Security of Computer Systems

Project Report

Authors:
Jakub, Bronowski, 193208
Piotr, Trybisz, 193557

Version: 1.2

Versions

Version	Date	Description of changes
1.0	08.04.2025	Creation of the document.
1.1	08.04.2025	Added pictures.
1.2	09.04.2025	Filling parts necessary to pass control term.

1. Project – control term

1.1 Description

The main aim of this project is to create an app which will allow user to sign and verify the PDF document signage. Also, app has to allow user to generate a pair of RSA keys to make all operations possible.

1.2 Results

App was created in Python with usage of PyQT5 UI library. App is also available via TUI, which was created earlier in the process.

App consists of 3 functional parts:

1. Key management (fig. 1)

Here, user can set paths to private and public keys. Colours of the text next to textboxes are signalising if keys are present in given places. Also, here user sets paths where new keys will be stored.

When user set paths, user is asked to enter the password (PIN, can contains letters) to unlock access to files or to create new set of keys. If password is incorrect, user is asked again and files are not accessible in the app. Worth mentioning is a fact that if user creates new pair of keys, user has to remember PIN number. It is impossible to set new password to already existing keys, so that's why in this case PIN popup window is showed just once. Field to enter the password is placed in popup window (fig. 3). Process of key generation takes some time, that is why popup windows informing about executed processes are displayed (fig. 4 and fig. 5).

2. Document signing (fig. 1)

Here user can enter path to file user wants to sign and where to store signed file. It is possible to overwrite not signed file. Operation status is shown to user in popup window (fig. 6). Signature is added to the end of the PDF file, past last *EOF* tag. Signature contains encrypted SHA-256 checksum of original PDF document. Only *.pdf* documents can be signed.

3. Document verifying (fig. 2)

Here user can verify if given document is signed and not modified after. User has to specify path to PDF file. If document wasn't changed after signing, positive result is shown to user in popup window (fig. 7). If PDF verification failed, popup window is also displayed, but with appropriate text (fig. 8). Only *.pdf* documents can be verified. Verification consist of splitting signed file to its content and signature. Encrypted SHA-256 checksum is calculated and checked if matches file signature.

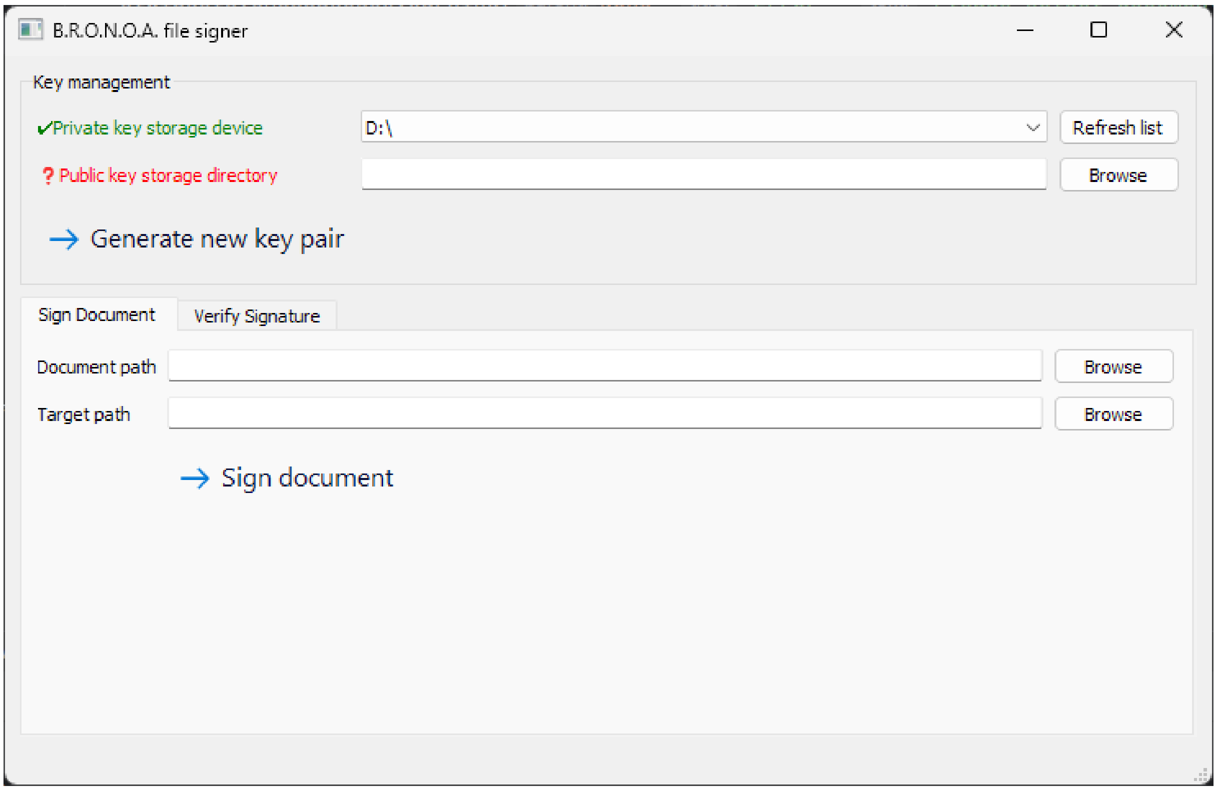


Fig. 1 – App main screen.

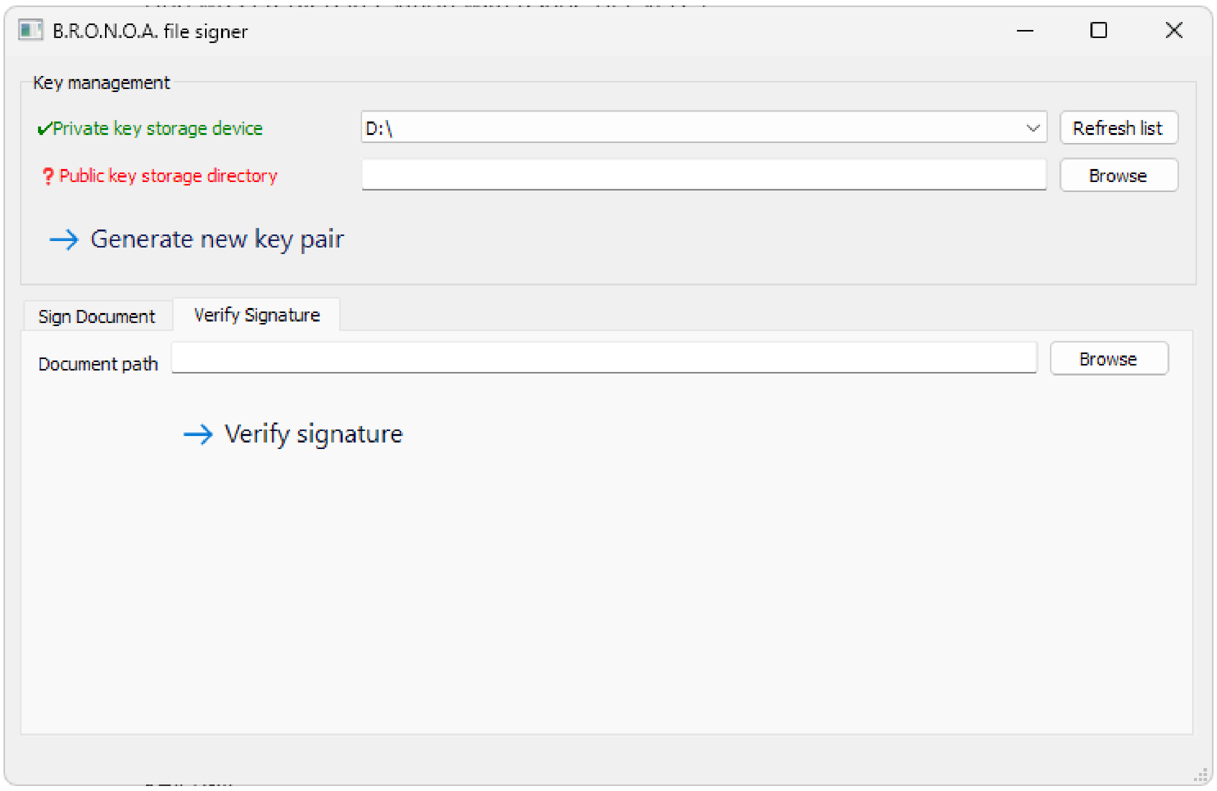


Fig. 2 – Signature verifying screen.

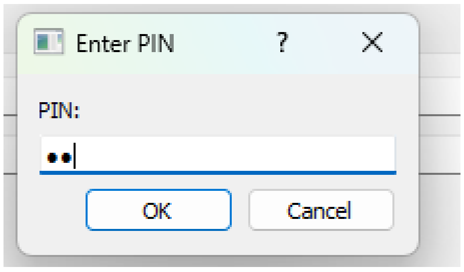
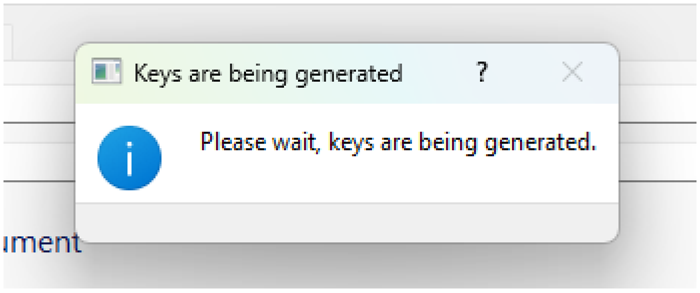
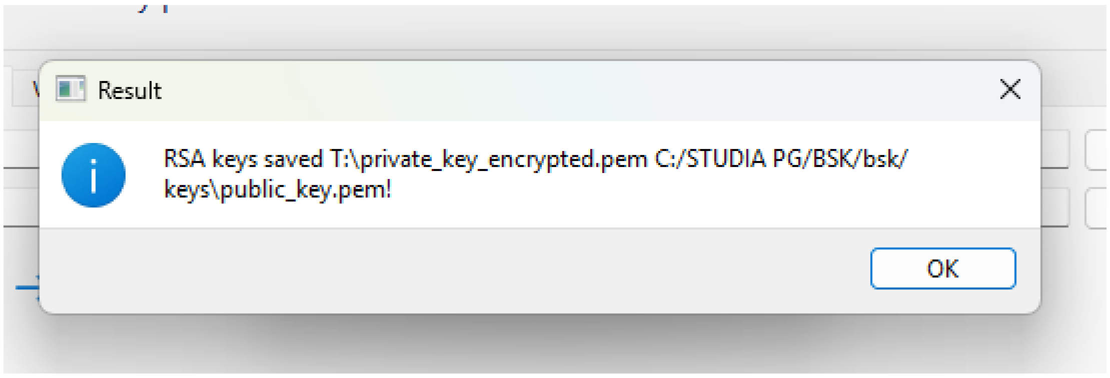


Fig. 3 – PIN window.


Keys are being generated

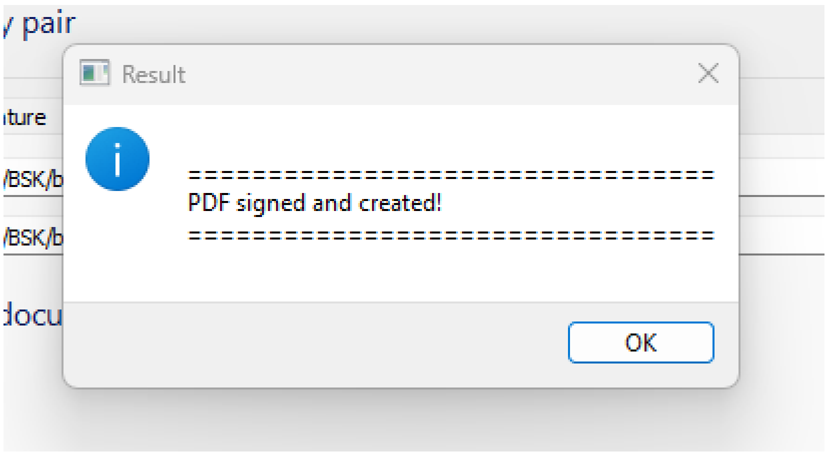
Please wait, keys are being generated.

Fig. 4 – Key generation popup window.


Result

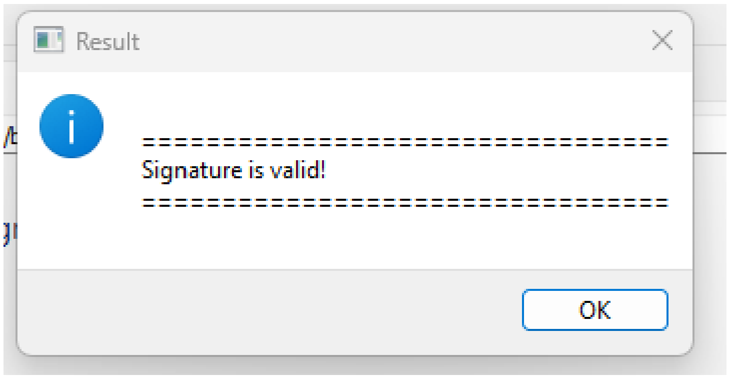
RSA keys saved T:\private_key_encrypted.pem C:/STUDIA PG/BSK/bsk/keys\public_key.pem!

Fig. 5 – Success of key generation.


Result

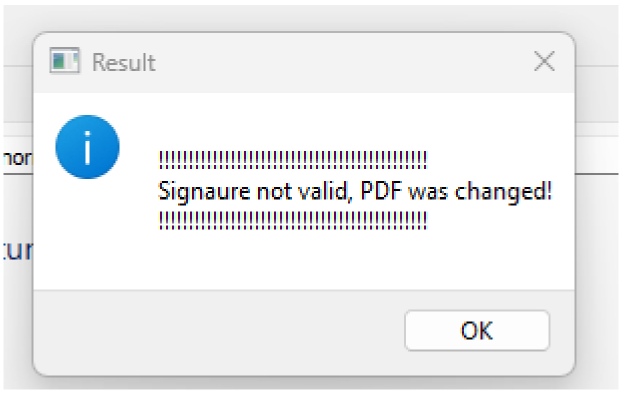
```
==================================
PDF signed and created!
==================================
```

Fig. 6 – Success of creating signed PDF.



```
==================================
Signature is valid!
==================================
```

Fig. 7 – Success of PDF verification.



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Signaure not valid, PDF was changed!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Fig. 8 – Failure of PDF verification.

1.3 Summary

Application works in expected way. All critical methods are implemented using *try… except* mechanism to handle errors during its execution. App was tested manually.

2. Project – Final term

2.1 Description

Content

2.2 Code Description

Content

```
/*!
  *  A list of events:
  *  <ul>
  *  <li> mouse events
  *     <ol>
  *     <li>mouse move event
  *     <li>mouse click event<br>
  *         More info about the click event.
  *     <li>mouse double click event
  *     </ol>
  *  <li> keyboard events
  *     <ol>
  *     <li>key down event
  *     <li>key up event
  *     </ol>
  *  </ul>
  *  More text here.
  */
```

List. 1 – Code listing [2].

Final Content.

2.3 Description

Content

2.4 Results

Content

2.5 Summary

Content

3. Literature

[1] https://en.wikipedia.org/wiki/PAdES [accessed on 08.04.2025]

[2] Project instruction.