

# Electronic Identification and Signature – ONE ID

## *Project, Application and Technical Explanation*

**Students: Genti Nuhui (221525) and Jahjanur Zulbeari (221544)**

### **- Project Explanation**

**One Id Electronic Identification and Signature** is a Spring Boot application that provides a secure, automated process for verifying an individual's identity and enabling electronic document signing.

#### **1. Biometric and Document-Based Authentication**

- The system captures a **live photo** of the user to prevent identity fraud.
- The system saves **user's personal information**.
- An **ID card** is scanned and verified using AI-driven recognition techniques.

#### **2. Public and Private Key Generation**

- After successful verification, the system generates a **public-private key pair**.
- The **private key** is downloaded locally and it is deleted from the application immediately. It is used for identification or signature.
- The **public key** allows verification of signed documents, ensuring authenticity.

#### **3. Secure Digital Signing**

- Users can sign digital documents electronically using their **verified account**.
- This ensures tamper-proof, legally binding agreements without requiring physical presence.

#### **4. SSL Certificate for Secure Communication**

- The application is protected by an **SSL certificate**, ensuring that all data transmitted between the user and the server is encrypted.
- SSL prevents **man-in-the-middle attacks, data interception, and unauthorized access**.

## 5. Compliance with Security Standards

- The system adheres to **global encryption standards** (RSA) to ensure security.
- The user's **password** is encrypted when entered in the database.
- The system uses **Two-Factor Authentication** when logging in or registering.
- **The private key** is uploaded via text file to prevent **Keylogging**.

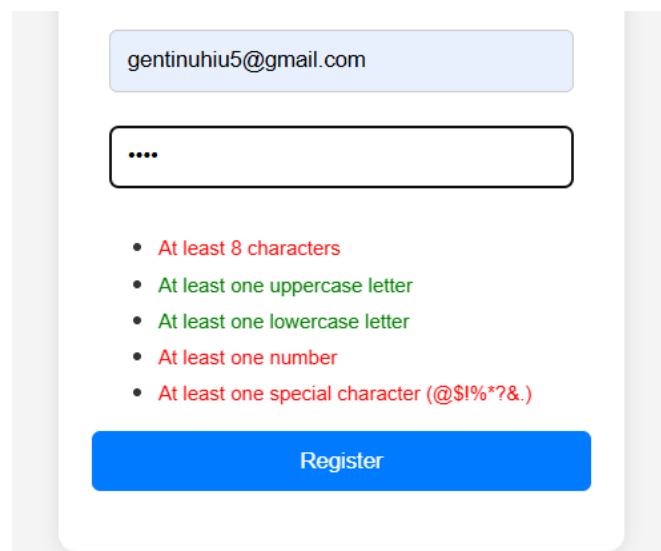
### Importance of Digital Identity Verification and Electronic Signatures

With increasing cybersecurity threats and regulatory requirements, digital identity verification and electronic signatures offer:

- **Improved security** against identity theft and document fraud.
- **Efficiency** in remote transactions without physical paperwork.
- **Legal validity** for online agreements and contracts.
- **User convenience** with a protected verification and signing process.

### - Application Features

1. When registering an user, the system asks for a **strong password**.



The image shows a user registration form. At the top, there is a light blue rounded rectangle containing the email address "gentinuhui5@gmail.com". Below this is a white rounded rectangle with a black border, containing four dots "...." to represent a password field. Under the password field is a list of five password requirements, each preceded by a red dot. The requirements are: "At least 8 characters" (red text), "At least one uppercase letter" (green text), "At least one lowercase letter" (green text), "At least one number" (red text), and "At least one special character (@\$!%\*?&.)" (red text). At the bottom of the form is a solid blue rounded rectangle with the word "Register" in white text.

2. **Two-Factor Authentication** when logging in or registering.

### Verify Your Email

*gentinuhui5@gmail.com*

Verification code expires in 55 seconds

Enter verification code

Verify

3. User dashboard for managing information. The user will be verified after all input fields are filled and **confirmed with ID Card**.

## Identification Process

You need to complete the verification steps below in order to verify your identity! X

### Step 1: Personal Information

Username:

Email:


First Name:  X

Last Name:  X

SSN:  X


EDIT

### Step 2: Live Camera Capture



SCAN

### Step 3: Upload ID Card



CHOOSE A FILE

4. The use of private-public keys and document signing is allowed only after the user has been **verified**.

### Step 1: Personal Information

Username:

Email:

First Name:  ✓

Last Name:  ✓

SSN:  ✓

[EDIT](#)

### Step 2: Live Camera Capture

[SCAN](#)

### Step 3: Upload ID Card

[CHOOSE A FILE](#)

## Key Generator

You can use your Public and Private Keys for authentication! ✓

#### PUBLIC KEY

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmH28iafkHxb0haiDDOfVY/78eg1YRv2MYRzCgTu/ci9YFAtz8LRGsurOS7Rvo95DcqMrhQHO1JdYQi6iFkK8x..

[Download Public Key](#)

#### PRIVATE KEY

MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCYfbyJp+QfFvSFqIMM59Vj/vx6DVhG/YxhHMKBO79yL1gUC3PwtEay6s5LtG+j3kNyoyuFAC7UI1h...

[Download Private Key](#)

5. The user must download the private key because it will be **lost** immediately after the page refreshes.

6. Verified users can sign documents and can mark them as **Confidential** or **Public**.

## Key Generator

You can use your Public and Private Keys for authentication! ✓

#### PUBLIC KEY

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmH28iafkHxb0haiDDOfVY/78eg1YRv2MYRzCgTu/ci9YFAtz8LRGsurOS7Rvo95DcqMrhQHO1JdYQi6iFkK8x..

[Download Public Key](#)

#### PRIVATE KEY



MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCYfbyJp+QfFvSFqIMM59Vj/vx6DVhG/YxhHMKBO79yL1gUC3PwtEay6s5LtG+j3kNyoyuFAC7UI1h...

[Download Private Key](#)

### Upload PDF for Signing

[Choose File](#) Document.pdf ☒ Confidential ☐ Public [Upload & Sign](#)

7. Each signed document will have an **identifier**, which can be used to verify the authenticity of the person who signed the document.

1 / 2 | - 100% + |  


SIGNED BY Genti Nuhiu, ID:7

Genti Nuhiu

Software Engineering Student in Skopje, North Macedonia

A software engineering student passionate about IT solutions, complex algorithms, and innovative system design. Always exploring new technologies, optimizing processes, and solving challenging problems through code.

Projects



Contact

Phone Number: 072/594-651

Email: gentinuhui5@gmail.com

8. The verification of **confidential** documents shows the masked username of the person and the timestamp.

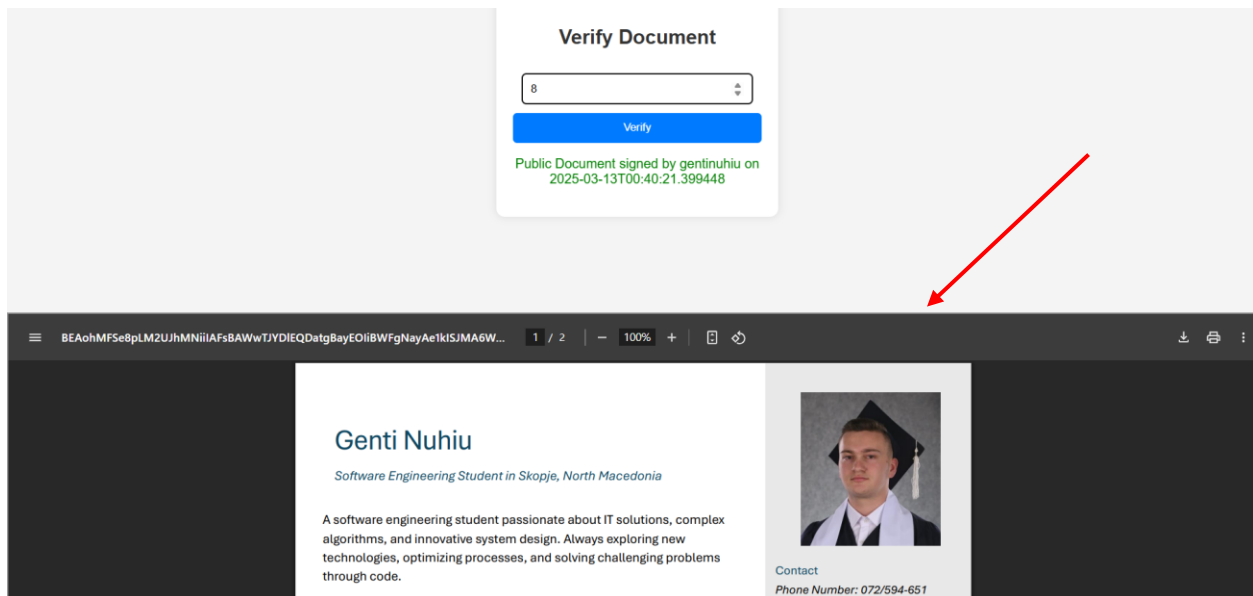
Verify Document

7

Verify

Confidential Document signed by gen\*\*\*  
on 2025-03-13T00:35:42.401319

9. The verification of **public** documents shows us the whole username of the person, the timestamp as well as **the document itself**.



## - Technical Explanation

### 1. Introduction

OneID and Electronic Signature is a secure authentication and identity verification system designed to enhance user verification and data security. The system integrates multiple authentication methods, including password-based authentication and private key-based authentication. It also employs biometric verification, such as facial recognition and Optical Character Recognition (OCR), to validate identity documents.

This system ensures secure user identification and document verification while providing a seamless login experience for end users. It is particularly useful in environments where security and identity verification are critical, such as online banking, government portals, and enterprise applications.

### 2. System Overview

The OneID system provides a multi-faceted approach to authentication and identity verification. Users can log in through traditional credential-based authentication or utilize RSA-based private key authentication. The system also verifies user identity by comparing live facial images with those on ID cards and extracting data from identity documents through OCR.

#### Key components of the system:

- **User Authentication:** Credential-based login and private key authentication.

- **Identity Verification:** Validation using ID card images, OCR, and facial recognition.
- **Electronic Signatures:** Users can securely sign documents with cryptographic keys.
- **Secure Data Handling:** User information, passwords, and private keys are securely encrypted and stored.

### 3. Technology Stack

- **Backend:** Spring Boot (Java), Hibernate (JPA), Spring Security
- **Frontend:** HTML, JavaScript, Bootstrap
- **Database:** H2 Database (for development), MySQL (for production)
- **Security:** RSA Encryption, SSL/TLS
- **OCR and Face Recognition:** Tesseract OCR, Python Face Recognition

### 4. Key Functionalities

#### a) User Registration

- Users sign up with their username, email, and password.
- Strong password validation is enforced.
- A verification email with a security code is sent to verify the email address.

#### b) Login Mechanisms

- **Credential-based authentication:** Users log in with their username and password.
- **Private Key authentication:** Users authenticate by signing a random string with their private key.

#### c) Identity Verification

- **OCR-based ID card scanning:** Extracts name, surname, and SSN from the uploaded ID card.
- **Facial Recognition:** Matches the user's live photo with the ID card image.

#### d) Electronic Signature

- Users can digitally sign documents using their private keys.

- The system verifies the signature using the corresponding public key stored in the database.

#### **e) Data Security**

- All sensitive data is encrypted.
- Public and private key pairs are generated for each user.
- SSL/TLS is enforced for secure communication.

### **5. Authentication Mechanisms**

#### **Credential-Based Authentication**

1. User enters their username and password.
2. Password is validated against the stored hash.
3. If correct, a verification code is sent to the user's email.
4. User enters the code to complete the login process.

#### **Private Key Authentication**

1. User uploads their private key file.
2. A random challenge string is generated.
3. The user signs the string with their private key.
4. The signature is sent to the server.
5. The server verifies the signature using the stored public key.
6. If valid, the user is authenticated.

### **6. Security Features**

#### **SSL/TLS Implementation**

- Enforces encrypted communication using SSL.
- Uses a self-signed certificate for development and production SSL certificates for secure environments.

#### **Public/Private Key Generation**

- Uses RSA (2048-bit) encryption to generate key pairs.
- Public keys are stored securely in the database.



- Private keys are stored on the user's local machine.

### **Encryption of Sensitive Data**

- Passwords are stored as hashed values using bcrypt.
- Sensitive user data (e.g., ID card images) is stored in binary format with access restrictions.

## **7. User Management and Data Handling**

- Users can update their profile information.
- Admins can validate users based on ID verification.
- If name, surname, and SSN match, the account is verified.
- If facial recognition succeeds, the account is fully validated.

## **8. Biometric Verification and OCR**

### **OCR (Optical Character Recognition)**

1. Extracts text from the uploaded ID card image.
2. Compares the extracted text with user-provided data.
3. If the data matches, the system marks the ID as verified.

### **Face Recognition**

1. Captures a live photo from the user.
2. Compares it with the photo on the ID card.
3. If the match is successful, the system marks the face as verified.

## **9. Services and Business Logic**

### **UserService**

- Handles user creation, updating, and validation.

### **KeyService**

- Generates and manages public/private keys for each user.

### **OCRService**

- Extracts user data from ID cards and matches it with stored records.

## **ImageService**

- Compares ID card photos with live photos for identity verification.

## **AuthService**

- Manages authentication, including password-based and private key authentication.

## **10. Controllers and API Endpoints**

### **AuthController**

- Handles user login and registration.
- Manages verification codes and validation.

### **DataController**

- Handles uploading ID cards and live photos.

### **VerificationController**

- Validates private key authentication.
- Checks whether a signed challenge is correct.

### **PdfController**

- Handles document uploads and digital signing.
- Stores signed documents securely.
- Provides endpoints for document retrieval and download.

### **VerifyController**

- Verifies signed documents.
- Allows users to check document authenticity.
- Supports confidential and public document validation.

## **11. Frontend Implementation**

### **Credential-Based Login**

```
<form action="/login" method="post">
```

```
  <input type="text" name="username" placeholder="Username" required>
```

```
  <input type="password" name="password" placeholder="Password" required>
```

```
<button type="submit">Login</button>
</form>
```

### **Private Key Authentication**

```
async function encryptAndLogin(event) {
  let privateKeyFile = document.getElementById("private-key").files[0];
  let privateKey = await readFile(privateKeyFile);
  let signature = await signWithPrivateKey(privateKey, "challenge_string");
  fetch("/oneid/verifyLogin", {method: "POST", body: JSON.stringify({username,
signature}})});
}
```

## **12. Deployment and Configuration**

- **Application Configuration:**

```
spring.datasource.url=jdbc:h2:file:~/data/users
```

```
spring.jpa.hibernate.ddl-auto=update
```

```
server.ssl.enabled=true
```

- **Running the Application:**

```
mvn spring-boot:run
```