Security of Computer Systems

**Project Report**

Authors:

Julian Wasylka 193223

Igor Józefowicz 193257

Version: 1.1

**Versions**

| **Version** | **Date** | **Description of changes** |
| --- | --- | --- |
| 1.0 | 13.04.2025 | Creation of the document |
| 1.1 | 02.06.2025 | Completion of the document |

## Project – control term

### Description

This phase focused on implementing an RSA-4096 key generator. The application securely stores private keys using AES-256-EAX encryption, with the key derived from a PIN provided by a user through SHA-256. The GUI allows users to input a PIN, select an output directory (ideally a pendrive), and generate cryptographic key pairs. Private keys are never stored in plaintext.

### Results

The application generates:

* rsa\_private.bin: the AES-encrypted RSA private key,
* rsa\_public.pem: the corresponding public key.

Key generation runs in a separate thread to prevent GUI freezing, with progress displayed via a progress bar.

### Summary

The key generator successfully meets all functional and security requirements for cryptographic key generation. Keys are strongly encrypted, and the application effectively handles errors and validates user input.

## 2. Project - final term

### 2.1 Description

The final project is a complete PAdES-compliant digital signature tool. It allows users to sign and verify PDF documents using RSA-4096 and SHA-256. Private keys are loaded from USB pendrives, decrypted using the user PIN, and used to sign PDF hashes. The application uses *PyQt5* for its GUI and supports both signing and signature verification workflows.

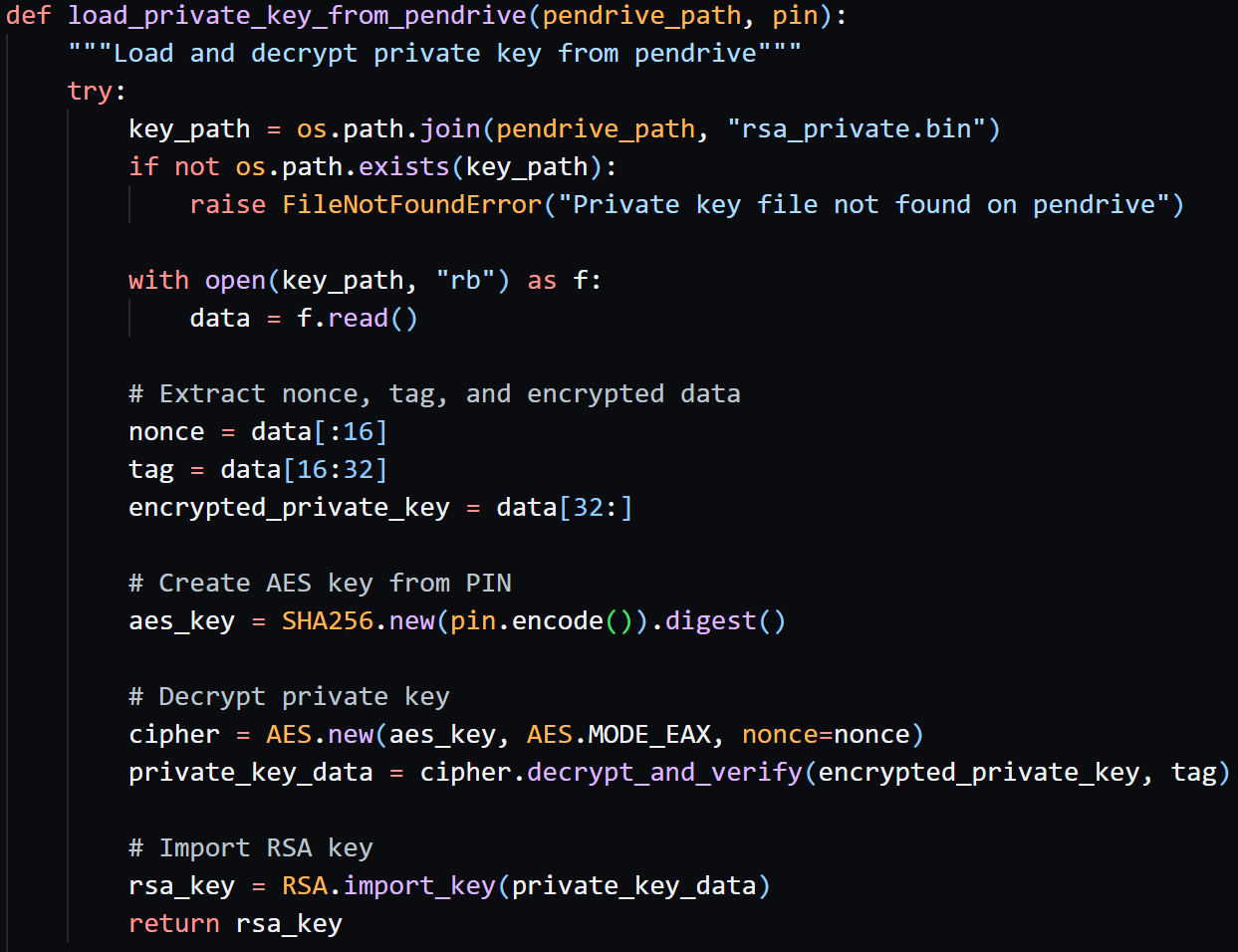
The complete source code of the application is available on GitHub:

<https://github.com/julianwasylka/Emulating-the-PAdES-Qualified-Electronic-Signature>

### 

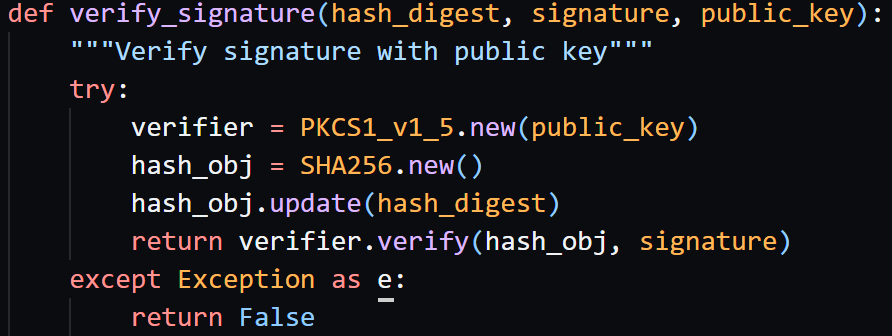
### 2.2 Code Description

##### Listing 1 – Private key decryption and loading from pendrive



This function loads the encrypted RSA private key from the pendrive, decrypts it using a key derived from the user's PIN via SHA-256, and imports it using the RSA module. It ensures that the private key never exists unencrypted on disk.

##### Listing 2 – Signature verification logic



This function verifies a digital signature by recalculating the PDF's hash and checking it with the provided signature. It ensures the integrity and authenticity of the signed document.

##### Listing 3 – PDF signing process



This function signs a PDF document by generating a digital signature and embedding it into the PDF metadata. All original pages are preserved, and signature-related information is added in a PAdES-compliant manner.

##### Listing 4 – Pendrive detection logic



This function detects connected removable drives by scanning system disk partitions.

### 2.3 Tests

The application has been tested with the following scenarios:

* RSA key generation with various PIN lengths
* Document signing with different PDF sizes and formats
* Signature verification with valid signatures
* Detecting changes in signed documents
* Pendrive detection across different USB devices
* Error handling for invalid PINs and missing keys

### 2.4 Results

* Successful integration of GUI, file handling, encryption, and signature generation.
* User feedback by progress bars and status indicators.
* Correct cryptographic processing using *PyCryptodome* and secure storage of keys.
* Fully working PDF signing and verification.

### 2.5 Summary

The final project meets all functional and security requirements. It implements a secure, user-friendly PAdES-compliant signing tool with support for key management, digital signatures, GUI interface, and error handling. The modular architecture ensures maintainability and clarity.

## 3. Literature

1. Online Doxygen documentation, <https://www.doxygen.nl/manual/lists.html>
2. AES Encryption & Decryption In Python, <https://onboardbase.com/blog/aes-encryption-decryption>
3. PyCryptodome documentation, https://pycryptodome.readthedocs.io/en/latest/src/public\_key/rsa.html
4. Encrypt and decrypt using PyCrypto AES-256, https://stackoverflow.com/questions/12524994/encrypt-and-decrypt-using-pycrypto-aes-256
5. Python progress bar, [https://stackoverflow.com/questions/3160699/python-progress-bar](https://stackoverflow.com/questions/3160699/python-progress-bar/)
6. Platform module, <https://www.geeksforgeeks.org/platform-module-in-python>
7. The complete PyQt5 tutorial — Create GUI applications with Python, <https://www.pythonguis.com/tutorials/pyqt-basic-widgets>
8. Encryption and Decryption of PDFs with PyPDF2, <https://pypdf2.readthedocs.io/en/3.x/user/encryption-decryption.html>