CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Secure Password-less Authentication

A Multi-Factor Approach Using Facial Recognition, FIDO2 standard, and Dynamic OTP

Master's thesis in Computer Systems and Networks

Ayush Siddharth, Deaa Khankan

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Secure Password-less Authentication

A Multi-Factor Approach Using Facial Recognition, FIDO2 standard, and Dynamic OTP

AYUSH SIDDHARTH

DEAA KHANKAN

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Secure Password-less Authentication: A Multi-Factor Approach Using Facial Recognition, FIDO2 standard, and Dynamic OTP

Supervisor: Rhouma Rhouma, CSE
Examiner: Ahmed Ali-Eldin Hassan, CSE

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2025

Secure Password-less Authentication: A Multi-Factor Approach Using Facial Recognition, FIDO2 standard, and Dynamic OTP
A Multi-Factor Approach Using Facial Recognition, FIDO2 standard, and Dynamic OTP

AYUSH SIDDHARTH
DEAA KHANKAN

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In the rapidly advancing era of digitization, traditional password-based authentication systems are becoming insufficient to secure online services. The use of weak passwords introduces various vulnerabilities including brute-force attacks, credential theft, and phishing. Moreover, the usage of complex passwords is not user-friendly and leads the user to reuse the same password across multiple services. The adoption of biometric authentication systems, especially facial recognition-based ones, is becoming more common in handheld devices like mobile phones and laptops. However, it remains limited in online services due to several security challenges, such as spoofing, privacy concerns regarding user data embeddings, and the reliability of securing sensitive information. These challenges underscore the need for more robust and user-friendly authentication solutions to protect sensitive data. This thesis aims to design and develop a multi-factor authentication system inspired by FIDO2 standard. by combining facial recognition, Dynamic One-Time passwords alongside FIDO2 standard which utilize a standard USB stick for secure public-key cryptography. This study evaluates the performance and the overall security of the proposed system. The prototype developed in this thesis is secure against several attacks, including brute-force attacks, phishing, human negligence, and SQL injection. That said, it still has some limitations due to the lack of necessary hardware. For example, the prototype remains insecure against spoofing attacks and can be easily cracked due to the absence of a camera capable of processing the depth of the face in the current frame to detect liveness. Despite the limitations caused by hardware constraints, the findings highlight the potential of the proposed prototype to be a secure authentication system. Future work could focus on integrating more advanced hardware to overcome these challenges, making the system a viable solution for secure and scalable authentication in real-world applications

Keywords: FIDO2, Face Recognition, Computer Science, Authentication System, Security, Biometric Authentication.

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Rhouma Rhouma, for their invaluable guidance, support, and encouragement throughout this thesis. Their expertise, insightful feedback, and unwavering patience have been instrumental in shaping this work. We sincerely appreciate the time and effort they dedicated to helping us develop our ideas and refine our research. This thesis would not have been possible without their continuous support and mentorship.

Ayush Siddharth, Deaa Khankan, Gothenburg, 2025-02-15

Contents

List of Figures

List of Tables

1

Introduction

In the era of digitization that is growing rapidly more now than ever, the traditional password-based authentication system mechanisms are not enough to secure online systems and services. Choosing between complex and weak passwords involves a trade-off between security and usability. Complex passwords are harder to guess and provide better protection against attacks but are difficult to remember, often leading users to reuse them or store them insecurely. Weak passwords are easier to recall but significantly increase vulnerability to brute-force and dictionary attacks. Moreover, susceptibility to phishing[1], brute force attacks[2], credential theft[3] are some of the shortcomings of the old system that has led to more exploration on how to make such systems and services more secure and at the same time keeping it user-friendly.

In majority of the handheld devices, biometric authentication is included nowadays. It relies on capturing unique features of an individual for identification such as fingerprints and facial recognition. Facial recognition is one of the most commonly used methods. This method verifies the identity of an individual through unique facial features, and utilizing some involuntary actions like eyes blinking to provide a more secure way of accessing systems. In biometric authentication the user-friendliness is addressed as the users don't need to remember the passwords and offers a more seamless authentication mechanism.

Facial-based authentication systems are also not 100 percent secure and have been found vulnerable to few specific attacks, Including spoofing attacks using deepfake technologies to impersonate another individual. However, the proposed prototype in this thesis does not solely relies on facial-recognition technology but also introduces a second factor of authentication using Dynamic One-Time Password (OTP). The whole approach towards multifactor authentication in the proposed prtotype also takes inspiration from Fast Identity Online 2(FIDO2) standard which makes the prototype more secure and robust.

FIDO2 protocol is a web authentication protocol and is widely supported by modern web browsers and services. It is based on public-key cryptography and addressees many of the shortcomings of the traditional password based approach as well. FIDO2 is often used in combination with secure hardware devices, like token security keys, Android key store on smartphones and trusted platform module on computers. In this thesis the hardware used to store the credential is a normal USB stick instead of token security key. When the user logins, the authenticator where the private key is stored securely, is used to verify the possession of the private key without it being sent over the internet. This standard is thus more resistant to phishing attacks. Moreover, the extra layer of incorporating Dynamic One-Time Passwords which uses short-lived passwords that are sent to users over the email which only users have access to, provides an additional depth in cases

where the adversary might be able to attack the facial recognition detection mechanism.

The purpose of this thesis is to develop and analyze the advantage and limitations of a multi-factor authentication system inspired by FIDO2 standard (a possession factor) by integrating facial recognition (an inherence factor), and OTP (a knowledge factor), taking into account resource usage, accuracy, conveniency and the security of the whole environment. The client application will only be a mock-up login screen and the focus will be on the security aspect of the prototype. A normal USB stick will be used to store and retrieve the authentication credential securely.

Chapter 2, Cryptographic methods for User Authentication presents a literature overview of the authentication methods and cryptographic tools used to develop such a system. The chapter begins by describing the authentication methods and explores the authentication and registration ceremonies in the FIDO2 standard. OAuth 2.0 is another state of the art method that is used for authentication and is briefly explained to understand the MFA system alternative for comparison. Further, it covers the principles and theory behind the cryptographic tools. Chapter 3 discusses the facial recognition models. It provides an outline for one traditional model and one state-of-the-art model. Finally, it describes the metrics that will be utilized for comparing such models.

Chapter 4 provides a high-level overview of the system architecture design, while Chapter 5 outlines the low-level implementation, focusing on the authenticator. The results are also introduced, describing the end product of the prototype, the accuracy metrics, and the threat model.

The next chapter discusses about the results obtained from the security and performance analysis for our system. It also refers to various research done on facial recigntion systems to better understand the accuracy of such systems. Finally, future work is mentioned on how to further improve the system before concluding in Chapter 7.

2

Cryptographic methods for User Authentication

Digital security relies heavily on authentication, which allows systems to confirm users' identities before allowing them access to private data. Most of the authentication techniques that meet the standard security requirements include Time-Based One-Time Passwords (TOTP), FIDO2, and OpenID Connect. These techniques, however, are not impervious to flaws that could jeopardise the integrity of authentication, such as phishing attempts, credential thefts, and facial spoofing. Cryptographic tools are crucial for guaranteeing both confidentiality and authenticity in order to reduce these dangers. The two encryption methods used to protect data are symmetric encryption, which uses a common key to encrypt and decrypt data, and asymmetric encryption, which uses a pair of public and private keys. Sophisticated cryptographic algorithms, such as Password Based Key derivation Function(PBKDF2), Hash Based Message Authentication Codes(HMACs), and Adavnced Encryption Standard(AES), provide extra security layers to safeguard sensitive data and authentication procedures. The Python cryptography library is essential for putting these cryptographic methods into practice since it provides developers with reliable ways to incorporate authentication and encryption into their apps. This section examines these cryptographic tools, vulnerabilities, and authentication techniques, giving a summary of the key technologies that guarantee safe access and data security in systems.

2.1 Authentications Methods

Authentication essentially refers to a process where the identity of a user or a device is verified. With everything getting digitalized, it has gained a lot of attention, especially for protecting sensitive data. It helps protect personal information and prevents unauthorized access. There are several ways through which this process is implemented with the most simple, traditional one being the username/password authentication, where the user only has the knowledge about the credentials. The traditional methods are getting replaced with advanced ones like biometrics authentication.

To add more layers of security, multi-factor authentications are becoming the standard now for most industries and organizations[4]. In this method, a combination of authentication methods are used. This essentially means that even if one method is compromised, the adversary still needs to break others to be able to get access. On a broad level, these different methods can be classified into three factors: knowledge, possession and inherence[5].

The knowledge factor refers to the information that only user knows about, such as password or a short-lived one time password which the user can obtain through SMS, email or through an app. The latter is also cited as Dynamic One-Time Passwords or Time based One-Time Passwords(TOTP). Creating and remembering a complex passwords are very cumbersome and as a result the password based factor is considered less secure[6]. On the other hand, OTPs act as a great layer of security with other authentication methods. Since they expire quickly, generally within seconds, even if an adversary gets hold of it will expire in seconds. The OTPs are widely used in financial organizations[7].

The possession factor mainly involves having a security token, or a USB key or a smart phone device. It requires special hardware configurations and strong encryption to store the keys. The possession factor also comes under the category of password-less authentication. The main challenge that this raises is physical access to the USB key or device where the authentication data is saved should be kept safely. Yubikey, developed by Yubico[8] is one such hardware that can be used as a security token.

The inherence factor is based on features that are unique to us, like fingerprints, iris patterns, voice or facial features. This comes with additional privacy requirements and it is easier and convenient way of authentication compared to other factors[9]. In this method of biometric authentication, as it is called, the data collected is personal and extra care should be taken to store it securely. All smartphones are enabled with biometric authentication to protect sensitive data. The adversary can still in some cases use spoofing techniques or deepfake technologies to bypass biometric authentication layer. In the next few sections, some of the authentication methods or standards are explored.

2.1.1 TOTP

Time-based One-Time Passwords(TOTP) are short-lived unique codes that is used to verify user's identity in addition to using other factors of authentication. TOTP can be implemented across different mediums such as email, sms, or app-based authenticators. It relies on time-synchronized cryptographic hashing and for implementing such functionality, the prover and the verifier must know or be able to derive the current Unix time using the OS UNIX epoch time for generating OTP[10].

In the TOTP setup, a unique secret key, encoded in Base32 is generated by the service provider. The user does not need to know the secret key explicitly. The secret key is then shared by utilizing the medium it is implemented on. The key acts a basis for generating secure codes. It uses Hash-Based Message Authentication Code (HMAC) with SHA-1 as its hashing algorithm. The current time along with the unique secret key are hashed to generate a unique HMAC output. This output is further truncated to produce a six-digit or eight-digit OTP, depending on the configuration required[11]. The whole algorithm can be summarized in Equation (2.1).

$$TOTP = Truncate(HMACSHA1(secret, timeinterval)) \tag{2.1}$$

The TOTP code is valid only under a specific time interval which is generally between 30 and 60 seconds. This ensures that even if the OTP is leaked, it becomes invalid after that time period. After the user enters the OTP code, the server recalculates the TOTP using the current time interval before comparing the results. This time-bound feature

makes it difficult for any adversary since the code usually expires in seconds and becomes invalid[12].

For using email as a medium, the OTP code generated by the server is sent to the user's registered email. This method relies on the server to calculate a new OTP every time the user wants to authenticate. Another medium can be an application which is required to be downloaded by the user in order to recieve the OTP. The OTP can be shared with the user through QR code that the user scans with an authenticator app (like Google Authenticator) to make the process more secure[13].

PyOTP is one such library in Python that provides the above functionality. It provides an interface which is easy to use and can generate and verify OTP codes[14]. In this project, a Simple Mail Transfer Protocol (SMTP) server will be used along with py-OTP, to send emails to the user's registered email. To allow for slight discrepancies in time-synchronization between the prover and the verifier, PyOTP also allows a validity window[14].

When TOTP codes are sent over SMS or email, the security is only strong as these channels. These are particularly vulnerable to interception, man-in-the middle and phishing attacks.

2.1.2 FIDO2 Standard

FIDO stands for Fast Identity Online. FIDO2 is a standard for password-less authentication created by the FIDO alliance in collaboration with the World Wide Web Consortium(W3C). The Fido Alliance industry is a well-reputed consortium that develops open authentication standards to reduce the dependency on traditional password-based authentication. W3C maintains the web guidelines standards. FIDO2, their latest standard promotes password-less security[15].

The FIDO2 standard leverages asymmetric encryption to authenticate users without passwords. It consists of two primary ceremonies, Registration and Authentication. In the registration ceremony, the authenticator uses public key cryptography to establish secure credentials with the Relying Party(RP), which is initiated by the website or service requesting secure authentication. The RP stores the user's public key for future verifications and the authenticator stores the private key securely. In the authentication ceremony, when the user attempts to login, the RP sends back a challenge to the Authenticator via the RP Javascript application. After which the Authenticator signs using the private key before sending it back to the RP for verification[15].

The cryptographic credentials that are stored on the authenticator are popularly known as passkeys and are often backed by biometrics or PIN. It can be stored on an external entity as well such as on a USB security key, if it supports FIDO2. The external storage makes it difficult for the adversary to get hold of the key since this is not accessible over the internet. For this thesis, facial-recognition biometric method will be used to access the passkey.

Security keys are physical devices which are used to enable public cryptography. The users must insert, tap or NFC-scan the key to make the whole system work as the private key resides within the unique physical security key that only the user possesses. The requirement of an external physical device also ensures protection against remote

attacks[15].

The FIDO2 standard is based on two main components: WebAuth(Web Authentication) standard and CTAP2(Client to Authenticator) Protocol. WebAuthn is a browser-based API standard for passwordless authentication, while CTAP2 enables communication between external authenticators and devices, together forming the foundation of the FIDO2 authentication framework[15].

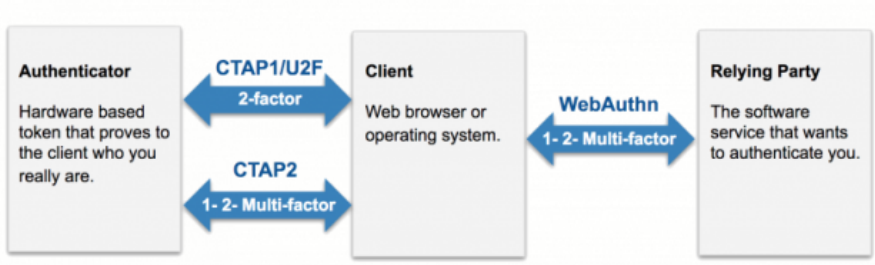


Figure 2.1: FIDO2 Building Block [16]

2.1.2.1 Web Authentication

The Web Authentication(WebAuth) is a W3C standard for password-less authentication using asymmetric encryption. It enables interaction between Relying Parties (website or the application) and users devices like security keys or smartphones, that act as authenticators[17].

WebAuth enables both registration and authentication ceremonies. There are three main components involved in the whole workflow of WebAuth. The software service, which is also known as the Relying Party(RP), is where the users registers and authenticate their accounts. The RP stores the user's public key provided by the authenticator during the registration ceremony, which helps in validating the response. This association of the public key with the user's account enables the RP to authenticate users during their future visits without needing a password. To prevent the credentials reuse for multiple sites, the RP is uniquely identified by its domain.

The second component is the client, which is the browser or the app that acts as an intermediary which helps in facilitating the communication between the RP and the authenticator. The main role of the browsers or the app here is to help in exchanging the data securely during both the ceremonies of registration and authentication. When the RP initiates the register or login processes, the client sends the request to the authenticator and then forwards the response from the authenticator back to the RP. It is the client responsibility to make sure that the structure of the exchanged data is correct. WebAuth is supported by modern browsers such as Chrome, Firefox, and Safari[17].

The third component is the Authenticator, where the private key is stored to prove to the client, the identity. It performs cryptographic operations, such as encrypting and storing the user's private key securely and signing the challenge from the RP. This Authenticator can either be platform based or physical security keys. The platform authenticators stores private keys in a specialized hardware-based security chip embedded into the user's device called Trusted Platform Module(TPM). These are usually integrated with the device's operating system and the private keys are kept isolated in a hardware-protected

environment. The physical security keys on the other hand can be used across different client devices offering more flexibility to the users. Portable devices, like special USB sticks or NFC keys are some examples of such authenticators[17].

The WebAuth Workflow for both the ceremonies, Registration and Authentication is described in more detail below.

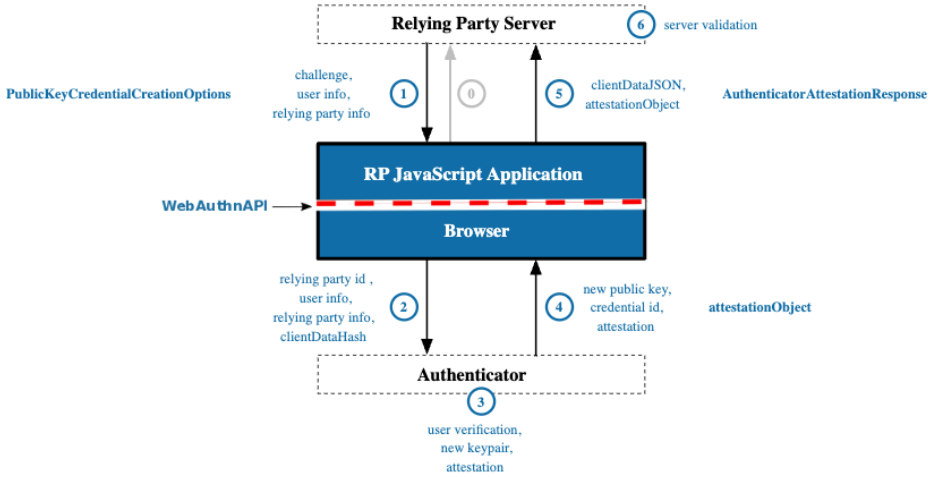


Figure 2.2: WebAuth Registration Ceremony [18]

- Registration Ceremony

Step 0 In the first step, the RP initiates the registration process through the RP JavaScript application that is running on the users browser or app.

Step 1 After the client receives the request from the RP server, which consists of a challenge (a unique random value), user-specific information and other information about the RP like its domain or properties that will be used in the cryptographic encryption by the authenticator later.

Step 2 All the above information is forwarded to the Authenticator using WebAuth API supported by the browsers. In this step, the communication is initiated between the users' browser or app to the authenticator.

Step 3 Now, the authenticator processes the request, does user verification and generates a new key-pair. The user verification in this Thesis is focused on facial recognition. Before sending the response, it stores the private key.

Step 4 Once the face is successfully captured, and the key-pair is generated, the public key, along with user info and attestation object is sent back to the client. The attestation object serves as a cryptographic proof that the key-pair was generated by a trusted authenticator and provides assurance to the RP.

Step 5 Here the data received by the browser or app (Client) is processed and then forwarded to the RP server for further validation.

Step 6 In this final step of registration, the RP server verifies the challenge using the received public key. If the public key is valid, it is stored in the RP credential

repository. This public key is used in future verifications when users try to identify themselves[18].

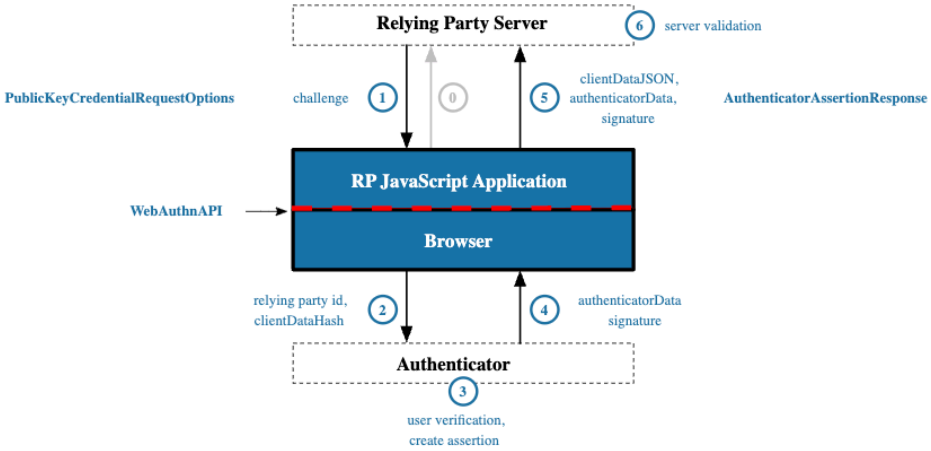


Figure 2.3: WebAuth Authentication Ceremony [19]

Authentication Ceremony

Step 0 Now, after the user is registered, the authentication process can be started by the user through interacting with the RP Java Script Application running in the browser or app. The user Id or email, which is a unique identifier for the user is provided.

Step 1 A challenge is generated by the RP server after receiving the login request and is sent back to the client.

Step 2 After receiving the challenge and user information, the client forwards the request to the authenticator using the WebAuth API through browser/app where the service is running. The payload consists of the RP unique identifier, which is used by the authenticator to validate the RP. The payload also includes a challenge for the authenticator to sign. The challenge signing and RP unique identifier prevents misuse of the credentials across different sites.

Step 3 Next step involves authenticator performing the verification. Once the user verification is successful, the private key access is permitted and the authenticator generates an assertion by signing the challenge received from the RP using the private key.

Step 4 All the information is now sent back to the browser in this step, which contains the signature and the authentication data.

Step 5 Here, the client compiles the data received and forwards it to the RP server.

Step 6 This is the step where the validation happens. First, the signed challenge is verified that it corresponds to the original challenge. Once this is done, the authentication data is validated to check if it hasn't been tampered with. If all checks are satisfied, the user is authenticated[19].

2.1.2.2 CTAP2

Client to Authenticator Protocol 2 (CTAP2) is one of the core components of FIDO2 standard and defines the protocol for secure communication between client devices and authenticators. In other words, this protocol handles real-time interaction between them, thus enabling password-less authentication. The client can be a platform (an Operating System such as Microsoft Windows), a browser (such as Google Chrome), or an application (such as an SSH client). In this process, different types of authenticators can be used. For a FIDO security key, USB or NFC is used typically[20].

The older version of CTAP1 does not support user verification using a PIN or biometric. While on the other hand, CTAP2 is a more versatile protocol and supports user verification. CTAP2 defines a set of commands that standardizes the communication. The two key commands are "authenticatorMakeCredential", which is used to generate a unique public-private key pair and "authenticatorGetAssertion", which is used for signing a challenge [20]. The key difference between CTAP2 and WebAuth is depicted in 2.1.

CTAP2 relies on asymmetric encryption to create unique credentials for each user and each relying party. During registration, the authenticator provides an attestation statement. When CTAP2 protocol is used, attestation are typically X.509 certificates that are issued by the authenticator manufacturer and helps the RP to verify the device's legitimacy. During authentication, the authenticator signs a challenge from the client using ECDSA(Elliptical Curve Digital Signature Algorithm). Elliptical Curve Cryptography(ECC) is advantageous due to its strong security with smaller key sizes. On the other hand, for the challenge data, CTAP2 uses SHA-256 for hashing. The payloads are hashed with SHA-256 before the authenticator signs it[21].

There are various security benefits because of the high standard the protocol maintains. Having a unique key-pair makes it secure and the RP domain verification helps to mitigate phishing attacks. The authenticator can detect when a malicious site attempts to impersonate a legitimate relying party. In such cases, it will safely deny the request to ensure security. By utilizing the challenge creation and response mechanism, replay attacks could be mitigated, since each challenge is unique. On top of that the attestation certificates adds another layer by ensuring the RP can safely trust the authenticator[21]. Additionally, the hashing mechanism ensures that data in transit is not altered ensuring integrity as well.

2.1.3 Yubikey

Yubikey is a hardware based security device developed by Yubico that strengthens the MFA by acting as a possession factor. It is designed to provide an alternative user-friendly approach compared to traditional username and password. It supports multiple protocols such as FIDO2/WebAuth and OTP and is compatible across major platforms (Windows, macOS, iOS, Android). Moreover, it is also designed to be waterproof, crust-resistant and long lasting. The latest Yubico Bio Series, shown is Figure2.4 supports fingerprint recognition but still facial recognition is not supported[22].

Figure 2.4: Yubico Bio Series with fingerprint sensors [23]

2.1.4 Open ID Connect

Open ID Connect(OIDC), the identity layer built on top of Open Authorization 2.0, or OAuth2.0. OAuth2.0 is a standard which is widely used in authentication and authorization systems. It enables secure delegated access to resources without exposing user credentials. For granting the access to another service, the user explicitly grants permission. It is token-based authentication, where a token is used instead of username and password. Before the token can be issued, OIDC can enforce MFA and can work together to make the system more secure[24].

OAuth2.0 consists of four main components in OAuth2.0 : user who owns the data (Resource Owner), the delegate or third party that is requesting to access the resource(Client), the server which issues the tokens (Authorization server) and the server where the resources are hosted(Resource Server)[25]. In a typical flow, the client redirects the user to the authorization server, and first requesting permission to grant access to resources. Once the user grants permission, the authorization server provides an authorization code to the client which is used to receive an access token. The access token can then be used by the third party to access user's resources from the resource server. The flow is shown in Figure 2.5. The tokens usually have an expiry date and should be refreshed or generated again for regaining the access.

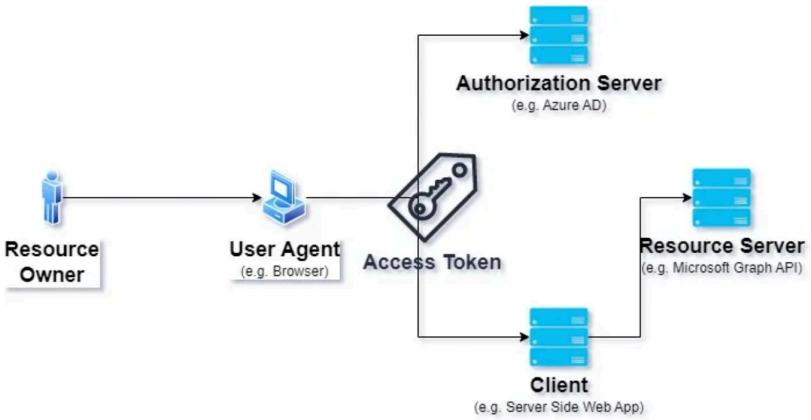


Figure 2.5: OAuth2.0 [26]

2.1.5 Authentications Vulnerabilities

Authentication vulnerabilities are flaws that allow attackers to masquerade as legitimate users or illegally access sensitive resources. This can arise due to several causes such as errors in the code, weak credential usage by the users as well as if the system is poorly implemented and contains logical errors. Some vulnerabilities may even allow attackers to get access to the desired resources without even interacting with the login page. This can lead breaching of sensitive information, masquerades as a legitimate user and in some cases can lead to gaining control over the whole system and destroy it. such vulnerabilities can be very costly for companies, especially if the breaches violate General Data Protection Regulation (GDRP) or California Privacy Rights Act (CCPA) regulations. Some of the most known attacks will be mentioned below.

Brute-Force and dictionary attacks: A Brute-Force attack is an attempt to gain unauthorized access by systematically automating an excessive number of login attempts using every possible combination of characters until the correct credentials are found. Such attacks can be computationally expensive and time-consuming, and therefore impractical in real-world scenarios, especially for complex passwords with a mix of characters and symbols. However, dictionary attacks rely on attempting pre-compiled lists of common passwords, such as "admin" or "123456." These lists are used by attackers to gain access to accounts with weak, guessable credentials. The attacker can create customized lists and include known information about the victim, such as pet names or birth dates. Such information can be mutated to generate additional passwords to add to the dictionary. A Rainbow Table attack is a variant of the dictionary attack, where precomputed hash values for common passwords are used to speed up the attack process. Rainbow tables could also be used if the attacker gains access to the username/password file, which is usually stored in a hashed form[27].

If no protection mechanisms exist to prevent such attacks, such as account lockouts, CAPTCHA, or salt when hashing, attackers can take advantage and gain access despite the use of complex credentials and reliable cryptographic operations.

SQL Injection is a vulnerability vector that allow the attacker to manipulate SQL queries and execute malicous commands on the database. Such commands can be used to gain access to sensitive information without even providing any credentials

bypassing authentication mechanisms[27]. Taking a simple scenario where the user attempt to login to a service by providing its username and password. The SQL validation query might look like this:

```
SELECT UserName FROM Users WHERE UserName = 'user_input_username'
↪  AND Password = 'user_input_password';
```

The query checks if the username and password exist in the database and returns the username or other relevant data, which allows the user to log in to the service.

An attacker can inject malicious SQL code to manipulate the query and bypass authentication. For example, an attacker could enter the username as "admin' OR 1=1 – " and any password. This query sets the username to admin and returns true directly, bypassing the rest of the query due to the usage of –, which comments out the remaining part of the query. The executed query would become:

```
SELECT UserName FROM Users WHERE UsenName = admin or 1=1;
```

which returns data related to "admin" account without providing any password credentials.

Insecure password change and recovery features can be serious points of attack if not implemented correctly. Such as using weak security questions, the absence of CAPTCHAs as well as using emails with long timeouts. Such vulnerabilities can be combined with others such as brute force to compromise accounts which is considered safe in normal circumstances[27].

Subdomain Takeover is a vulnerability that occurs when an attacker gains unauthorized control over a legitimate subdomain due to misconfiguration or an unclaimed subdomain. This vulnerability usually happens when a subdomain is registered and points to a resource or service, such as Bluehost, AWS, or other providers, that is no longer associated with any resources. The attacker can find these "orphaned" subdomains through DNS enumeration or subdomain scanning tools. Once the unclaimed subdomains are found, the attacker can attempt to claim them by registering new services/websites with the same provider and linking them to the unclaimed subdomain. The providers usually allow anyone to create resources and associate them with any domain/subdomain. he attacker can then exploit the vulnerability and control the subdomain to serve malicious content such as phishing sites and malware, thereby compromising the security and reputation of the organization.

Facial Spoofing is a term when the attacker impersonates the identity of another person by mimcking the facial features of an autherized user. This can be done if the system has no active livness detection to detect if the person persent in the current frame is a real person or a spoofed identity by deepfake, masks, filters or mannequins.

Human negligence is one of the most reasons to data breaches. Errors caused by humans is one of the easiest and most efficient attacks exists nowadays in the wild where technical knowledge is not required. By using what is called "social engineering", credentials can be extracted from users by luring them through fake emails, calls or by sending links that contain malware. Furthermore, users may leave

their token keys unattended, losing them or using weak credentials when creating accounts. There are unlimited possibilities on how a human, unintentionally can leak sensitive information or grant access to illegal parties. Therefore, security should never rely on users and MFA should always present[27].

2.2 Cryptographic tools for Confidentiality

Having good cryptography is the core of any good secure systems. Cryptography protects data from unauthorized access, ensuring confidentiality, integrity, and authenticity. It does so by converting the data into unreadable formats so that it can only be accessed by authorized parties. There are multiple types of techniques that can be used to convert the data such that it becomes difficult for any adversary to access it.

2.2.1 Symmetric Encryption

Symmetric encryption refers to the form of encryption where only a single cryptographic key is used for both encryption and decryption. Due to its efficiency and speed, it is widely used for protecting data in applications like database, secure file storage and network communications. Using the one secret key as shown in Figure 2.6, it converts the plain text (readable) into a cipher text (unreadable/encrypted). In the case of asymmetric encryption, the authorized parties must securely manage and protect the key[28].

Figure 2.6: Symmetric Encryption [28]

Depending on the required encryption strength, the length of the secret is decided as a first step in the process. The key length directly impacts security, longer keys make brute-force attacks much more difficult. After the secret key is generated, the plaintext is then transformed into a cipher text using it. There are various algorithms that can be used for converting the plain text. The same key is then used to decrypt the cipher text, which is fed into the algorithm, converting it back to the plain text. The algorithm either processes the data in block modes[29] or in bit-mode[30].

Block ciphers encrypt the data block-by-block. The block segments are usually of 64-bit or 128-bit. Each block in this mode is encrypted independently and it is linked together to provide additional security. AES(Advanced Encryption Standard) is an example of one such algorithms that operate in block cipher mode. There several ways that the block cipher modes can operate in. The basic mode called, ECB(Electronic Code Book) is discouraged because identical plain text blocks produce identical cipher text blocks, making it vulnerable to replay and frequency attacks. Other modes, CBC (Cipher Block Chaining)[31] and GCM (Galois/Counter Mode)[32], are better where an IV(initialization

vector) and Galois Field multiplication operations are used, respectively to enhance the security. In CBC mode, with the use of IV, the processing time increases since the decryption process has to start from the first block to maintain the chain. It is impossible to decrypt multiple blocks parallely. The GCM mode is even more advanced and it combines the CTR (Counter) mode along with the Galois field multiplication operation. Using a CTR mode, a counter value is used which is encrypted and incremented for reach block and then is XORed with the plain text, to produce the cipher text as shown in Figure 2.7. When combining this mode with Galois field, in GCM mode, it produces both cipher text and an authentication tag. This makes it more robust against tampering.

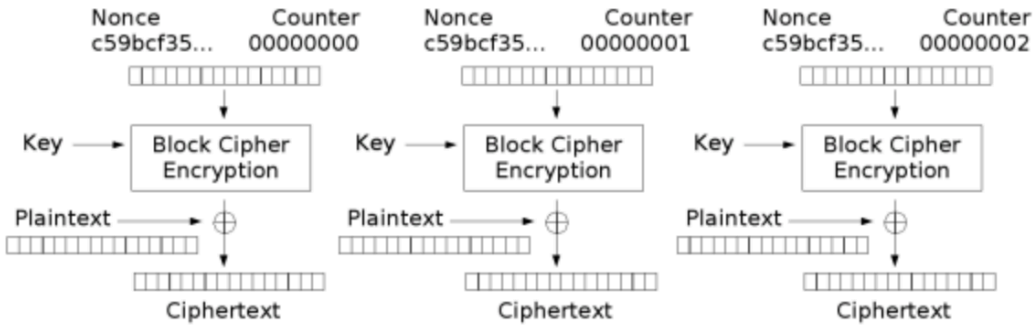


Figure 2.7: CTR Mode Encrpytion [33]

The Stream ciphers, rather than encrypting the data in blocks encrypts the data bit-by-bit. In this mode, a pseudo random stream of bits is generated which is XORed with the plain text bit-by-bit, producing the cipher text. The security of such symmetric encryption entirely relies on keeping the secret key securely and safely. When distributing the secret key, secure channels should be always used. Symmetric encryption provides confidentiality, but on its own, it does not ensure data integrity. However, modes like GCM include features to verify data integrity as well.

2.2.2 Advanced Encryption Standard

AES(Advanced Encryption Standard) is a symmetric encryption standard that uses a single key to encrypt and decrypt data. It is fast and highly efficient and was standardized by NIST(National Institute of Standards and Technology). AES can operate on different key lengths of 128, 192, or 256 bits (depicted in Figure2.8), but comes with a trade off for speed. The larger the key is used, the more slower the algorithm works[34].

AES is a block cipher(encrypts the data in blocks) and can run on three different modes: ECB, CBC, and GCM. It operates on the data in multiple rounds of encryption. In every round, data is converted in a way that makes it more difficult for anyone to decrypt, without the correct key. In the beginning of every round, from a predefined substitution box, each byte in the data block is substituted. After which every row is shifted cyclically. To add diffusion in the next step, all the columns are now mixed, later XORed with a round key derived from the original encryption key before another round begins. The algorithm is shown in Figure 2.9 and the detailed implementation can be found in Appendix A.2 [34].

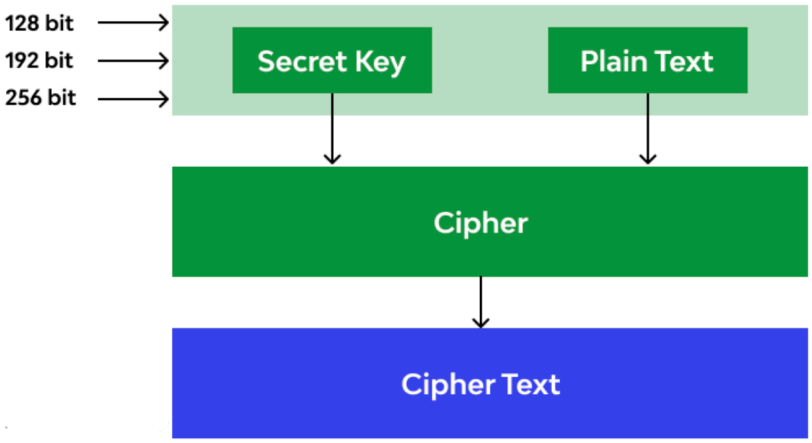


Figure 2.8: AES Design [34]

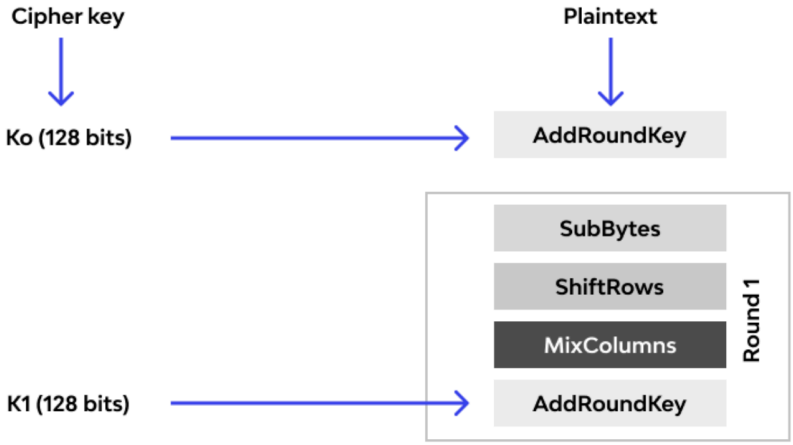


Figure 2.9: AES Encryption Process [34]

2.2.3 Password-Based Key Derivation Function

Password-Based Key Derivation Function(PBKDF) is a function that is used to generate cryptographic keys from passwords. In cases, where a cryptographic key is required, a user's password alone is seldom not sufficient for direct use as a cryptographic key. PBKDF acts as a stretcher for the password and makes it more resilient to brute-force attacks[35].

In the process of stretching the user's password, a salt is added to the password. This random salt ensures that for two similar passwords, the cryptographic keys generated are unique. Additionally, salt also helps to protect against precomputed attacks. Once the salted password is generated, it is passed through multiple iterations of a hash function such as SHA-256 to increase the computational cost for brute-force attacks as shown in Figure 2.10. Then the final output gives a cryptographic key that can be used in encryption. The complete algorithm is summarized in Appendix A.3[36].

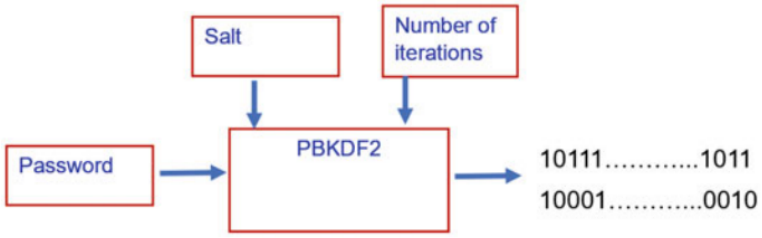


Figure 2.10: Password-based key derivation function [37]

When using PBKDF in systems where user passwords are stored as hashes in databases, it is essential to avoid excessively high iteration counts. This ensures the system remains efficient while maintaining security. Common recommendations suggest 10,000+ iterations, but it also depends on the application and the computing power[38].

2.2.4 Hash-based Message Authentication Codes

HMAC(Hash-Based Message Authentication Codes), unlike encryption which aims to conceal data, is used to verify that the data has not been altered and is from a correct source. It is a symmetric cryptographic hash function, like SHA-256, that combines the hash function with a secret key to create digital signatures[39]. In this thesis, HMAC will be used for implementing the time-based one-time passwords(TOTP).

The algorithm works by first taking the data to be authenticated and a secret key as an input. The secret key is generated randomly and is shared between the sender and the receiver. In the next step, a string cryptographic hash function, like SHA-256 is selected and the secret key is hashed or padded to match the block size of the chosen hash function. If the key is longer, it's hashed to the required length, and if the key is shorter, it is padded with zeros to the right. The algorithm proceeds with XORing it with two fixed constants: an inner padding(ipad) and an outer padding(opad) as shown in the Equation (2.2). In the equation, M is the data to be authenticated and K is the secret key. The final output, which is obtained by concatenating acts as a unique fingerprint[41].

$$HMAC(K, m) = H((K \oplus opad)||H(K \oplus ipad)||m)) \tag{2.2}$$

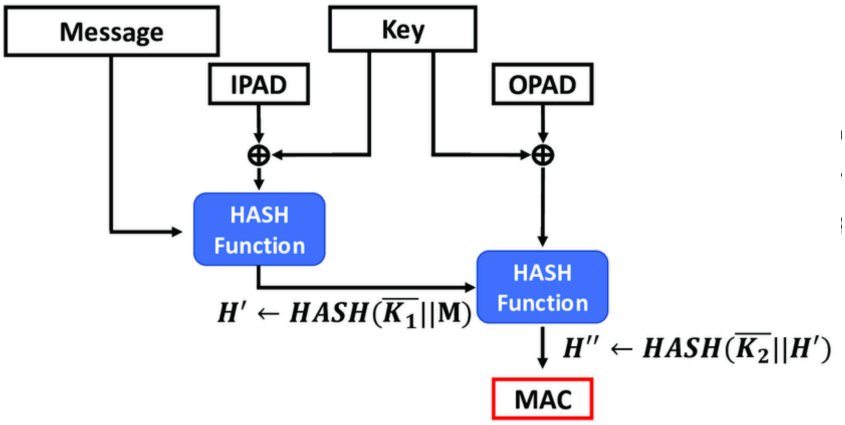


Figure 2.11: HMAC Algorithm on a high level[40]

2.2.5 Asymmetric Encryption

In this technique, two mathematically related keys are generated, namely public key and private key. Public key, as the name suggests is public and is shared openly. Unlike symmetric encryption, the public and private keys are different. The basic workflow is shown in 2.12.

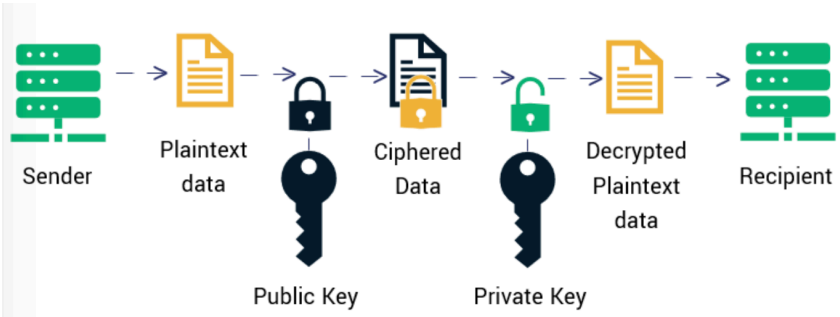


Figure 2.12: Asymmetric Encryption [28]

The plain text data is encrypted by the sender using the public key received from the receiver of the data. After receiving the data the receiver then uses the private key to decrypt. In a similar way, the digital signatures can be implemented using asymmetric encryption and is shown in the Figure 2.13. Here, the sender signs the data with their private key instead of using public key to encrypt, and creates a digital signature. The verification can be performed later using the sender's public key, ensuring both data integrity and authenticity[28].

A couple of common algorithms used are the Riverst-Shamnir-Adleman(RSA) and Elliptical Curve Cryptography(ECC). RSA is based on the mathematical challenge of factoring large numbers. For such reasons, keys are generally large, 2048 or 4096 bits, for providing strong security. But this can also affect the performance. On the other hand, ECC uses the properties of elliptic curves over a finite fields to create secure keys. ECC algorithms like Elliptic Curve Digital Signature Algorithm, provides similar security to RSA with smaller keys[42].

Elliptic Curve Digital Signature Algorithm(ECDSA) is a type of asymmetric encryption algorithm that utilizes mathematics of the elliptic curves over finite fields[44]. The math-

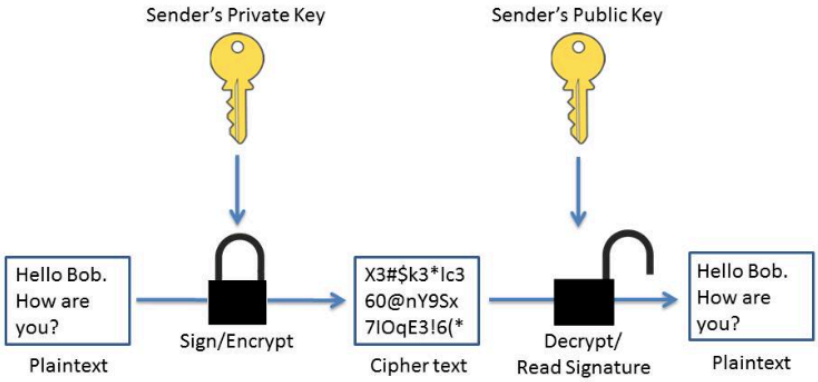


Figure 2.13: How Digital Signatures work [43]

ematical properties of elliptic curve allow strong security with significantly shorter keys than the RSA algorithm, thus improving the efficiency in terms of speed and storage. Smaller keys reduce the computational load, and thereby enables faster digital signature verification that is crucial in securing web communications[42].

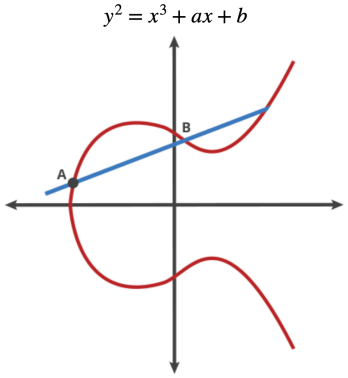


Figure 2.14: An example of Elliptic Curve [44]

The algorithm works by generating the private key and a public key using an elliptic curve. First, the private key is chosen randomly and kept secret. Now, using this private key and a base point defined by the elliptic curve, a public key can be generated. The process creates a point on the elliptic curve that acts as a public key. To sign a message, the hash is generated using functions like SHA-256, to generate a digest of fixed length. The signature is then generated using the hash, a nonce and the private key. This signature is unique to the combination of the message and the private key. The nonce ensures that in case the signing happens more than once, the signature will also be different. Finally, the receiver of the signed message uses the public key and the mathematics of an elliptic curve to verify the signature if it was actually generated by the correct private key. If the calculated value differs, the signature is rejected. Generally, some standard curves like sec256k1 or P-256 are used. which are widely trusted for their cryptographic properties and robustness[45]. It is important to note that not all elliptic curves provide a similar security.

Nonce Security, Elliptic Curve Selection, along with hash functions enables ECDSA algorithm to create unforgeable signatures, providing high efficiency in terms of performance

and space[46]. It scales well across devices with limited power, memory or bandwidth which is essential for applications that require high-performance security.

Asymmetric encryption is ideal for open networks where sharing of the private keys is risky. Additionally, it helps in data integrity by allowing the senders proves the data's origin by supporting digital signatures[47].

2.3 Python Cryptography Library

The python cryptography library provides a wide range of cryptographic primitives and algorithms to support encryption, decryption, hashing, key management, and digital signatures[48]. The library supports symmetric encryption such as AES using the *AES* module, key derivation functions such as PBKDF2 using the *PBKDF2HMAC* moodule, hashing algorithms and asymmetric encryption that includes public key cryptography and digital signature algorithms using the *ec* from *asymmetric* module. The python cryptographic library is easy to use, maintainable, compliant with modern security standards and ensures high performance[49].

2.4 Conclusion

The above sections outlines the authentication methods and the cryptographic tools integrated into the project to enhance security and authentication. HMACs are utilized as part of the TOTP mechanism to ensure the generated code is confidential and unique. TOTP provides an added layer of security. FIDO2 standard incorporates multiple cryptographic tools that will be utilized to achieve the same level of security as provided by FIDO2 standard. symmetric encryption algorithms like AES secures data through encryption to enhance the password security and ECDSA ensures integrity of data with secure digital signatures which is an essential part of the FIDO2 standard. Additionally, hashing algorithms like SHA-256 provide irreversible representations of sensitive information. These tools collectively establish a strong cryptographic framework for the project.

3

Facial Recognition

Facial recognition is a technology that leverages advanced algorithms to identify or verify users based on their facial features. This chapter explores the critical components that make facial recognition systems effective, beginning with the facial recognition pipeline, which breaks down the process from detecting and processing facial images to verifying identities. It then delves into the facial recognition libraries, highlighting the tools and frameworks that enable seamless implementation of these systems. Finally, the last section in this chapter discusses about facial recognition metrics, providing an understanding of how facial recognition systems are evaluated. Together, these sections offer a comprehensive guide to understanding and working with facial recognition technology.

3.1 Facial Recognition pipeline

Facial Recognition is one of the biometric methods through which authentication can be performed using the facial features of the person. Using this method, a camera is required to capture the face of the person. The captured image is then processed and is sent for analysis. The specific, unique features such as the distance between the eye, the shape of the nose, lips and jawline. The output is a mathematical representation which are called face embeddings. These embeddings are used later for verification. The facial recognition pipeline typically consists of four key phases: face detection, alignment and resizing, representation, and verification[50].

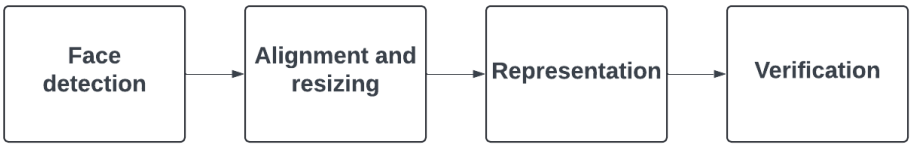


Figure 3.1: Typical Facial Recognition Pipeline

3.1.1 Face detection

In the first phase of facial recognition pipeline, the system uses a camera to identify and locate the face. Traditionally, Haar features are used to detect the face and identify its face unique features in real time. Viola-Jones algorithm is one example of facial detection algorithm that uses Haar features.

Haar features are sequence of rescaled square shaped functions. *(These are very similar to convolutional kernels found in CNN.)* [51] Example of Haar features are shown in 3.2.

There are edge features((1) and (2)), which can be used to detect the eyebrows, then there are line feature(3), that can be used to detect the lips. These algorithms can rapidly detect faces.

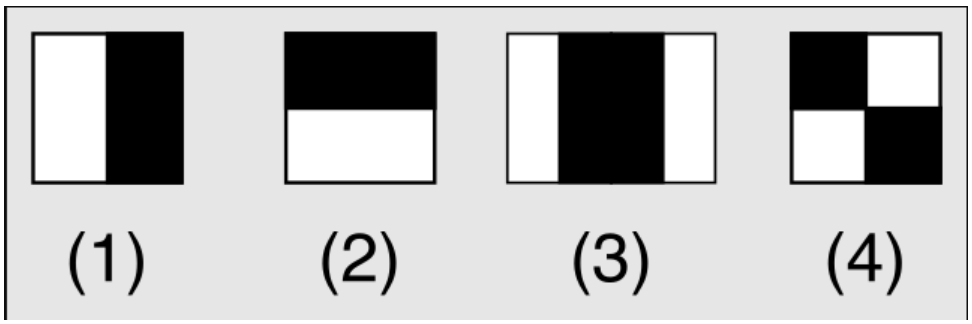


Figure 3.2: Example of Haar feature[52]

Apart from the traditional method, modern deep learning models also allows facial recognition. They are more accurate at recognizing facial emotions, orientations, and lighting situations because they employ Convolutional Neural Networks (CNN). They accomplish this by processing spatial hierarchies of features, including edges, forms, and intricate patterns, through a sequence of layers. Furthermore, CNNs can more precisely translate complicated facial traits into a consistent format after being trained on a large dataset of diverse faces in order to provide dependable embeddings[53].

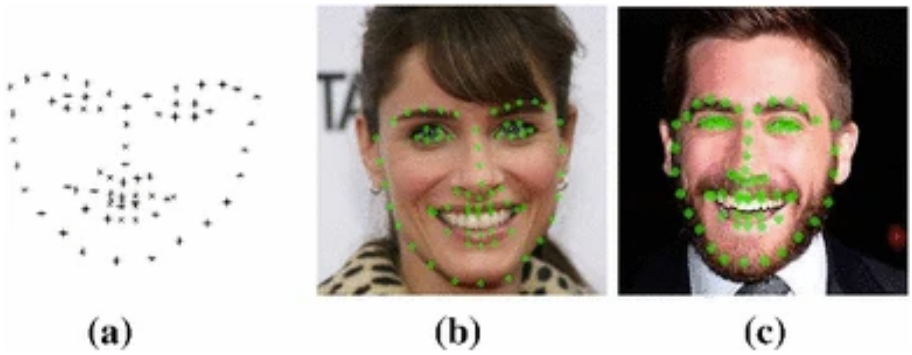


Figure 3.3: An example of facial landmarks coordinates being assigned to faces[53]

Face landmarks of particular features, such as the mouth, nose, and eyes, are frequently included in face detection to aid in subsequent stages. By giving each landmark a coordinate, this procedure makes it easier to align facial features consistently. The computing load for the subsequent stages is decreased by cropping the identified face out of the frame[50].

3.1.2 Alignment and resizing

In this phase, the image is rotated to align both eyes horizontally using the landmarks created in the other steps. To guarantee that the face traits are consistent between samples, the rotation is necessary. Trigonometry can be used to calculate the angle at which the image should be rotated. The formula for calculating the degree of rotation using trigonometry is shown in equation (3.1) [54].

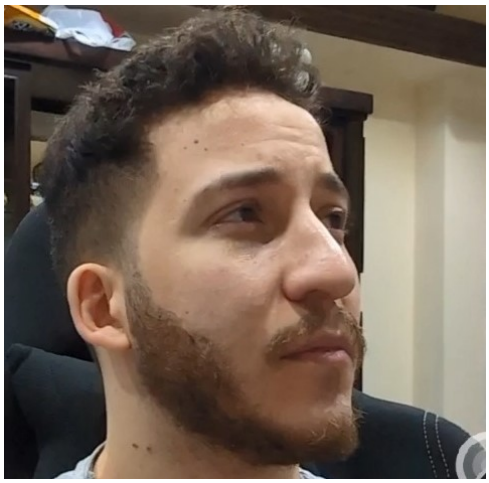
$$A^\circ = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \times \frac{180}{\pi} \tag{3.1}$$

Figure 3.4: Representation of a right-angled triangle stemming from the facial landmarks

Following alignment, scaling is carried out in accordance with the specifications of facial recognition systems. Images in Facebook's DeepFace model are usually shrunk to 152X152 or 224X224 pixels, depending on the model[55], but images in the Python face-recognition library are usually resized to 150X150 pixels. In the parts that follow, the two models will be covered in further detail. Although it requires more processing power, a higher resolution aids in collecting finer details of the facial anatomy.

3.1.3 Face Representation

In this step of the pipeline, facial embeddings are generated. The method for generating embeddings depends on the model used. For example, HOG (Histogram of Oriented Gradients) calculates embeddings by extracting gradient-based features, emphasizing edges and shapes, and is used in traditional models. In contrast, CNNs (Convolutional Neural Networks) generate embeddings by learning hierarchical features through multiple layers, capturing complex patterns such as textures and facial structures. The face embedding is a numerical vector as shown in Figure 3.5 representing the face. Different inputs yield different embeddings, allowing for efficient comparison.



$$embedding = [-0.08856192976236343, 0.1964293271303177, 0.03771832212805748, \dots]$$

Figure 3.5: The face in the image and its corresponding facial embedding list generated by Python Face-Recognition library

3.1.4 Face verification

In this last phase, pre-stored embedding is used to compare the user's face embedding to confirm or deny the identity. The comparison is based on the similarity or distances between embeddings to decide whether it is the same or different face. The lower the distance, the higher the similarity and vice versa[56].

The distance can be calculated using different metrics. Some of the common metrics includes, Euclidean Distance, Cosine similarity and Manhattan distance[56]. Euclidean Distance measures the direct distance between points in multi-dimensional space. In Equation (3.2), p and q represent the two face embeddings, resulting in a numerical value representing similarity.

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.2}$$

Cosine similarity uses the cosine of the angle between the embeddings instead of calculating the direct distance. It assesses similarity regardless of the magnitude. It is more effective when the embeddings have been normalized. While the Manhattan distance adds the absolute differences between embeddings to provide a different measure of similarity. Manhattan distance is also refered to as taxicab or city block distance. Manhattan distance provides an alternative to Eucledian distance, especially useful for higher-dimensional data[56].

Finally, after the distance is calculated, it is compared to a threshold. If the distance is below the threshold, verification succeeds, otherwise it fails. The threshold is usually selected by evaluating the verification system's accuracy. This should be optimized to balance the False Acceptance Rate (FAR)(allowing access to impostors) and False Reject Rate (FRR)(denying access to legitimate users). The Equal Error Rate(EER) point, is often chosen as an optimal threshold where FAR and FRR are balanced. A low threshold value reduces the FAR while a high threshold value lowers FRR[54]. The approach to set a threshold should be based on the specific use case requirements with usability

being taken into account. For a high security environment, a lower threshold may be preferred**cite {easyai2024facial_recognition}**.

```
distances = face_recognition.face_distance(known_face_encodings,
↪  face_encoding)

if distance < threshold
    verification succeeds
```

In this project, two facial recognition libraries will be tested, namely, Python Face-Recognition library and DeepFace, both of them are based on the Convolutional Neural Network(CNN) models.

3.2 Facial Recognition Libraries

Face recognition libraries simplifies the development process for integrating facial recognition capabilities. It is integrated with powerful pre-trained models like Histogram of Oriented Gradients (HOG), and Convolutional Neural Network (CNN) and is optimized for performance and accuracy. Modern software libraries provide end-to-end pipelines, including face detection, alignment and resizing, feature extraction and verification.

3.2.1 Python Face-Recognition

The Python Face-Recognition library[57] is a user-friendly library to do facial recognition tasks. Under the hood, it uses Dlib's deep learning-based CNN model. The Dlib library is written in C++ with Python bindings[58]. A python package appropriately wraps the dlib's facial-recognition functions into a simple, easy to use API. The library is open-source and is known for its extensive usage in computer vision, facial recognition, and object detection. It provides highly optimized implementations, thereby, boosting the performance efficiency.

The library offers two main models: HOG and CNN-based. The first one is faster as it uses gradient directions and edge analysis to detect the face. The latter one, is a deep-learning model which is more accurate but comes with the cost of computational power. The CNN model is based on the ResNet-34 architecture, and the model is trained to generate 128-dimensional face embeddings[59].

For face landmark detection, the library includes a model that can detect 68 facial landmarks, such as eyes, nose, lips and ears. These 68 features enhance the accuracy helping in correct facial alignment. Lastly, for comparison of the embeddings, the library utilizes the Euclidean distance to determine the similarity. Depending on the set threshold, the matches can be classified. The default model the library uses is based on HOG.

HOG is a feature descriptor used for object detection tasks such as face detection. Unlike deep-learning based methods, HOG does not rely on training neural network which makes it more computationally efficient. It is a lightweight, effective solution for detecting face features and is one of the models provided by Python face-recognition's library.

HOG works by analyzing the gradients. By focusing on gradients, it makes it more useful

for detecting faces across different light conditions. Gradients are the differences in pixel intensity and is based on the direction and intensity of pixel values. These gradients orientations are then compiled into a histogram, providing enhanced detail capture, which represents the distribution of edges and outlines within a region[60].

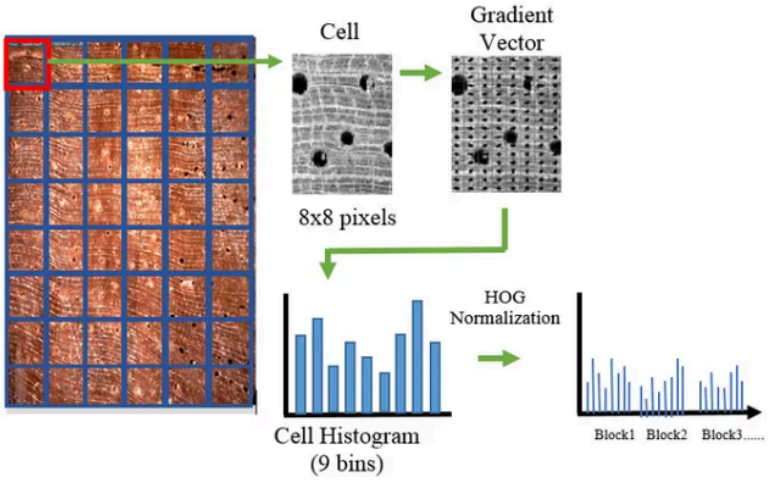


Figure 3.6: Steps to calculate HOG high-dimensional vector [61]

The HOG algorithm can be broken down into several steps as depicted in Figure 3.6. In the first pre-processing step, the image captured is standardized for the quality and the noises are reduced by improving the contrast and the lighting. After that, in the next step for the processed input image, the gradient direction and magnitude are calculated for each pixel and are grouped into bins. The image itself is divided into several small cells, which are further combined into blocks. Once this is done, normalization using blocks to ensure consistency across the image. Finally in the last step, the histograms from all the cells are concatenated to generate a unique high-dimensional vector (also called descriptor) that represents the overall structure of the image[62]. HOG maintains it own glossary with pre-existing descriptors in a database that helps to classify the image as "face" or "non-face".

3.2.2 DeepFace

Another Python library, DeepFace was developed by Facebook[63], offering an easy to use API to do facial recognition and analysis. The DeepFace library leverages state-of-the-art models for face recognition. By default, the library uses VGG-Face as its primary model for processing and recognition tasks. Each of the models has its own specific strength, making DeepFace a great choice for facial recognition-based applications and systems.

VGG-Face model uses a deep-learning CNN model for performing face recognition and is known for its feature extraction through multiple convolutional layers. Google's FaceNet uses distance metrics and creates a 128-dimensional face embeddings. Another library inspired by FaceNet, is OpenFace. OpenFace library is based on deep-learning neural networks as well. The first model to use deep learning for facial recognition, DeepID, is another pioneering model that is specialized in using multiple hidden layers to learn highly discriminative features for user verification. The last, ArcFace, as the name suggests uses the angular distances in the embeddings to improve the distinction between similar faces.

DeepFace integrates all the above mentioned pre-trained models architectures to generate face embeddings, offering more flexibility[63]. In the DeepFace face recognition pipeline, the algorithm begins with the face detection and alignment where it uses MTCNN(Multi-Task Cascaded Convolutional Networks) or OpenCV's face detector. OpenCV's face detection offers two choices for face detection, naemly, Haar features and deep-learning while MTCNN is based on deep learning solely. MTCNN identifies faces by using three stages of cascaded CNNs. Cascades are just like smaller CNNs in sequences[63].

After alignment, DeepFace sends the image through a deep learning models(such as VGG-Face, FaceNet, etc) to create face embeddings[64]. Finally, for the verification step, embeddings are compared utilizing distance metrics such as Euclidean, Cosine Similarity or Manhattan distances. The calculated distances are compared to the set threshold. The deep learning models mentioned before uses a specialized type of neural network called Convolutional Neural networks (CNNs) that are designed to process structured data like images and videos.

For doing so, CNNs processes the image in multiple layers as shown in Figure 3.7. The first core layer which are the *Convolutional Layers*, apply filter to an image to detect patterns and complex features. Each filter, or kernel, slides over the image, performing a dot product highlighting specific features. Other core layers like *Activation Layers* and *Pooling Layers* introduce non-linearity and reduce the dimensionality, respectively. This allows the neural network to learn more complex features and also making it more computationally efficient. Toward the end, *Fully Connected Layers* processes the extracted feature to generate the face embeddings[65].

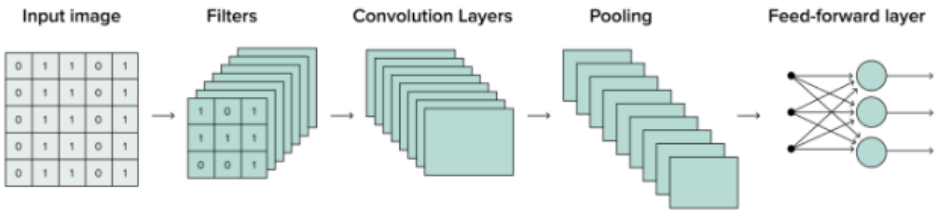


Figure 3.7: 3 layers of CNN [66]

ResNet (Residual Network) is one such example of a CNN architecture. This model is known for its residual connections where generally one or more layers are skipped[67]. This allows the model to learn the differences more easily without losing information or getting stuck during training. ResNet-34 for example, has 34 layers and uses complex feature extraction to produce the face embeddings that capture unique facial features. In libraries like DeepFace, Res-Net models are used to extract facial features and generate embeddings.

Another CNN architecture designed for real-time face detection and alignment is the MTCNN (Multi-Task Cascaded Convolutional Neural Network).This model uses a cascaded structure with three CNNs[68]. It operates in three stages. In the first stage, the image is scanned and the non-face regions are discarded. The second stage works more on the refinement for the regions extracted in the first stage. In the final stage, the facial landmarks are identified. In addition to the detection, MTCNN also allows the model to align the face. MTCNN's cascased structure makes it computationally more efficient to detect and align faces in real-time.

Convolutional Neural Networks (CNNs) have become a foundational technology in facial recognition systems. CNNs are a specialized type of artificial neural network which is suited for processing grid-like data, such as images. It analyzes the image of a face, extracts useful information, and creates an "embedding", a multi-dimensional vector that is a unique representation of the face.

Along with facial recognition, DeepFace also offers functionality to detect age, gender and emotions. The library is as a result quite versatile and have pretty good accuracy with minimal tuning owing to its pre-trained networks. It also includes an anti-spoofing module to understand whether a given image is fake or real[63]. With such features, DeepFace becomes computationally intensive on the other hand. It can face challenges with requiring GPU's, or occlusions under low-light settings.

3.3 Facial Recognition Performance Metrics

Evaluating the performance of face recognition models is essential to understand their accuracy, reliability, and suitability for real-world applications. Each application may prioritize different aspects of performance, such as reducing false positives in security systems or improving detection accuracy in image tagging. There are several different metrics that can be used for such facial recognition models [69].

A **True Positive (TP)** occurs when the model correctly matches the face to the correct person. A high TP means that the model is able to identify or detect faces accurately. The face in Figure 3.9 is correctly identified with matching face in Figure 3.8.

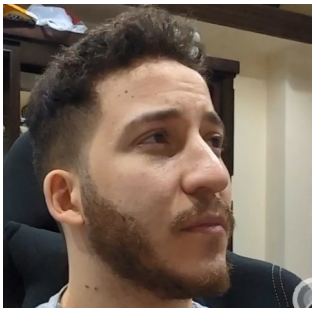


Figure 3.8: Correct face

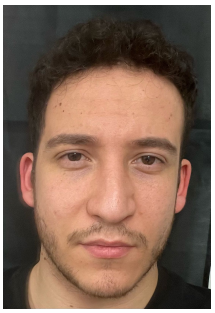


Figure 3.9: Correctly matched face (TP)

Figure 3.10: Incorrectly matched face (FP)

False Positive (FP), as opposed to TP, incorrectly identifies or detects the face when there is none. High FP can lead to security risks, as the model for which the metric is calculated can grant unauthorized access. An example of the TP scenario is depicted in Figures 3.8, 3.9 and 3.10. Here the facial features like a beard, lighting or facial orientation are getting mistakenly recognized as matching by the model.

True Negative (TN) is another metric which occurs when a model correctly identifies that there are no faces in an empty region. In other words, a TN would mean the model has accurately rejected an unregistered face. High TN is good as it indicates that the model is not mis-classifying non-faces or unknown faces[70].

The fourth metric, **False Negative (FN)** occurs when the model fails to recognize a known person. Having a high FN would essentially mean that the legitimate individuals

accesses are denied.

For representing all the above four metrics a Confusion Matrix is used shown in Figure 3.11. Each cell in the matrix represents TP, FP, FN and TN. Combining all the above four metrics gives a total number of predictions made by the model. And using the TPs and TNs along with the total predictions, the Accuracy can be calculated as shown in (3.3)[71]. It gives an overall sense of how the model performs but may not be suitable for imbalanced datasets where TPs and TNs dominate over FPs and FNs.

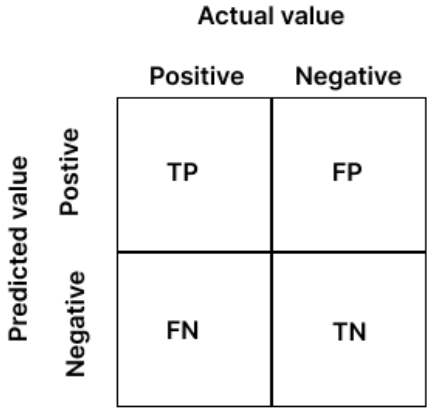


Figure 3.11: Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

Lighting robustness and pose robustness can be ascertained using the accuracy[72]. Lighting robustness refers to a model's capacity to perform well under various lighting conditions, including bright sunlight, dim light, and shadows. Real-world surroundings often offer lighting abnormalities that can change features such as highlights, shadows, and skin tone. Accuracy is calculated as the proportion of faces successfully identified in each lighting condition. Lighting robustness is evaluated by comparing accuracy in these scenarios. Conversely, pose robustness evaluates a model's ability to recognise faces in a range of head orientations, such as side views, tilts, and rotations. Postural changes can obscure facial features, making recognition more challenging. In this instance, accuracy is calculated as the percentage of faces that are properly detected for each pose. Occlusion handling[73] is another helpful statistic that assesses a model's capacity to identify faces that are partially hidden by hands, masks, or other environmental elements. In these situations, accuracy is calculated as the ratio of successfully detected faces with occlusions to faces without occlusions.

Apart from these, **Expression handling** is yet another useful way to assess the models. Expression handling evaluates a model's ability to identify faces with various facial emotions, including grins, frowns, and astonishment. The geometry of facial landmarks can be changed by expressions, making recognition more difficult making this a good indicator to assess the strength of the models. **Precision**, **Recall**, and **F1-score** are often used to evaluate expression handling, as they provide a more nuanced view of the model's performance across multiple expression classes. **Recall** uses the TP and FN, and can be

calculated as shown in equation (3.4)[71]. A low Recall indicates that the FN is high. If FP are used instead of FN, then the value which is known as **Precision** can be obtained. It will give the ratio of TPs to all predicted positives. The formula to calculate is shown in equation (3.5)[71]. The precision metric is also crucial in access control applications where mis-classifying error could lead to security breaches. High precision indicates that the model's prediction are mostly correct.

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

It is essential to balance recall and precision. For doing so, **F1 - Score** can be used as an analysis metric, providing a single measure of performance when both FP and FN are equally important. The F1-Score can be calculated as shown in equation (3.6)[71]. This metric is particularly useful when there is an uneven distribution, like more negatives than positives.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision * Recall} \tag{3.6}$$

For the evaluation of all these metrics, various kinds of datasets are used and Table A.7 provides a comprehensive summary of some of the datasets available for testing the models.

3.4 Conclusion

The topics discussed in this chapter focuses on the details of the implementation and evaluation of facial recognition systems using two libraries: Python Face-Recognition and Facebook's DeepFace. These libraries are utilized to analyze and compare their performance in recognizing faces. FIDO2 standard will be mimicked to carry out the authentication analysis performed by using these facial recognition models. The evaluation metrics discussed is performed by using multiple facial recognition metrics, including True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). In addition, metrics such as precision, recall, and the F1 score are used to provide a comprehensive assessment. Together, these libraries and metrics offer a detailed analysis of the effectiveness of facial recognition in various scenarios.

4

Methodology

This chapter provides a high-level overview of the environment architecture. The system consists of three layers, including the client application, the relying party, and the authenticator. The interactions between these layers will be outlined, as well as how each layer contributes to creating a secure and seamless authentication environment.

4.1 System Architecture

The access control system prototype developed in this project relies on public key-based authentication, inspired by FIDO2 standard. Where a challenge is signed by the authenticator using a secured private key to prove the authenticity of the user. The prototype emulates two ceremonies similar to those found in FIDO2 standard, specifically the registration and authentication processes. The difference lies in the communication approach between the authenticator and the relying party. The communication between the parties is facilitated through HTTPS requests where the exchange of data is done in JSON format instead of relying on the browser to coordinate WebAuthn and CTAP2 calls. To ensure that the system aligns with the FIDO2 standard, even though WebAuthn and CTAP2 are not utilized, the implementation of this prototype mirrors the creation and management of passkeys as accurately as possible in addition to several major security mechanisms. This includes domain comparison with the relying party ID, using Options structure as defined by FIDO2 standard (see A.1 for more details), utilizing ECC for public-private key pair generation and implementing proper mechanisms to secure the private key. The face recognition authentication system is integrated into the authenticator as an additional layer of security and is used prior to any signing process. Additionally, TOTP is integrated into the authenticator to provide further security mechanism, particularly for making more secure decisions under specific conditions. The authentication and Registration ceremonies are explained below.

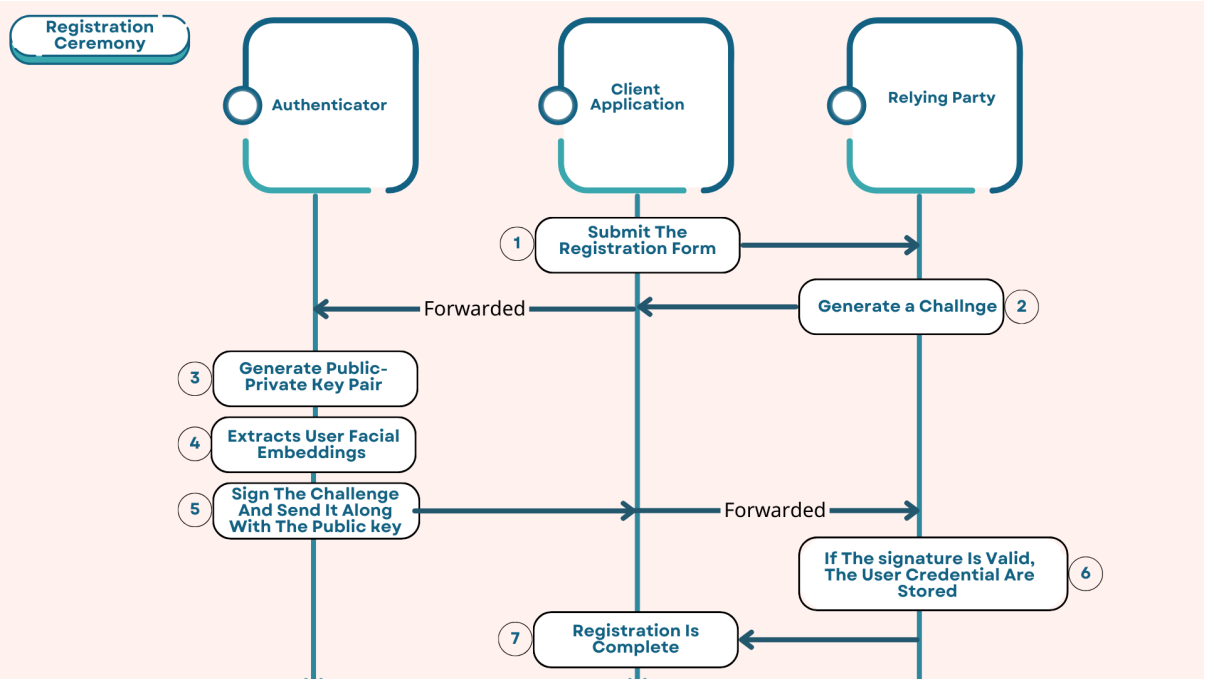


Figure 4.1: Registration Ceremony Flow

- Registration Ceremony

Step 1 The user initiates the registration ceremony on the relying party by interacting with the client application.

Step 2 The relying party generates a challenge and sends it back to the client application, which then forwards it to the authenticator.

Step 3 The authenticator generates a public-private key pair specific to the user.

Step 4 The authenticator extracts the user facial embeddings and saves them along with the private key on the database.

Step 5 The authenticator signs the received challenge and sends it, along with the corresponding public key, to the client application, which then forwards it to the relying party.

Step 6 The relying party checks the authenticity of the public key by validating the signed challenge.

Step 7 If the signature is valid, the registration ceremony is complete and the new account is added to the database

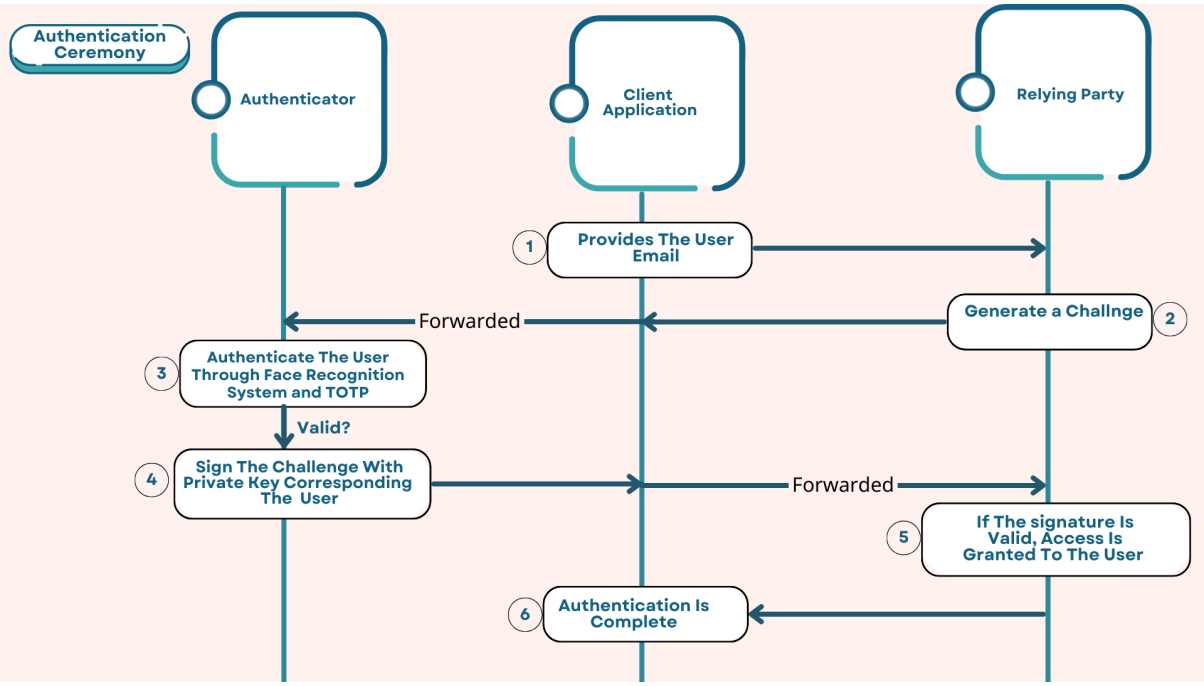


Figure 4.2: Authentication Ceremony Flow

- Authentication Ceremony

Step 1 The authentication ceremony begins when the user provides their email in the client application, which is then forwarded to the relying party.

Step 2 The relying party generates a challenge and sends it, along with the users email, to the client application, which is then forwarded to the authenticator.

Step 3 The authenticator authenticates the user through the face recognition system by extracting user's facial embeddings and comparing them to the faces in the current frame.

Step 4 If the user is authenticated, the received challenge is signed using the user-specific private key and sent back to the client application, which then forwards the signed challenge to the relying party.

Step 5 The signature is validated using the users public key.

Step 6 If the signature is valid, the user is granted access to the requested resources

4.2 Client Application

The client application serves as the front-end interface, that interacts with the users and the middle layer between the relying party and the authenticator. In this prototype, the primary use of the client application is to collect the necessary data from the users to complete the authentication and registration ceremonies. The client application depends on the relying party to make access decisions. Both the registration and authentication ceremonies are initiated from the client application through an interaction by the user.

The registration ceremony begins when the user submits the web form on the client application. This action sends the submitted information to the relying party and starts the registration process. For the authentication ceremony, the user only needs to provide their email which is forwarded from the client application to the relying party for further processing. The client application manages communication between the relying party and the authenticator, handling the exchange of data whenever necessary as visualized in fig 4.2 and fig 4.1.

4.3 Relying party

The relying party is the back-end application on the server side which facilitates the authentication and registration ceremonies. When a signed challenge is retrieved from the authenticator via the client application, the relying party verifies the authenticity of the signature and grants access accordingly. Similar mechanism is applied in registration ceremony to verify the genuinity of the received public key. The overall design of the relying party is divided into three components:

1. Application Layer: The application layer handles the core business logic between the rest of the components and the client application. This component offers an API which can be invoked by the client application to conduct specific actions. Its main responsibility is to coordinate the registration and authentication ceremonies by retrieving the user credentials from Credential Repository component and forwarding it to the Identity Provider.
2. Credential Repository: This repository stores the users data including the user email and the public key which were sent to the relying party during the registration process. This repository can operate by only storing the user email and its corresponding public key.
3. Identity Provider: This is the entity which grants user access based on the authenticator response. Token-based access can be used here but a basic customized solution is implemented instead since this is not the core focus of this project.

4.4 Authenticator

The authenticator in this prototype is a software-based face recognition system. It is responsible for generating the public-private key pair when the registration ceremony is initiated by the relying party side. The security of this prototype relies primarily on this layer, as it is responsible for managing and securing the private key. It is important to note that no Trusted Platform Module or similar hardware was utilized to store

users' credentials, as this prototype is intended to be integrated on a special USB stick with its own hardware designed to handle cryptographic operations in future work. The architecture of the authenticator is structured into five main components.

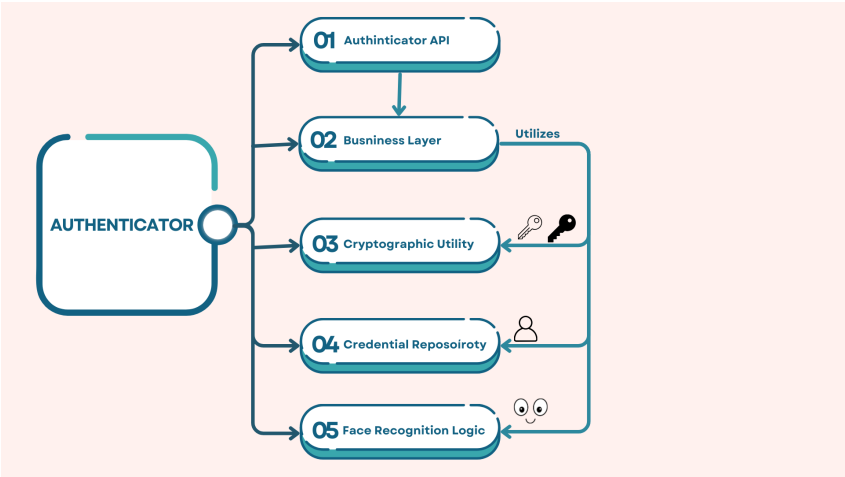


Figure 4.3: Authenticator Components

1. Authenticator API: This component provides an API endpoint that handles the interaction with the client application which is initiated by the relying party. It hides the actual implementation of the authenticator to simplify the interaction. It allows the client application to trigger the registration and authentication ceremonies on the authenticator side through the business layer

2. Business Layer: similar to the application layer in the relying party, this component handles the coordination between the internal processes within the authenticator. It manages the interaction between the sub-components to ensure that the ceremonies flow is executed seamlessly.

3. Cryptographic Utility: This component compensates the absence of the TPM to perfume essential cryptographic operations required for secure authentication. It handles the public-private key pair generation based on ECC, generating cryptographic keys, encryption and decryption of the private key and digital signature of received challenges. This component utilizes Python Cryptography library which adhere to industry standard to ensure integrity and confidentiality

4. Credential Repository: This repository interacts with the database to securely store and retrieve user credentials such as private keys. Unlike the relying party credential repository, this component should be highly secured. A leak of this repository exposes the private key which can lead to compromising the whole authentication environment.

5. Facial Recognition Logic: This core module is responsible for executing facial recognition processes to decide whether the user should be granted access. It is also responsible for picture capturing, extracting the face embeddings and forwarding them to be stored on the database and to be used for authentication in the future.

6. TOTP: This component adds an extra layer of security to the authentication system, especially considering that the current prototype has limitations in terms of dedicated hardware for cryptographic operations and an advanced camera. It will

be responsible for sending an OTP to the user's email once the user is authenticated through face recognition.

4.5 Summary

The proposed prototype architecture consists of three main components; the client application, the relying party, and the authenticator. The client application interacts directly to initiate registration and authentication ceremonies and acts as a channel between the relying party and the authenticator. The authenticator, which is based on a face recognition system, handles the key pair generation and signing procedure of the challenges provided by the relying party to complete the authentication process. TOTP is integrated within the authenticator as an additional layer of security to be used under certain conditions. The relying party on the server side, verifies the signed challenges during both ceremonies to grant access to the requested resources. The system is designed to ensure secure and seamless user authentication despite not having a dedicated hardware for cryptographic operations such as Trusted Platform Module. FIDO2 standard was used as an inspiration to enforce a robust public key-based authentication system.

5

Implementation and Results

In this chapter, the low-level implementation of the system is described with a focus on the authenticator side. It is important to note that this implementation does not utilize the WebAuthn API and CTAP2, and the interactions between the parties are conducted using HTTPS calls. In addition to the implementation details, this chapter also presents the outcomes of the designed and implemented face recognition-based access control system. The functionality of the system will be over-viewed, as well as its adherence to FIDO2 standard and its ability to provide secure and password-less authentication. Furthermore, the result of the metrics analysis will be introduced, alongside an overview of the threat model.

5.1 The System Workflow

The prototype is working seamlessly in both registration and authentication ceremonies. When the registration ceremony is initiated, a pop-up on the user device is shown to get the confirmation of capturing 6 pictures and converting them into embeddings. If the user's face is not present in the frame or more than 2 faces are detected, the system pauses and no pictures are captured until a face appears in the frame. When the embeddings are successfully created, the user's credentials are stored on the relying party and the authenticator side. The user can login to the service and initiate the authentication ceremony by providing their username on the client application. If the user exists in the database, the camera on the user side starts capturing frames of the user. The user should blink at least twice in 5 seconds to prove its liveness. additionally, the similarity score of the detected face should exceeds the integrated threshold to be granted access which is 70 percent. The environment, face expressions or angles might affect the similarity score. Therefore, if the similarity score is between 65 and 70, the user will get an OTP on the email as an additional security mechanism.

Figure 5.1: Login and sign-up pages

5.2 Relying Party

The core responsibility of the relying party is to generate challenges to be signed by the authenticator and to verify whether the signature is valid or not, to grant access accordingly. A relying party object is created utilizing Python FIDO 2 library which then can be used to generate (*PublicKeyCredentialCreationOptions*) object as described in A.1. The Options object is a way to standardize and facilitate the interaction between authenticators and the relying party through the client application, ensuring compatibility and a consistent process for registration and authentication ceremonies. As mentioned before, this implementation does not use WebAuthn API or CTAP2 and the usage of Options structure is to ensure that the core features are integrated in this prototype without relying on the browser. The creation of the Options object is introduced below.

```
rp = PublicKeyCredentialRpEntity(name=local_host, id=local_host)
server = Fido2Server(rp)
    options = server.register_begin(
        {
            "id": user_id.encode('utf-8'),
            "name": user_name,
            "displayName": user_name,
        },
        user_verification="preferred"  # Adjust based on your security policy
    )
```

During the registration ceremony, the relying party creates the Options object. The object is sent to the authenticator via the client application to be signed using a private key generated by the authenticator. The authenticator signs the received challenge and sends it back along with a public key corresponding to the private key used for the signing process, which is specific to the current user. When the relying party receives the signed challenge and the public key, the public key is verified using a verification function from the cryptography library, specifically designed for ECC. If the public key is valid, a new account is registered on the database and the public key is stored along with the user credential; otherwise, it is ignored.

```
    try:
```

```
        public_key.verify(signature, challenge, ec.ECDSA(hashes.SHA256()))
        print("Signature is valid.")
        return True
    except Exception as e:
        print("Signature is invalid:", e)
    return False
```

The same function and similar process are followed in the authentication ceremony. The relying party generates a challenge when the user initiates the authentication ceremony on the client application. The challenge is sent to the authenticator via the client application to be signed. The relying party then verifies the authenticity of the signature using the user-specific public key. If the signature is valid, the user is granted access; otherwise, the request is ignored.

5.3 Authenticator

The integration of the authenticator and how it fulfill its role in the authentication and registration ceremonies will be outlined here. The authenticator in this prototype is software-based, developed with a focus on secure authentication mechanisms and reliable face recognition. This section describes how the authenticator interacts with the relying party via the client application as a middle layer. It outlines the cryptographic operations done to achieve security in a password-less environment, relying on public-private key authentication and face recognition system. This overview demonstrates how the design is practically realized in a software prototype, emphasizing modularity, security, and adherence to cryptographic standards.

5.3.1 Cryptographic Operations

When the response is received by the authenticator, it compares the domain of the client that sent the request with the relying party ID included in the request. If they do not match, the request is ignored. Otherwise, the authenticator generates a public-private key pair using ECC with the SECP256R1 curve. The private key is immediately encrypted using AES, with an encryption key derived through PBKDF. In this prototype, the password that is fed to PBKDF is derived from the USB unique information such as serial number and identifier due to the absence of the necessary hardware to handle secure cryptographic operations and management.

```
        private_key = ec.generate_private_key(ec.SECP256R1(),
        ↪  default_backend())
        public_key = private_key.public_key()
```

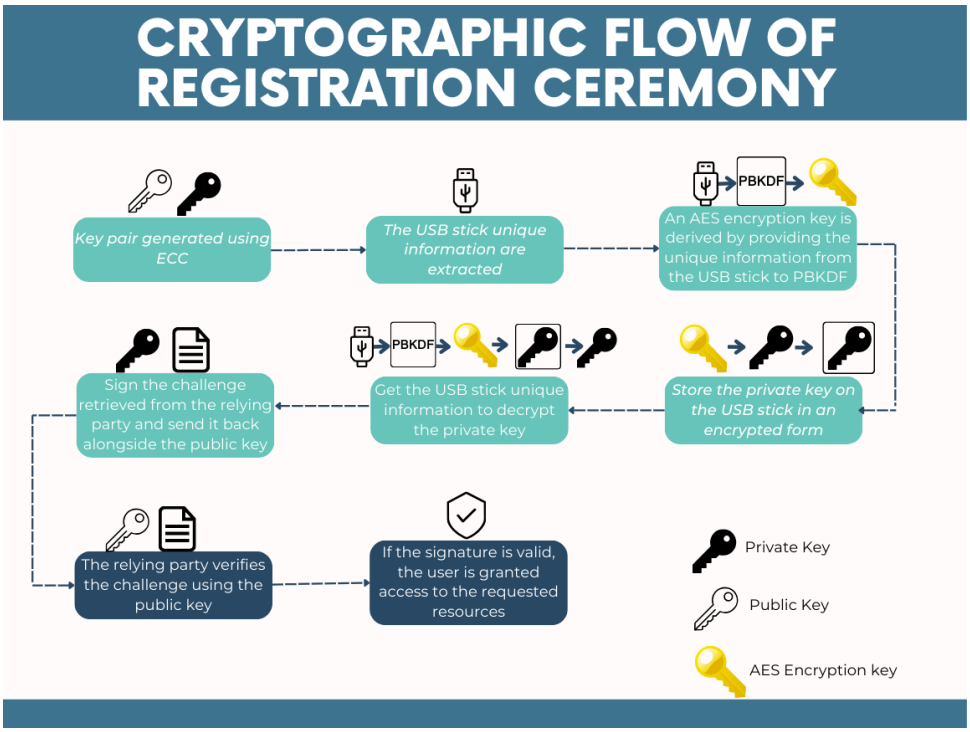


Figure 5.2: Cryptographic flow of the registration ceremony

Once the credentials are successfully retrieved, the authenticator initiates the process of extracting facial embeddings of the user. Following this, the private key is decrypted to sign the challenge received from the relying party. The decryption of the private key is done using the same procedure done to encrypt it since symmetric algorithm is used for encryption. Note that the password used in PBKDF to generate the key for AES is generated using the USB unique information and this is just for demonstration purposes as it relies on information set by the manufacture which can be accessed or used. Once the challenge is signed, it is returned along with user data and its corresponding public key to the relying party for verification.

```
    pbkdf_key = generate_pbkdf(USB_information,salt)

    # Create a cipher object using the provided key and IV
    cipher = Cipher(algorithms.AES(pbkdf_key), modes.CBC(iv),
    ↪  backend=default_backend())
    decrypter = cipher.decryptor()

    # Decrypt the ciphertext
    padded_data = decrypter.update(encrypted_private_key) +
    ↪  decrypter.finalize()

    # Remove padding
    unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
```

```
    decrypted_private_key = unpadder.update(padded_data) +
    ↪  unpadder.finalize()

    return decrypted_private_key
```

The authentication ceremony on the authenticator begins when the relying party forwards the user credential along with the challenge. The authenticator prompts the user for facial recognition verification to confirm the legitimacy of the user. The facial recognition system analyzes the user's face and calculates a similarity score to determine access eligibility. If the similarity score exceeds the threshold, e.g. 70 percent, access is granted directly. If the score did not reach the threshold and ranged between 65 and 70 percent, an OTP will be sent to the user email from the authenticator as an additional layer of security. The threshold can be adapted to satisfy the security requirements of the system. Once the user successfully verifies their identity, the private key is decrypted and used to sign the challenge which is sent back to the relying party for validation. Upon successful verification of the signed challenge, the user is granted access.

5.3.2 Facial Recognition Setup

The following section describes the setup for a lightweight facial recognition model, python face-recognition and a more sophisticated library, namely, deepface.

5.3.2.1 Python face-recognition

Frames are continuously captured from the video stream using a while loop implemented with the OpenCV CV2 library, ensuring real-time processing of each frame for subsequent operations. The colors format is converted from BGR which OpenCV supports to RGB format which is normally used in machine learning operations. For faster processing, the images are resized to be 0.3 of the original size. The face locations are detected first and forwarded to extract the face embeddings. The model used to detect faces can be determined by changing the parameter in the face_locations function.

```
    # Resize frame for faster face recognition processing
    small_frame = cv2.resize(rgb_frame, (0, 0), fx=0.30, fy=0.30)
```

```
    face_locations = face_recognition.face_locations(small_frame, 1,"CNN")
    face_encodings = face_recognition.face_encodings(small_frame,
    ↪  face_locations)
```

Once the embeddings are extracted from a captured frame, the frame is scaled back to enhance the visualization experience of the user while attempting to get access. Following this, the user embeddings are retrieved from the database to determine the access eligibility of the user. During this step, the embeddings stored from the registration ceremony are retrieved and iterated over and compared with the face embeddings which is present in the current frame. To calculate the similarity score, face_distance function is utilized to calculate Euclidean distance between the stored embeddings and the embeddings the current frame. The distance is then negated to derive the similarity score. If the score exceeds 70 percent, the user is granted access directly. However, if the score is between 65 and 70 percent, an OTP will be sent to the users email as an additional verification

step to confirm that the person presenting the authenticator is indeed authorized.

```
        distances = face_recognition.face_distance(known_face_encodings,
        ↪  face_encoding)

        min_distnce = np.min(distances)

        similarity_score = 1 - min_distnce
        if min_distnce > 0.70:
            similar_face.append((user_account, f"{similarity_score:.2f}"))
        elif 0.65 < min_distnce < 0.70:
            similar_face.append(("OTP",f"{similarity_score:.2f}"))
        else:
            similar_face.append(("UNKOWN",f"{similarity_score:.2f}"))
```

During the biometric authentication process, liveness detection is also necessary. It ensures that only genuine, live persons can authenticate. That means that the facial recognition systems are strengthened and made more robust. Liveness detection prevents common attacks like using printed photos, digital images or videos that can be used to spoof the system. It adds another layer in the press of face detection that verifies the realness of the face. There are multiple live detection techniques. One of the basic techniques involves detecting specific actions from the user, like blinking, or smiling, or analyzing the skin texture to confirm the liveness. Other advanced methods use infrared sensors, to even detect blood flow in the face.

To detect if the person has blinked, the Eye Aspect Ratio (EAR) is a commonly used metric. The EAR is calculated based on six eye landmarks, denoted as p1 through p6 as shown in Figure 5.3. These landmarks represent key points on upper and lower eyelids, and in the corners. The ratio is then calculated for vertical to horizontal distances. The formula[74] is shown in Equation (5.1). ||p2-p6|| and ||p3 - p5|| are the euclidean distances between the vertical eye landmarks while, ||p1-p4|| is the horizontal Euclidean distance. When the EAR value remains relatively stable, it indicates that the eye is open. When the eye is closed, the vertical distance between the eyelid decreases[75]. The interval used in this implementation is 2 blinks in every 5 seconds since a human blink about 15 to 20 times per minute. It is important to note that the frame processing is performed only once in each five frames to enhance system speed.

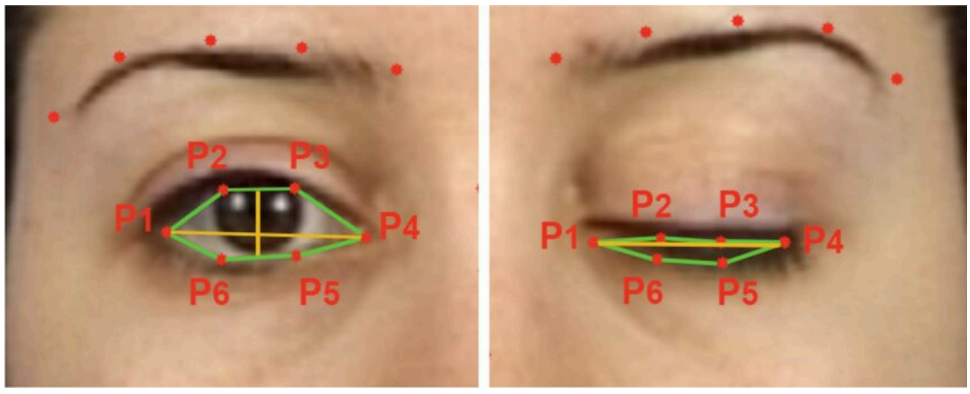


Figure 5.3: EAR calculated from the facial points p1, p2, p3, p4, p5 and p6 [74]

$$EAR = \left(\frac{||p2 - p6|| + ||p3 - p5||}{2 \times ||p1 - p4||} \right) \tag{5.1}$$

5.3.2.2 Deepface

Similar to python face recognition library, openCV library is used for the frame capturing. Deepface library can handle the alignment, colors and resizing. Before the facial embeddings can be extracted, a check is performed using the extract_faces method. The function detects, extracts, and preprocesses faces from an image. The detector_backend parameter specifies that a Multi-Task Cascaded Convolutional Network (MTCNN) algrorithm is used to locate faces in the image. This method can also be used to verify the presence of multiple faces in an image.

```
faces = DeepFace.extract_faces(img_path=image_path,
↪  detector_backend='mtcnn')
```

After the face is successfully detected, the next step is to extract facial embeddings from the image. The face recognition system utilizes the represent fucntion from the Deepface library. The model_name specifies the pre-trained model to be used for generating the facial embedding. The cropped face is passed through the specified model (example, Facenet) and it outputs a 128-dimensional vector that represents the unique features of the face.

```
face_encoding = DeepFace.represent(img_path=image_filename,
↪  model_name="Facenet", detector_backend="mtcnn")
```

Finally, to compare embeddings, euclidean distance is calculated using the scipy package from python. Deepface also analyses the emotions but liveness detection is not offered. The blink eye detection is also used with Deepface to check the liveness.

```
euclidean_distance = scipy.spatial.distance.euclidean(face_encoding,
↪  known_encoding)
```

5.3.3 OTP Generation and Validation

The TOTP is implemented on the authentication side utilizing the Pyotp library. The process is invoked when the similarity score of the face in the current frame is between 65 and 75 percent and liveness detection layer is bypassed. The integration of this security layer is simple, a hashcode is generated and sent to the user's mail which then must be entered by the user to be verified and gain access to the requested resources. The generation and verification of the hashcode can be easliy done using functions provided by PyOTP.

```
#OTP object is created with an interval of 60 seconds
totp_object = pyotp.TOTP('base32secret3232',interval=60)

#Generating the otp code
```

```
otp_code = totp_object.now()

#Verfying the OTP code
totp_object.verify(otp_code)
```

To be able to send the OTP to the user's email, a Gmail account is created with an app-passowrd configuration which allows the email to be accessed via third-party applications.

```
    from_email = "fido2zx@gmail.com"
    password = "qurc ejlo dhor nqaa"
```

The email body, as well as the Multipurpose Internet Mail Extensions (MIME) are created and then sent via the SMTP server to the target emails. The authenticator prompts the user to enter the received OTP which is valid for 60 seconds. if the OTP is valid, the challenge is signed and the user gains access from the relying party.

5.4 Threat Model

The authentication system is a core component in any system where sensitive resources should be secured. Given its critical function, it is essential to identify, assets and resolve the issues which can expose the integrity, confidentiality, or availability to be compromised. In this section, the prototype will be reviewed against common authentication system vulnerabilities that outline its strengths and limitations.

5.4.1 Facial Spoofing

The current prototype is not fully secure against facial spoofing. Deepfake videos could potentially deceive the system if it is well-crafted and simulate realistic facial features and movements including natural blinking intervals posing a significant threat to the system.

The camera used to build this prototype is a standard camera and lacks the ability to capture the depth of the subjects to detect livness in the frames. Unlike level 3 classified cameras which usually are equipped with infrared sensors to enhance robustness against spoofing attacks. the absence of such capabilities in the current implementation of the proposed prototype limits its ability to mitigate spoofing attacks.

Tests of the current implementation, which integrates EAR as a liveness detector, showed that the system is not robust against spoofing attacks. When a static image of a registered user is presented to the camera and slightly shaken, the perceived positions of facial landmarks are altered. This change, particularly in the detected eye positions, causes the system to misinterpret the shake as a valid blink, bypassing the liveness detector and allowing unauthorized access using only a static image.

5.4.2 Subdomain Takeover

FIDO2 based authentication system are vulnerable against subdomain takeover if the subdomain is not correctly configured or unclaimed. The attacker can use the gained

subdomain and create a malicious website which has a valid interaction with the authenticator. This subdomain can then be used to create convincing phishing campaigns misleading users to disclose their personal data. Additionally, subdomain takeover can lead to session hijacking if cookies are shared across the subdomains within the site domain. For example, if the session cookie is set for a domain like .exampel.com, that cookie will be valid and accessible for all subdomains such as auth.exampel.com or login.exampel.com. allowing the attacker to gain full access to the secured resources and escalate the attack by changing the user credentials.

5.4.3 Environmental Limitations

If the user is trying to get access from a completely dark room with a camera without flash functionality, the user will not be able to log in since facial recognition is the first layer of authentication and the similarity score must in the worst case, at least exceed the threshold to initiate the TOTP, such environmental circumstances lead to unavailability to the requested resources which can be vital in some cases where the resources need to be fetched immediately.

5.4.4 Leakage of the private key

The private key is securely stored on the USB stick in an encrypted form. To sign challenges received from the relying party, the private key must be decrypted first, which can be dangerous if done in an insecure environment. The current prototype runs on a standard USB stick which does not have dedicated hardware for cryptographic operations. When the signatures need to be signed, the private key is decrypted on the fly and temporarily stored on the RAM. This can be exploited by an attacker by dumping the content of the RAM and extracting the private key for a specific user, leading to compromise the authentication system for that specific user. The proposed prototype is intended to work on a USB stick with detected hardware for cryptographic operations to ensure that sensitive information is not leaked via side-channel attacks. A temporary solution for this is to override the value of the sensitive information directly in the RAM to help reduce the risk of such attacks.

5.4.5 ZAP Scan

Zed Attack Proxy (ZAP) by Checkmarx, one of the most widely used application vulnerability scanners in the world, was used to scan for vulnerabilities. The scanner was employed to detect potential common vulnerabilities, including SQL injections, Cross-Site Scripting (XSS), Buffer Overflow, and Log4Shell. The scanner detected a potential time-based SQL injection due to the lack of proper sanitization and input validation.

Time-based SQL injection can be used to blindly, manipulate the database to process queries with delayed responses based on certain conditions, which may leak sensitive information from the time it takes to process the queries.ZAP scanner calculate the time it takes to process the original query when logging in or registering and compares it to a modified query. The vulnerability exists if the modified query takes significantly longer to execute. The original query took 409 milliseconds according to ZAP, while the injected caused a delay of 20016 milliseconds (20 seconds). This delay indicates the presence of an SQL injection vulnerability. It confirms that the attacker's commands are being executed

on the database server, even though the attacker cannot directly see the output of the queries This issue was resolved by adding validation functions to ensure the username and email address are properly validated during both the authentication and registration processes. The validation function uses a whitelist approach, where only inputs that follow the provided rules are accepted, while all others are rejected.

```
def is_valid_username(user_name):
    # Allow only alphanumeric characters, underscores, and hyphens
    return bool(re.match(r'^[a-zA-Z0-9_-]{3,30}$', user_name))

def is_valid_email(email):
    # Simple regex to check if the email follows the general email pattern
    return bool(re.match(r"[^@]+@[^@]+\.[^@]+", email))
```

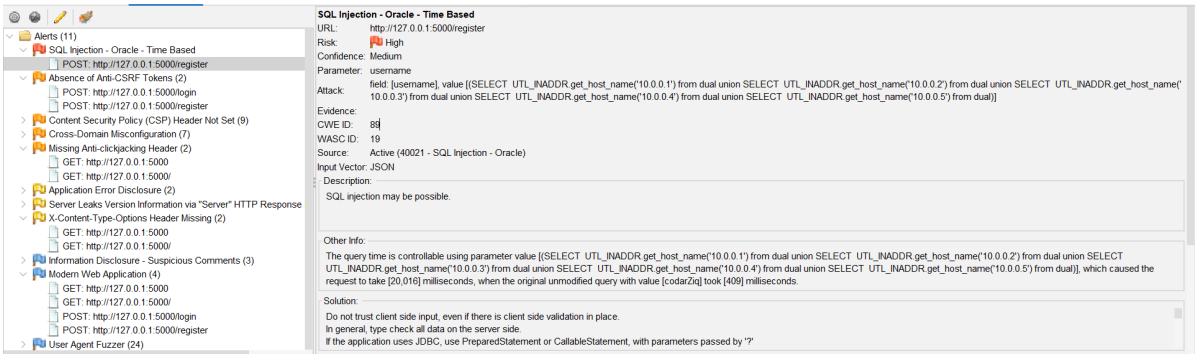


Figure 5.4: A ZAP screenshot showing alerts and potential time-based SQL injection attack.

5.5 Performance Metrics

For determining the performance in terms of memory allocation, CPU usage and elapsed time of the Python face recognition and deepface, python libraries like Tracemalloc, Psutil and time are used. Library tracemalloc, provides functions which allow to monitor memory usage over time and focuses on python's internal memory management. Calling the below *start* method, initializes the memory tracking so that later the information can be gathered about memory allocated at specific points in the code. Further, *get_traced_memory()* is invoked to get the peak memory usage during the program execution.

```
    tracemalloc.start()
    current, peak = tracemalloc.get_traced_memory()
```

CPU usage percentage at the current moment is measured using the cpu _percent function of the library psutil. It takes an interval parameter. The interval, 0.1 specifies that the function will measure CPU usage over a 100 millisecond interval to provide more stable result. It returns a percentage of total CPU usage across all cores. The function call looks like:

```
    cpu_start = psutil.cpu_percent(interval=0.1)
```

On the other hand, the function *virtual_memory* measures the physical memory usage in megabytes (MB). It gives the system wide memory usage that helps to identify if processes are consuming too much memory on a system-wide level. The last metric for determining the elasped time uses the *time* function to capture the current timstamp in seconds.

```
    mem_start = psutil.virtual_memory().used / (1024 ** 2)  # Memory in MB
    start_time = time.time()
```

6

Discussion

This chapter will discuss the security and performance of the proposed prototype, comparing it to common authentication systems while outlining its strengths, weaknesses, and limitations.

6.1 Security Discussion

The goal of this prototype is to provide a secure password-less authentication system. To achieve that, multiple authentication mechanisms were integrated including something you have(a USB stick), something you are (facial features) and in some cases, something you know (an OTP). The OTP is used primarly as a backup to improve the availability of resources when the environment conditions are not good and affects the accuracy of the facial recognition and validation system. Additionally, a FIDO2-inspired setup runs in the background to ensure a seamless public key authentication process. Compared to standard authentication systems using HTTPS requests and relying on a username/password approach, the prototype developed in this thesis has several advantages and is more resilient to common attacks. Since the sensitive credentials (e.g., the private key) are neither stored on the client application nor the relying party database, SQL injection attacks are not applicable to break the system. When the user logs in, the only credential required by the client application is the username, which it uses to interact with the relying party. The relying party database does not include critical credentials but only the public key and non-sensitive information such as birth date and email address. While a breach of such database can lead to privacy violations, the authentication system remains secure and trustworthy. In contrast, in standard username/password-based authentication systems, database security which the client application interacts with, is critical to maintaining the system's robustness, as both the username and password are considered sensitive credentials. Leakage of the username can make brute-force attacks easier since only the password is required to be guessed to get authenticated. And if the password is leaked, the entire authentication process is compromised. Even if these credentials are stored in hash form rather than plain text, rainbow tables can be used to reverse the hashes. Additionally, if the hash algorithm used is outdated or weak (e.g., MD5), it may be easier to crack the hash directly.

The prototype is even more robust against human negligence compared to username/password approach. The passwords set by users to secure online services are often weak and reused over multiple services , which makes the security of the service dependent on the user's knowledge in security best practices. Even when complex passwords are used, these are often saved on a paper or a note-taking application, which are considered insecure en-

vironments. Furthermore, passwords can be easily phished by fake websites, keyboard trackers or even sneaky glances.

Unlike the proposed prototype, where common phishing attacks are not applicable due to the nature of authentication systems based on face recognition and public key cryptography, The sensitive content on the USB stick can only be decrypted if the user authenticates through the face recognition system. Meaning that the authentication process will not be initiated if the content is copied to another USB stick, since the data is encrypted using AES with an encryption key derived from PBKDF, where the USB serial number and model are used as input. This makes the database unusable on another hard drive and increases the resilience of the authentication system due to the presence of randomness connected to the hardware. This is essential because of the absence of detected hardware to handle cryptographic operations.

Due to the absence of passwords in the prototype, there is no need for frequent password changes or recovery which can become a serious point of attack if not implemented correctly. However, a facial-based authentication system may require updates to the stored embeddings to maintain reliability as users age or when facial features change due to natural aging, facial surgeries, or other factors. Therefore, the system must offer a secure and reliable method for updating these features. This can be solved by offering re-enrollment sessions where users can verify their identity by a combination of facial recognition and other authentication models to ensure security during the update process.

One common weakness in facial recognition-based authentication systems is spoofing. This prototype was developed with a standard webcam which lacks the ability to process the depth of the image to detect liveness in the faces of the current frame, unlike cameras which are classified with level 3 security. Blinking detection using Eye Aspect Ratio EAR was implemented as an attempt to mitigate spoofing attacks. However, this mechanism failed against simple attacks. One such attack involved showing a picture of the valid user and shaking it slightly to change the eyes position, which leads the system to incorrectly identify as a blink. Additional mechanisms could be added, such as requiring the user to conduct a specific behavior, like opening their mouth or smiling. While this action can improve the security against spoofing attacks through pictures, they could be easily bypassed if the attack uses deepfake videos that mimic the required behavior to gain access.

Additionally, subdomain takeover could be a concern if there is a lack of periodic maintenance or if misconfiguration exists. In the event of such attacks, the authenticator could be interacted with from a malicious wep application, bypassing the domain comparison provided by the FIDO2 standard. This vulnerability could be exploited to conduct Denial-of-service (DOS) attacks by constantly making the authenticator busy responding to fake requests and filling its memory with fake accounts. As a result, legitimate users could be unable to create new accounts or access the service.

Despite the above-mentioned limitations, which can be addressed with advanced hardware, the prototype still offers advantage over common passphrase-based keys such as non-biometric YubiKeys. Non-biometric only requires the possession of the USB stick to gain access. If such keys are stolen, the attacker gains access to all registered accounts. Unlike the prototype where the USB stick is unusable unless the user is authenticated through the integrated face recognition system. Even if the non-biometric keys are com-

bined with PIN codes, having a biometric layer with facial features is more secure and reliable, particularly for high-security facilities and services.

The image embeddings are stored in plain form on the USB stick, meaning that these can only be extracted and accessed if the USB stick is stolen. To further enhance security, these embeddings could be stored in an encrypted format, similar to the private key, ensuring an additional layer of protection. Having this type of information on the USB stick can be crucial for users afraid of data breaches on web applications or the relying party can have. Moreover, securing this type of sensitive information aligns with data protection regulations such as GDPR and CCPA, ensuring compliance and instilling user confidence in the system's privacy measures.

Another common type of authentication like OpenID connect also uses username password as its first layer of authentication before generating tokens for authorization. A token is a digital artifact that can be stolen, shared, or copied. It does not rely on the physical presence of a person, increasing risks like stolen credentials. The facial recognition biometric are tied to an individual and cannot be shared like passwords or tokens. Depending on the authentication criteria, token based approached are more suitable such as in the case of distributed systems, where secure token management should be robust. To have an advantage of both the systems in the OpenID connect, the authentication can leverage the facial recognition for initial strong authentication instead of using the traditional username password method. The hybrid model will ensure both physical presence verification and secure session handling.

Today, some services rely solely on using TOTP to provide access to resources, such as Claude AI. The user only needs to provide their email to receive a link that forwards them to a page containing the TOTP. This implies that the service's security is reliant on the email provider's security. To access any services that solely rely on TOTP, one only needs to compromise the email account or be able to monitor its content. Apart from human error, technical flaws like replay attacks, man-in-the-middle attacks, and sporadic clock synchronisation problems can also affect TOTP. The suggested prototype has TOTP included as an extra security measure. At first, TOTP was only supposed to be implemented when face recognition failed due to environmental constraints. However, TOTP was imposed as a permanent layer that is constantly invoked because of the limits in liveness detection that were found and the unreliability of the current implementation.

Similar to the suggested prototype, there are popular devices like YubiKey available on the market. Without the need for extra authentication procedures, the version without fingerprint authentication can be utilised straight away to access protected services. If the client application does not impose additional security levels, the hardware can be taken and utilised directly, increasing the danger of human carelessness attacks. In contrast to the suggested prototype, which has extra layers to counteract human error, like face recognition and the authenticator-side TOTP layer.

6.2 Performance

The facial detection and verification step in the multi-factor authentication system plays an important role. With careful considerations, the facial recognition models should be adapted as no models provide an accuracy of 100 percent and the resource usage for

each different models vary. Due to restrictions in camera hardware, lighting conditions, and demographic factors, high system requirements, the algorithm could fail to correctly identify the user. Table A.4 lists the performance of HOG-based face recognition models and Table A.5 describes the performance of the CNN based models.

The HOG-based models such as one offered by python face recognition have demonstrated sturdy performance in controlled environments for the registration and login flow. The registration and login time remained below 6 seconds and showed only occasional spikes as shown in the graph figures 6.1 and 6.2. On the Yale[76] dataset, it has also been demonstrated to attain an accuracy of 99.98% with an 80% train-test split, whilst the ORL[77] dataset obtained 96.25% accuracy in comparable circumstances. These highlight the HOG models' dependability, especially given their low pose, lighting, and expression variability. Additionally, the facial recognition HOG model did not consume a lot of memory resources, as evidenced by the flat memory usage of about 13 MB for both the registration and login flows. Figures 6.5 and 6.6 display the iterations for the memory usage of both flows.

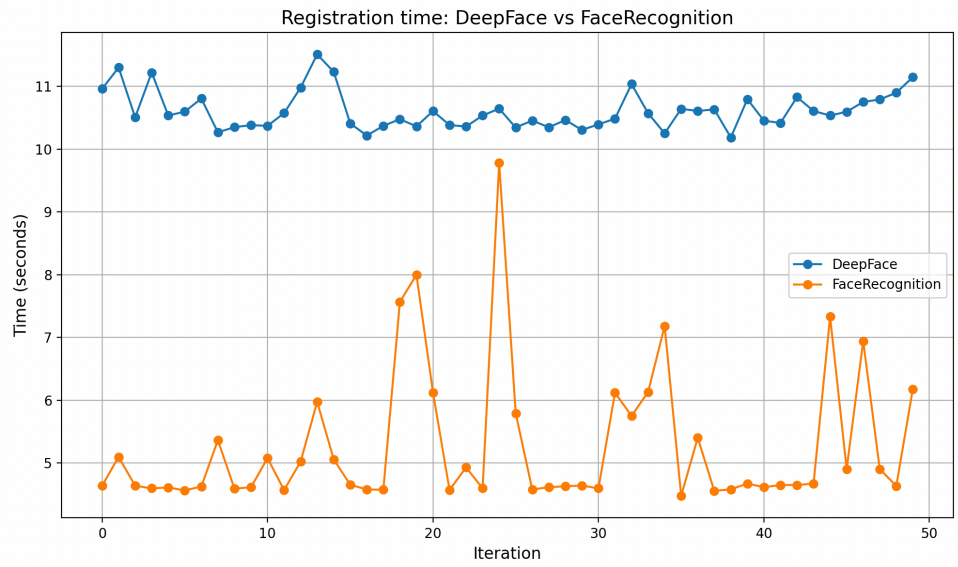


Figure 6.1: Registration Time Comparison

Utilizing the Mediu Staff database, which replicates unregulated real-world circumstances in a more uncontrolled environment, the HOG method demonstrated a 97.6% accuracy rate. This demonstrates how versatile HOG-based models are when used with other feature extraction methods. The resilience against significant position fluctuations and occlusion is still moderate, though. Although the models are computationally inexpensive with CPU usage hovering around 5-6%, which makes them perfect for real-time applications like attendance and anti-spoofing systems. Figures 6.3 and 6.4 illustrate the CPU usage for the HOG-based facial recognition model's registration and login flow. In very complex or dynamic conditions, their performance may decline.

Because of its simplicity of use and ability to adjust to variations in lighting, HOG feature extraction is a common choice for lightweight systems. However, when the environment gets more complicated, further advancements are required to maintain accuracy, including the use of preprocessing based on deep learning.

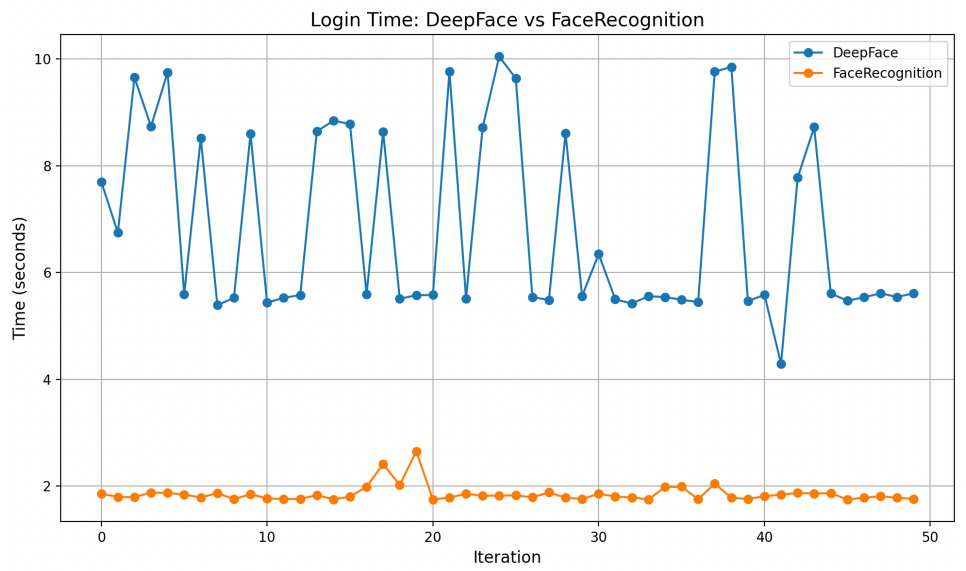


Figure 6.2: Login Time Comparison

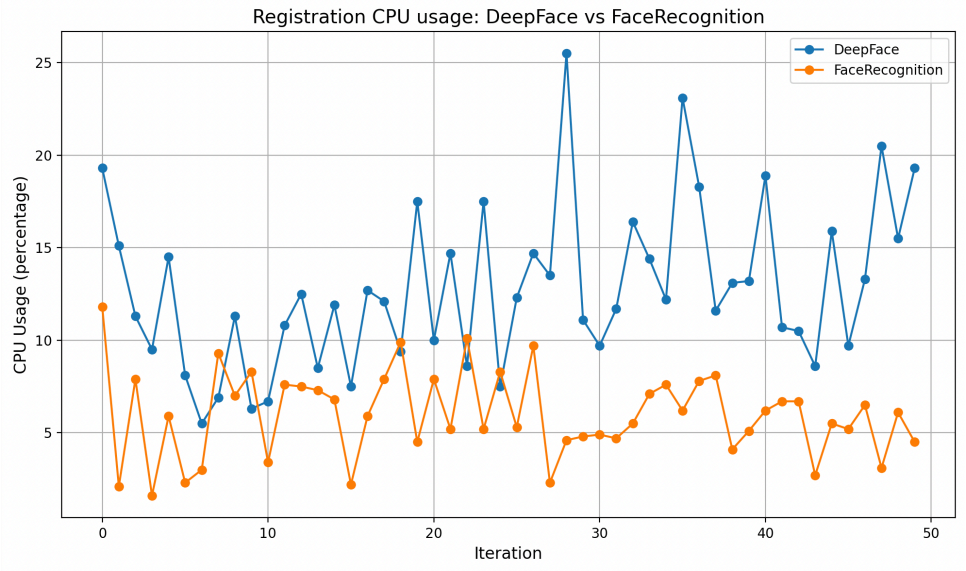


Figure 6.3: Registration CPU Usage Comparison

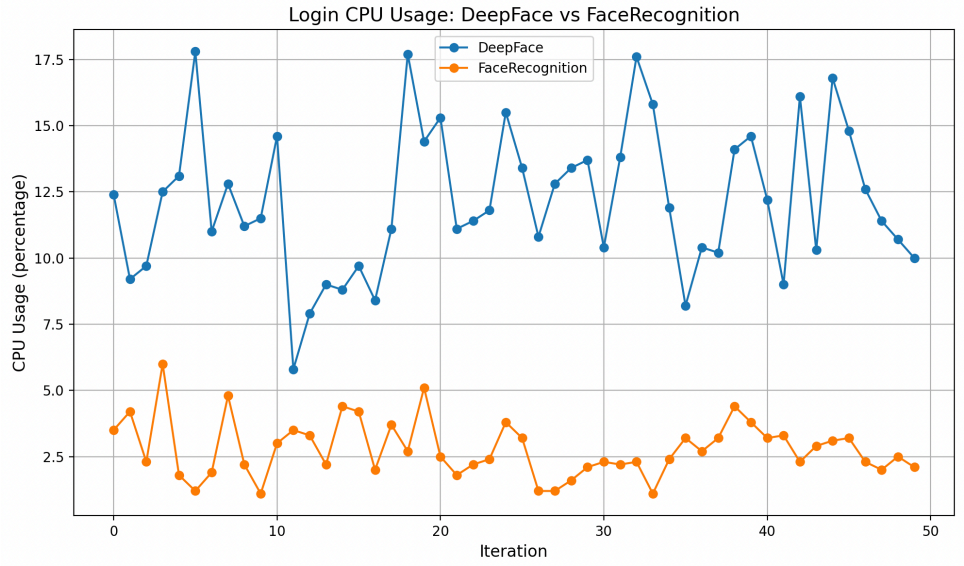


Figure 6.4: Login CPU Usage Comparison

The performance of CNN-based models on a variety of datasets shows how well they handle complex and varied settings. Compared to the HOG based model, the time taken to complete the registration and login flow for a Facenet512 model was found to be relatively high when analysed on the CNN-based deepface. The login and registration time are shown in the figures 6.1 and 6.2. When it comes to adding another layer of complexity like emotion detection, CNNs obtained a 98.77% accuracy across seven emotion classes on the CK+ dataset intended for emotion recognition. Similarly, FaceNet with Inception ResNet v1 demonstrated its capacity to handle real-world datasets with varying lighting, expressions, and occlusions by achieving 96.51% accuracy on the Indonesian Faces dataset[78].

As demonstrated by their 100% accuracy on the ORL dataset, CNN models performed better under controlled conditions. On the Yale dataset, however, the challenges posed by low-resolution images resulted in an accuracy drop of 85.71%, suggesting that CNN performance is significantly influenced by preprocessing and dataset quality. As seen in figures 6.5 and 6.6, the peak memory usage for deepface varied between 400 and 500MB during the registration flow, which is comparable to the login phase and illustrates the demanding nature of such models. Additionally, the VGGFace and ArcFace models' accuracy was 97.78% and 87.8%, respectively, across a range of datasets, demonstrating CNNs' adaptability. While HOG-based models shown a small decline in performance, CNN-based models also demonstrated strong robustness above 85% accuracy with obstructed pictures. Moreoover, when it comes to expression handling, the CNN-based models achieved F1-Scores above 95% on CK+ dataset for recognizing expressions like happiness and surprise, while HOG based models excelled only for neutral or subtle expressions.

Despite their high accuracy, CNN-based models are computationally intensive, requiring significant resources for both training and deployment. As seen in figures 6.3 and 6.4, the CPU usage for these models was likewise much higher than that of the HOG model, peaking at about 20% and fluctuated significantly. Their ability to withstand occlusions, location changes, and illumination conditions makes them suitable for large-scale

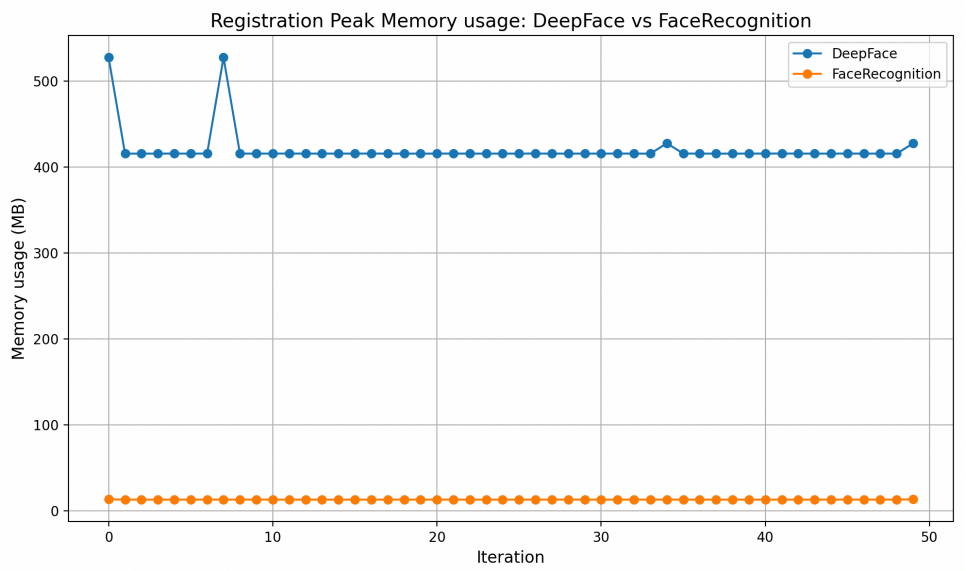


Figure 6.5: Registration Memory Usage Comparison

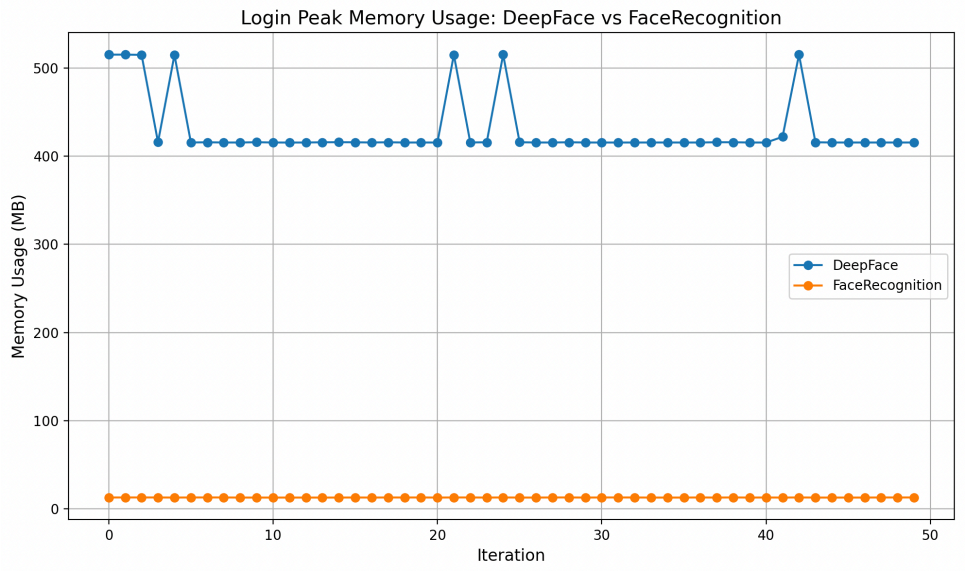


Figure 6.6: Login Memory Usage Comparison

applications such as biometric verification and emotion recognition. However, because of the higher processing required, this restricts their feasibility for lightweight, real-time systems.

A CNN model such as Facenet512 generates a facial embedding of size 512. Each value has 4 bytes, which means that a single facial embedding of size 512 will be 2 KBy in size. On the other hand, the HOG model generates a facial embedding of size 128, which comes out to be 0.5 KB. The storage requirement for other models are given in Table A.6. A typical yubikey has a storage size of 51KB [79] and the private key occupies a size of 0.234KB. Therefore, it leaves ample space for having accounts for multiple relying parties and has a limit on the storage as well. A light-weight model could optimize on the storage space but at the same time wont perform very well in an uncontrolled environment.

6.3 Future work

The current prototype did demonstrate the feasibility of integrating a facial recognition-based authentication system in combination with OTP and FIDO2 standard on a passkey for secure authentication. However, several areas can be improved in future work to ensure and enhance the security and performance of the proposed prototype.

Integration of Secure Communication Protocols: The current version of the prototype uses HTTP for communication, which transmits data in plain text. This approach is used to make it easier to monitor the packets in the development phase and is insecure. In future work, secure communication protocols should be integrated to enhance the overall security. An example of such protocols can be TLS (Transport Layer Security) to ensure that all data exchanged between the authenticator and the relying party is encrypted preventing eavesdropping and altering. IPsec could also be utilized on the network layer, particularly for IoT devices to provide more security features.

Full Integration of the FIDO2 Standard: The prototype utilizes many core features of the FIDO2 standard such as public key-based authentication and domain validation. However, it still does not cover all the standard capabilities due the presence of limitations in both time and hardware. Particularly, this does affect the browser role which is normally a core component in FIDO2-based authentication system. Future work could focus on fully integrating FIDO2 standard specification including WebAuth and CTAP2. Allowing the browser to detect the passkey an option for the authentication. Making the system more robust and provide a seamless user experience which is provided by modern browsers and operating systems.

Penetration Testing and Vulnerability Assessment: A comprehenisve penetration testing should be conducted to identify potential vulnerabilities. The current prototype did not go through a rigorous penetration testing and further investigation is needed to capture and patch vulnerabilities to reduce the attack surface. This can be done using more advanced tools such as Burbsuit, Wireshark and Metaspolit.

Hardware and System Optimization for Real-World Deployment: A normal USB stick and a standard webcamera were used to develop the current version of the prototype. The normal USB stick does have a detected hardware for cryptographic operations which makes the operation runs on the CPU of the host device. If the

host resources are limited, this will hinder the current prototype from working. Additionally, the RAM of the host device can be dumped in side-channel attack to extract sensitive information such as the private key and compromise the authentication system for the corresponding user. Therefore, a special USB stick with detected hardware should be utilized to process cryptographic operations just like those used by Yubikey. To complete a task with the least amount of CPU and memory utilization, several operations can be modified and optimized for the USB. Furthermore, level 3 classified cameras are not compatible with the facial recognition technology currently in use. For a more reliable authentication system, the logic should be changed to incorporate this aspect in livness detection.

7

Conclusion

The Thesis proposes an authentication system and evaluates its security and performance, outlining its strength and limitations. In today's digital world, various authentication systems have several vulnerabilities discussed in this Thesis. The traditional username password-based approach has several limitations, such as being prone to brute force and dictionary attacks, and SQL injection. If the user forgets the password, it further generates an extra complexity to recover it for the user securely. In majority of the cases where OpenID connect is used, which uses the OAuth 2.0 protocol for authorization, the first layer of authentication also uses the username password approach.

Biometric based authentication systems are becoming popular due to its user friendly nature, high accuracy and better security. The Artificial Intelligence(AI) advancements in such technologies has made it quite reliable to use these biometric form of authentication. The fingerprints-based authentication system is also not hundred percent secure. For example, Yubikey require an extra hardware to work and is also vulnerable to spoofing attack. Another biometric form uses the face recognition technology and also requires a security token to store the keys and a camera to detect the user. The pre-trained models available for multiple face recognition models are quite advanced and has proven to provide a high accuracy even in low lighting conditions. The face recognition technology can also check for fake images or videos by using liveness detection and anti-spoofing techniques. The CNN models provide high accuracy for occlusions, low-lighting, in case of different poses, but also comes with the cost of resource usage of execution. HOG comparatively works better when it comes to resource usage and uses traditional techniques that is suitable for a more controlled environment. The time taken to execute the ceremonies, the CPU usage as well as the memory usage was found to be lower for HOG based models compared to CNN. With continuous research that is ongoing in the field of artificial intelligence, especially in the fields of CNNs and improving the current models for facial recognition has a very good scope of achieving outstanding results in the upcoming years. With advancements of such facial recognition systems, the proposed prototype will work even more accurately.

For implementing such systems, the framework was inspired by FIDO2 standard. It outlines the communications between different layers of the systems minding good security practices. Strong cryptography is one of the core requirements of the standard. For running a digital signature algorithm, the key pair is generated using the advanced elliptic curve cryptography and symmetric encryption utilizing AES to enhance security. Even to further strengthen the system, PBKDF is used for stretching the password to make it more difficult to access for the attacker. The password used in the proposed prototype is extracted from the unique information of the USB stick, such as the serial number and

the model name. This add a uniqueness to each USB stick when generating AES keys to encrypt the private key and store it securely. For demonstration purposes, the proposed authentication system uses an https call for communication with the normal USB stick and in future should be replaced by the FIDO stick, which is a special USB stick designed for storing the identity.

A typical FIDO2 stick such as Yubikey contains 51KB of space for persistent storage. The current prototype using the HOG model can store more than 50 accounts depending on what metadata needs to be stores. A typical private key is of size 0.24 KB and a facial embedding generated by the model is of 0.5KB. The additional space can be utilized for storing user attributes like username, email or even date of birth depending on the requirement. It can also further be used to store certificates that can help in more secure communication over the net using the https standard.

To have extra protection on top of the current facial recognition systems, an additional layer of TOTP based verification is also enforced. The basic working condition of TOTP is that the email or the phone number should only be accessible to the user. This ensures that even if the facial recognition technology fails to determine with good accuracy if the face is of the correct user, the TOTP layer will ensure it by sending a one time password that has a very short expiration to confirm. TOTP uses HMACs to ensure that the one time code cannot be compromised.

The multi factor approach of authentication using FIDO2, face recognition and TOTP, in spite of using strong cryptography tools which comes with certain security considerations. For example, a misconfiguration of the subdomain could make the system vulnerable to subdomain takeover and give the attacker the ability to bypass the domain verification feature. Another area of consideration is the liveness detection, which could also lead to problems if the security stick is stolen. The deepfakes are becoming widely popular and such videos can easily deceive the system, if anti-spoofing techniques are not implemented properly. No one can ever be sure of human negligence, but even in the case of stolen keys, the system can be made secure enough such that the credentials are only accessible if the user verification passes through the facial recognition and TOTP layers.

Apart from the physical security of the device, the communication between the relying party and the authenticator uses the REST API framework using HTTP POST requests to interact which is not a reliable option but was used to make the development process smoother. Additionally, more comprehensive penetration testing is needed to detect and patch vulnerabilities to make the system more reliable. ZAP provides a good overview of potentials vulnerabilities and alert but it does not work as a stand alone solution.

Thus, the proposed authentication system provides a user friendly and secure approach to authenticate users, but comes with the requirement of the security token and an advanced camera for best results. The choice of facial recognition model for verification also plays a crucial role in deciding the accuracy of the whole system. The system is robust against most of the attacks on other authentication systems used today, especially username/password based one. The scenario where the facial recogntition cannot figure out if it is the correct user accurately, an additional layer of TOTP is used to ensure the system is more strengthened. Following good API security principles, strong cryptography protocols, and FIDO2 standard approach, the proposed system is a better alternative with the use of special hardware, the FIDO stick.

Bibliography

[1] R. K. Ayeni, A. A. Adebiyi, J. O. Okesola and E. Igbekele, "Phishing attacks and detection techniques: A systematic review", in *2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG)*, 2024, pp. 1–17. DOI: 10.1109/SEB4SDG60871.2024.10630203.

[2] P. W. Security, *Password-based authentication*, Accessed: 2025-01-02, 2025. [Online]. Available: https://portswigger.net/web-security/authentication/password-based.

[3] JumpCloud, *Credential theft: How it works and how to mitigate it*, Accessed: 2025-01-02, 2025. [Online]. Available: https://jumpcloud.com/blog/credential-theft-mitigation.

[4] JumpCloud, *2024 multi-factor authentication (mfa) statistics & trends*, Accessed: 2025-01-02, 2024. [Online]. Available: https://jumpcloud.com/blog/multi-factor-authentication-statistics.

[5] P. I. Certification, *Understanding the three factors of authentication*, Accessed: 2025-01-02, 2025. [Online]. Available: https://www.pearsonitcertification.com/articles/article.aspx?p=1718488.

[6] N. Y. Post, *Your complicated passwords may not be as effective as you think*, Accessed: 2025-01-02, 2025. [Online]. Available: https://nypost.com/2025/01/02/tech/complicated-passwords-may-not-be-as-effective-as-you-think/.

[7] Robots.net, *What does otp mean in banking?*, Accessed: 2025-01-02, 2024. [Online]. Available: https://robots.net/fintech/what-does-otp-mean-in-banking/.

[8] Yubico, *Passwordless account login with yubikey*, https://www.yubico.com/solutions/passwordless/, Last accessed: November 7, 2024.

[9] Identity.com, *Privacy concerns with biometric data collection*, Accessed: 2025-01-02, 2025. [Online]. Available: https://www.identity.com/privacy-concerns-with-biometric-data-collection/.

[10] A. Wiki, *What is time-based one-time password (totp)?*, Accessed: 2025-01-02, 2024. [Online]. Available: https://auth.wiki/totp.

[11] D. M'Raihi, S. Machani, M. Pei and J. Rydell, *Totp: Time-based one-time password algorithm*, RFC 6238, Section 4: Detailed Description of the Algorithm, May 2011. DOI: 10.17487/RFC6238. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6238#section-4.

[12] P. Identity, *The one-time password (otp) ultimate guide*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.pingidentity.com/en/resources/blog/post/one-time-password-ultimate-guide.html.

[13] Kaspersky, *How secure are authenticator apps?*, Accessed: 2025-01-02, 2023. [Online]. Available: https://www.kaspersky.com/blog/authenticator-apps-and-security/47426/.

[14] PyOTP Contributors, *Pyotp: Python one-time password library*, https://github.com/pyauth/pyotp, Last accessed: November 6, 2024.

[15] F. Alliance, *Fido alliance specifications*, https://fidoalliance.org/specifications/, Last accessed: November 7, 2024.

[16] Yubico, *Webauthn: Web authentication documentation by yubico*, https://developers.yubico.com/WebAuthn/, Last accessed: November 7, 2024.

[17] W. W. W. C. (W3C), *Web Authentication: Use Case - Registration*, https://www.w3.org/TR/webauthn/#sctn-usecase-registration, Last accessed: November 7, 2024.

[18] Yubico, *Webauthn developer guide: Webauthn client registration*, https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/WebAuthn_Client_Registration.html, Last accessed: November 7, 2024.

[19] Yubico, *Webauthn developer guide: Webauthn client authentication*, https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/WebAuthn_Client_Authentication.html, Last accessed: November 7, 2024.

[20] Yubico, *Client to authenticator protocol (ctap)*, https://developers.yubico.com/CTAP/, Last accessed: November 7, 2024.

[21] W. W. W. C. (W3C), *Web authentication: An api for accessing public key credentials level 2*, https://www.w3.org/TR/webauthn-2/, Last accessed: November 7, 2024.

[22] Yubico, *What is a yubikey?*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.yubico.com/.

[23] Yubico, *Yubico products: Secure solutions for strong authentication*, https://www.yubico.com/products/, Last accessed: November 7, 2024.

[24] Curity, *Issuing oauth and openid connect tokens*, Accessed: 2025-01-02, 2024. [Online]. Available: https://curity.io/docs/idsvr/latest/token-service-admin-guide/token-issuance/index.html.

[25] D. Hardt, *The oauth 2.0 authorization framework*, RFC 6749, Oct. 2012. DOI: 10.17487/RFC6749. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749.

[26] ITNEXT, *Understanding oauth 2: Architecture, use cases, benefits, and limitations (part 1)*, https://itnext.io/understanding-oauth-2-architecture-use-cases-benefits-and-limitations-part-1-fbc3056e30b8, Last accessed: November 7, 2024.

[27] J. Martinez, *11 common authentication vulnerabilities you need to know*, Accessed: 2024-10-21, 2024. [Online]. Available: https://www.strongdm.com/blog/authentication-vulnerabilities.

[28] S. Store, *Types of encryption: What to know about symmetric vs asymmetric encryption*, https://sectigostore.com/blog/types-of-encryption-what-to-know-about-symmetric-vs-asymmetric-encryption/, Last accessed: November 7, 2024.

[29] GeeksforGeeks, *Block cipher modes of operation*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.geeksforgeeks.org/block-cipher-modes-of-operation/.

[30] GeeksforGeeks, *Symmetric key cryptography: Stream ciphers*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.geeksforgeeks.org/symmetric-key-cryptography/.

[31] D. Security, *Cbc mode initialization vector (iv)*, Accessed: 2025-01-02, 2024. [Online]. Available: https://defuse.ca/cbcmodeiv.htm.

[32] D. A. McGrew and J. Viega, *The galois/counter mode of operation (gcm)*, Accessed: 2025-01-02, 2004. [Online]. Available: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf.

[33] C. 101, *What are block ciphers?*, https://ctf101.org/cryptography/what-are-block-ciphers/, Last accessed: November 7, 2024.

[34] Wallarm, *What is aes (advanced encryption standard)?*, https://www.wallarm.com/what/what-is-aes-advanced-encryption-standard, Last accessed: November 7, 2024.

[35] PullRequest, *Understanding the benefits of key derivation functions: A deep dive into pbkdf2*, Accessed: 2025-01-02, 2023. [Online]. Available: https://www.pullrequest.com/blog/understanding-the-benefits-of-key-derivation-functions-a-deep-dive-into-pbkdf2/.

[36] P. Gutmann, *Pkcs #5: Password-based cryptography specification version 2.1*, RFC 8018, Section 5.1: PBKDF2 - Password-Based Key Derivation Function 2, Jan. 2017. DOI: 10.17487/RFC8018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8018#section-5.1.

[37] G. Ranganathan, X. Fernando, F. Shi and Y. E. A. Editors, "Advances in intelligent systems and computing", in *Soft Computing for Security Applications*, ser. Advances in Intelligent Systems and Computing, G. Ranganathan, X. Fernando, F. Shi and Y. E. Allioui, Eds., vol. 1397, Springer, Year of publication. DOI: 10.1007/978-981-16-5301-8_12. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-16-5301-8_12.

[38] N. Ferguson, B. Schneier and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing, 2010, Includes detailed discussions on ECDSA and its security properties, ISBN: 978-0470474242.

[39] N. I. of Standards and T. (NIST), *Hash-based key derivation function*, https://csrc.nist.gov/glossary/term/hash_based_key_derivation_function, Last accessed: November 6, 2024.

[40] C. U. of Technology, *Cryptography*, Accessed: 2025-01-02, 2025. [Online]. Available: https://chalmers.instructure.com/courses/20957.

[41] H. Krawczyk, M. Bellare and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, Feb. 1997. DOI: 10.17487/RFC2104. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2104.

[42] J. C. Villanueva, *Should we start using 4096 bit rsa keys?*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.jscape.com/blog/should-i-start-using-4096-bit-rsa-keys.

[43] B. Wyro, *Encrypting vs signing with openpgp: What's the difference*, Accessed: 2025-01-02, 2025. [Online]. Available: https://blog.mdaemon.com/encrypting-vs-signing-with-openpgp-whats-the-difference.

[44] Chalmers University of Technology, *Research notes*, Accessed: https://chalmers.instructure.com/courses/20957/files/2469209?module_item_id=328713, Last accessed: November 6, 2024.

[45] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, Refer to elliptic curve properties and efficiency details, ISBN: 9780849385230. [Online]. Available: https://www.cacr.math.uwaterloo.ca/hac/.

[46] NotSoSecure, *Ecdsa nonce reuse attack*, Accessed: 2025-01-02, 2023. [Online]. Available: https://notsosecure.com/ecdsa-nonce-reuse-attack.

[47] Identity, *What are digital signatures? the complete guide*, Accessed: 2025-01-02, 2024. [Online]. Available: https://www.identity.com/what-are-digital-signatures/.

[48] C. Contributors, *Cryptography: Python library for cryptographic operations*, https://cryptography.io/en/latest/, Last accessed: November 7, 2024.

[49] Python Cryptographic Authority (PyCA), *Cryptography: Python library for cryptographic recipes and primitives*, https://pypi.org/project/cryptography/, Last accessed: November 7, 2024.

[50] EasyAI, *Understand face recognition in one article (4 features + 4 steps)*, Accessed: 2025-01-02, 2024. [Online]. Available: https://easyai.tech/en/ai-definition/facial-recognition/.

[51] Y.-Q. Wang, "An analysis of the viola-jones face detection algorithm", *Image Processing On Line*, vol. 4, pp. 128–148, 2014. DOI: 10.5201/ipol.2014.104. [Online]. Available: https://doi.org/10.5201/ipol.2014.104.

[52] *Vj feature types*, https://upload.wikimedia.org/wikipedia/commons/thumb/3/31/VJ_featureTypes.svg/640px-VJ_featureTypes.svg.png, Last accessed: November 6, 2024.

[53] Y. Wu and Q. Ji, "Facial landmark detection: A literature survey", *International Journal of Computer Vision*, vol. 127, no. 2, pp. 115–142, Feb. 2019. DOI: 10.1007/s11263-018-1097-z. [Online]. Available: https://doi.org/10.1007/s11263-018-1097-z.

[54] S. Serengil and A. Ozpinar, "A benchmark of facial recognition pipelines and co-usability performances of modules", *Gazi University Journal of Science*, pp. 95–107, 2024. DOI: 10.17671/gazibtd.1399077. [Online]. Available: https://doi.org/10.17671/gazibtd.1399077.

[55] W. contributors, *Deepface — wikipedia*, https://en.wikipedia.org/wiki/DeepFace, Last accessed: November 6, 2024, 2024.

[56] A. Greche, M. Jazouli, N. Es-Sbai, A. Majda and A. Zarghili, "Comparison between euclidean and manhattan distance measure for facial expressions classification", in *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, 2017, pp. 1–4. DOI: 10.1109/WITS.2017.7934618. [Online]. Available: https://doi.org/10.1109/WITS.2017.7934618.

[57] A. Geitgey, *Face recognition: A simple face recognition library built using dlib's state-of-the-art face recognition*, https://github.com/ageitgey/face_recognition, Last accessed: November 6, 2024.

[58] D. E. King, *Dlib c++ library*, https://dlib.net/, Last accessed: November 6, 2024.

[59] A. Rosebrock, *Face detection with dlib (hog and cnn)*, https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/, Last accessed: November 6, 2024, 2021.

[60] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

[61] B. Sugiarto, E. Prakasa, R. Wardoyo *et al.*, "Wood identification based on histogram of oriented gradient (hog) feature and support vector machine (svm) classifier", in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2017, pp. 337–341. DOI: 10.1109/ICITISEE.2017.8285523.

[62] O. Contributors, *Hogdescriptor struct reference*, https://docs.opencv.org/3.4/d5/d33/structcv_1_1HOGDescriptor.html, Last accessed: November 6, 2024.

[63] S. I. Serengil, *Deepface: A lightweight face recognition and facial attribute analysis library*, https://github.com/serengil/deepface, Last accessed: November 6, 2024.

[64] F. Schroff, D. Kalenichenko and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[65] O. M. Parkhi, A. Vedaldi and A. Zisserman, "Deep face recognition", in *Proceedings of the British Machine Vision Conference (BMVC)*, 2015, pp. 41.1–41.12. DOI: 10.5244/C.29.41.

[66] Serokell, *Introduction to convolutional neural networks*, https://serokell.io/blog/introduction-to-convolutional-neural-networks, Last accessed: November 7, 2024.

[67] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[68] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks", *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016. DOI: 10.1109/LSP.2016.2603342.

[69] eljko Vujovi *et al.*, "Classification model evaluation metrics", *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[70] Microsoft, *Characteristics, limitations, and how to measure accuracy when using the face service*, Accessed: 2025-01-02, 2023. [Online]. Available: https://learn.microsoft.com/en-us/legal/cognitive-services/face/characteristics-and-limitations.

[71] T. R. Project, *Confusion matrix: Qwraps2 package*, https://search.r-project.org/CRAN/refmans/qwraps2/html/confusion_matrix.html, Last accessed: November 22, 2024.

[72] C. 2018, *Pose-robust face recognition via deep residual equivariant mapping*, Accessed: 2025-01-02, 2018. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Cao_Pose-Robust_Face_Recognition_CVPR_2018_paper.pdf.

[73] I. J. of Research Publications and Reviews, *Enhancing occlusion handling in face recognition: A performance evaluation*, Accessed: 2025-01-02, 2023. [Online]. Available: https://ijrpr.com/uploads/V5ISSUE10/IJRPR33894.pdf.

[74] *Eye aspect ratio calculated from facial points*, https://www.researchgate.net/figure/Eye-aspect-ratio-calculated-from-the-facial-points-p-1-p-2-p-3-p-4-p-5-and-p-6_fig2_365907177, Last accessed: November 6, 2024.

[75] T. Soukupová and J. ech, "Real-time eye blink detection using facial landmarks", in *Proceedings of the Computer Vision Winter Workshop*, 2016.

[76] O. Belitskaya, *Yale face database*, https://www.kaggle.com/datasets/olgabelitskaya/yale-face-database, Last accessed: November 7, 2024.

[77] A. Hamdy, *Orl dataset*, https://www.kaggle.com/datasets/ahmad5hamdy12/orl-dataset, Last accessed: November 7, 2024.

[78] A. Nuriman, *Indonesian public figure faces dataset*, https://www.kaggle.com/datasets/arifnuriman/indonesian-public-figure-faces/data, Last accessed: November 7, 2024.

[79] Yubico, *Certificate size for piv applications*, Accessed: 2025-01-02, 2025. [Online]. Available: https://docs.yubico.com/yesdk/users-manual/application-piv/cert-size.html#:~:text=Total%20space%20available%20for%20certificates&text=Note%20that%20that%20total%20amount,.)%20is%20about%2051%2C000%20bytes..

[80] A. S. Hmazah, S. S. Hreshee and A. Aldhahab, "Optimizing face recognition accuracy with hog features and svm classifier: A study on orl and yale databases", *4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, pp. 748–754, 2024. DOI: 10.1109/ICSINTESA62455.2024.10748014.

[81] H. Mady and S. M. S. Hilles, "Face recognition using hog and lbp features on real-world databases", *Mediu Staff Database Study*, 2024, Accessed from internal resources.

[82] D. Sena, *Ck+ dataset*, https://www.kaggle.com/datasets/davilsena/ckdataset, Last accessed: November 7, 2024.

[83] P. Samal and M. F. Hashmi, "Facial expression recognition using combined feature and convolutional neural networks", *CK+ Dataset Study*, pp. 112–120, 2024.

[84] M. Bilal, S. Razzaq, N. Bhowmke, A. Farooq, M. Zaib and S. Soaiq, "Performance of cnn models on orl and yale datasets", *MDPI Journal of Computational Intelligence*, pp. 65–72, 2024, Accessed from public database.

[85] M. A. Faqih, Hendrawan, E. Mulyana and W. Hermawan, "On the performance of face recognition models for local faces dataset", *17th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pp. 20–28, 2023. DOI: 10.1109/TSSA59948.2023.10366896.

[86] J. Li, *Lfw dataset*, https://www.kaggle.com/datasets/jessicali9530/lfw-dataset, Last accessed: November 7, 2024.

[87] A. Firmansyah, T. F. Kusumasari and E. N. Alam, "Comparison of face recognition accuracy of arcface, facenet, and facenet512 models on deepface framework", *DeepFace Framework Study*, 2024, Accessed from internal resources.

[88] A. L. Cambridge, *Orl dataset*, https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, Accessed: 2024-12-14, 1994.

[89] S. D. University of California, *Yale face database*, http://vision.ucsd.edu/content/yale-face-database, Accessed: 2024-12-14, 1997.

[90] P. Lucey, J. F. Cohn, T. Kanade *et al.*, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 486–498, 2010. DOI: 10.1109/TPAMI.2010.112.

[91] U. of Massachusetts Amherst, *Labeled faces in the wild (lfw)*, http://vis-www.cs.umass.edu/lfw/, Accessed: 2024-12-14, 2007.

[92] H. Ahamed *et al.*, "Face detection using hog and lbp in real-world video sequences", in *International Conference on Computer Vision Applications*, IEEE, 2019, pp. 23–29. DOI: 10.1109/ICCV19.8893214.

[93] W. Hermawan, Hendrawan *et al.*, "A comprehensive study of indonesian faces for biometric recognition", *Indonesian Journal of Artificial Intelligence*, vol. 7, no. 2, pp. 102–112, 2021. DOI: 10.1109/TSSA.2021.8642989.

A

Appendix 1

A.1 FIDO2 Options Dictionary

```
options_dict = {
        "challenge": base64.b64encode(options.challenge).decode('utf-8'),
        "rp": {
            "name": options.rp.get("name", ""),
            "id": options.rp.get("id", "")
        },
        "user": {
            "id": options.user.id.decode('utf-8'),
            "name": options.user.name,
            "display_name": options.user.display_name
        },
        "pubKeyCredParams": [
            {
                "type": param["type"],
                "alg": param["alg"]
            } for param in options.pub_key_cred_params
        ],
        "timeout": options.timeout,

    }
```

A.2 AES Algorithm

```
Input:
  P = Plaintext (128-bit block)
  K = Cipher Key (128, 192, or 256 bits)
  Nr = Number of Rounds (10 for 128-bit keys, 12 for 192-bit keys, 14 for
  ↪  256-bit keys)

Output:
  C = Ciphertext (128-bit block)

Steps:
1. Key Expansion:
```

```
   - Expand the cipher key (K) into an array of round keys (Nr + 1 keys, each
   ↪  128 bits) using the key schedule algorithm.

2. Initial Round:
   - AddRoundKey(P, RoundKey[0]):
     XOR the plaintext (P) with the first round key (RoundKey[0]).

3. Main Rounds (Repeat for Nr - 1 rounds):
   For each round (i = 1 to Nr - 1):
   a. SubBytes:
      Substitute each byte in the state using a fixed substitution box
      ↪  (S-Box).
   b. ShiftRows:
      Rotate rows of the state to the left by 0, 1, 2, and 3 bytes
      ↪  respectively.
   c. MixColumns:
      Apply a linear transformation on each column of the state using a fixed
      ↪  matrix.
   d. AddRoundKey:
      XOR the state with the current round key (RoundKey[i]).

4. Final Round (Nr):
   - Perform SubBytes, ShiftRows, and AddRoundKey (skip MixColumns).

5. Output the final state as the ciphertext (C).
```

A.3 PBKDF Algorithm

```
        Input:
  P = Password (user input)
  S = Salt (randomly generated, unique for each password)
  c = Iteration Count (number of hash iterations, e.g., 10,000+)
  dkLen = Desired Key Length (length of derived key in bytes)
  PRF = Pseudo-Random Function (e.g.,SHA256)

Output:
  DK = Derived Key of length dkLen

Algorithm:
1. Split the desired key length (dkLen) into blocks:
   - Let l = ceil(dkLen / hLen), where hLen is the output length of the PRF.
   - Let r = dkLen - (l - 1) * hLen (remaining bytes for the last block).

2. Initialize an empty output buffer for the derived key (DK):
   - DK = ""

3. For each block i from 1 to l:
```

```
   a. Compute the initial hash for the block:
      U1 = PRF(P, S || INT(i))
      - S || INT(i) is the salt concatenated with the block index (i) encoded
      ↪  as a 32-bit integer.

   b. Set the intermediate block value T to U1:
      T = U1

   c. Perform (c - 1) additional hash iterations:
      For j = 2 to c:
          Uj = PRF(P, Uj-1)
          T = T XOR Uj
      - XOR the result of each iteration with the running value of T.

   d. Append T to the derived key (DK):
      DK = DK || T

4. Truncate the final derived key to dkLen bytes if necessary.

5. Return DK.
```

A.4 Performance of HOG based Face recognition models for multiple datasets

Table A.1: Performance of HOG-Based Face Recognition Models

Metric	Dataset	Performance	Source
ORL Accuracy	ORL	96.25%	[80]
Yale Accuracy	Yale	99.98%	[80]
Mediu Staff Accuracy	Mediu Staff	97.6%	[81]
Lighting Robustness	All Datasets	High (Proven robust under lighting changes)	[81]
Pose Robustness	ORL, Yale	Moderate to High (Handles small pose changes effectively)	[80], [81]
Occlusion Handling	ORL, Yale	Moderate (HOG features perform reasonably well under partial occlusion)	[80], [81]
Expressions Handling	Mediu Staff	High	[81]
Computational Cost	All Datasets	Low (Efficient for real-time use)	[80], [81]
Applications	All Datasets	Real-time applications (e.g., anti-spoofing, attendance systems)	[80], [81]

A.5 Performance of CNN based Face recognition models for multiple datasets

Table A.2: Performance of CNN-Based Face Recognition Models

Metric	Dataset	Performance	Source
CK+ Accuracy	CK+[82]	98.77%	[83]
ORL Accuracy	ORL	Up to 100% (High-performing CNN models)	[84]
Yale Accuracy	Yale	Up to 85.71% (Low-resolution adaptations)	[84]
FaceNet Accuracy	Indonesian Faces	96.51% (ResNetv1 with Euclidean threshold 0.75)	[85]
VGGFace Accuracy	LFW[86]	97.78% (DeepFace Framework)	[85]
ArcFace Accuracy	Diverse Datasets	87.8% (DeepFace Framework)	[87]
Lighting Robustness	CK+, Indonesian Faces	Very High (Demonstrated in tests)	[83], [87]
Pose Robustness	CK+, Indonesian Faces	Very High (Proven in multiple datasets)	[83], [87]
Occlusion Handling	CK+, Indonesian Faces	High (CNNs handle occlusions effectively)	[83], [85], [87]
Expressions Handling	CK+	Very High (Explicitly tested on expressions)	[83]
Computational Cost	All Datasets	High (Resource-intensive, especially for large-scale CNNs)	[83], [84], [87]
Applications	All Datasets	Large-scale systems (e.g., biometric authentication, emotion recognition)	[83], [87]

A.6 Facial embedding sizes for different Face Recognition models

A.7 Datasets Used in Evaluation

Model	Embedding Size	Model Type
DeepFace (VGG-Face)	16 KB	CNN
DeepFace (Facenet)	0.5 KB	CNN
DeepFace (OpenFace)	0.5 KB	CNN
DeepFace (DeepID)	2 KB	CNN
DeepFace (ArcFace)	2 KB	CNN
face_recognition	0.5 KB	HOG
DeepFace (Facenet512)	2 KB	CNN

Table A.3: Embedding Sizes and Model Types for Various Face Recognition Models

Table A.4: Datasets Used in Evaluation

Dataset	Images	Variations	Primary Purpose	Source
ORL	400	Facial expressions, lighting, pose	Benchmark for small, controlled datasets	[88]
Yale	165	Lighting, expressions, occlusions	Testing lighting and occlusion handling	[89]
CK+	10,000	Expressions, lighting, pose	Facial expression recognition	[90]
LFW	13,233	Real-world lighting, pose, occlusions, expressions	Generalization to real-world conditions	[91]
Mediu Staff	Few thousand	Real-world variations in lighting, occlusions, pose	Uncontrolled environment testing	[92]
Indonesian Faces	5,00010,000	Ethnic diversity, lighting, pose, expressions	Adaptability to ethnic and diverse groups	[93]