BSK 2025

Generated by Doxygen 1.13.2

Chapter 1

BSK_project_2025

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 auxiliary_app Namespace Reference

Classes

- class AuxiliaryApp

Variables

- level
- root = Tk()
- k = AuxiliaryApp(root)

5.1.1 Variable Documentation

5.1.1.1 k

```
auxiliary_app.k = AuxiliaryApp(root)
```

5.1.1.2 level

```
auxiliary_app.level
```

5.1.1.3 root

```
auxiliary_app.root = Tk()
```

5.2 functionality Namespace Reference

Functions

- generate_rsa_key ()
- derive_aes_key (pin)
- encrypt_private_key (private_key, aes_key)
- decrypt_private_key (encrypted_data, aes_key)
- create_keys (pin, file_path)
- save_public_key (public_key, output_file)
- save_encrypted_private_key (encrypted_private_key, output_file)
- prepare_public_key (file_path)
- load_and_decrypt_private_key (file_path, pin)
- verify_pdf (pdf_file_path, public_key)
- create_cert (private_key, save=False)
- sign_pdf (pdf_file_path, cert, key, change_name=False)
- verify_is_pdf_signed (pdf_file_path)
- sign_pdf_full (pdf_file_path, key)

Variables

- level

5.2.1 Function Documentation

5.2.1.1 create_cert()

```
functionality.create_cert (
              private_key,
              save = False)
```

```
Generate a self-signed X.509 certificate using the provided RSA private key.

Parameters:
    private_key (rsa.RSAPrivateKey): The private key to sign the certificate.
    save (bool): If True, saves the certificate to a file named 'rsa_cert.pem'.

Returns:
    x509.Certificate: A self-signed X.509 certificate.

Side Effects:
    - Optionally writes the certificate to 'rsa_cert.pem'.
    - Logs debug messages including key types and certificate creation steps.
```

5.2.1.2 create_keys()

```
functionality.create_keys (
              pin,
              file_path)
```

```
Generate RSA key pair, derive AES key from PIN, encrypt the private key, and save both keys to files.

Parameters:
    pin (str): User-provided PIN used to derive the AES key.
    file_path (str): Path where the encrypted private key will be saved (PEM format).

Side Effects:
    - Saves the encrypted private key to `file_path`.
    - Saves the public key to `file_path` with `_pub.pem` suffix.
    - Logs debug messages about the operation.
    - Logs errors if key creation or file operations fail.

Returns:
    None

Raises:
    Logs exceptions internally but does not propagate them.
```

5.2.1.3 decrypt_private_key()

```
functionality.decrypt_private_key (
              encrypted_data,
              aes_key)
```

```
Decrypt private RSA key using AES key.

Parameters:
    encrypted_data (bytes): byte string consisting of a 16-byte iv and AES-encrypted RSA key
    aes_key (bytes): The AES key (32 bytes) used for decryption.

Returns:
    rsa.RSAPrivateKey: RSA private key object.
```

5.2.1.4 derive_aes_key()

```
functionality.derive_aes_key (
              pin)
```

```
Derive a 256-bit SHA key.

Parameters:
    pin (str): User-provided pin.

Returns:
    bytes: A 32-byte AES key derived from the SHA-256 hash of the pin.
```

5.2.1.5 encrypt_private_key()

```
functionality.encrypt_private_key (
              private_key,
              aes_key)
```

```
Encrypt private RSA key using AES key.

Parameters:
    private_key (rsa.RSAPrivateKey): The RSA private key object to encrypt.
    aes_key (bytes): AES bytes.

Returns:
    bytes: A byte string with initialization vector and encrypted RSA private key
```

5.2.1.6 generate_rsa_key()

```
functionality.generate_rsa_key ()
```

```
Generate a 4096-bit RSA private key.

Uses a public exponent of 65537 and the default cryptographic backend.

Returns:
    rsa.RSAPrivateKey: A newly generated RSA private key object.
```

5.2.1.7 load_and_decrypt_private_key()

```
functionality.load_and_decrypt_private_key (
              file_path,
              pin)
```

```
Reads encrypted private key from .pem file and returns it in DER encoding

Parameters:
    file_path (str): Path to the PEM-formatted public key file.

Returns:
    rsa.RSAPrivateKey: RSA private key object.

Side Effects:
    - Logs debug messages about the operation.
```

5.2.1.8 prepare_public_key()

```
functionality.prepare_public_key (
              file_path)
```

```
Reads public key from .pem file and returns it in DER encoding

Parameters:
    file_path (str): Path to the PEM-formatted public key file.

Returns:
    bytes: RSA public key bytes with DER encoding format.

Side Effects:
    - Logs debug messages about the operation.
```

5.2.1.9 save_encrypted_private_key()

```
functionality.save_encrypted_private_key (
              encrypted_private_key,
              output_file)
```

```
Save an AES-encrypted RSA private key to a binary file.

Parameters:
    encrypted_private_key (bytes): Encrypted private key data.
    output_file (str): Path to the output file.

Side Effects:
    - Writes the encrypted key to the file.
    - Logs the save operation.

Returns:
    None
```

5.2.1.10 save_public_key()

```
functionality.save_public_key (
              public_key,
              output_file)
```

```
Save an RSA public key to a PEM-formatted file.

Parameters:
    public_key (rsa.RSAPublicKey): The RSA public key to save.
    output_file (str): Path to the output file.

Side Effects:
    - Writes the public key in PEM format to the file.
    - Logs the save operation.

Returns:
    None
```

5.2.1.11 sign_pdf()

```
functionality.sign_pdf (
              pdf_file_path,
              cert,
              key,
              change_name = False)
```

```
Digitally sign a PDF using an X.509 certificate and RSA private key.

Parameters:
    pdf_file_path (str): Path to the PDF file to be signed.
    cert (x509.Certificate): The X.509 certificate used for signing.
    key (rsa.RSAPrivateKey): The RSA private key corresponding to the certificate.
    change_name (bool): If True, output file will be named with '_signed.pdf' suffix.

Side Effects:
    - Writes a new signed PDF file to disk (overwrites input if `change_name` is False).
    - Logs and suppresses exceptions during signing.

Returns:
    bool: information if PDF was signed
```

5.2.1.12 sign_pdf_full()

```
functionality.sign_pdf_full (
              pdf_file_path,
              key)
```

```
Create a certificate and use it to sign a PDF.

This function generates a self-signed certificate from the given RSA private key
and uses it to apply a digital signature to the specified PDF file.

Parameters:
    pdf_file_path (str): Path to the PDF file to be signed.
    key (rsa.RSAPrivateKey): The RSA private key used for signing.

Side Effects:
    - Modifies the PDF file at ‘pdf_file_path‘ by adding a digital signature.
```

5.2.1.13 verify_is_pdf_signed()

```
functionality.verify_is_pdf_signed (
              pdf_file_path)
```

```
Check if chosen PDF is signed.

Parameters:
    pdf_file_path (str): Path to the signed PDF file.

Returns:
    bool: True if the PDF contains any signature.
```

5.2.1.14 verify_pdf()

```
functionality.verify_pdf (
              pdf_file_path,
              public_key)
```

```
Verify the digital signature of a signed PDF against a provided public key.

Parameters:
    pdf_file_path (str): Path to the signed PDF file.
    public_key (Crypto.PublicKey.RSA.RsaKey): RSA public key to compare with the signer’s certificate.

Returns:
    bool: True if the signature is cryptographically valid and matches the provided public key, False otherwis

Side Effects:
    - Logs debugging information about the verification process.
    - Reads and parses the PDF file.

Notes:
    - The function compares SHA-256 hashes of the provided public key to hash from the certificate.
    - Requires the PDF to have exactly one embedded signature.
```

5.2.2 Variable Documentation

5.2.2.1 level

```
functionality.level
```

5.3 main_app Namespace Reference

Classes

- class EncryptionApp

Variables

- level
- root = Tk()
- app = EncryptionApp(root)

5.3.1 Variable Documentation

5.3.1.1 app

```
main_app.app = EncryptionApp(root)
```

5.3.1.2 level

```
main_app.level
```

5.3.1.3 root

```
main_app.root = Tk()
```

Chapter 6

Class Documentation

6.1 auxiliary_app.AuxiliaryApp Class Reference

Public Member Functions

- __init__ (self, root)
- submit (self)
- create_keys (self)
- choose_location (self)

Public Attributes

- str pin = ""
- file_path = None
- root = root
- message_key = StringVar()
- label_key = Label(root, textvariable=self.message_key)
- message_pin = StringVar()
- label_pin = Label(root, textvariable=self.message_pin)
- pin_var = StringVar()

6.1.1 Constructor & Destructor Documentation

6.1.1.1 __init__()

```
auxiliary_app.AuxiliaryApp.__init__ (
              self,
              root)
```

```
Initialize the RSA Key Generator app.
This class sets up a simple Tkinter GUI to collect a 4-digit PIN and a file path to save a generated RSA priva

Parameters:
    root (Tk): The root Tkinter window.

Attributes:
    root (Tk): The root Tkinter window.
    message_key(StringVar): Stores the status or instruction message for key / key path.
    label_key(Label): Displays messages to the user.
    message_pin(StringVar): Stores the status or instruction message for pin.
    label_pin(Label): Displays messages to the user.

    pin_var (StringVar): Tkinter variable linked to the PIN entry widget.
    pin (str): 4-digit PIN entered by the user.
    file_path (str or None): Path to save the generated private key.
```

6.1.2 Member Function Documentation

6.1.2.1 choose_location()

```
auxiliary_app.AuxiliaryApp.choose_location (
              self)
```

```
A file save dialog is shown for the user to select a `.pem` file location.

Side Effects:
    - Updates `self.file_path` with the selected or provided path.
    - Logs debug messages about file selection.
```

6.1.2.2 create_keys()

```
auxiliary_app.AuxiliaryApp.create_keys (
              self)
```

```
Create RSA keys using the provided PIN and file path.

This method delegates key generation to the `functionality.create_keys()`
function, passing the  PIN and selected file path and
updates the GUI message before and after key creation.

Side Effects:
    - Calls `functionality.create_keys(self.pin, self.file_path)` to generate keys.
    - Updates the status message shown in the GUI via `self.message`.
    - Logs debug messages about key creation.
```

6.1.2.3 submit()

```
auxiliary_app.AuxiliaryApp.submit (
              self)
```

```
Handle the submission of the PIN and key file location.

Validates that:
    - The PIN is not empty.
    - The PIN has 4 characters.
    - A file path has been selected.

If all checks pass, it calls `create_keys()` to generate RSA keys.
Otherwise, displays appropriate messages to the user.

Side Effects:
    - Updates `self.pin`, `self.message`, and `self.file_path`.
    - Clears the PIN entry field after key creation.
```

6.1.3 Member Data Documentation

6.1.3.1 file_path

```
auxiliary_app.AuxiliaryApp.file_path = None
```

6.1.3.2 label_key

```
auxiliary_app.AuxiliaryApp.label_key = Label(root, textvariable=self.message_key)
```

6.1.3.3 label_pin

```
auxiliary_app.AuxiliaryApp.label_pin = Label(root, textvariable=self.message_pin)
```

6.1.3.4 message_key

```
auxiliary_app.AuxiliaryApp.message_key = StringVar()
```

6.1.3.5 message_pin

```
auxiliary_app.AuxiliaryApp.message_pin = StringVar()
```

6.1.3.6 pin

```
auxiliary_app.AuxiliaryApp.pin = ""
```

6.1.3.7 pin_var

```
auxiliary_app.AuxiliaryApp.pin_var = StringVar()
```

6.1.3.8 root

```
auxiliary_app.AuxiliaryApp.root = root
```

The documentation for this class was generated from the following file:

- auxiliary_app.py

6.2 main_app.EncryptionApp Class Reference

Public Member Functions

- __init__ (self, root)
- get_usb_drives (self)
- monitor_usb (self)
- on_usb_inserted (self, drive_path)
- find_pem_files (self, directory)
- submit_pin (self)
- prepare_private_key (self)
- choose_pdf (self)
- choose_pub_key (self)
- prepare_public_key (self, file_path)
- sign_pdf (self)
- verify_pdf (self)

Public Attributes

- root = root
- message_pdf = StringVar()
- message_private_key = StringVar()
- message_public_key = StringVar()
- message_general = StringVar()
- str pin = ""
- str pdf_file_path = ""
- private_key_path = None
- public_key_path = None
- private_key = None
- public_key = None
- cert = None
- pin_var = StringVar()
- pin_entry = Entry(root, textvariable=self.pin_var, font=("calibre", 10, "normal"), show="∗")
- submit_pin = Button(root, text="Submit", command=self.submit_pin)
- choose_pdf_btn = Button(root, text="Choose PDF", command=self.choose_pdf)
- sign_pdf_btn = Button(root, text="Sign", command=self.sign_pdf)
- verify_btn = Button(root, text="Verify", command=self.verify_pdf)
- choose_pub_key_btn = Button(root, text="Choose public key", command=self.choose_pub_key)
- label_pdf = Label(root, textvariable=self.message_pdf)
- label_private_key = Label(root, textvariable=self.message_private_key)
- label_public_key = Label(root, textvariable=self.message_public_key)
- label_general = Label(root, textvariable=self.message_general)
- detected_drives = set(self.get_usb_drives())
- monitor_thread = threading.Thread(target=self.monitor_usb, daemon=True)
- on_usb_inserted

6.2.1 Constructor & Destructor Documentation

6.2.1.1 __init__()

```
main_app.EncryptionApp.__init__ (
              self,
              root)
```

```
GUI application for PDF signing and signature verification using RSA and AES encryption.

This application provides a graphical interface for:
- Entering a PIN to decrypt an RSA private key.
- Selecting a PDF file to sign or verify.
- Choosing a public key file for signature verification.
- Automatically detecting USB drives to retrieve keys.

Parameters:
    root (Tk): The root Tkinter window.

Attributes:
    root (Tk): Main Tkinter window.
    pin (str): PIN used to derive AES key.
    pdf_file_path (str): Path to the selected PDF file.
    private_key_path (str or None): Path to the detected encrypted private key file.
    public_key_path (str or None): Path to the selected public key file.
    private_key (RSAPrivateKey or None): Decrypted private RSA key.
    public_key (RSAPublicKey or None): Loaded public RSA key.
    cert (x509.Certificate or None): Certificate generated for signing.
    pin_var (StringVar): Linked to the PIN entry widget.
    pin_entry (Entry): PIN input widget.
    submit_pin (Button): Button to submit the PIN.
    choose_pdf_btn (Button): Button to select a PDF for signing/verifying.
    sign_pdf_btn (Button): Button to sign the selected PDF.
    verify_btn (Button): Button to verify the PDF signature.
    choose_pub_key_btn (Button): Button to select a public key file.

    message_pdf (StringVar): Status or instruction message_pdf displayed to the user.
    message_private_key (StringVar): Message displaying key detection info.
    message_public_key (StringVar): Message displaying chosen public key info.
    message_general (StringVar): Message displaying general info.

    label_pdf (Label): Displays general messages to the user.
    label_private_key (Label): Displays key detection status.
    label_public_key (Label): Displays key detection status.
    label_general (Label): Displays general messages to the user.

    detected_drives (set): Set of USB drives detected on launch.
    monitor_thread (Thread): Background thread to monitor USB drive changes.
```

6.2.2 Member Function Documentation

6.2.2.1 choose_pdf()

```
main_app.EncryptionApp.choose_pdf (
              self)
```

```
Open a file dialog to choose PDF.

Side Effects:
- PDF file path is saved.
Logs debug information.
```

6.2.2.2 choose_pub_key()

```
main_app.EncryptionApp.choose_pub_key (
              self)
```

```
Open a file dialog to choose public key.

Side Effects:
- Public key is loaded into memory.
- Logs debug messages about the operation.
```

6.2.2.3 find_pem_files()

```
main_app.EncryptionApp.find_pem_files (
              self,
              directory)
```

```
Continuously monitor USB drives for new insertions.

Parameters:
    directory (string): path to directory with private key.

Checks all files in given directory and returns ones with .pem extension.

Returns:
    list: A list of files with .pem extension.
```

6.2.2.4 get_usb_drives()

```
main_app.EncryptionApp.get_usb_drives (
              self)
```

```
Detect currently mounted USB drives.

Uses `psutil.disk_partitions()` to scan for removable drives by checking
the 'opts' field of each partition.

Returns:
    list: A list of mount points for all detected USB drives.
```

6.2.2.5 monitor_usb()

```
main_app.EncryptionApp.monitor_usb (
              self)
```

```
Continuously monitor USB drives for new insertions.

This method runs in a background thread and periodically (every 2 seconds) checks
the currently mounted USB drives by calling `get_usb_drives()`.

When a new USB drive is detected, it schedules a call to `on_usb_inserted` on the main
Tkinter thread using `root.after` to safely handle UI updates.

This loop runs indefinitely as a daemon thread.
```

6.2.2.6 on_usb_inserted()

```
main_app.EncryptionApp.on_usb_inserted (
              self,
              drive_path)
```

```
Continuously monitor USB drives for new insertions.

Parameters:
    drive_path (string): path to detectedd USB.

This method runs in a background thread and periodically (every 2 seconds) checks
the currently mounted USB drives by calling `get_usb_drives()`.

When a new USB drive is detected, it schedules a call to `on_usb_inserted` on the main
Tkinter thread using `root.after` to safely handle UI updates.

This loop runs indefinitely as a daemon thread.
```

6.2.2.7 prepare_private_key()

```
main_app.EncryptionApp.prepare_private_key (
              self)
```

```
Load and decrypt private key using pin.

Side Effects:
- Private key is decrypted and loaded into memory.
- Logs encryption status.
```

6.2.2.8 prepare_public_key()

```
main_app.EncryptionApp.prepare_public_key (
              self,
              file_path)
```

```
Open a file dialog to choose public key. Key format is checked and if it is supported key is loaded into memor

Parameters:
    file_path (str): Path to public key.

Side Effects:
- Public key is loaded into memory.
- Logs debug messages about the operation.
```

6.2.2.9 sign_pdf()

```
main_app.EncryptionApp.sign_pdf (
              self)
```

```
Sign pdf with `sign_pdf_full()` function.

Side Effects:
- PDF is signed.
- Logs debug messages about the operation.
- Shows approriate messages.
```

6.2.2.10 submit_pin()

```
main_app.EncryptionApp.submit_pin (
              self)
```

```
Handle the submission of the PIN, private key must be detected for the function to work.

Loads private key from detected file with  `prepare_private_key()` function.

Side Effects:
- Private key is loaded into memory.
- Logs pin length and key.
- Appropriate message_pdf is shown.
- PIN input is cleared.
```

6.2.2.11 verify_pdf()

```
main_app.EncryptionApp.verify_pdf (
              self)
```

```
Verify pdf with `verify_pdf()` function.

Side Effects:
- PDF is verified.
- Logs debug messages about the operation.
- Shows approriate messages.
```

6.2.3 Member Data Documentation

6.2.3.1 cert

```
main_app.EncryptionApp.cert = None
```

6.2.3.2 choose_pdf_btn

```
main_app.EncryptionApp.choose_pdf_btn = Button(root, text="Choose PDF", command=self.choose_↩
pdf)
```

6.2.3.3 choose_pub_key_btn

```
main_app.EncryptionApp.choose_pub_key_btn = Button(root, text="Choose public key", command=self.↩
choose_pub_key)
```

6.2.3.4 detected_drives

```
main_app.EncryptionApp.detected_drives = set(self.get_usb_drives())
```

6.2.3.5 label_general

```
main_app.EncryptionApp.label_general = Label(root, textvariable=self.message_general)
```

6.2.3.6 label_pdf

```
main_app.EncryptionApp.label_pdf = Label(root, textvariable=self.message_pdf)
```

6.2.3.7 label_private_key

```
main_app.EncryptionApp.label_private_key = Label(root, textvariable=self.message_private_key)
```

6.2.3.8 label_public_key

```
main_app.EncryptionApp.label_public_key = Label(root, textvariable=self.message_public_key)
```

6.2.3.9 message_general

```
main_app.EncryptionApp.message_general = StringVar()
```

6.2.3.10 message_pdf

```
main_app.EncryptionApp.message_pdf = StringVar()
```

6.2.3.11 message_private_key

```
main_app.EncryptionApp.message_private_key = StringVar()
```

6.2.3.12 message_public_key

```
main_app.EncryptionApp.message_public_key = StringVar()
```

6.2.3.13 monitor_thread

```
main_app.EncryptionApp.monitor_thread = threading.Thread(target=self.monitor_usb, daemon=True)
```

6.2.3.14 on_usb_inserted

```
main_app.EncryptionApp.on_usb_inserted
```

6.2.3.15 pdf_file_path

```
main_app.EncryptionApp.pdf_file_path = ""
```

6.2.3.16 pin

```
str main_app.EncryptionApp.pin = ""
```

6.2.3.17 pin_entry

```
main_app.EncryptionApp.pin_entry = Entry(root, textvariable=self.pin_var, font=("calibre", 10,
"normal"), show="*")
```

6.2.3.18 pin_var

```
main_app.EncryptionApp.pin_var = StringVar()
```

6.2.3.19 private_key

```
main_app.EncryptionApp.private_key = None
```

6.2.3.20 private_key_path

```
main_app.EncryptionApp.private_key_path = None
```

6.2.3.21 public_key

```
main_app.EncryptionApp.public_key = None
```

6.2.3.22 public_key_path

```
main_app.EncryptionApp.public_key_path = None
```

6.2.3.23 root

```
main_app.EncryptionApp.root = root
```

6.2.3.24 sign_pdf_btn

```
main_app.EncryptionApp.sign_pdf_btn = Button(root, text="Sign", command=self.sign_pdf)
```

6.2.3.25 submit_pin

```
main_app.EncryptionApp.submit_pin = Button(root, text="Submit", command=self.submit_pin)
```

6.2.3.26 verify_btn

```
main_app.EncryptionApp.verify_btn = Button(root, text="Verify", command=self.verify_pdf)
```

The documentation for this class was generated from the following file:

- main_app.py

Chapter 7

File Documentation

7.1 auxiliary_app.py File Reference

Classes

- class auxiliary_app.AuxiliaryApp

Namespaces

- namespace auxiliary_app

Variables

- auxiliary_app.level
- auxiliary_app.root = Tk()
- auxiliary_app.k = AuxiliaryApp(root)

7.2 functionality.py File Reference

Namespaces

- namespace functionality

Functions

- functionality.generate_rsa_key ()
- functionality.derive_aes_key (pin)
- functionality.encrypt_private_key (private_key, aes_key)
- functionality.decrypt_private_key (encrypted_data, aes_key)
- functionality.create_keys (pin, file_path)
- functionality.save_public_key (public_key, output_file)
- functionality.save_encrypted_private_key (encrypted_private_key, output_file)
- functionality.prepare_public_key (file_path)
- functionality.load_and_decrypt_private_key (file_path, pin)
- functionality.verify_pdf (pdf_file_path, public_key)
- functionality.create_cert (private_key, save=False)
- functionality.sign_pdf (pdf_file_path, cert, key, change_name=False)
- functionality.verify_is_pdf_signed (pdf_file_path)
- functionality.sign_pdf_full (pdf_file_path, key)

Variables

- functionality.level

7.3 main_app.py File Reference

Classes

- class main_app.EncryptionApp

Namespaces

- namespace main_app

Variables

- main_app.level
- main_app.root = Tk()
- main_app.app = EncryptionApp(root)

7.4 README.md File Reference

Index