Security of Computer Systems

Project Report

Authors:
Paweł Wawrzyński, 193270
Filip Jezierski, 196333

Version: 2.1

Versions

Version	Date	Description of changes
1.0	08.03.2025	First version of the document, description of application for generating keys
1.1	10.04.2025	Finished section 1. Project - control term, added link to GitHub repository
2.0	01.06.2025	Added description of application for signing PDF document and signature verification and all necessary information for the final term
2.1	07.06.2025	Added description of the code for the signature_app.py

1. Project – control term

1.1 Description

The main task of the project is to design and develop an application to make a qualified electronic signature according to PAdES (PDF Advanced Electronic Signature) standard concept. The project consists of two applications - main application verifying qualified electronic signatures and the other one for generating keys which are encrypted with the user's PIN number.

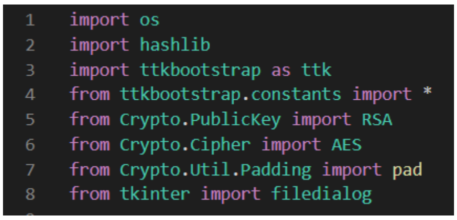
1.2 Results

GitHub repository

https://github.com/MrKtosiek/BSK-PDF-Signature

Technologies used

Both applications are written in Python with usage of Python Cryptography Toolkit (pycrypto) library used for encrypting data and generating RSA keys. For user interface we used ttkbootstrap library.



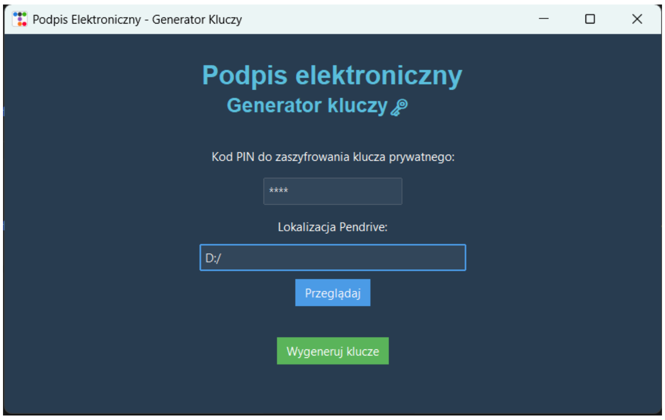
```python
import os
import hashlib
import ttkbootstrap as ttk
from ttkbootstrap.constants import *
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from tkinter import filedialog
```

Used dependencies

Application for generating keys

This application is used for generating a public key which is saved in the .pem format and a private key which is encrypted with the 256-bit AES cipher algorithm. The user has to enter a PIN number - it's important not to forget it because later the PIN is used to decrypt the private RSA key before signing a document.

- Encrypted private key is stored on a pendrive and can be used in order to sign a PDF document.
- Public key is used to verify the authenticity of a signed document



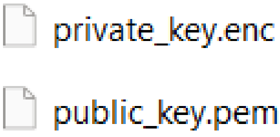
Podpis Elektroniczny - Generator Kluczy

Podpis elektroniczny

Generator kluczy 🔑

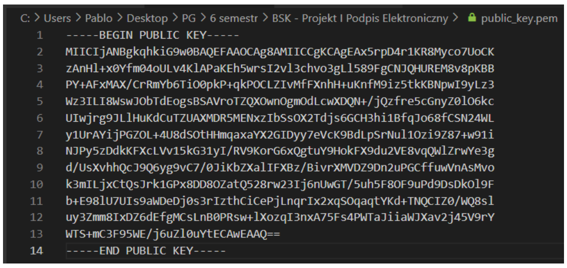
Kod PIN do zaszyfrowania klucza prywatnego: ****

Lokalizacja Pendrive: D:/



private_key.enc

public_key.pem



C: > Users > Pablo > Desktop > PG > 6 semestr > BSK - Projekt I Podpis Elektroniczny > public_key.pem

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAx5rpD4r1KR8Myco7UoCK
zAnHl+x0Yfm04oULv4KlAPaKEh5wrsI2vl3chvo3gLl589FgCNJQHUREM8v8pKBB
PY+AFxMAX/CrRmYb6TiO0pkP+qkPOCLZIvMfFXnhH+uKnfM9iz5tkKBNpwI9yLz3
Wz3ILI8WswJObTdEogsBSAVroTZQXOwnOgmOdLcwXDQN+/jQzfre5cGnyZ0lO6kc
UIwjrg9JLlHuKdCuTZUAXMDR5MENxzIbSsOX2Tdjs6GCH3hi1BfqJo68fCSN24WL
y1UrAYijPGZOL+4U8dSOtHHmqaxaYX2GIDyy7eVcK9BdLpSrNul1Ozi9Z87+w91i
NJPy5zDdkKFXcLVv15kG31yI/RV9KorG6xQgtuY9HokFX9du2VE8vqQWlZrwYe3g
d/UsXvhhQcJ9Q6yg9vC7/0JiKbZXalIFXBz/BivrXMVDZ9Dn2uPGCffuwVnAsMvo
k3mILjxCtQsJrk1GPx8DD8OZatQ528rw23Ij6nUwGT/5uh5F8OF9uPd9DsDkOl9F
b+E98lU7UIs9aWDeDj0s3rIzthCiCePjLnqrIx2xqSOqaqtYKd+TNQCIZ0/WQ8sl
uy3Zmm8IxDZ6dEfgMCsLnB0PRsw+lXozqI3nxA75Fs4PWTaJiiaWJXav2j45V9rY
WTS+mC3F95WE/j6uZl0uYtECAwEAAQ==
-----END PUBLIC KEY-----
```

1.3 Summary

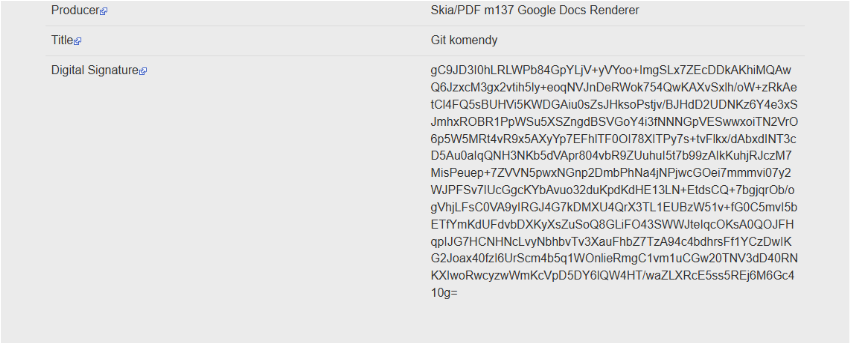
In the first stage of the project (control term), we worked on implementing an application for generating keys. The next stage will be to create an application for making electronic signatures and verifying them.

2. Project – Final term

2.1 Description

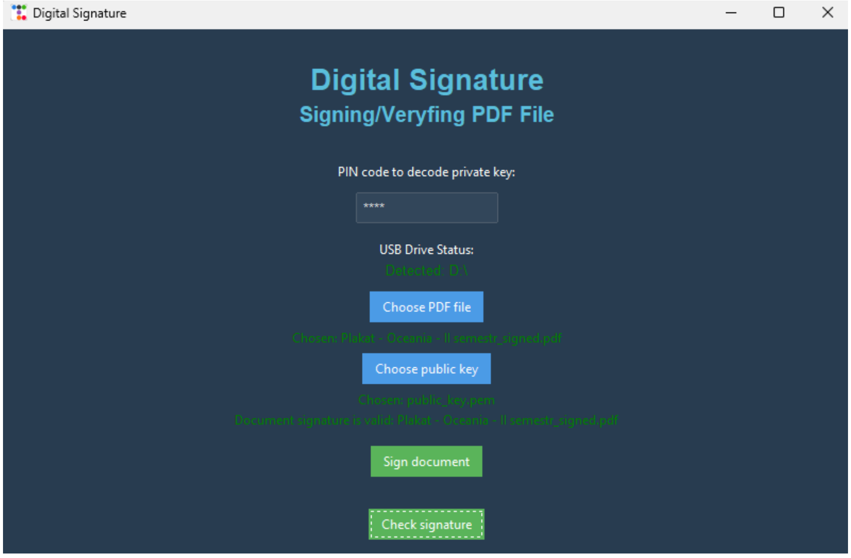
In this stage of the project we worked on implementing an application responsible for signing documents and verifying signatures. The application includes features such as automatic detection of a USB drive and encryption of the private key with a PIN code. After signing a document a new copy is being created (name+"_signed".pdf) - the content of the document remains unchanged, however inside the metadata a new field with the digital signature is being created.

Digital signature inside the metadata of the signed file:



Producer	Skia/PDF m137 Google Docs Renderer
Title	Git komendy
Digital Signature	gC9JD3I0hLRLWPb84GpYLjV+yVYoo+ImgSLx7ZEcDDkAKhiMQAw Q6JzxcM3gx2vtih5ly+eoqNVJnDeRWok754QwKAXvSxlh/oW+zRkAe tCl4FQ5sBUHVi5KWDGAiu0sZsJHksoPstjv/BJHdD2UDNKz6Y4e3xS JmhxROBR1PpWSu5XSZngdBSVGoY4i3fNNNGpVESwwxoiTN2VrO 6p5W5MRt4vR9x5AXyYp7EFhITF0OI78XITPy7s+tvFlkx/dAbxdINT3c D5Au0alqQNH3NKb5dVApr804vbR9ZUuhul5t7b99zAlkKuhjRJczM7 MisPeuep+7ZVVN5pwxNGnp2DmbPhNa4jNPjwcGOei7mmmvi07y2 WJPFSv7IUcGgcKYbAvuo32duKpdKdHE13LN+EtdsCQ+7bgjqrOb/o gVhjLFsC0VA9yIRGJ4G7kDMXU4QrX3TL1EUBzW51v+fG0C5mvI5b ETfYmKdUFdvbDXKyXsZuSoQ8GLiFO43SWWJtelqcOKsA0QOJFH qpIJG7HCNHNcLvyNbhbvTv3XauFhbZ7TzA94c4bdhrsFf1YCzDwIK G2Joax40fzl6UrScm4b5q1WOnlieRmgC1vm1uCGw20TNV3dD40RN KXIwoRwcyzwWmKcVpD5DY6lQW4HT/waZLXRcE5ss5REj6M6Gc4 10g=

Digital signature successful verification:



PIN code to decode private key: ****

USB Drive Status:
Detected: D:\

Chosen: Plakat - Oceania - II semestr_signed.pdf

Chosen: public_key.pem

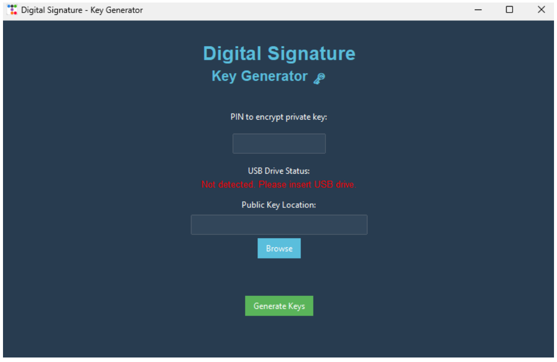
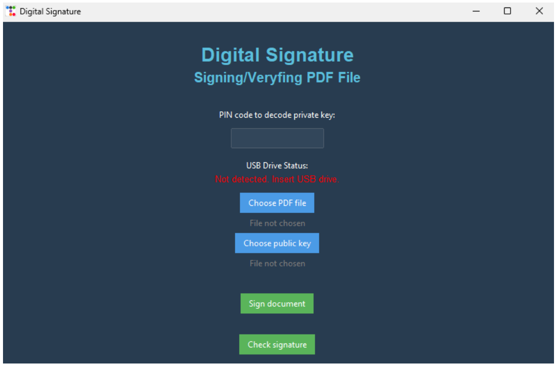
Document signature is valid: Plakat - Oceania - II semestr_signed.pdf

2.2 Code Description

Application for generating keys (generate_keys.py)	
Function	**Description**
auto_check_pendrive()	Automatically checks for pendrive every second.
check_pendrive()	Checks for pendrive presence and updates status.
find_pendrive():	Finds the removable drive (pendrive) connected to the system.
generate_and_save_keys()	Handles the process of generating and saving RSA keys.
save_keys(encrypted_private_key, public_key, pendrive_path, public_key_path, message_label)	Saves encrypted private key to pendrive and public key to specified location.
encrypt_private_key_with_pin(private_key, pin)	Encrypts the private key using AES encryption with a user-defined PIN.
generate_rsa_keys()	Generates a pair of RSA keys (private and public).

Application for signing PDF document and signature verification (signature_app.py)	
Function	**Description**
find_pendrive()	Finds the removable drive (pendrive) connected to the system.
load_private_key_from_pendrive(pendrive_path)	Loads an encrypted private key from a connected USB pendrive.
decrypt_private_key(pin, encrypted_private_key)	Decrypts an RSA private key using a PIN-derived key.
sign_pdf(file_path, private_key)	Digitally signs a PDF file using the provided RSA private key.
check_signature(file_path, key_path)	Verifies the digital signature of a PDF file using the provided RSA public key.

2.3 Description

Application for generating public and private keys *(generate_keys.py)*	**Application for signing PDF document and signature verification** *(signature_app.py)*
 Digital Signature - Key Generator Digital Signature Key Generator PIN to encrypt private key: USB Drive Status: Public Key Location: Browse Generate Keys Not detected. Please insert USB drive.	 Digital Signature Digital Signature Signing/Veryfing PDF File PIN code to decode private key: USB Drive Status: Choose PDF file Choose public key Sign document Check signature Not detected. Insert USB drive. File not chosen File not chosen
● generation of public key (not encrypted) ● generation of private key (encrypted with user's PIN)	● signing a document with the private key (user has to enter PIN to decrypt it) ● veryfying a signed document (only public key is required)
● automatically detecting user's pendrive	

Code for the application responsible for generating keys is in the file named "generate_keys.py". Code for the other application (verifying and signing the document) is in the file "signature_app.py". All functions have wider descriptions which are present in the documentation generated with Doxygen.

Example scenario of application usage:

1. User A opens the key generation application.
2. User A generates a public and private key pair, the private key is encrypted with a PIN code and saved to a USB drive.
3. The public key is saved and can be shared with other users.
4. User A opens the signing application.
5. The application automatically detects the USB drive with the encrypted private key.
6. User A enters the correct PIN code.
7. User A selects a document and signs it using the private key.
8. The signature is saved in the document's metadata.
9. User B receives the signed document and the public key from User A.

10. User B opens the application and selects the signed document.
11. User B selects the public key received from User A.
12. The application reads the signature from the document's metadata.
13. The application verifies the signature using the public key.
14. The application confirms whether the signature is valid or not.

2.4 Results

GitHub repository
https://github.com/MrKtosiek/BSK-PDF-Signature

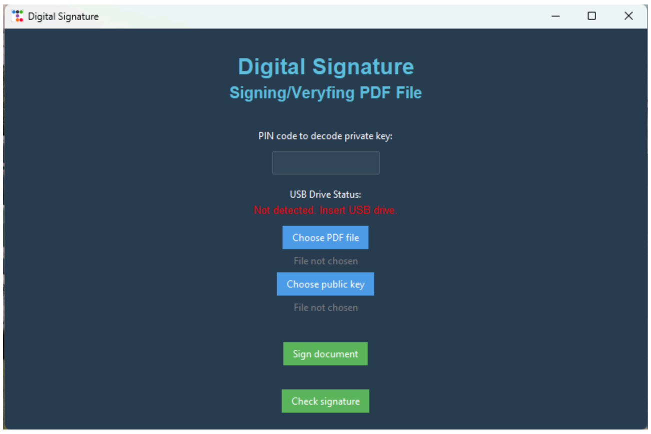
Application for generating keys
Functionalities of this application haven't been changed since the first stage of the project. It allows generating a public and a private key. Changes in code include slight refactorization and comments for functions (for Doxygen documentation).

Application for signing PDF document and signature verification

This application includes the following functionalities:

- signing a document after generating a public and a private key (the private key is encrypted with a PIN)
- verifying a signed document using the public key shared by the person who signed it
- automatically detecting a connected USB drive with the private key when trying to sign a document
- storing the signature in the file's metadata so that the internal structure of the file is not changed

State before pendrive detection:



Digital Signature

Digital Signature
Signing/Veryfing PDF File

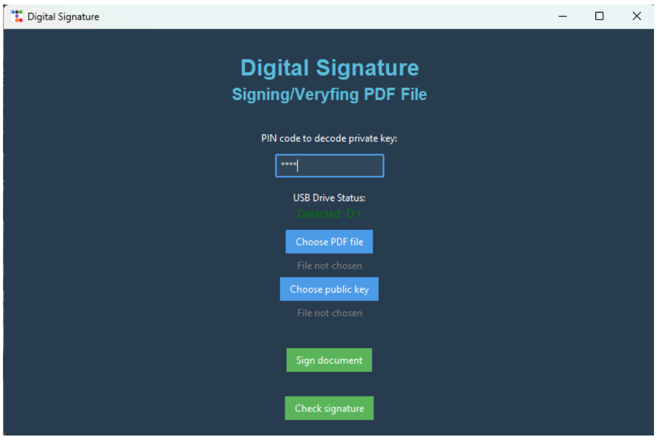
PIN code to decode private key:

USB Drive Status:
Not detected. Insert USB drive.

File not chosen

File not chosen

After pendrive detection:



Digital Signature

Digital Signature
Signing/Veryfing PDF File

PIN code to decode private key:
****

USB Drive Status:
Detected: D:\

File not chosen

File not chosen

2.5 Summary

Both applications are simple but useful and work as intended. They allow the user to sign as well as verify the signature - it’s a common practice especially if documents are important ex. VAT invoices/payment confirmations etc. After verifying the signature you can be sure that the document is legitimate. The project can definitely be considered a success. We managed to create a simple and lightweight application for electronic signatures.

3. Literature

[1] https://ttkbootstrap.readthedocs.io/en/latest/

[2] https://www.doxygen.nl/manual/lists.html

[3] https://pypdf2.readthedocs.io/en/3.x/

[4] https://docs.python.org/3/library/crypto.html