та його нормалізація по середньому вектор, який був обчислений на етапі навчання. Потім знаходимо вагу зображення та обчислюємо помилку між вагами зображення та всіх навчальних зображень. Помилка може бути сумою простих чи евклідових відстаней кожної пари терезів. Якщо найменша помилка нижче заданого порога, зображення ідентифікується як зображення з навчального набору з

найменшою помилкою, інакше людина на зображенні позначається як невідомий. В OpenCv алгоритм реалізований у класі EigenFaceRecognizer.

Алгоритм Fisherface був створений для підвищення точності методу Eigenface. Аналіз основних компонентів, що становить ядро алгоритму Eigenface, фокусується на максимальній дисперсії зображення, яке добре представляє дані та не вимагає багато пам'яті. Коли деякі компоненти вибираються, а деякі відкидаються, губиться багато розрізняючої інформації. З іншого боку, метод Fisherface застосовує лінійний дискримінантний аналіз (LDA) для зменшення розмірності та отримання характеристик, які найкраще поділяють людей. Ціль LDA полягає в тому, щоб максимізувати співвідношення між індивідуальним розкидом і всередині індивідуальним розкидом, тим самим отримуючи найбільш корисні функції для розрізнення.

Для досягнення незалежності від умов освітлення важливо надати кілька зображень до навчальної бібліотеки з усіма можливими станами та виразами обличчя кожної людини [1]. Цей метод покращує точність Eigenfaces в основному для ситуацій, коли Eigenfaces не справлявся найчастіше: зображення з різними умовами освітленнями [2].

Алгоритм працює з безліччю C класів / індивідів $X = \{X1,. . . , XC\}$, клас Xi - це вектор довжини N, створений зображенням, що згладжує i з навчальної вибірки: $Xi = \{x1,. . . , xn\}$. Як і в методі Eigenfaces, обчислюється середнє всіх векторів μ, а також μ_i середнє значення класу i: $i \in \{1,. . . , C\}$.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \,.$$

$$\mu_i = \frac{1}{|X_i|}\sum_{x_j \in X_i} x_j \,.$$

Використання μ і μ_i дозволяє обчислити розкид між класами SB та розкид всередині класів SW, де μ - загальне середнє значення, μ_i - середнє значення класу i, обчислений на попередньому кроці. Змінна c - це кількість класів, N -

кількість вибірок для класу i, X_i - це одна вибірка з класу i, x_j показує приклад цього X_i [7].

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T;$$

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_i - \mu_i)^T.$$

На наступному етапі обчислень він шукає оптимальну Fisherface $Wopt$, яка максимізує ставлення детермінанта між розкидом усередині класу SW та розкидом між класами SB і, таким чином, поділом класів. Це неможливо вирішити, тому що SW має розмір не більше (N - c), N - це кількість вибірок, а c - кількість класів, що робить матрицю розкиду SW сингулярної. Зразки проектуються в (N - c) –мірне простір отримати узагальнені власні вектори Wfld.

Остаточна матриця W, яка проектує зразок у (c-1) –мірне простір, що виходить у рівнянні 2.10 [4].

$$W_{opt} = \arg\max_w \frac{\left|W^T S_B W\right|}{\left|W^T S_w W\right|};$$

$$W_{pca} = \arg\max_w \left|W^T S_T W\right|;$$

$$W_{opt} = \arg\max_w \frac{\left|W^T W_{pca}^T S_B W_{pca} W\right|}{\left|W^T W_{pca}^T S_W W_{pca} W\right|};$$

$$W = W_{fld}^T W_{pca}^T.$$

Подібно до методу Eigenfaces, Fisherface обчислює лінійний дискримінантний аналіз, який дозволяє порівнювати результати тестованого зображення та зображень, збережених під час фази навчання. Чим більша різниця при прогнозуванні двох векторів, що містять ознаки, які поділяють

індивідів *W*, тим нижча впевненість у результатах. У класі OpenCV, що реалізує метод Fisherface, називається FisherFaceRecognizer.

2.3 Метод розпізнавання особи Віоли-Джонса

Хоча метод був розроблений і представлений в 2001 році Полом Віолой і Майклом Джонсом [1, 2], він до цих пір є основним для пошуку об'єктів на зображенні в реальному часі [2].

Основні принципи , на яких заснований метод, такі:

- використовується зображення в інтегральному представленні , що дозволяє швидко вичислити необхідні об'єкти;
- використовуються признаки Хаара , за допомогою яких відбувається пошук цього потрібного об'єкта (в контексті, обличчі та його рис);
- використовується бустинг (від англ. boost – покращення, підсилення) для вибору найбільш підходящих ознак для іншого об’єкта на цій частині зображення;
- всі ознаки йдуть на вхід класифікатора , який дає результат «вірно» або «хибно»;
- використовуються каскади ознак для швидкого видалення вікон, де не знайдено обличчя.

Навчання класифікаторів йде дуже повільно, але результати пошуку обличчя дуже швидкі , саме тому був обраний даний метод розпізнавання осіб на зображенні. Віола-Джонс є одним із найкращих за співвідношенням показників ефективності розпізнавання/швидкості роботи. Також цей детектор має надзвичайно низьку ймовірність хибного виявлення особи. Алгоритм навіть добре працює і розпізнає риси обличчя під невеликим кутом, приблизно до 30 градусів. При нахилі кута більше 30 градусів процент виявлення різко падає. І це не дозволяє в стандартній реалізації виявити зворотне обличчя людини під довільним кутом, що в значній мірі ускладнює або робить неможливим

використання алгоритму в сучасних виробничих системах з їх зростаючими потребами.

Потрібен детальний розбір принципів, на яких заснований алгоритм Віоли-Джонса. Данний метод в загальном вигляді шукає обличчя і риси обличчя за загальним принципом скануючого вікна .

У загальному вигляді задача виявлення особи і рисунок особи людини на цифровому зображенні виглядає саме так:

- маємо зображення , на котрому потрібно шукати обєкти . Воно представлено двомірною матрицею пікселів розміром *w***h*, у якій кожен піксель має значення:
- — від 0 до 255, якщо це чорно-біле зображення;
- —якщо це кольорове зображення (компоненти R, G, B).от 0 до 255^3;
- в результаті своєї роботи алгоритм повинен визначити обличчя та його риси і помітити їх – пошук здійснюється в активній області зображення прямоугольними визнаннями , за допомогою яких описується знайдене обличчя та його характеристики: (1.1) де *x, y* – координати центру *i*-го прямокутника , *w* – ширина, *h* – висота, а – кут нахилу прямокутника до вертикальної осі зображення.

$$rectangle_i = \{x,y,w,h,a\},$$

Іншими словами, переважно до малюнків і фотографій, використовується підхід на основі скануючого вікна (вікно сканування) : сканується зображення вікна пошуку (так зване, вікно сканування), а потім застосовується класифікатор до кожного положення. Система навчання і вибір найбільш значущих прмзнаків повністю автоматизована і не вимагає впливу людини, тому цей підхід працює швидко.

Задача пошуку та знаходження особи на зображенні за допомогою цього принципу часто буває черговим кроком на шляху до розпізнавання характерних рис, до прикладу, перевірки людини за розпізнаним обличчям або розпізнавання миміки обличчя.

Для того, щоб зробити які-небудь дії з даними, використовується інтегральне представлення зображень [3] в методі Віоли-Джонса. Таке представлення часто використовується і в інших методах, до прикладу, у вейвлет-преутвореннях, SURF і багатьох інших розроблених алгоритмах. Інтегральне зображення дозволяє швидко розрахувати суммарну яскравість довільного прямокутника на даному зображенні, причому який би прямокутник не був, час розрахунку незмінний.

Інтегральне зображення – це матриця, що складається за розмірами з вихідного зображення . У кожному її елементі зберігається сума інтенсивності всіх пікселів, що знаходяться лівіше і вище даного елемента. Елементи матриці розраховуються за наступною формулою:

$$L(x, y) = \sum_{i=0,\, j=0}^{i \le x,\, j \le y} * I(i, j), \tag{2.8}$$

де $I(i,j)$ — яскравість пікселя вихідного зображення.

Кожен елемент матриці $L[x,y]$ являє собою суму пікселів у прямокутнику від (0,0) до (x,y), тобто значення кожного пікселя (x,y) дорівнює сумі значень усіх пікселів лівіше і вище даного пікселя (x,y). Розрахунок матриці займає лінійний час, пропорційний числу пікселів у зображенні, тому інтегральне зображення розраховується за один прохід.

Розрахунок матриці можливий за формулою 9:

$$L(x,y) = I(x,y) – L(x-1,y-1) + L(x,y-1) + L(x-1,y) \tag{2.9}$$

По такій інтегральній матриці можна дуже швидко вирахувати суму пікселів довільного прямокутника, довільної площі. Нехай в прямокутнику ABCD нас цікавить об’єкт D (рисунок 2.1).

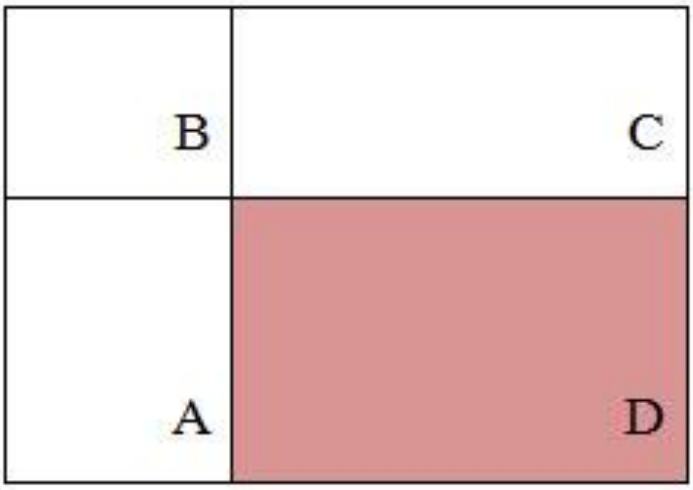

Рисунок 2.1 – Прямокутник ABCD

З рисунка зрозуміло, що суму всередині прямокутника можна виразити через суми та різницю суміжних прямокутників за наступною формулою:

$$S(ABCD) = L(A) + L(C) — L(B) — L(D) \quad (2.10)$$

Ознака — відображення f: $X => D_f$, де D_f — безліч допустимих значень ознаки. Якщо задані ознаки $f_1,\ldots,f_n$, то вектор ознак $x = (f_1(x),\ldots,f_n(x))$ називається визнаним описом об'єкта $x \in X$. Ознакові описи допустимо співставляти із самими об'єктами. При цьому множина $X = D_{f1} * \ldots * D_{fn}$ називають визначеним простором [1].

Признаки поділяються на наступні типи в залежності від множини D_f:

- бінарна ознака, $D_f = \{0,1\}$;
- номінальний признак: D_f — кінцева множина;
- порядковий признак: D_f — кінцева впорядкована множина;
- кылькысний признак: D_f — множина дійсних чисел.

Звичайно, бувають прикладні завдання з різними визнаннями, для їх вирішення підходять далеко не всі методи.

У стандартному методі Віоли – Джонса використовуються прямокутні признаки, зображені на рисунку нижче, вони називаються примітивами Хаара (рисунок 2.2).

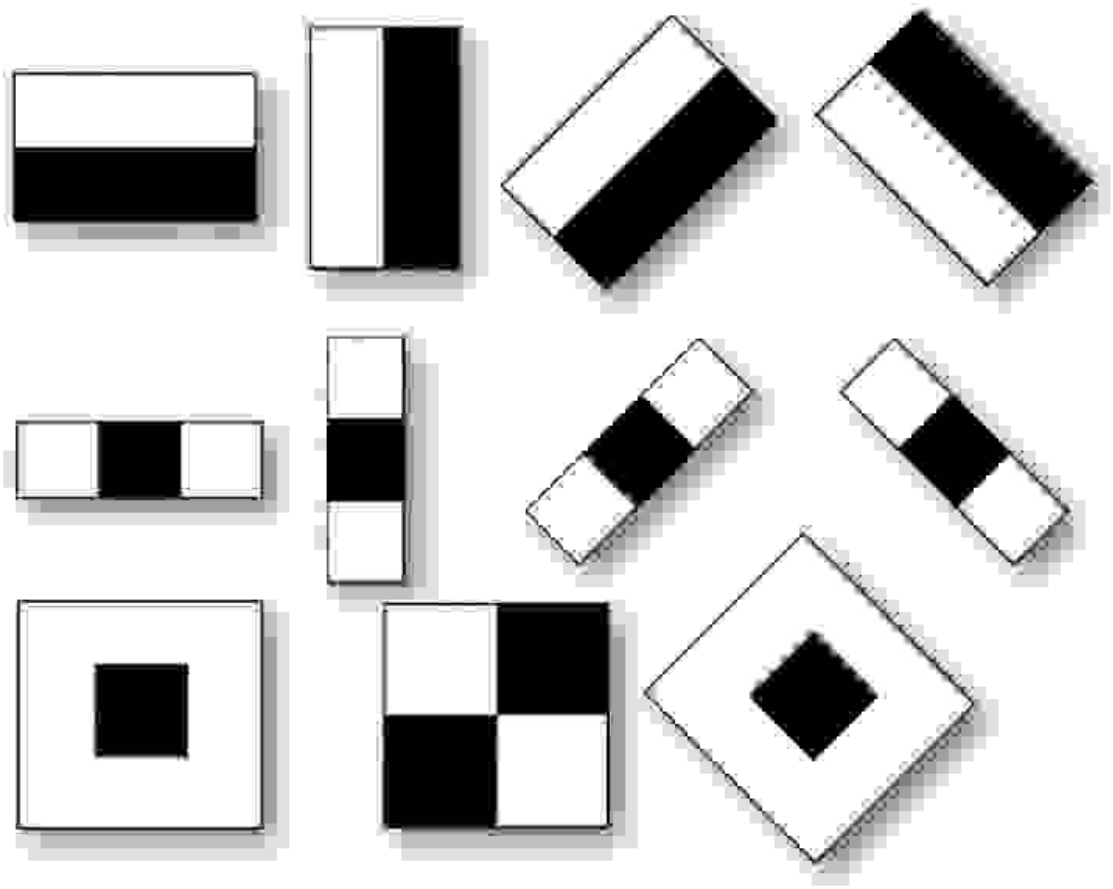

Рисунок 2.2 – Примітиви Хаара

У розширеному методі Віоли – Джонса, що використовується в бібліотеці OpenCV, використовуються додаткові признаки: визначеним значенням такого визнання буде.

$$F = X\text{-}Y, \tag{2.11}$$

де X – сума значень яскравості точок закритих світлою частиною ознаки , а Y – сума значень яскравості точок закритих темної частини ознаки. Для їх вирахування використовується інтегральне поняття зображення, розглянуте вище.

Признаки Хаара дають точне значення перепаду яскравості по осі X і Y відповідно. Візуалізація сканованого вікна в програмі приведена на рисунку 2.3.

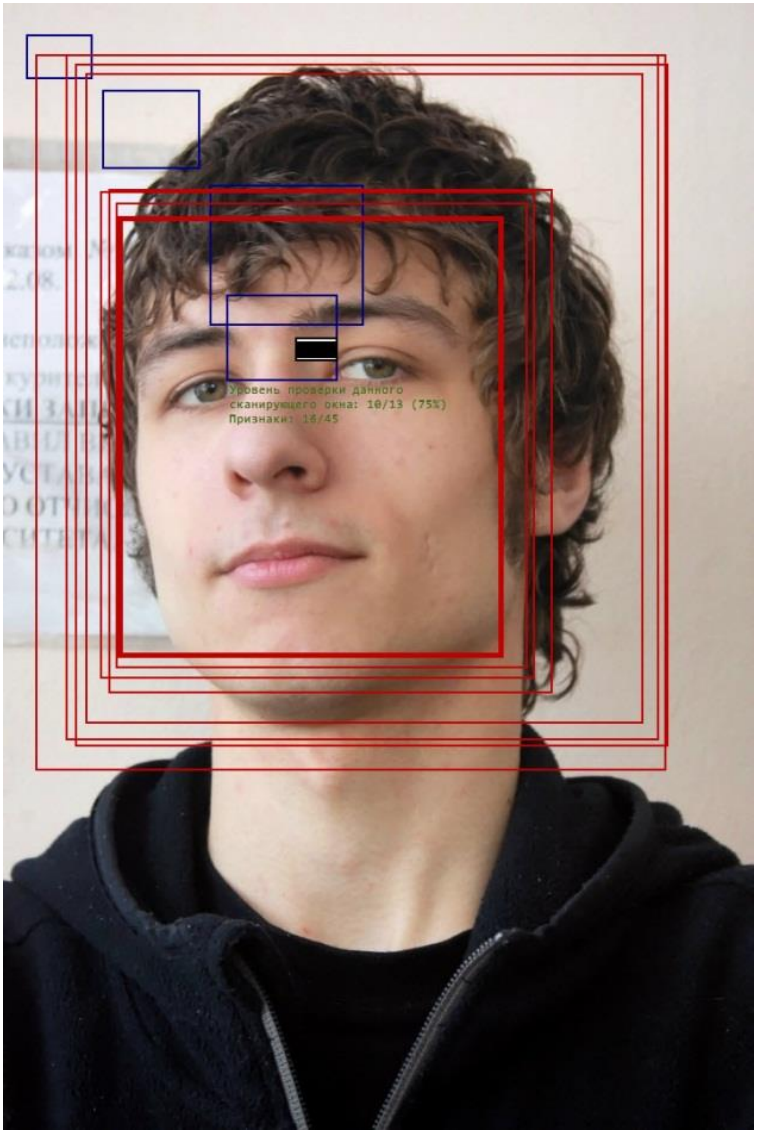

Рисунок 2.3 – Ковзаючі вікна

Алгоритм сканування вікна з позначками має наступні кроки:

- є досліджуване зображення, вибране вікно сканування, вибрані використовувані позначки;

- далі вікно сканування починає послідовно рухатися по зображенню з кроком в 1 комірку вікна (допустим, розмір самого вікна є 24*24 ячейки);

- при скануванні зображення в кожному вікні вираховується приблизно 200 000 варіантів розташування позначок, за рахунок масштабу вікно та їх положень у вікні сканування;

- сканування проводиться послідовно для різних масштабів;

- масштабується не саме зображення, а скановане вікно (змінюється розмір ячейки);

- всі знайдені визнання потрапляють до класифікатора, який «виносить вердикт».

2.4 Алгоритм гістограми локальних бінарних шаблонів

Гістограми локальних бінарних шаблонів (LBPH) це алгоритм для розпізнавання осіб, який представляє локальні об'єкти, такі як краї на обличчях, і тому не так схильний до помилок, викликаних різними умовами висвітлення. Попередні два алгоритми, Eigenfaces та Fisherfaces розглядають дані як вектор у просторі зображень високої розмірності ця багатовимірність призвела до того, що довелося шукати підпростору низької розмірності, щоб уникнути роботи з переважними структурами. LBPH призначений для розпізнавання людей у різних умовах освітлення і на задньому фоні, і на відміну від Fisherfaces йому не потрібно багато зображень однієї й тієї ж людини з усіма можливими умовами, які можуть виникнути. Конкретні риси людини у цьому алгоритмі представлені за допомогою простого вектора даних.

Для навчання алгоритму нам знадобиться набір даних з кількома зображеннями осіб людини, яких ми хочемо розпізнати, можливо, з різним розташуванням та виразом. Обсяг зображень для кожного людини істотно впливає точність [5]. Алгоритм перетворює зображення у відтінки сірого, де кожне зображення представлене значенням інтенсивності в діапазоні від 0 до 255. Потім для кожного вікна розміром 3x3 пікселя він порівнює центральне значення з іншими 8 значеннями, якщо значення краю нижче, то встановлює 0, інакше 1. Потім видаляється центральне значення. Вікно тепер містить лише двійкові значення 0 і 1, тому можна об'єднати ці 8 значень в одне нове значення, яке можна розглядати як двійкове число, і це число міститься як значення для пікселів у центрі. Весь процес показано рисунку 2.4.

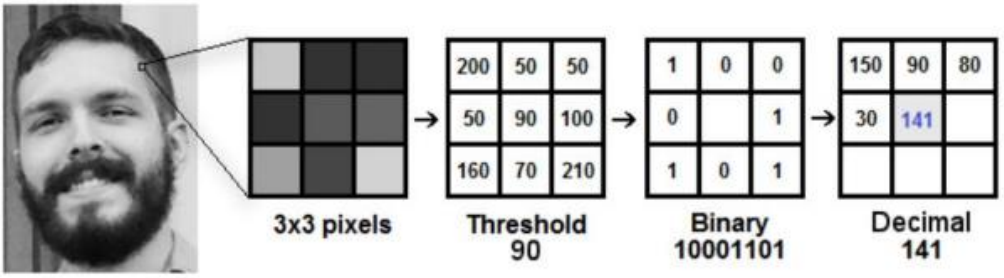

Рисунок 2.4 - Принцип роботи алгоритму LBPH із зображенням

Принцип алгоритму LBPH із зображення береться блок 3x3 пікселів у відтінках сірого, показаний у діапазоні 0-255, якщо значення вище, ніж центральний піксель, результат на пікселі дорівнює 1, інакше 0. Наприкінці двійкове значення об'єднується та поміщається як у центральний піксель.

На цьому етапі кадр виглядає як зображення в градаціях сірого, якому виділяються швидкі зміни інтенсивності, такі як краї або кути. На наступному етапі зображення з останнього кроку ділиться на кілька сіток, кількість сіток залежить від необхідної точності, швидкості обчислень чи вимог до пам'яті. Для кожної сітки ми створюємо гістограму (значення від 0 до 255, оскільки ми працюємо в градаціях сірого), що показує, як Часто це значення / інтенсивність пікселів зустрічається в сітці. Об'єднуючи кожну гістограму, ми отримуємо нову велику гістограму з 256 * $GridnX$ * $GridY$ позиціями, яка представляє конкретну особливість людини. Гістограми з кількох навчальних зображень тих самих людей зберігаються та порівнюються окремо для досягнення вищої точності. Після того, як ми навчили алгоритм із зображеннями людей, яких ми хочемо розпізнати, він може перейти до фактичного розпізнавання. Той самий процес, що описаний у попередніх параграфах, виконується для отримання гістограми. Порівняння цієї гістограми з гістограмами з навчання виконується за сумою евклідових відстаней або різниці в абсолютних значення між кожною окремою позицією між двома гістограмами. У Зокрема, в OpenCV це робиться за допомогою евклідових відстаней. Метод вибирає найменшу різницю з усіх гістограм і повертає особистість людини з навчального набору, якому належала гістограма з найменшою відмінністю та рівнем точності, як числова різниця між двома гістограмами.

У OpenCV клас, специфічний для розпізнавання облич LBPH, називається LBPHFaceRecognizer. Цей клас надає всі необхідні функції для навчання та запуску розпізнавання осіб та налаштування розпізнавання LBPH. В цьому класі можна встановити максимальну різницю між діаграмами, на яких розпізнавання повертає позитивне значення, значення $GridnX$ та $GridY$ або радіус, тому вікно з виявленням країв змінюється з 3x3 пікселів на більше вікна.

Детально розглянуто алгоритми виявлення та розпізнавання обличчя людини у бібліотеці OpenCV. У числі розглянутих алгоритмів такі, як: виявлення об'єктів за допомогою класифікатора Віоли-Джонса, розпізнавання осіб із використанням алгоритмів Eigenfaces, Fisherfaces та LBPH.

3 РЕАЛІЗАЦІЯ СИСТЕМИ АУТЕНТИФІКАЦІЇ

3.1 Алгоритм аутентифікації на основі розпізнавання осіб

При проектуванні системи аутентифікації з використанням біометричних дани було класифіковано системи ідентифікації та аутентифіції з біометричним розпізнаванням осіб. Розроблена схема приведена на рисунку 3.1.

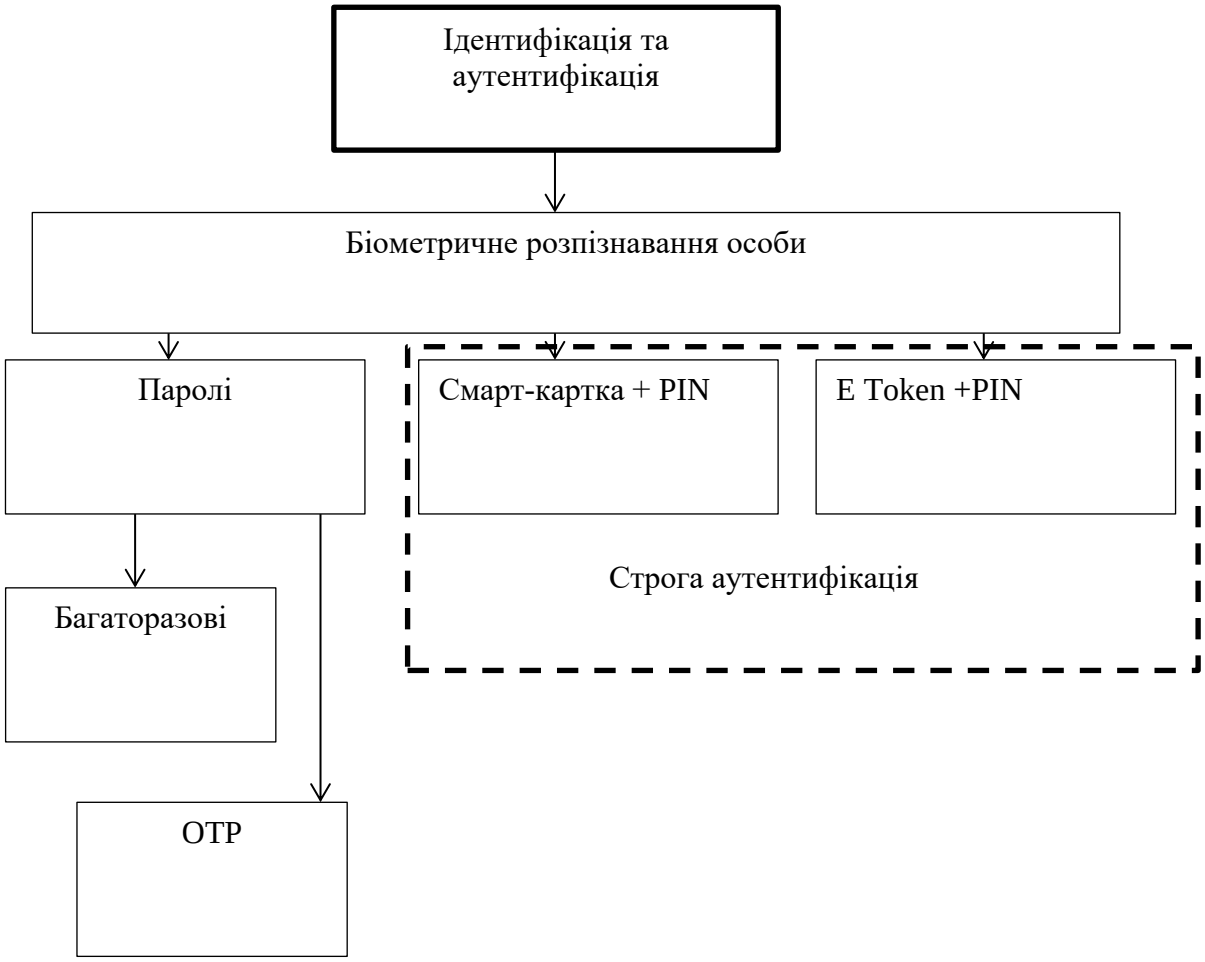

Рисунок 3.1 - Схема аутентифікації з біометричними даними

Проектована система аутентифікації базується на пошуку відповідних біометричних відбитків в базі даних та співставлення їх з ідентифікатором користувача(логіном), як першим етапом аутентифікації. На рисунку 3.2 приведений алгоритм проведення біометричної аутентифікації розробленою системою.

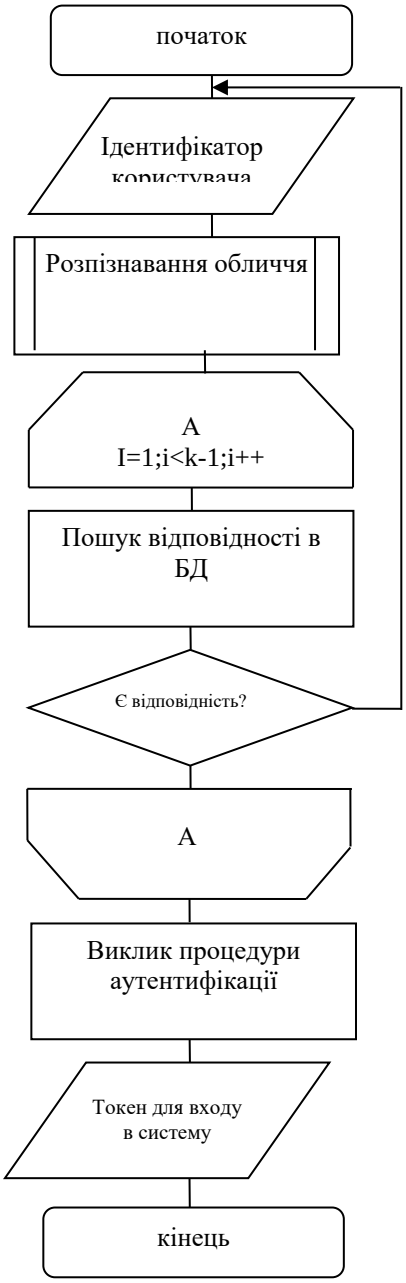

Рисунок 3.2 - Алгоритм аутентифікації з використанням біометричних даних

Процес аутентифікації користувача в системі здійснюється згідно наступної послідовності дій:

1) На вхід підсистеми збору даних надходить біометричний образ користувача(опрацьовується зображення), де він піддається оцифровці.

2) Оцифрований біометричний образ передається на вхід підсистеми обробки сигналів, де він піддається операції нормалізації та виділення біометричні параметру вектор(контрольні точки рис обличчя).

3) Вектор біометричних параметрів надходить до модуля аутентифікації, який виконує перевірку введеного ідентифіатора та ідентифікатора що в БД відповідає біометричним даним.

4) Введення пароля.

На рисунку 3.3 приведена спрощена схема біометричної автентифікації, що дозволяє відобразити зв’язки компонентів системи.

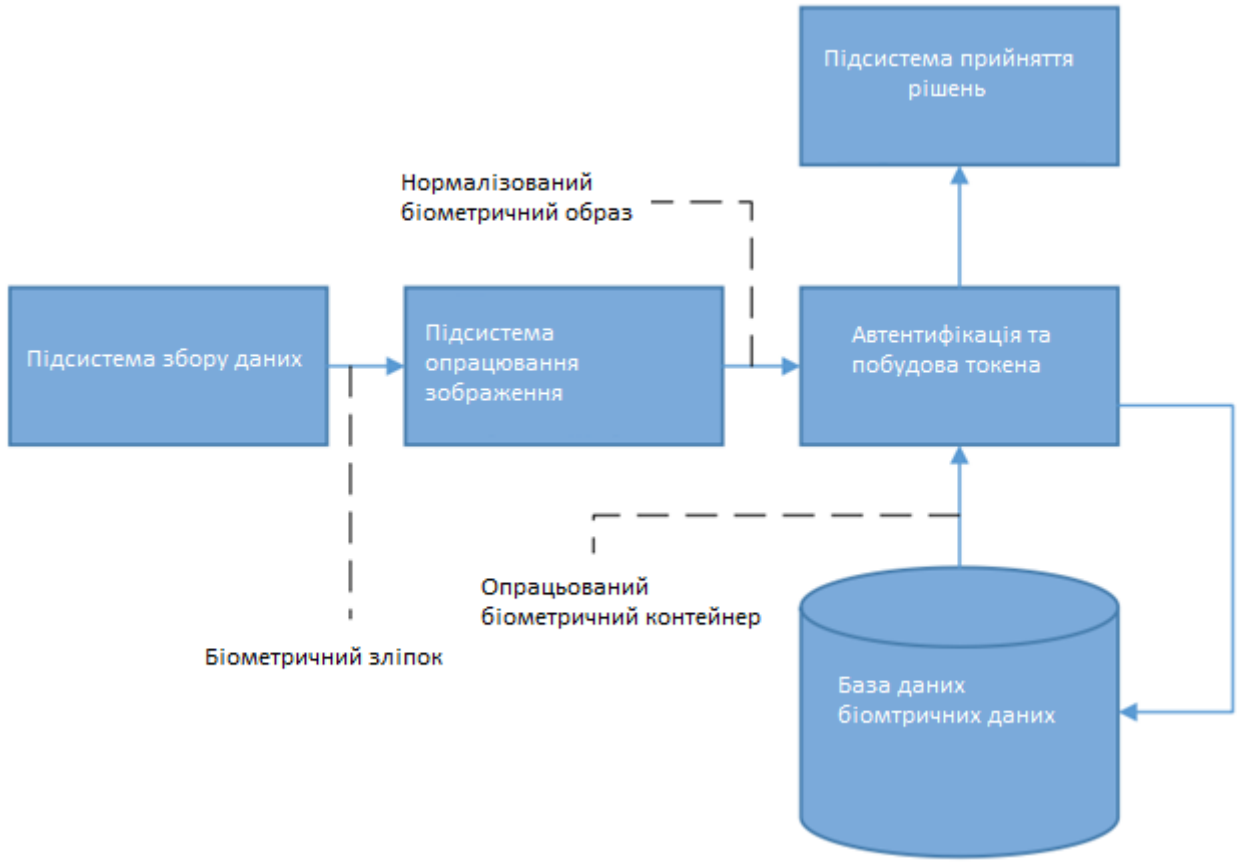

Рисунок 3.3 - Спрощена схема біометричної автентифікації

Основна ідея запропонованого рішення базується на отриманні ідентифікатора, який згодом перевіряється з ідентифікатором введеним користувачем. Для отримання ідентифікатора у процесі реєстрації, образ, що надходить на вхід підсистеми аутентифікації, також подається модуль розпізнавання обличчя.

На рисунку 3.4 приведена розробблена діаграма послідовності для процесу реєстрації користувача.

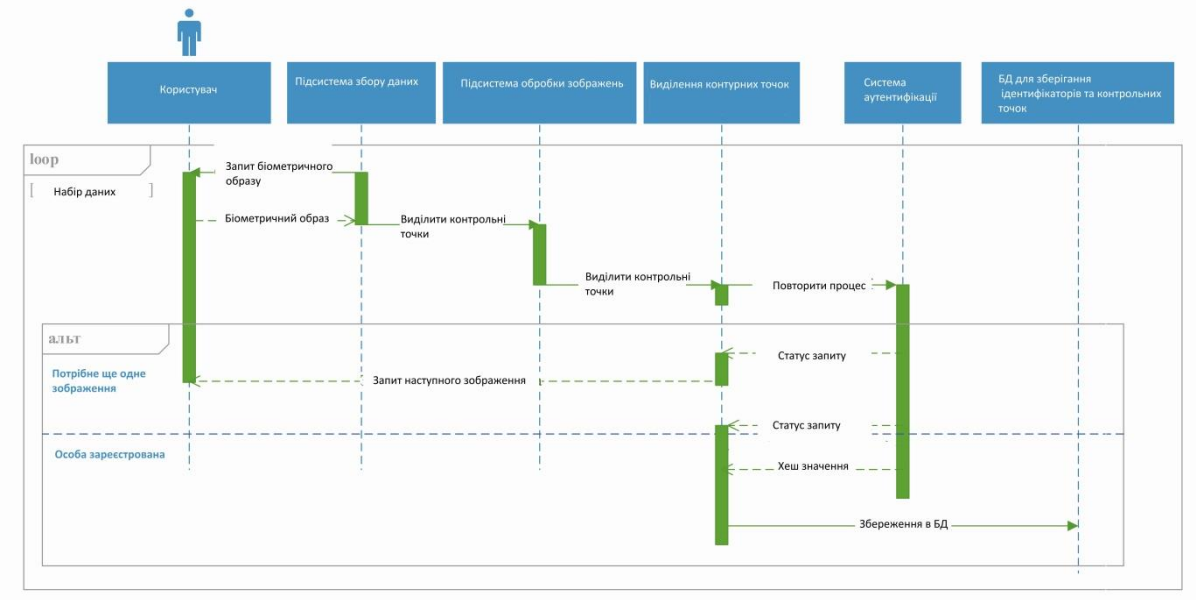

Рисунок 3.4 -Процес реєстрації користувачів

Діаграма послідовності для процесу біометричної аутентифікації користувача приведена на рисунку 3.5.

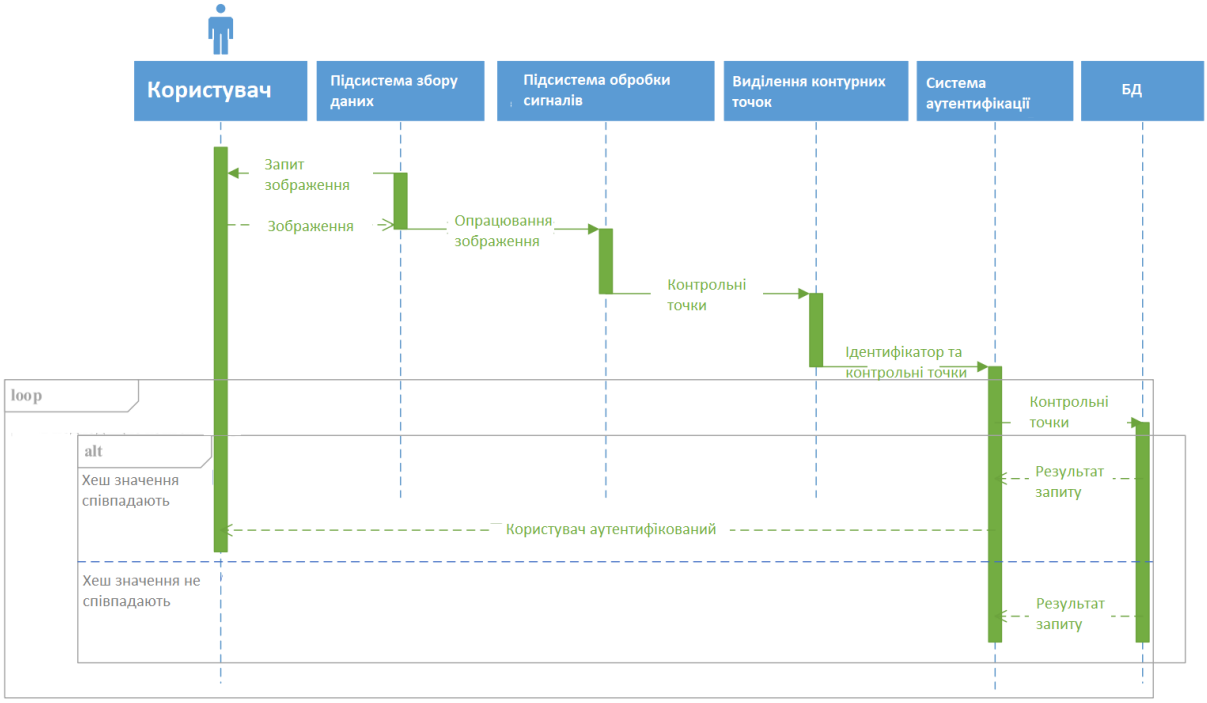

Рисунок 3.5 – Процес аутентифікації користувача

3.2 Можливості бібліотеки OpenCV

OpenCV - це open source бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. Широко використовується у таких мовах як C, C++, Python та Java.

Розглянемо принципи роботи з пік селями та колірними просторами. Перед тим як перейти до практики, нам потрібно трохи розібратися з теорією. Кожне зображення складається із набору пікселів. Піксель – це будівельний блок зображення. Якщо уявити зображення як сітки, кожен квадрат у сітці містить один піксель, де точці з координатою (0, 0) відповідає верхній лівий кут зображення. Наприклад, уявимо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків та 300 стовпців. У сукупності у зображенні є 400*300 = 120000 пікселів(рисунок 3.6).

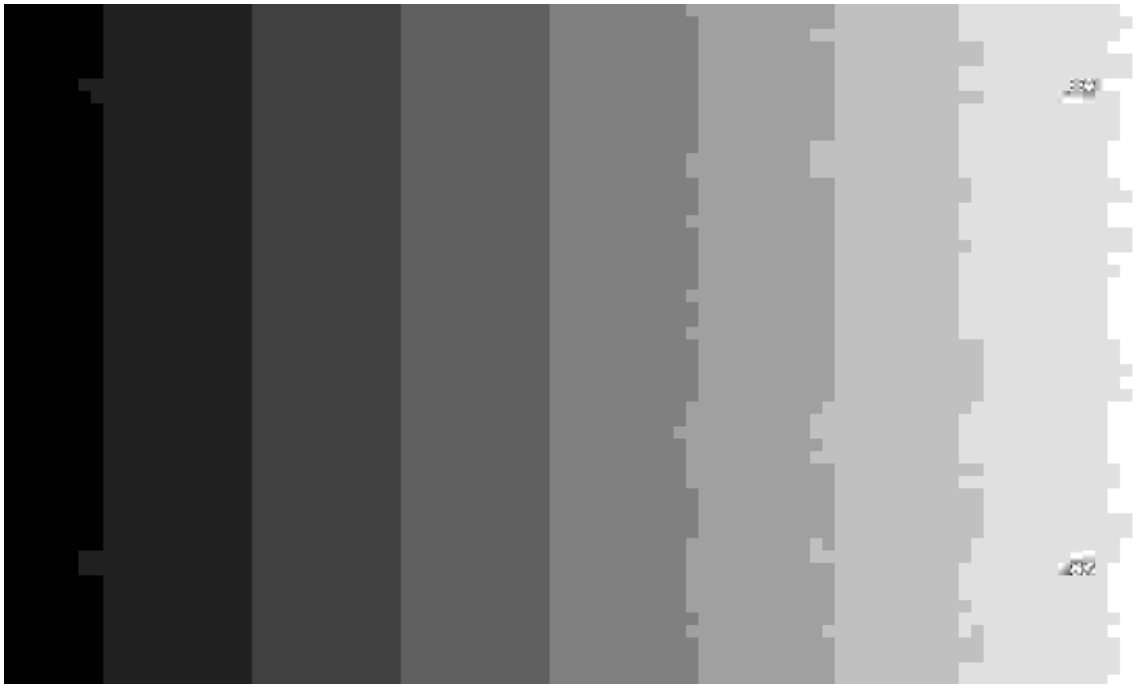

Рисунок 3.6 – Відтінки сірого

У більшості зображень пікселі представлені двома способами: у відтінках сірого та у колірному просторі RGB. У зображеннях у відтінках сірого кожен

піксель має значення між 0 і 255 де 0 відповідає чорному, а 255 відповідає білому. А значення між 0 і 255 приймають різні відтінки сірого, де значення ближче до 0 темніші, а значення ближче до 255 світліші:

Кольорові пікселі зазвичай представлені в колірному просторі RGB(red, green, blue — червоний, зелений, синій), де одне значення для червоної компоненти, одне для зеленої та синій. Кожна з трьох компонентів представлена цілим числом в діапазоні від 0 до 255 включно, яке вказує як «багато» кольору міститься. Виходячи з того, що кожна компонента представлена в діапазоні [0,255], для того, щоб уявити насиченість кожного кольору, нам буде достатньо 8-бітного цілого беззнакового числа. Потім ми об'єднуємо значення всіх трьох компонентів у кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонентів повинна дорівнювати 255: (255, 255, 255). Тоді, щоб отримати чорний колір, кожна з компонентів повинна бути рівною 0: (0, 0, 0). Нижче наведені поширені кольори, представлені у вигляді кортежів RGB (рисунок 3.7).

Рисунок 3.7 – RGB кортежі

3.3 Встановлення OpenCV

Розглянемо процес імпорту бібліотеки OpenCV. Тепер перейдемо до практичної частини. Перше, що нам необхідно зробити, — це імпортувати бібліотеку. Є кілька шляхів імпорту, найпоширеніший — це вираз:

```
import cv2
```

Також можна зустріти наступну конструкцію для імпорту цієї бібліотеки:

```
from cv2 import cv2
```

Завантаження, відображення та збереження зображення:

```
def loading_displaying_saving():
    img = cv2.imread('girl.jpg', cv2.IMREAD_GRAYSCALE)
    cv2.imshow('girl', img)
    cv2.waitKey(0)
    cv2.imwrite('graygirl.jpg', img)
```

Для завантаження зображення використовуємо функцію cv2.imread(), де першим аргументом вказується шлях до зображення, а другим аргументом, який є необов'язковим, вказуємо, у якому кольоровому просторі хочемо вважати наше зображення. Щоб рахувати зображення в RGB – cv2.IMREAD_COLOR, у відтінках сірого – cv2.IMREAD_GRAYSCALE. За умовчанням цей аргумент набуває значення cv2.IMREAD_COLOR. Ця функція повертає 2D (для зображення у відтінках сірого) або 3D (для кольорового зображення) масив NumPy. Форма масиву для кольорового зображення: висота x ширина x 3, де 3 це байти, по одному байти на кожну з компонентів. У відтінках сірого все трохи простіше: висота помножити на ширину.

За допомогою функції cv2.imshow()відображаємо зображення на екрані. Як перший аргумент передаємо функції назву нашого вікна, а другим

аргументом зображення, яке ми завантажили з диска, проте, якщо далі не вкажемо функцію cv2.waitKey(), то зображення миттєво закриється. Ця функція зупиняє виконання програми до натискання кнопки, яку потрібно передати першим аргументом. Для того, щоб будь-яка клавіша була зарахована, передається 0. Зліва представлено зображення у відтінках сірого, а праворуч у форматі RGB (рисунок 3.8).

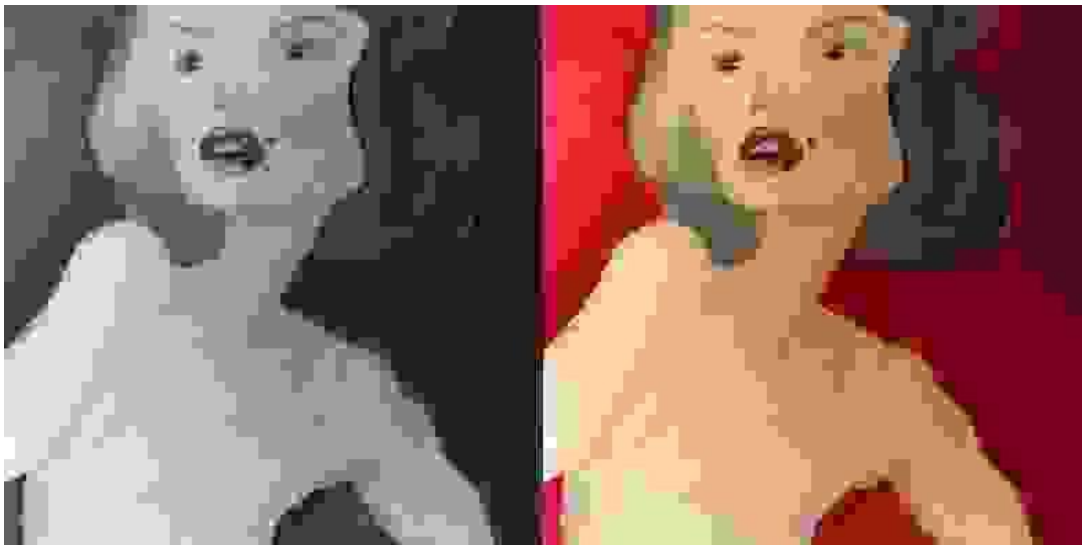

Рисунок 3.8 – Зображення в відтінках сірого та форматі RGB

І, нарешті, за допомогою функції cv2.imwrite() записуємо зображення у файл у форматі jpg (дана бібліотека підтримує всі популярні формати зображень: png, tiff, jpeg, bmp і т. д., тому можна було зберегти наше зображення у будь-якому з цих форматів), де першим аргументом передається безпосередньо сама назва та розширення, а наступним параметром зображення, яке ми хочемо зберегти.

Доступ до пікселів та маніпулювання ними здійснюється спеціальними командами. Для того, щоб дізнатися висоту, ширину та кількість каналів зображення можна використовувати атрибут shape:

```
print("Висота:"+str(img.shape[0]))
print("Ширина:" + str(img.shape[1]))
print("Кількість каналів:" + str(img.shape[2]))
```

Важливо пам'ятати, що зображення відтінків сірого img.shape[2] недоступні, оскільки дані зображення представлені у вигляді 2D масиву.

Щоб отримати доступ до значення пікселя, нам просто потрібно вказати координати *x* та *y* пікселя, який нас цікавить. Також важливо пам'ятати, що бібліотека OpenCV зберігає канали формату RGB у зворотному порядку, у той час як ми думаємо в термінах червоного, зеленого та синього, то OpenCV зберігає їх у порядку синього, зеленого та червоного кольорів:

```
(b, g, r) = img[0, 0]
print("Красный: {}, Зелёный: {}, Синий: {}".format(r, g, b))
```

Спочатку ми беремо піксель, розташований у точці (0,0). Цей піксель, та й будь-який інший піксель, представлені у вигляді кортежу. Зауважте, що назва змінних розташована в порядку b, g і r. У наступному рядку виводимо значення кожного каналу на екран. Як можна побачити, доступ до значень пікселів досить простий, також можна і маніпулювати значеннями пікселів:

```
img[0, 0] = (255, 0, 0)
(b, g, r) = img[0, 0]
 print("Червоний: {}, Зелений: {}, Синій: {}".format(r, g, b))
```

У першому рядку ми встановлюємо значення пікселя (0, 0) рівним (255, 0, 0), потім знову беремо значення даного пікселя і виводимо його на екран, в результаті мені на консоль вивелося наступне:

```
Червоний: 251, Зелений: 43, Синій: 65
Червоний: 0, Зелений: 0, Синій: 255
```

3.4 Розпізнавання зображення

Детекція (тобто перебування) і розпізнавання обличя — дуже популярне завдання в сучасному світі: у метро вже давно стежать за нашими пересуваннями, а в заміських будинках все частіше ставлять «розумні вічка».

Для роботи нам знадобиться:

1. Google Colab – середовище для роботи з Python у браузері. Вони ще надають доступ до GPU дають (а для нейронних мереж це дуже потрібно).
2. Програмні модулі проекту.

Тож розпочати можна з підключення модулів. Перед тим, як розпочати роботу, давайте змінимо середовище виконання на «Прискорення GPU». У майбутньому нам це знадобиться для роботи нейронної мережі. Для цього Google Colab натискаємо «Середовище виконання» → «Змінити середовище виконання» → «Апаратний прискорювач GPU».

Тепер підключимо необхідні бібліотеки. У Colab більшість бібліотек вже встановлено, тому нам залишилося їх лише імпортувати:

- cv2- для завантаження та обробки зображень;
- dlib- для детекції осіб;
- numpy- для роботи з матрицями.

Крім цього, за допомогою команди pip install face_recognitionми встановлюємо бібліотеку face_recognition— з її допомогою ми оброблятимемо обличчя нейронною мережею.

Загалом зробити повну обробку можна всього за кілька рядків коду, тому що в бібліотеці face_recognitionбагато процесів (наприклад, детекція осіб) вже загорнуті у функції.

Наступним етапом буде первинне налаштування бібліотеки. Тепер нам потрібно зробити невелике налаштування, щоб потім повноцінно використовувати всі інструменти та кодову базу.

Почнемо з налаштування детектора облич. Спочатку скачуємо файл із готовою моделлю за допомогою команди wget та розпаковуємо файл у формат .dat.

```
wget http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2
bunzip2 shape_predictor_68_face_landmarks.dat.bz2
```

Далі створюємо детектор облич і передаємо завантажену модель в shape_predictor, який передбачатиме ключові точки людського обличчя – контур голови, очі, ніс та рот.

```
face_detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

Відмінно, тепер готові переходити до змістовної частини обробки фотографій.

Наступним етапом буде завантаження та обробка фотографій. Перед тим, як завантажувати фотографію в змінну Python, її потрібно завантажити у файлове сховище Google Colab. Для цього достатньо натиснути на іконку Папки на лівому сайдбарі та вибрати Upload (рисунок 3.9).

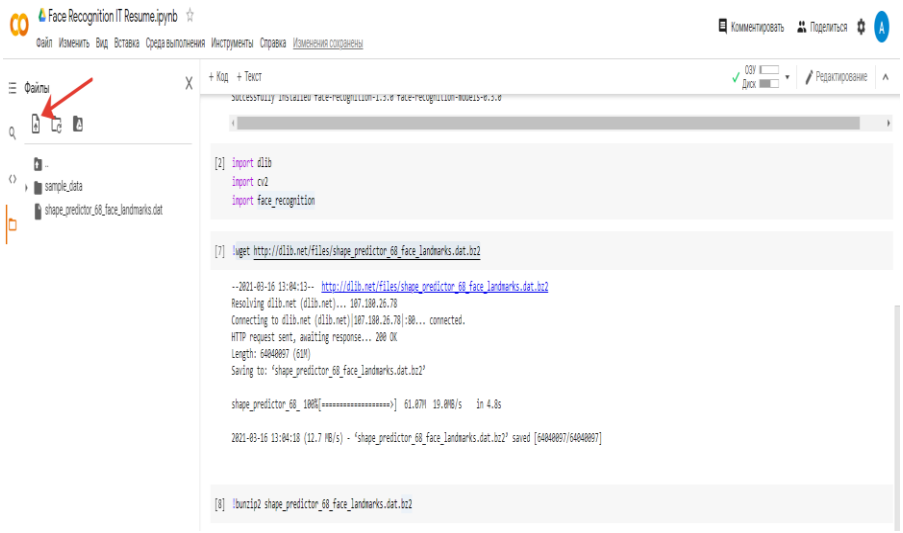

Рисунок 3.9 –Вікно завантаження зображення

Для прикладу, завантажимо фотографію та проведемо маніпуляції з розпізнавання образів доступні у бібліотеці (рисунок 3.10).

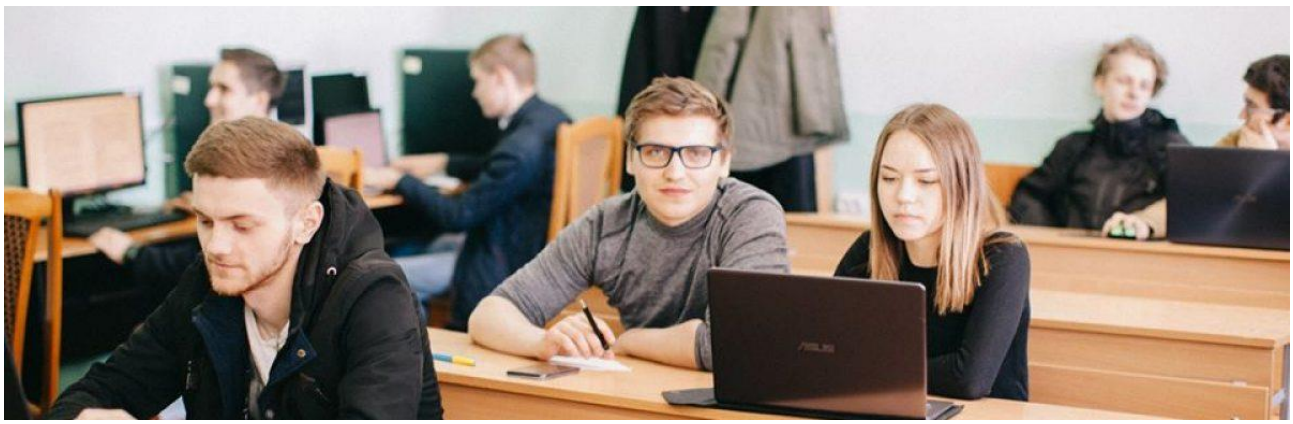

Рисунок 3.10 – Тестова фотографія

Відмінно тепер треба «прочитати» зображення в змінну Python. Це можна зробити кількома способами, але зручніше представити зображення одразу у вигляді матриці. Це можна зробити за допомогою функції imreadз модуля cv2.

Якщо завантажувати кольорове зображення, то «матриця» матиме розмірність 3*height*width, де 3 — кількість колірних каналів (RGB), a width і height— розмірність зображення в пікселях. Перевірити це можна за допомогою методу shape. Кожне число в матриці значення пікселя по конкретному колірному каналу.

```
img = cv2.imread('conor.jpg')
img.shape #  перевіряємо розмірність матриці
#(434, 770, 3)
```

Щоб подивитися на завантажену картинку, потрібно:

1. Імпортувати команду cv2_imshow: from google.colab.patches import cv2_imshow.

2. Передати на функцію cv2_imshow необхідну картинку: cv2_imshow(img).

Деякі завдання мають сенс обробляти не кольорове зображення, а чорно-біле. Наприклад, на детекцію облич наявність кольору мало впливає — контури обличчя і так добре видно. А ось для нейронної мережі це може бути важливий момент - мало, людей виглядають однаково, а колір шкіри різний.

Тому давайте переведемо нашу кольорову картинку в чорно-білу:

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Тепер переходимо до наступного етапу – детекція особи. Наступним етапом буде розпізнавання для систем авторизації користувачів. Детектор осіб у нас вже налаштований, тож можемо переходити безпосередньо до детекції. Робити це будемо у 2 етапи:

1. Шукаємо прямокутники, у яких знаходяться особи.
2. По цих прямокутниках передбачаємо контури та ключові точки обличчя.

За допомогою detector отримаємо координати потрібного прямокутника. Якщо осіб кілька, то наборів точок буде кілька. Щоб вибрати конкретну особу, потрібно вказати індекс (у нашому випадку нуль).

```
face_rect = detector(img, 1)[0]
```

Щоб подивитися на результат, можна намалювати рамку навколо обличчя за допомогою методу rectangleта переглянути фото за допомогою cv2_imshow. Рамка залишиться на фото назавжди, тому краще потім завантажити зображення заново (рисунок 3.11).

Рисунок 3.11- Результат face_rect = detector(img, 1)[0]

Тепер за знайденими квадратами потрібно "передбачити" ключові точки обличчя. Для цього використовуємо predictor, а потім витягуємо координати точок.

```
points = predictor(img, face_rect)
landmarks = np.array([*map(lambda p: [p.x, p.y], points.parts())])
```

Тут, щоб отримати координати точок, ми використали відразу кілька корисних операцій Python:

- лямбда-функції;
- функцію map;
- розпакування.

На виході у нас вийшло 68 пікселів. Тепер ми маємо всю необхідну інформацію, щоб обробити зображення нейронною мережею — і зображення, і ключові точки обличчя. Переходимо до наступного етапу.

Наступним етапом буде. Обробка нейронною мережею. Основна ідея обробки мережею полягає в тому, що з вихідного «великого» зображення виділяються лише ключові ознаки, які згодом і допоможуть нам відрізнити одну людину від іншої. Іншими словами, багатовимірна матриця-картинки перетворюється на відносно невеликий числовий вектор, який характеризує обличчя з картинки (рисунок 3.12).

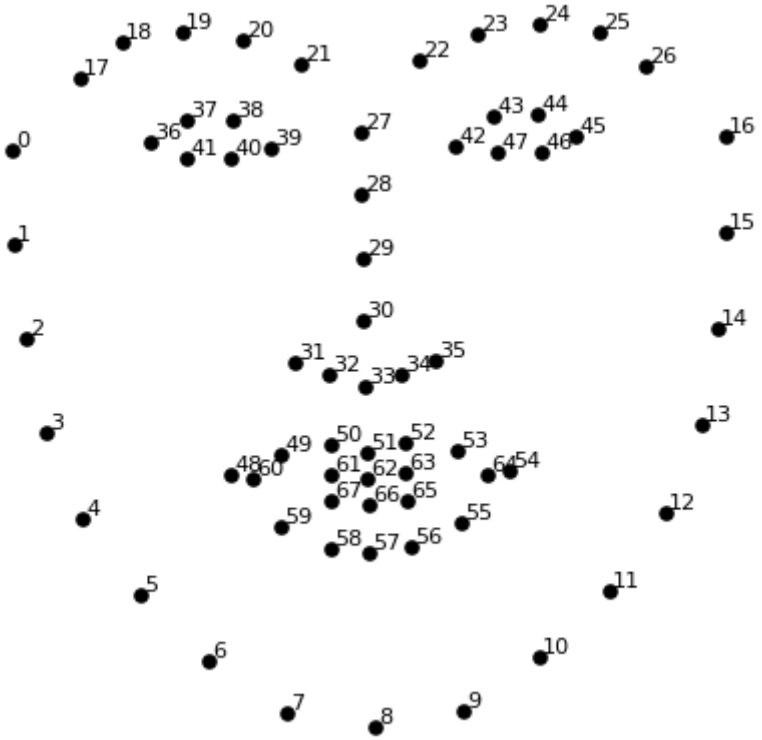

Рисунок 3.12 – Результат опрацювання нейронною мережею

Отже, отримати підсумковий вектор дуже просто - достатньо викликати функцію face_encodings.

vector = face_recognition.face_encodings(img)[0]

Зверніть увагу, що як аргументи передали лише зображення, а отримані раніше ключові точки не використовували. Причина дуже проста — функція face_encodingsробить ті самі дії, які проробили вручну раніше для розуміння процесу «зсередини».

На виході отримуємо вектор розмірності 128, тобто. 128 чисел, які описують особу на фотографії. Що означає кожне з цих чисел — невідоме: це певна абстракція, відновити і осмислити яку неможливо — надто багато математичних перетворень лежить усередині цього процесу.

Однак, від цього вектора нам мало толку — щоб оцінити ефективність «нейромережевих технологій», потрібно порівняти два вектори для двох різних фотографій. Давайте трохи поекспереметнуємо - подаватимемо різні фото на вхід і порівнюватимемо з початковим фото.

Наступний етап порівняння векторів. Щоб порівнювати особи між собою, будемо діяти так:

1. Завантажуємо нове фото та робимо обробку, яку вже розібрали на попередніх етапах. На виході отримуємо вектор.
2. Порівнюємо вектор між собою.
3. На підставі відстані приймаємо рішення — одна й та сама людина на фото або різні.

Постає питання — що означає «порівняти вектор»? Найпростіший спосіб - порахувати відстань у деякій метриці. Ми виберемо звичайну евклідову метрику. Обчислювати її без зайвого коду нам дозволить pdistфункція scipy.spatial.distance.

Отже, спочатку завантажимо ще одну фотографію. Зберігаємо отриманий вектор для першої фотографії в vector1, для другої - vector2і обчислюємо відстань:

```
from scipy.spatial.distance import pdist
pdist([vector1, vector2], 'euclidean')
```

Отримуємо відповідь - 0,48. Це досить близька відстань, з чого робимо висновок, що це та сама людина. Мережа впоралася чудово.

Проведемо, ще кілька експериментів — давайте візьмемо знову Конора, але ускладнимо завдання — нехай він буде в окулярах (рисунок 3.13).

Рисунок 3.13- Тестування зображення

Результат - 0,56. Знову позитивний результат (розробник бібліотеки face_recognition рекомендує дотримуватись порогу 0,6 – якщо більше 0,6, то це інша людина).

Знову проводимо вже відому процедуру проте з невідомим зображенням і отримаємо відповідь… — 0.72. За затвердженим раніше поріг, робимо висновок.

Розберемо код, який можна завантажити з face_detect.py, зображення abba.png і haarcascade_frontalface_default.xml .

```
# Отримати надані користувачем значення
imagePath = sys.argv[1]
cascPath = sys.argv[2]
```

Спочатку передаємое зображення та імена каскаду як аргументи командного рядка. Будемо використовувати зображення ABBA, а також стандартний каскад для виявлення облич, наданий OpenCV.

```
# Створюємо каскад Хаара
faceCascade = cv2.CascadeClassifier(cascPath)
```

Створюємо каскад і ініціалізуємо його каскадом обличчя. Це завантажує каскад облич у пам’ять, щоб він був готовий до використання. Пам’ятайте, що каскад – це лише файл XML, який містить дані для виявлення облич.

```
# Читаємо малюнок
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Читаємо зображення та перетворюємо його на відтінки сірого. Багато операцій в OpenCV виконуються в градаціях сірого.

```
# Виявляти обличчя на зображенні
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

Ця функція визначає фактичне обличчя та є ключовою частиною коду, тому давайте розглянемо параметри:

1. Ця detectMultiScale функція є загальною функцією, яка виявляє об’єкти. Оскільки ми називаємо це каскадом на обличчі, це те, що він виявляє.
2. Перший варіант – це зображення в градаціях сірого.
3. Другий - це scaleFactor. Оскільки деякі обличчя можуть бути ближче до камери, вони виглядатимуть більшими, ніж обличчя позаду. Коефіцієнт масштабу це компенсує.
4. Алгоритм виявлення використовує рухоме вікно для виявлення об’єктів. minNeighborsвизначає, скільки об’єктів буде виявлено поблизу

поточного, перш ніж буде оголошено знайдене обличчя. minSize, тим часом, дає розмір кожного вікна.

Були обрані загальновживані значення для цих полів. При реалізації проекту проводились експерименти із різними значеннями для розміру вікна, коефіцієнта масштабування тощо, доки не знайдено найбільш ефективні параметри.

Функція повертає список прямокутників, у яких, на її думку, знайдено обличчя. Далі ми переглянемо там, де він думає, що щось знайшов.

```
print "Found {0} faces!".format(len(faces))
# Draw a rectangle around the faces
for (x, y, w, h) in faces:
   cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Ця функція повертає 4 значення: x_i урозташування прямокутника, а також ширину та висоту прямокутника (w, h).

Використаємо ці значення, щоб намалювати прямокутник за допомогою вбудованої rectangle()функції.

```
cv2.imshow("Faces found", image)
cv2.waitKey(0)
```

ВИСНОВКИ

Проведено дослідження процесів аутентифікації та авторизації, для того щоб виділити основні функціональні відмінності.

Досліджено протоколи аутентифікації із симетричними та асиметричними алгоритмами шифрування і хеш функціями для виділення сильних та слабких сторін використання обраних видів автентифікації.

Виконано формалізацію процесу авторизації на основі біометричних даних для побудови алгоритму біометричної автентифікації.

Досліджено алгоритми розпізнавання осіб: в бібліотеці OpenCV, гістограм локальних бінарних шаблонів та метод розпізнавання Віоли-Джонса. Детально розглянуто алгоритми виявлення та розпізнавання обличчя людини у бібліотеці OpenCV. У числі розглянутих алгоритмів такі, як: виявлення об'єктів за допомогою класифікатора Віоли-Джонса, розпізнавання осіб із використанням алгоритмів Eigenfaces, Fisherfaces та LBPH.

Реалізувано систему авторизації за допомогою систем розпізнавання осіб на мові програмування python та проведено тестування розроблених програмних модулів.

Формалізовано та побудовано обчислювальні вирази побудови моделі розпізнавання осіб для систем автентифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

4. Стефурак Н.А., Продан Т.І., Дослідження створення цифрового підпису засобами C#. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ - 2021), Тернопіль, 2021. 143 -144 с.

5. Продан Т.І., Івасьєв С.В., Сучасні методи біометричної ідентифікації. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2022), Тернопіль, 2022. 62-65 с.

6. Кузик В.М., Продан Т.І., Івасьєв С.В., Слєпцова О.Я. Біометрична система аутентифікації з використанням голосових даних. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та компютерно-інтегровані технології»(КБКІТ-2020), Тернопіль, 2020. – С.56-59.

7. Шкіра Ю.Р. , Гевко Н.І., Гавриків Н.Г., Осадчук О.Я. Алгоритми та засоби автоматичної детекції обличчя в відео потоці. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та компютерно-інтегровані технології»(КБКІТ-2020), Тернопіль, 2020. – С.19.

8. Ahonen T., Hadid A., Pietikäinen M. Face recognition with local binary patterns – Computer vision-eccv 2004.

9. Basarat S. Beginning Node.js 1st ed. Edition //Springer Science. – 50.

10. Belhumeur P. N., Hespanha J. P., Kriegman D. J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection – IEEE Transactions on pattern analysis and machine intelligence. – 1997.

11. Berezsky O. Biomedical image search and retrieval algorithms / O. Berezsky, G. Melnyk, Yu Batko // Комп'ютинг. – 2008. – Т. 7, Вип. 1. – С. 108–113.

12. Bradsky G. Learning OpenCV – O'Reilly / Kaehler A. – O'Reilly Media, 2008 – 580c.

13. Brunelli R., Poggio T. Face recognition through geometrical features – Computer Vision—ECCV'92. – Springer Berlin/Heidelberg, 1992.

14. Hrytsyk V. V. Modeling and synthethis of complex symmetrical images / V. V. Hrytsyk, K. M. Berezska, O. M. Berezsky // International journal of pattern recognition and artificial intelligence. – 2004. – V. 18, № 2. – P. 175–195.

15. Kanade T. Picture processing system by computer complex and recognition of human faces. – 1974.

16. Laurenz Wiskott Face Recognition by Elastic Bunch Graph Matching / Jean-Marc Fellous Norbert Kruger – CRC Press,1999 – 396

17. Lienhart, R. An extended set of Haar-like features for rapid object detection / Maydt, J. ICIP02, 2002 – 250

18. Messer K. et al. Performance characterization of face recognition algorithms and their sensitivity to severe illumination changes – ICB. – 2006.

19. Peter N. Belhumeur Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection / David J. Kriegman, 1997 – 64

20. Raudys S. J. et al. Small sample size effects in statistical pattern recognition: Recommendations for practitioners //IEEE Transactions on pattern analysis and machine intelligence. – 1991.

21. Shan C. Facial expression based on Local Binary Patterns: A comprehensive study.Image and Vision Computing / Gong S., McOwan P.W., 2009 – 10

22. Turati C. et al. Newborns' face recognition: Role of inner and outer facial features //Child development. – 2006. – Т. 77. – №. 2. – С. 297-311.

23. Turk M., Pentland A. Eigenfaces for recognition – Journal of cognitive neuroscience. – 1991.

24. Wiskott L. et al. Face recognition by elastic bunch graph matching //IEEE Transactions on pattern analysis and machine intelligence. – 1997.

25. Zhao W. et al. Face recognition: A literature survey – ACM computing surveys (CSUR). – 2003.

26. Березька К. М. Редактор синтезу та моделювання складних зображень симетричної структури / К. М. Березька, О. М. Березький // Вісник Національного університету «Львівська політехніка». Комп’ютерна інженерія та інформаційні технології. – 2001. – № 433. – С. 134–137.

ДОДАТОК А

Код програмного засобу

```
# importing libraries
import tkinter as tk
from tkinter import Message, Text
import cv2
import os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font
from pathlib import Path

window = tk.Tk()
window.title("Face_Recogniser")
window.configure(background='white')
window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)
message = tk.Label(
    window, text="Face-Recognition-System",
    bg="green", fg="white", width=50,
    height=3, font=('times', 30, 'bold'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="No.",
               width=20, height=2, fg="green",
               bg="white", font=('times', 15, ' bold '))
lbl.place(x=400, y=200)

txt = tk.Entry(window,
               width=20, bg="white",
               fg="green", font=('times', 15, ' bold '))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Name",
                width=20, fg="green", bg="white",
                height=2, font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)
```

```
txt2 = tk.Entry(window, width=20,
                bg="white", fg="green",
                font=('times', 15, ' bold '))
txt2.place(x=700, y=315)

# The function below is used for checking
# whether the text below is number or not ?


def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False
# Take Images is a function used for creating
# the sample of the images which is used for
# training the model. It takes 60 Images of
# every new user.


def TakeImages():

    # Both ID and Name is used for recognising the Image
    Id = (txt.get())
    name = (txt2.get())

    # Checking if the ID is numeric and name is Alphabetical
    if(is_number(Id) and name.isalpha()):
        # Opening the primary camera if you want to access
        # the secondary camera you can mention the number
        # as 1 inside the parenthesis
        cam = cv2.VideoCapture(0)
        # Specifying the path to haarcascade file
        harcascadePath = "data\haarcascade_frontalface_default.xml"
        # Creating the classier based on the haarcascade file.
```

```
detector = cv2.CascadeClassifier(harcascadePath)
# Initializing the sample number(No. of images) as 0
sampleNum = 0
while(True):
    # Reading the video captures by camera frame by frame
    ret, img = cam.read()
    # Converting the image into grayscale as most of
    # the the processing is done in gray scale format
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # It converts the images in different sizes
    # (decreases by 1.3 times) and 5 specifies the
    # number of times scaling happens
    faces = detector.detectMultiScale(gray, 1.3, 5)

    # For creating a rectangle around the image
    for (x, y, w, h) in faces:
        # Specifying the coordinates of the image as well
        # as color and thickness of the rectangle.
        # incrementing sample number for each image
        cv2.rectangle(img, (x, y), (
            x + w, y + h), (255, 0, 0), 2)
        sampleNum = sampleNum + 1
        # saving the captured face in the dataset folder
        # TrainingImage as the image needs to be trained
        # are saved in this folder
        cv2.imwrite(
            "TrainingImage\ "+name + "."+Id + '.' + str(
                sampleNum) + ".jpg", gray[y:y + h, x:x + w])
        # display the frame that has been captured
        # and drawn rectangle around it.
        cv2.imshow('frame', img)
    # wait for 100 milliseconds
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is more than 60
    elif sampleNum > 60:
        break
# releasing the resources
cam.release()
# closing all the windows
cv2.destroyAllWindows()
# Displaying message for the user
res = "Images Saved for ID : " + Id + " Name : " + name
# Creating the entry for the user in a csv file
row = [Id, name]
```

```
        with open('UserDetails\UserDetails.csv', 'a+') as csvFile:
            writer = csv.writer(csvFile)
            # Entry of the row in csv file
            writer.writerow(row)
        csvFile.close()
        message.configure(text=res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text=res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text=res)

# Training the images saved in training image folder


def TrainImages():
    # Local Binary Pattern Histogram is an Face Recognizer
    # algorithm inside OpenCV module used for training the image dataset
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    # Specifying the path for HaarCascade file
    harcascadePath = "data\haarcascade_frontalface_default.xml"
    # creating detector for faces
    detector = cv2.CascadeClassifier(harcascadePath)
    # Saving the detected faces in variables
    faces, Id = getImagesAndLabels("TrainingImage")
    # Saving the trained faces and their respective ID's
    # in a model named as "trainner.yml".
    recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImageLabel\Trainner.yml")
    # Displaying the message
    res = "Image Trained"
    message.configure(text=res)


def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    # creating empty ID list
    Ids = []
    # now looping through all the image paths and loading the
    # Ids and the images saved in the folder
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
```

```
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids
# For testing phase


def TrackImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    # Reading the trained model
    recognizer.read("TrainingImageLabel\Trainner.yml")
    harcascadePath = "data\haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    # getting the name from "userdetails.csv"
    df = pd.read_csv("UserDetails\UserDetails.csv")
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for(x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if(conf < 50):
                aa = df.loc[df['Id'] == Id]['Name'].values
                tt = str(Id)+"-"+aa
            else:
                Id = 'Unknown'
                tt = str(Id)
            if(conf > 75):
                noOfFile = len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown\Image" +
                            str(noOfFile) + ".jpg", im[y:y + h, x:x + w])
            cv2.putText(im, str(tt), (x, y + h),
                        font, 1, (255, 255, 255), 2)
        cv2.imshow('im', im)
        if (cv2.waitKey(1) == ord('q')):
            break
    cam.release()
    cv2.destroyAllWindows()
```

```
takeImg = tk.Button(window, text="Sample",
                    command=TakeImages, fg="white", bg="green",
                    width=20, height=3, activebackground="Red",
                    font=('times', 15, ' bold '))
takeImg.place(x=200, y=500)
trainImg = tk.Button(window, text="Training",
                     command=TrainImages, fg="white", bg="green",
                     width=20, height=3, activebackground="Red",
                     font=('times', 15, ' bold '))
trainImg.place(x=500, y=500)
trackImg = tk.Button(window, text="Testing",
                     command=TrackImages, fg="white", bg="green",
                     width=20, height=3, activebackground="Red",
                     font=('times', 15, ' bold '))
trackImg.place(x=800, y=500)
quitWindow = tk.Button(window, text="Quit",
                       command=window.destroy, fg="white", bg="green",
                       width=20, height=3, activebackground="Red",
                       font=('times', 15, ' bold '))
quitWindow.place(x=1100, y=500)


window.mainloop()
```

ДОДАТОК Б

Світокопія публікацій

ЗАХІДНОУКРАЇНСЬКИЙНАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІВАНО-ФРАНКІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ НАФТИ І ГАЗУ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
НАДВІРНЯНСЬКИЙ КОЛЕДЖ НТУ
ГАЛИЦЬКИЙ КОЛЕДЖ ІМ. В. ЧОРНОВОЛА

Проблемно-наукова міжгалузева конференція

АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

(АКІТ – 2021)

20—22 лютого 2021 року

Тернопіль

УДК 004.56.5(043.2)

Стефурак Н.А.[1], Продан Т.І.[2]
[1]Галицький коледж ім.. В. Чорновола
[2]Західноукраїнський національний університет

ДОСЛІДЖЕННЯ СТВОРЕННЯ ЦИФРОВОГО ПІДПИСУ ЗАСОБАМИ C#

Вступ. Криптографічні цифрові підписи використовують алгоритми з відкритим ключем для забезпечення цілісності даних. Якщо підписуються дані за допомогою цифрового підпису, інша сторона може перевірити підпис і переконатися в тому, що дані надійшли від відправника і не були змінені після підписування [1, 2].

Мета: Дослідити та проаналізувати процес створення і перевірки цифрових підписів за допомогою класів простору імен System.Security.Cryptography .

1. Створення підписів засобами C#

Цифрові підписи зазвичай застосовуються до хеш-значень, які представляють масиви даних великого розміру. У прикладі демонструється застосування цифрового підпису до хеш-значення. Спочатку створюється екземпляр класу RSA з метою формування набору, що складається з відкритого і закритого ключів. Потім екземпляр RSA передається в новий екземпляр класу RSAPKCS1SignatureFormatter. При цьому примірнику RSAPKCS1SignatureFormatter передається закритий ключ, який і створює цифровий підпис. Перед тим як підписати хеш-код, вам необхідно вказати використовуваний хеш-алгоритм. У цьому прикладі використовується алгоритм SHA1. Далі, викликається метод CreateSignature для виконання підпису.

Через проблеми з алгоритмом SHA1 рекомендується використовувати SHA256 або більш високий рівень.

```
using System;
using System.Security.Cryptography;
class Class1
{   static void Main()
  {     //The hash value to sign.
    byte[] hashValue = {59,4,248,102, 77,97,142,201,210,12, 224,93,25,41,100,
197,213,134,130,135};      //The value to hold the signed value.
    byte[] signedHashValue;
    //Generate a public/private key pair.
    RSA rsa = RSA.Create();
    //Create an RSAPKCS1SignatureFormatter object and pass it the
    //RSA instance to transfer the private key.
    RSAPKCS1SignatureFormatter rsaFormatter = new
RSAPKCS1SignatureFormatter(rsa);
    //Set the hash algorithm to SHA1.
    rsaFormatter.SetHashAlgorithm("SHA1");
    //Create a signature for hashValue and assign it to
    //signedHashValue.
    signedHashValue = rsaFormatter.CreateSignature(hashValue);   }}
```

2. Перевірка підписів

Для перевірки того, чи підписані дані певною стороною, необхідні такі відомості:

– відкритий ключ з боку, з котрого підписали дані;

- цифровий підпис;
- підписані дані;
- хеш-алгоритм, який використовувався при створенні підпису.

Для перевірки підпису, створеного за допомогою класу RSAPKCS1SignatureFormatter , використовується клас RSAPKCS1SignatureDeformatter. Класу RSAPKCS1SignatureDeformatter необхідно надати відкритий ключ підписує боку. Для RSA потрібні значення модуля і експоненти, щоб вказати відкритий ключ. Спочатку необхідно створити RSA об'єкт для зберігання відкритого ключа, який буде перевіряти підпис, а потім ініціалізує RSAParameters структуру значеннями модуля і експоненти, які задають відкритий ключ.

В коді показано створення структури RSAParameters. Властивостям присвоюється байтовий масив. ModulusmodulusDataExponentexponentData:

```
RSAParameters rsaKeyInfo;
rsaKeyInfo.Modulus = modulusData;
rsaKeyInfo.Exponent = exponentData;
```

Після створення RSAParameters об'єкта можна форматувати новий екземпляр класу реалізації, застосовуючи значення, RSA зазначені в RSAParameters. RSA примірник, в свою чергу, передається конструктору об'єкта RSAPKCS1SignatureDeformatter для передачі ключа.

Дистанційна сторона підписала об'єкт за допомогою алгоритму SHA1, що створює цифровий підпис. RSAPKCS1SignatureDeformatter.VerifySignature метод перевіряє, чи є цифровий підпис допустимим і використовувався для підписування.

Через проблеми з алгоритмом SHA1 рекомендується використовувати SHA256 або більш високий рівень. Однак якщо для створення підпису використовувався алгоритм SHA1, для перевірки підпису необхідно використовувати SHA1.

```
RSA rsa = RSA.Create();
rsa.ImportParameters(rsaKeyInfo);
RSAPKCS1SignatureDeformatter rsaDeformatter = new
RSAPKCS1SignatureDeformatter(rsa);
rsaDeformatter.SetHashAlgorithm("SHA1");
if(rsaDeformatter.VerifySignature(hashValue, signedHashValue))
{   Console.WriteLine("The signature is valid.");}
else{   Console.WriteLine("The signature is not valid.");}
```

Якщо підпис дійсний, цей фрагмент коду видасть повідомлення « The signature is valid», а в іншому випадку - повідомлення « The signature is not valid».

Висновок. Проведено дослідження та проаналізовано процес створення і перевірки цифрових підписів за допомогою класів простору імен System.Security.Cryptography, що дозволяють зробити висновок про досить легку реалізацію алгоритмів цифрового підпису засобами C#.

Перелік використаних джерел.

1. FIPS PUB 186-3, Digital Signature Standart, National Institute of Standarts and Technology.

2. ДСТУ 4145-2002, «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, заснована на еліптичних кривих. Формування та перевірка».

КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

КБКІТ-2022

науково-практична конференція
молодих вчених
аспірантів та студентів

м.Тернопіль

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ГАЛИЦЬКИЙ ФАХОВИЙ КОЛЕДЖ ІМ. В. ЧОРНОВОЛА

КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ (КБКІТ – 2022)

науково-практична конференція
молодих вчених, аспірантів та студентів

29–31 серпня 2022
Тернопіль

УДК 004.56.5(043.2)

Кузик В.М.[1] Продан Т.І.[2], Івасьєв С.В.[2], Слєпцова О.Я.[1]

[1]Галицький фаховий коледж імені В'ячеслава Чорновола
[2]Західноукраїнський національний університет

БІОМЕТРИЧНА СИСТЕМА АУТЕНТИФІКАЦІЇ З ВИКОРИСТАННЯМ ГОЛОСОВИХ ДАНИХ

Вступ. Ідентифікація людини за голосом є одним із біометричних систем аутентифікації. Біометрична система аутентифікації – система перевірки особистості (аутентифікації) людини за її біометричними показниками Біометричний параметр – параметр, що є частиною самої людини.

Біометричні системи аутентифікації досить зручні для користувачів. На відміну від парольних або ключових систем автентифікації, біометричні використовують параметри, які неможливо забути або втратити. Проблеми збереження автентифікаційних даних не виникає.

Мета роботи полягає у тому щоб розглянути один із способів побудови біометричної системи ідентифікації користувача за голосом, яка є незалежною від ключової парольної фрази. Основними перевагами використовуваного підходу є простота реалізації та висока надійність.

1. Біометричний метод аутентифікації за голосом

Біометричний метод аутентифікації за голосом є простим у застосуванні. Він не вимагає спеціальної апаратури, достатньо лише мікрофона та звукової плати. В даний час технологія розвивається, так як цей метод широко використовується у сучасних бізнес-центрах. Основним недоліком методу голосової автентифікації є його низька точність. Голос людини може змінюватись залежно від стану здоров'я, настрою, віку тощо. Крім того, не варто забувати про сторонні шуми. Через високу ймовірність помилок другого роду його застосовують переважно у приміщеннях середнього рівня безпеки [1]. Один із способів аутентифікації людини за голосом – це завдання системі безлічі зразків голоси однієї і тієї ж людини для порівняння з автентифікаційним ключем, що отримується в майбутньому. Далі існує безліч способів порівняння [1, 2].

Система, яку передбачається розробити, має бути незалежною від конкретної фрази. Через це може виникнути проблема з підміною голосу за допомогою звукозаписних пристроїв, наприклад, диктофонів. Однак цю проблему можна обійти за допомогою генерації простих фраз, які людина має вимовити. Не можна заздалегідь передбачити, яка фраза буде затребувана системою, що розпізнає, отже зловмисник не зможе записати ключову фразу. Такий підхід додає завдання перекладу мови до тексту, проте вона не є основною і може бути вирішена за допомогою сторонніх бібліотек [3].

На вхід системи ідентифікації надходить запис ключового повідомлення користувача. Одним із простих форматів для обробки є WAV.WaveformAudioFileFormat – формат файлу-контейнера для зберігання записи оцифрованого аудіопотоку, підвид RIFF. Цей контейнер, як правило, використовується для зберігання несжатого звуку в імпульсно-кодовий модуляції.

Отримані значення амплітуд можуть не збігатися для двох однакових записів через зовнішній шум, різні гучності вхідного сигналу і так далі. Одним із найбільш ефективних способів попередньої підготовки звуку є нормалізація [4].

Нормалізація звуку – процес вирівнювання частотних характеристик студійного звукозапису на магнітний носій. Корекція необхідна, оскільки процес намагнічування покриття плівки відбувається нерівномірно стосовно спектру аудіочастот. Якщо не проводити корекцію, навіть перше відтворення запису звучатиме несхоже на оригінал.

Існують два способи нормалізації [4]:

- пікова нормалізація – це спосіб нормалізації, за якого рівень звукового сигналу піднімається до максимально можливого значення цифрового звуку без появи спотворень. Орієнтиром служить найвищий пік амплітуд. Цей спосіб повністю виключає обмеження амплітуди сигналу (кліпінг), проте, за наявності у файлі сильно виділяється піку, то нормалізація за його рівнем може призвести до тому, що звуковий сигнал залишиться досить тихим, незважаючи на досить високу гучність оригіналу. Розмір звуку при пікової нормалізації вимірюється у відсотках;
- RMS-нормалізація - нормалізація за середньоквадратичним значенням рівня звуку у файлі. Є повною протилежністю до пікової нормалізації. При цьому способі величина звуку вимірюється у децибелах. Цей спосіб найбільш підходить для людського вуха, однак за високої гучності можливий кліпінг.

Оскільки передбачається, що людина вимовлятиме ключове вираз спокійний, то очевидна перевага у пікової нормалізації внаслідок того, що в необробленому звуку не повинно бути надто великих перепадів амплітуд. Так як розмір унікальних характеристик навіть для секундного зразка звуку величезний, то робити складні операції над такими обсягами даних неможливо. Крім того, не зовсім зрозуміло, як порівнювати об'єкти з різною кількістю унікальних характеристик.

Обчислювальну складність завдання можна зменшити, розбивши її на менш складні підзавдання. Це дозволить за допомогою встановлення фіксованого розміру підзадачі та усереднення результатів обчислень за всім завданням одержати наперед задану кількість ознак для класифікації. Як розбиття мається на увазі використання поділу звукового сигналу на звані кадри певної довжини. Кадри повинні перекривати один одного, тому що у випадку, якщо вони будуть поруч один з одним, то звук спотворюватиметься.

Для усунення небажаних ефектів при обробці кадрів кожен елемент кадру множиться на вікно. Вікно – вагова функція, яка використовується для керування ефектами, зумовленими наявністю бічних пелюсток у спектральних оцінках (розтікання спектра).

У більшості завдань цифрової обробки немає можливості досліджувати сигнал на безкінечному інтервалі. Немає можливості дізнатися, який був сигнал до увімкнення пристрою і який він буде в майбутньому. Також обмеження інтервалу дослідження може бути обумовлено нестаціонарністю досліджуваного сигналу.

Обмеження інтервалу аналізу рівносильне добутку вихідного сигналу на віконну функцію. Таким чином, результатом віконного перетворення Фур'є не спектр вихідного сигналу, а спектр твори сигналу та віконної функції. Спектр, отриманий за допомогою віконного перетворення Фур'є [5] є оцінкою спектра вихідного сигналу і принципово допускає спотворення.

Спотворення, що вносяться застосуванням вікон, визначаються розміром вікна та його формою. Виділяють дві основні властивості частотних характеристик вікон: ширина головної пелюстки та максимальний рівень бічних пелюсток. Застосування вікон, відмінних від прямокутних, обумовлене бажанням зменшити вплив бічних пелюсток за рахунок збільшення ширини головного.

Типи віконних функцій:

Прямокутне вікно

$$w(n) = \begin{cases} 1, n \in [0, N-1] \\ 0, n \notin [0, N-1] \end{cases}. \quad (1)$$

Вікно Ханна

$$w(n) = 0.5(1 - \cos(\frac{2\pi n}{N-1})). \quad (2)$$

Вікно Хемінга

$$w(n) = 0.53836 - 0.46164(\cos(\frac{2\pi n}{N-1})). \quad (3)$$

Вікно Блекмена

$$w(n) = a_0 - a_1 \cos(\frac{2\pi n}{N-1}) + a_2 \cos(\frac{4\pi n}{N-1}). \quad (4)$$

Вікно Кайзера

$$w(n) = \frac{\left| l_0(\beta \sqrt{1 - \left(\frac{2n - N + 1}{N-1} \right)^2}) \right|}{|l_0(\beta)|}. \quad (5)$$

Найбільш простий і підходящою для вирішення задачі є функція вікна Хеммінгу.

Далі потрібно отримати короткочасну спектрограму кожного кадру окремо. Для цього використовується перетворення Фур'є.

Перетворення Фур'є (Fouriertransform) - це розкладання функцій на синусоїди (далі косинусні функції теж називаємо синусоїдами, оскільки вони відрізняються від «справжніх» синусоїд тільки фазою). Існує кілька видів перетворення Фур'є [5].

1. Неперіодичний безперервний сигнал можна розкласти в інтеграл Фур'є.

2. Періодичний безперервний сигнал можна розкласти у нескінченний ряд Фур'є.

3. Неперіодичний дискретний сигнал можна розкласти на інтеграл Фур'є.

4. Періодичний дискретний сигнал можна розкласти в кінцевий ряд Фур'є.

Комп'ютер здатний працювати лише з обмеженим обсягом даних, отже, реально він здатний обчислювати лише останній вид перетворення Фур'є. Отже, використовуватиметься дискретне перетворення.

На сьогоднішній день найбільш успішними є системи розпізнавання голосу,

які використовують знання про слуховий апарат. Велике поширення при розпізнаванні людського мовлення набула mel-шкала, лінійна при частотах нижче 1кГц і логарифмічна при частотах вище 1кГц. Mel-шкала була отримана в результаті експериментів із зразковими тонами (синусоїдами) в яких з випробовуваних вимагалося розділити дані діапазони частот на 4 рівні інтервалу або налаштувати частоту необхідного тону так, щоб він був в половину частоти вихідного. 1 mel визначається як 1 тисячна рівня тону на 1 кГц. як і в будь-яких інших спробах створити подібні шкали, розраховується, що шкала mel більш точно моделює чутливість людського вуха.

Перехід до нової шкали описується нескладною залежністю:

$$m = 1127 \ln (1 + \frac{f}{700}), \quad (6)$$

де m - Частота в крейдах; f – частота у герцах. Вектор ознак складатиметься з крейдяних коефіцієнтів, що розраховуються за формулою:

$$c_n = \sum_{k=1}^{K} (\log S_k) \left[n(k - \frac{1}{2}) \frac{\pi}{K} \right], \quad (7)$$

де c_n - Крейда-кепстральний коефіцієнт під номером n; S_k - Амплітуда kго значення в кадрі в крейдах; K – наперед задану кількість дрібнепстральних коефіцієнтів.

Висновок. Останньою стадією є класифікація того, хто говорить. Класифікація проводиться обчисленням міри схожості пробних даних та вже відомих. Міра схожості виражається відстанню від вектора ознак пробного сигналу до ознак уже класифікованого вектора.

Вектор ознак представляється як середнє арифметичне векторів, що характеризують окремі кадри мови. Для підвищення точності розпізнавання просто необхідно усереднювати результати не тільки між кадрами, а й враховувати показники кількох мовних зразків. Маючи кілька записів голосу, розумно не усереднювати показники одного вектора, а провести кластеризацію з допомогою нейронних мереж.

Перелік використаних джерел.

1. Болл Р. Руководство по биометрии / Р.М. Болл, Дж.Х. Коннел; [пер. с англ. под ред. Н. Агапова]. – М.: Техносфера, 2007. – 368 с.

2. Лакин Г. Биометрия / Лакин Г.Ф. – М.: Высшая школа, 1990. – 352 с.

3. Кауненко С.І., Колесніков К.В. Методи ідентифікації людини в інформаційних системах ISDMCI'2012: Интеллектуальные системы принятия решений и проблемы вычислительного интеллекта: Материалы международной научной конференции.- Херсон-Євпаторія ХНТУ,2012, 566с., С. 164-167.

4. Precise Biometrics // [Електр. Ресурс]. – Режим доступу: https://precisebiometrics.com/

5. An emerging biometric API industry standart // [Електр. Ресурс]. – Режим доступу: http://ieeexplore.ieee.org/document/820046/ 6. BioAPI Specification Version 1.1 // [Електр. Ресурс]. – Режим доступу: http://xml.coverpages.org/BIOAPIv11.pdf


АВТОМАТИЗАЦІЯ
ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ
ТЕХНОЛОГІЇ
проблемно-наукова міжгалузева
конференція молодих науковців
аспірантів та студентів
м. Тернопіль
2022

ЗАХІДНОУКРАЇНСЬКИЙНАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТЕФАНИКА
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
НАДВІРНЯНСЬКИЙ КОЛЕДЖ НТУ
ГАЛИЦЬКИЙ КОЛЕДЖ ІМ. В. ЧОРНОВОЛА

Проблемно-наукова міжгалузева конференція

АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

(АКІТ – 2022)

21—23 лютого 2022 року

Тернопіль

БЕЗПЕКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.56.5(043.2)

Продан Т.І.[1], Івасьєв С.В.[1]

[1]Західноукраїнський національний університет

СУЧАСНІ МЕТОДИ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ

Вступ. До сучасних методів аутентифікації відноситься перевірка автентичності на основі біометричних показників. При біометричній аутентифікації, секретними даними користувача можуть бути як очна сітківка, так і відбиток пальця. Ці біометричні образи є унікальними кожного користувача, що забезпечує високий рівень захисту доступу до інформації. Згідно з попередньо встановленими протоколами, біометричні зразки користувача реєструються в базі даних.

Мета: Дослідження систем біометричної ідентифікації, їх структури і технічних засобів для підвищення безпеки.

1. Особливості систем біометричної ідентифікації

Біометрична ідентифікація - це процес порівняння та визначення подібності між даними людини та її біометричним «шаблоном». Біометрія дозволяє ідентифікувати та провести верифікацію людини на основі набору специфічних та унікальних рис, властивих їй від народження. Цей метод розпізнавання прийнято вважати одним із найнадійніших, тому що на відміну від стандартних логіну та паролю біометричними даними набагато складніше несанкціоновано скористатися.

Розглянемо механізм дії біометричних систем. Спочатку в базі даних або захищеному переносному елементі, такому як смарт-карта, зберігається еталонна модель, заснована на біометричних характеристиках людини. Для цього можуть використовуватись один або кілька біометричних зразків. Збережені дані перетворюються на математичний код; таким чином формується база даних, що є набором кодів до 1000 біт, що фіксують унікальні біометричні характеристики користувачів. При зчитуванні відбитка пальців або райдужки ока сканер не розпізнає саме зображення, а перетворює його на цифровий код, який потім порівнює із завантаженою раніше еталонною моделлю.

Розглянемо типи систем біометричної ідентифікації. Найчастіше при думці про біометричні системи нам на думку приходять сканери відбитків пальців або системи розпізнавання облич. Можливо, ці типи систем здобули найбільшу популярність завдяки кінофільмам та телесеріалам. У будь-якому разі наука не стоїть на місці, і в останні роки межі біометрії стали набагато ширшими. На даний момент існують і активно застосовуються в безпеці, системах контролю доступу та запобігання крадіжкам вже 14 типів біометричних пристроїв.

Розглянемо ті методи та інструменти, які оцінюють біометричні параметри в статиці без розвитку в часі. Відбитки пальців Розпізнавання відбитків пальців є одним із перших біометричних методів. Він ґрунтується на визначенні структури ліній на подушечках пальців рук, інакше — папілярних візерунків. Після зчитування сканером унікальний малюнок трансформується на цифровий

біометричний шаблон, за допомогою якого система визначає, хто перед нею знаходиться. Такі сканери поділяються на два основні типи: оптичні та кремнієві (теплові та ємнісні).

Кожен із типів має свої переваги та недоліки. Наприклад, оптичні сканери є найбільш точними з погляду визначення візерунка, проте їх можна обдурити за допомогою силіконових або латексних накладок та інших нехитрих прийомів. Також вони швидко забруднюються на відміну від лінійних теплових і для виключення похибок їх доводиться очищати після кожного застосування. Для користувача відмінність полягає лише в тому, як взаємодіяти зі сканером - торкатися або проводити його. Зараз завдяки вбудованим у смартфони сканерам можна розблокувати таким чином мобільний пристрій, сплатити за покупки в інтернеті. У найближчому майбутньому планується впровадити подібні технології та інші пристрої загального користування, наприклад, у банкомати і навіть у метрополітені для заміни квитків (такий проект збираються реалізувати в Британії, щоб скоротити навантаження на турнікети).

Рисунок 1 – Класифікація систем автентифікації

Компанія Integrated Biometrics зі штаб-квартирою у Спартанбурзі, Південна Кароліна, була заснована у 2002 році. Integrated Biometrics продовжує рости як глобальний постачальник біометричних рішень. Компанія займається розробкою датчиків реєстрації та перевірки відбитків пальців. У продуктах Integrated Biometrics використовується плівка LES (світловипромінюючий датчик), що забезпечує швидкість, простоту використання та довговічність мобільних пристроїв біометричної верифікації. Інтегровані біометричні датчики відбитків пальців Sherlock, єдині сертифіковані ФБР, працюють під прямим сонячним промінням на сухих або вологих пальцях, стійкі до стирання і на 90-95 відсотків менше і легше традиційних оптичних сканерів. Вони більше підходять для мобільних пристроїв ніж кремнієві або традиційні оптичні датчики.

Сканери відбитків пальців СВП1 і СВП2 дозволяють отримати дактилоскопічні відбитки в різних режимах. Мають вбудований алгоритм перевірки якості відбитка та підсвічування для контролю стану сканування. Запуск сканування виконується автоматично при прикладанні пальця. Після отримання зображення сканер самостійно проводить перевірку якості. Кодування та декодування зображення здійснюються з використанням алгоритму WSQ (вейвлет-скалярне квантування). Цей тип є вдосконаленою версією попереднього. Зламати алгоритм його роботи значно важче, ніж за іншого біометричного сканування, оскільки вени знаходяться глибоко під шкірою. Інфрачервоні промені проходять через поверхню шкіри, де вони поглинаються венозною кров'ю.

Спеціальна камера фіксує зображення, оцифровує дані, а потім або зберігає їх або використовує для підтвердження особистості. Нещодавно технологія, що отримала назву FingoPay, була протестована. Система ідентифікує унікальний малюнок вен пальців рук і є майже досконалим пристроєм завдяки тому факту, що збіг структури малюнка вен у двох різних людей рівна 1 : 3,4 млрд.

Також аналогічна система виробника Hitachi була використана виробниками японських банкоматів. Bio-Plugin: біометрична система СКБД (M2SYS) Розробка M2SYS Bio-Plugin може розпізнавати як відбитки пальців, так і малюнок вен на пальцях та дозволяє підприємствам будь-якого масштабу та будь-якої галузі швидко інтегрувати в роботу систему біометричного програмного забезпечення. Bio-plugin може проводити зіставлення 100 мільйонів відбитків пальців на секунду на одному сервері та підтримує сумісність із Windows та веб-додатками.Перевагами системи є можливість швидкої інтеграції (кілька годин), відсутність компіляції ПЗ та глобальна інфраструктура підтримки.

Система була інтегрована в комплексне біометричне рішення виробника - гібридну біометричну платформу, мультимодальну систему, яка підтримує кілька форм біометрії, включаючи відбитки пальців, вени пальця та розпізнавання облич. Palm Jet (BioSmart) Однією з найновіших розробок компанії BioSmart є безконтактний сканер Palm Jet. Це комплексна біометрична система, яка позбавляє співробітників необхідності торкатися одних і тих же поверхонь. На даний момент використовується деякими підприємствами як заходи для профілактики COVID-19. Palm Jet сканує мережу підшкірних вен і порівнює результат із шаблоном у БД. Контрастний малюнок вен долоні формується рахунок різних коефіцієнтів поглинання випромінювання венами і тканинами долоні. Пристрій відмінно захищений від підлоги - його неможливо обдурити за допомогою силіконового муляжу або фотографії. Відстань сканування становить 40-100мм, при цьому швидкість розпізнавання менше секунди. Інші пристрої вендора: BioSmart PV-WTC-термінал для організації пропускного режиму та обліку робочого часу; BioSmart PV-WM та BioSmart DCR-P -зчитувачі вен долоні.

Висновки. Системи біометричної автентифікації набувають широкого розповсюдження та різноманіття технічних засобів. Розробка та дослідження систем біометричної ідентифікації безумовно актуальна задача.

Перелік використаних джерел.

1. Болл Р. Руководство по биометрии / Р.М. Болл, Дж.Х. Коннел; [пер. с англ. под ред. Н. Агапова]. – М.: Техносфера, 2007. – 368 с.

2. Кауненко С.І., Колесніков К.В. Методи ідентифікації людини в інформаційних системах ISDMCI'2012: Интеллектуальные системы принятия решений и проблемы вычислительного интеллекта: Материалы международной научной конференции.- Херсон-Євпаторія ХНТУ,2012, 566с., С. 164-167.

4. Precise Biometrics // [Електр. Ресурс]. – Режим доступу: https://precisebiometrics.com/

5. An emerging biometric API industry standart // [Електр. Ресурс]. – Режим доступу: http://ieeexplore.ieee.org/document/820046/