

PAdES signing app

1.0

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 frames Namespace Reference	9
5.2 frames.generate_window Namespace Reference	9
5.2.1 Variable Documentation	9
5.2.1.1 ACTIVATE_BUTTON_COLOR	10
5.2.1.2 BACKGROUND2_COLOR	10
5.2.1.3 BACKGROUND_COLOR	10
5.2.1.4 BLUE_BUTTON_COLOR	10
5.2.1.5 FOREGROUND_COLOR	10
5.2.1.6 PRIVATE_KEY_NAME	10
5.2.1.7 PUBLIC_KEY_NAME	10
5.3 frames.signing Namespace Reference	11
5.3.1 Variable Documentation	12
5.3.1.1 ACTIVATE_BUTTON_COLOR	12
5.3.1.2 BACKGROUND2_COLOR	12
5.3.1.3 BACKGROUND_COLOR	12
5.3.1.4 BLUE_BUTTON_COLOR	12
5.3.1.5 DEFAULT_WRAP_LENGTH	12
5.3.1.6 FOREGROUND_COLOR	13
5.3.1.7 GO_BACK_BUTTON_TEXT	13
5.3.1.8 INITIAL_STATUS_TEXT	13
5.3.1.9 LARGE_FONT_CONFIG	13
5.3.1.10 PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT	13
5.3.1.11 PATHS_REQUIRED_ERROR_TEXT	13
5.3.1.12 PDF_FILE_TYPES	14
5.3.1.13 PDF_READ_ERROR_TEXT	14
5.3.1.14 RETRY_BUTTON_TEXT	14
5.3.1.15 SELECT_SOURCE_BUTTON_TEXT	14
5.3.1.16 SELECT_SOURCE_PDF_TITLE	14
5.3.1.17 SELECT_TARGET_BUTTON_TEXT	14
5.3.1.18 SELECT_TARGET_PDF_TITLE	15
5.3.1.19 SIGN_BUTTON_INITIAL_TEXT	15

5.3.1.20 SIGNING_ERROR_TEXT	15
5.3.1.21 SIGNING_SUCCESS_TEXT	15
5.3.1.22 SOURCE_PDF_LABEL_TEXT	15
5.3.1.23 TARGET_PDF_LABEL_TEXT	15
5.3.1.24 UNEXPECTED_SIGNING_ERROR_TEXT	16
5.4 frames.start Namespace Reference	16
5.4.1 Variable Documentation	16
5.4.1.1 ACTIVATE_BUTTON_COLOR	16
5.4.1.2 BACKGROUND2_COLOR	17
5.4.1.3 BACKGROUND_COLOR	17
5.4.1.4 BLUE_BUTTON_COLOR	17
5.4.1.5 BUTTON_PADDING_Y	17
5.4.1.6 DEFAULT_PADDING_X	17
5.4.1.7 DEFAULT_PADDING_Y	17
5.4.1.8 DEFAULT_WRAP_LENGTH	17
5.4.1.9 FOREGROUND_COLOR	18
5.4.1.10 LABEL_TEXT	18
5.4.1.11 LARGE_FONT_CONFIG	18
5.4.1.12 SIGN_BUTTON_TEXT	18
5.4.1.13 VERIFY_BUTTON_TEXT	18
5.5 frames.usb_key_get Namespace Reference	18
5.5.1 Variable Documentation	20
5.5.1.1 ACTION_BUTTON_EXIT_TEXT	20
5.5.1.2 ACTION_BUTTON_INITIAL_TEXT	20
5.5.1.3 ACTION_BUTTON_RETRY_TEXT	20
5.5.1.4 ACTIVATE_BUTTON_COLOR	20
5.5.1.5 BACKGROUND2_COLOR	20
5.5.1.6 BACKGROUND_COLOR	20
5.5.1.7 BLUE_BUTTON_COLOR	21
5.5.1.8 BUTTON_PADDING_Y	21
5.5.1.9 DEFAULT_PADDING_X	21
5.5.1.10 DEFAULT_PADDING_Y	21
5.5.1.11 DEFAULT_WRAP_LENGTH	21
5.5.1.12 FOREGROUND_COLOR	21
5.5.1.13 INITIAL_INSTRUCTION_TEXT	22
5.5.1.14 INPUT_AREA_PADDING_Y	22
5.5.1.15 KEY_INVALID_MSG	22
5.5.1.16 KEY_OR_PIN_INVALID_MSG	22
5.5.1.17 LARGE_FONT_CONFIG	22
5.5.1.18 MULTIPLE_KEYS_MSG	22
5.5.1.19 NO_KEY_FILE_MSG	23
5.5.1.20 NO_USB_DRIVES_MSG	23

5.5.1.21 PIN_INVALID_FORMAT_MSG	23
5.5.1.22 PIN_LABEL_TEXT	23
5.5.1.23 PIN_REQUIRED_MSG	23
5.5.1.24 UNSUPPORTED_PLATFORM_MSG	23
5.6 frames.verify Namespace Reference	24
5.6.1 Variable Documentation	25
5.6.1.1 ACTIVATE_BUTTON_COLOR	25
5.6.1.2 BACKGROUND2_COLOR	25
5.6.1.3 BACKGROUND_COLOR	25
5.6.1.4 BLUE_BUTTON_COLOR	25
5.6.1.5 BUTTON_PADDING_Y	25
5.6.1.6 DEFAULT_PADDING_X	26
5.6.1.7 DEFAULT_PADDING_Y	26
5.6.1.8 DEFAULT_WRAP_LENGTH	26
5.6.1.9 FOREGROUND_COLOR	26
5.6.1.10 INITIAL_INSTRUCTION_TEXT	26
5.6.1.11 LARGE_FONT_CONFIG	26
5.6.1.12 NO_SIGNATURE_MSG	27
5.6.1.13 PATHS_REQUIRED_MSG	27
5.6.1.14 PDF_FILE_TYPES	27
5.6.1.15 PDF_INVALID_MSG	27
5.6.1.16 PUBLIC_KEY_FILE_TYPES	27
5.6.1.17 PUBLIC_KEY_INVALID_MSG	27
5.6.1.18 PUBLIC_KEY_NOT_FOUND_MSG	28
5.6.1.19 SECTION_SPACING_Y	28
5.6.1.20 SELECT_PDF_TO_VERIFY_TITLE	28
5.6.1.21 SELECT_PUBLIC_KEY_TITLE	28
5.6.1.22 SIGNATURE_INVALID_MSG	28
5.6.1.23 SIGNATURE_VALID_MSG	28
5.6.1.24 VERIFICATION_ERROR_MSG	29
5.6.1.25 VERIFY_BUTTON_GO_BACK_TEXT	29
5.6.1.26 VERIFY_BUTTON_INITIAL_TEXT	29
5.6.1.27 VERIFY_BUTTON_RETRY_TEXT	29
5.7 key_generate Namespace Reference	29
5.8 key_generate.AES_key_generator Namespace Reference	29
5.8.1 Function Documentation	30
5.8.1.1 aes_decrypt_file()	30
5.8.1.2 aes_encrypt_file()	30
5.8.1.3 hash_pin()	31
5.9 key_generate.RSA_key_generator Namespace Reference	31
5.9.1 Function Documentation	31
5.9.1.1 generate_keys()	31

5.10 key_getter Namespace Reference	32
5.11 key_getter.AES_PIN_decryptor Namespace Reference	32
5.11.1 Function Documentation	32
5.11.1.1 aes_decrypt_file()	32
5.11.1.2 hash_pin()	33
5.12 key_getter.key_getter Namespace Reference	33
5.12.1 Function Documentation	34
5.12.1.1 _get_key_linux()	34
5.12.1.2 _get_key_paths()	34
5.12.1.3 _get_key_windows()	35
5.12.1.4 get_key()	35
5.12.2 Variable Documentation	36
5.12.2.1 KEY_FILE_NAME	36
5.12.2.2 LINUX_PLATFORM_NAME	36
5.12.2.3 WINDOWS_PLATFORM_NAME	36
5.13 key_getter.usb_finder_linux Namespace Reference	36
5.13.1 Function Documentation	36
5.13.1.1 _get_usb_devices_linux()	37
5.13.1.2 get_usb_mount_paths_linux()	37
5.14 key_getter.usb_finder_windows Namespace Reference	37
5.14.1 Function Documentation	37
5.14.1.1 _get_usb_devices_windows()	38
5.14.1.2 get_usb_mount_paths_windows()	38
5.15 main Namespace Reference	38
5.15.1 Variable Documentation	39
5.15.1.1 APP_HEIGHT	39
5.15.1.2 APP_TITLE	39
5.15.1.3 APP_WIDTH	39
5.16 pdf_signer Namespace Reference	39
5.17 pdf_signer.signer Namespace Reference	39
5.17.1 Function Documentation	39
5.17.1.1 _generate_self_signed_cert()	39
5.17.1.2 sign()	40
5.18 pdf_signer.verifier Namespace Reference	40
5.18.1 Function Documentation	41
5.18.1.1 verify()	41
6 Class Documentation	43
6.1 main.App Class Reference	43
6.1.1 Detailed Description	44
6.1.2 Constructor & Destructor Documentation	44
6.1.2.1 __init__() [1/2]	44

6.1.2.2 <code>__init__()</code> [2/2]	44
6.1.3 Member Function Documentation	44
6.1.3.1 <code>_change_frame()</code>	44
6.1.3.2 <code>get_key_from_usb_result()</code>	45
6.1.3.3 <code>main_menu()</code>	45
6.1.3.4 <code>start_signing()</code>	45
6.1.3.5 <code>start_verifying()</code>	45
6.1.4 Member Data Documentation	45
6.1.4.1 <code>current_frame</code>	45
6.2 <code>frames.generate_window.GenerateKeys</code> Class Reference	46
6.3 <code>frames.usb_key_get.KeyFromUSBFrame</code> Class Reference	46
6.3.1 Detailed Description	46
6.3.2 Constructor & Destructor Documentation	46
6.3.2.1 <code>__init__()</code>	46
6.3.3 Member Function Documentation	47
6.3.3.1 <code>_process_pin_and_get_key()</code>	47
6.3.3.2 <code>_setup_ui()</code>	47
6.3.3.3 <code>_update_feedback()</code>	47
6.3.4 Member Data Documentation	48
6.3.4.1 <code>action_button</code>	48
6.3.4.2 <code>on_key_retrieved_callback</code>	48
6.3.4.3 <code>pin_entry</code>	48
6.3.4.4 <code>status_label</code>	48
6.4 <code>key_getter.key_getter.KeyInvalidException</code> Class Reference	48
6.4.1 Detailed Description	49
6.5 <code>key_getter.key_getter.KeyOrPinInvalidException</code> Class Reference	49
6.5.1 Detailed Description	49
6.6 <code>key_getter.key_getter.MultipleKeysFoundException</code> Class Reference	49
6.6.1 Detailed Description	49
6.7 <code>key_getter.key_getter.NoKeyFoundException</code> Class Reference	49
6.7.1 Detailed Description	50
6.8 <code>pdf_signer.verifier.NoSignatureFound</code> Class Reference	50
6.8.1 Detailed Description	50
6.9 <code>key_getter.key_getter.NoUSBDrivesFoundException</code> Class Reference	50
6.9.1 Detailed Description	50
6.10 <code>frames.signing.SigningFrame</code> Class Reference	51
6.10.1 Detailed Description	51
6.10.2 Constructor & Destructor Documentation	51
6.10.2.1 <code>__init__()</code>	51
6.10.3 Member Function Documentation	52
6.10.3.1 <code>_select_source_pdf_file()</code>	52
6.10.3.2 <code>_select_target_pdf_path()</code>	52

6.10.3.3 <code>_setup_ui()</code>	52
6.10.3.4 <code>_sign_pdf_document()</code>	53
6.10.3.5 <code>_update_feedback()</code>	53
6.10.4 Member Data Documentation	53
6.10.4.1 <code>end_signing_callback</code>	53
6.10.4.2 <code>private_key</code>	53
6.10.4.3 <code>sign_button</code>	54
6.10.4.4 <code>source_pdf_path_entry</code>	54
6.10.4.5 <code>source_pdf_path_var</code>	54
6.10.4.6 <code>status_label</code>	54
6.10.4.7 <code>target_pdf_path_entry</code>	54
6.10.4.8 <code>target_pdf_path_var</code>	54
6.11 <code>frames.start.StartFrame</code> Class Reference	54
6.11.1 Detailed Description	55
6.11.2 Constructor & Destructor Documentation	55
6.11.2.1 <code>__init__()</code>	55
6.11.3 Member Function Documentation	56
6.11.3.1 <code>_setup_ui()</code>	56
6.11.4 Member Data Documentation	56
6.11.4.1 <code>on_signing_chosen_callback</code>	56
6.11.4.2 <code>on_verifying_chosen_callback</code>	56
6.12 <code>key_getter.key_getter.UnsupportedPlatformException</code> Class Reference	56
6.12.1 Detailed Description	57
6.13 <code>frames.verifying.VerifyingFrame</code> Class Reference	57
6.13.1 Detailed Description	58
6.13.2 Constructor & Destructor Documentation	58
6.13.2.1 <code>__init__()</code>	58
6.13.3 Member Function Documentation	58
6.13.3.1 <code>_load_public_key()</code>	58
6.13.3.2 <code>_process_verification()</code>	59
6.13.3.3 <code>_select_file()</code>	59
6.13.3.4 <code>_select_pdf_to_verify_file()</code>	59
6.13.3.5 <code>_select_public_key_file()</code>	60
6.13.3.6 <code>_setup_ui()</code>	60
6.13.3.7 <code>_update_feedback()</code>	60
6.13.4 Member Data Documentation	60
6.13.4.1 <code>end_verifying_callback</code>	60
6.13.4.2 <code>pdf_to_verify_entry</code>	61
6.13.4.3 <code>pdf_to_verify_path_var</code>	61
6.13.4.4 <code>public_key_entry</code>	61
6.13.4.5 <code>public_key_path_var</code>	61
6.13.4.6 <code>status_label</code>	61

6.13.4.7 verify_button	61
7 File Documentation	63
7.1 generating/frames/generate_window.py File Reference	63
7.1.1 Detailed Description	63
7.2 generating/key_generate/AES_key_generator.py File Reference	64
7.3 generating/key_generate/RSA_key_generator.py File Reference	64
7.4 signing/frames/__init__.py File Reference	64
7.5 signing/services/key_getter/__init__.py File Reference	64
7.6 signing/services/pdf_signer/__init__.py File Reference	64
7.7 generating/frames/__init__.py File Reference	65
7.8 generating/key_generate/__init__.py File Reference	65
7.9 signing/frames/signing.py File Reference	65
7.9.1 Detailed Description	66
7.10 signing/frames/start.py File Reference	66
7.10.1 Detailed Description	67
7.11 signing/frames/usb_key_get.py File Reference	67
7.11.1 Detailed Description	69
7.12 signing/frames/verifying.py File Reference	69
7.12.1 Detailed Description	70
7.13 signing/main.py File Reference	70
7.13.1 Detailed Description	71
7.14 generating/main.py File Reference	71
7.14.1 Detailed Description	71
7.15 signing/services/key_getter/AES_PIN_decryptor.py File Reference	71
7.15.1 Detailed Description	72
7.16 signing/services/key_getter/key_getter.py File Reference	72
7.16.1 Detailed Description	73
7.17 signing/services/key_getter/usb_finder_linux.py File Reference	73
7.17.1 Detailed Description	74
7.18 signing/services/key_getter/usb_finder_windows.py File Reference	74
7.18.1 Detailed Description	74
7.19 signing/services/pdf_signer/signer.py File Reference	74
7.19.1 Detailed Description	75
7.20 signing/services/pdf_signer/verifier.py File Reference	75
7.20.1 Detailed Description	75
Index	77

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

frames	9
frames.generate_window	9
frames.signing	11
frames.start	16
frames.usb_key_get	18
frames.verifying	24
key_generate	29
key_generate.AES_key_generator	29
key_generate.RSA_key_generator	31
key_getter	32
key_getter.AES_PIN_decryptor	32
key_getter.key_getter	33
key_getter.usb_finder_linux	36
key_getter.usb_finder_windows	37
main	38
pdf_signer	39
pdf_signer.signer	39
pdf_signer.verifier	40

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Exception	
key_getter.key_getter.KeyInvalidException	48
key_getter.key_getter.KeyOrPinInvalidException	49
key_getter.key_getter.MultipleKeysFoundException	49
key_getter.key_getter.NoKeyFoundException	49
key_getter.key_getter.NoUSBDrivesFoundException	50
key_getter.key_getter.UnsupportedPlatformException	56
pdf_signer.verifier.NoSignatureFound	50
tk.Frame	
frames.generate_window.GenerateKeys	46
frames.signing.SigningFrame	51
frames.start.StartFrame	54
frames.usb_key_get.KeyFromUSBFrame	46
frames.verifying.VerifyingFrame	57
tk.Tk	
main.App	43
main.App	43

Chapter 3

Class Index

3.1 Class List

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

main.App	
Main application class that inherits from tkinter.Tk	43
frames.generate_window.GenerateKeys	46
frames.usb_key_get.KeyFromUSBFrame	
The KeyFromUSBFrame class handles the UI for retrieving a private key from a USB drive	46
key_getter.key_getter.KeyInvalidException	
Exception raised when the decrypted key data cannot be parsed as a valid RSA private key	48
key_getter.key_getter.KeyOrPinInvalidException	
Exception raised when the provided PIN is incorrect or the key file is corrupted/cannot be decrypted with the PIN	49
key_getter.key_getter.MultipleKeysFoundException	
Exception raised when the key file is found on multiple USB drives, creating ambiguity	49
key_getter.key_getter.NoKeyFoundException	
Exception raised when the key file is not found on any detected USB drives	49
pdf_signer.verifier.NoSignatureFound	
Exception raised when a PDF document does not contain any embedded digital signatures	50
key_getter.key_getter.NoUSBDrivesFoundException	
Exception raised when no USB drives are found connected to the system	50
frames.signing.SigningFrame	
The SigningFrame class provides the UI for the PDF signing process	51
frames.start.StartFrame	
The StartFrame class provides user interface for the application	54
key_getter.key_getter.UnsupportedPlatformException	
Exception raised when the current operating system is not supported for USB key retrieval	56
frames.verifyng.VerifyingFrame	
The VerifyingFrame class provides the UI for PDF signature verification	57

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

generating/main.py	
Entry point for the key generating application	71
generating/frames/__init__.py	65
generating/frames/generate_window.py	
GUI window for generating RSA key pairs	63
generating/key_generate/__init__.py	65
generating/key_generate/AES_key_generator.py	64
generating/key_generate/RSA_key_generator.py	64
signing/main.py	
Entry point for the signing application	70
signing/frames/__init__.py	64
signing/frames/signing.py	
A Tkinter Frame for selecting a PDF, choosing an output location, and signing the PDF	65
signing/frames/start.py	
A Tkinter Frame providing options to navigate to PDF signing or verification sections	66
signing/frames/usb_key_get.py	
A Tkinter Frame for prompting the user for a PIN to read and decrypt a private key from a USB drive	67
signing/frames/verifying.py	
A Tkinter Frame for selecting a PDF file and a public key to verify the PDF's digital signature	69
signing/services/key_getter/__init__.py	64
signing/services/key_getter/AES_PIN_decryptor.py	
Provides cryptographic utility functions for hashing and AES decryption	71
signing/services/key_getter/key_getter.py	
Retrieves and decrypts a private key from a USB drive	72
signing/services/key_getter/usb_finder_linux.py	
Provides functions to find USB device mount paths on Linux systems	73
signing/services/key_getter/usb_finder_windows.py	
Provides functions to find USB drive letters on Windows systems	74
signing/services/pdf_signer/__init__.py	64
signing/services/pdf_signer/signer.py	
Provides functions for signing PDF documents using RSA private keys	74
signing/services/pdf_signer/verifier.py	
Provides functionality to verify digital signatures in PDF documents	75

Chapter 5

Namespace Documentation

5.1 frames Namespace Reference

Namespaces

- [generate_window](#)
- [signing](#)
- [start](#)
- [usb_key_get](#)
- [verifying](#)

5.2 frames.generate_window Namespace Reference

Classes

- class [GenerateKeys](#)

Variables

- string [PRIVATE_KEY_NAME](#) = "private_key.key"
Default filename for the private key.
- string [PUBLIC_KEY_NAME](#) = "public_key.key"
Default filename for the public key.
- string [FOREGROUND_COLOR](#) = "#ffffff"
- string [BACKGROUND_COLOR](#) = "#1e1e1e"
- string [BACKGROUND2_COLOR](#) = "#2d2d2d"
- string [BLUE_BUTTON_COLOR](#) = "#007acc"
- string [ACTIVATE_BUTTON_COLOR](#) = "#005f99"

5.2.1 Variable Documentation

5.2.1.1 **ACTIVATE_BUTTON_COLOR**

```
string frames.generate_window.ACTIVATE_BUTTON_COLOR = "#005f99"
```

5.2.1.2 **BACKGROUND2_COLOR**

```
string frames.generate_window.BACKGROUND2_COLOR = "#2d2d2d"
```

5.2.1.3 **BACKGROUND_COLOR**

```
string frames.generate_window.BACKGROUND_COLOR = "#1e1e1e"
```

5.2.1.4 **BLUE_BUTTON_COLOR**

```
string frames.generate_window.BLUE_BUTTON_COLOR = "#007acc"
```

5.2.1.5 **FOREGROUND_COLOR**

```
string frames.generate_window.FOREGROUND_COLOR = "#ffffff"
```

5.2.1.6 **PRIVATE_KEY_NAME**

```
frames.generate_window.PRIVATE_KEY_NAME = "private_key.key"
```

Default filename for the private key.

5.2.1.7 **PUBLIC_KEY_NAME**

```
frames.generate_window.PUBLIC_KEY_NAME = "public_key.key"
```

Default filename for the public key.

5.3 frames.signing Namespace Reference

Classes

- class [SigningFrame](#)

The [SigningFrame](#) class provides the UI for the PDF signing process.

Variables

- tuple [LARGE_FONT_CONFIG](#) = ("TkDefaultFont", 16)
Font configuration for large text elements (e.g., status labels, buttons).
- int [DEFAULT_WRAP_LENGTH](#) = 750
Default wrap length in pixels for text in labels to ensure proper layout.
- list [PDF_FILE_TYPES](#) = ["PDF files", "*.pdf"]
File type filter used in file dialogs, restricting selection to PDF files.
- string [SELECT_SOURCE_PDF_TITLE](#) = "Select the PDF file to sign"
Title for the file dialog when selecting the source PDF to be signed.
- string [SELECT_TARGET_PDF_TITLE](#) = "Set target location and filename for signed PDF"
Title for the file dialog when selecting the output path for the signed PDF.
- string [INITIAL_STATUS_TEXT](#) = "Please choose the PDF file to sign and where the signed PDF file should be placed."
Initial instructional text displayed in the status label.
- string [SOURCE_PDF_LABEL_TEXT](#) = "Selected PDF file to sign:"
Text for the label indicating the selected source PDF file path.
- string [TARGET_PDF_LABEL_TEXT](#) = "Location where the signed PDF file will be saved:"
Text for the label indicating the selected target path for the signed PDF file.
- string [SELECT_SOURCE_BUTTON_TEXT](#) = "Select PDF file"
Text for the button used to trigger source PDF file selection.
- string [SELECT_TARGET_BUTTON_TEXT](#) = "Set target location"
Text for the button used to trigger target PDF path selection.
- string [SIGN_BUTTON_INITIAL_TEXT](#) = "Sign the PDF file"
Initial text for the button that initiates the PDF signing process.
- string [RETRY_BUTTON_TEXT](#) = "Try Again"
Text for the main action button when a retry is suggested after an error.
- string [GO_BACK_BUTTON_TEXT](#) = "Go back to main menu"
Text for the main action button to navigate back to the main menu after success.
- string [PATHS_REQUIRED_ERROR_TEXT](#) = "Both original and target PDF locations are required. Please select them."
Error message displayed when either source or target PDF paths are not selected.
- string [PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT](#) = "Original and target PDF locations cannot be the same. Please change one of them and try again."
Error message displayed when source or target PDF paths are the same.
- string [SIGNING_SUCCESS_TEXT](#) = "Successfully signed PDF and saved to the designated location."
Message displayed in the status label upon successful PDF signing.
- string [PDF_READ_ERROR_TEXT](#) = "Selected source file is not a valid PDF file. Please verify and try again."
Error message displayed if the selected source file is not a valid PDF.
- string [SIGNING_ERROR_TEXT](#) = "This PDF file may have already been signed or is unsuitable for signing. Please choose a different file."
Error message displayed if the PDF is already signed or unsuitable for signing.

- string `UNEXPECTED_SIGNING_ERROR_TEXT` = "An unexpected error occurred during signing: {error_type}. Please try again"
Error message template for unexpected errors during the signing process.
- string `FOREGROUND_COLOR` = "#ffffff"
- string `BACKGROUND_COLOR` = "#1e1e1e"
- string `BACKGROUND2_COLOR` = "#2d2d2d"
- string `BLUE_BUTTON_COLOR` = "#007acc"
- string `ACTIVATE_BUTTON_COLOR` = "#005f99"

5.3.1 Variable Documentation

5.3.1.1 ACTIVATE_BUTTON_COLOR

```
string frames.signing.ACTIVATE_BUTTON_COLOR = "#005f99"
```

5.3.1.2 BACKGROUND2_COLOR

```
string frames.signing.BACKGROUND2_COLOR = "#2d2d2d"
```

5.3.1.3 BACKGROUND_COLOR

```
string frames.signing.BACKGROUND_COLOR = "#1e1e1e"
```

5.3.1.4 BLUE_BUTTON_COLOR

```
string frames.signing.BLUE_BUTTON_COLOR = "#007acc"
```

5.3.1.5 DEFAULT_WRAP_LENGTH

```
frames.signing.DEFAULT_WRAP_LENGTH = 750
```

Default wrap length in pixels for text in labels to ensure proper layout.

5.3.1.6 FOREGROUND_COLOR

```
string frames.signing.FOREGROUND_COLOR = "#ffffff"
```

5.3.1.7 GO_BACK_BUTTON_TEXT

```
frames.signing.GO_BACK_BUTTON_TEXT = "Go back to main menu"
```

Text for the main action button to navigate back to the main menu after success.

5.3.1.8 INITIAL_STATUS_TEXT

```
frames.signing.INITIAL_STATUS_TEXT = "Please choose the PDF file to sign and where the signed  
PDF file should be placed."
```

Initial instructional text displayed in the status label.

5.3.1.9 LARGE_FONT_CONFIG

```
frames.signing.LARGE_FONT_CONFIG = ("TkDefaultFont", 16)
```

Font configuration for large text elements (e.g., status labels, buttons).

5.3.1.10 PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT

```
frames.signing.PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT = "Original and target PDF locations cannot  
be the same. Please change one of them and try again."
```

Error message displayed when source or target PDF paths are the same.

5.3.1.11 PATHS_REQUIRED_ERROR_TEXT

```
frames.signing.PATHS_REQUIRED_ERROR_TEXT = "Both original and target PDF locations are required.  
Please select them."
```

Error message displayed when either source or target PDF paths are not selected.

5.3.1.12 PDF_FILE_TYPES

```
frames.signing.PDF_FILE_TYPES = [("PDF files", "*.pdf")]
```

File type filter used in file dialogs, restricting selection to PDF files.

5.3.1.13 PDF_READ_ERROR_TEXT

```
frames.signing.PDF_READ_ERROR_TEXT = "Selected source file is not a valid PDF file. Please  
verify and try again."
```

Error message displayed if the selected source file is not a valid PDF.

5.3.1.14 RETRY_BUTTON_TEXT

```
frames.signing.RETRY_BUTTON_TEXT = "Try Again"
```

Text for the main action button when a retry is suggested after an error.

5.3.1.15 SELECT_SOURCE_BUTTON_TEXT

```
frames.signing.SELECT_SOURCE_BUTTON_TEXT = "Select PDF file"
```

Text for the button used to trigger source PDF file selection.

5.3.1.16 SELECT_SOURCE_PDF_TITLE

```
frames.signing.SELECT_SOURCE_PDF_TITLE = "Select the PDF file to sign"
```

Title for the file dialog when selecting the source PDF to be signed.

5.3.1.17 SELECT_TARGET_BUTTON_TEXT

```
frames.signing.SELECT_TARGET_BUTTON_TEXT = "Set target location"
```

Text for the button used to trigger target PDF path selection.

5.3.1.18 SELECT_TARGET_PDF_TITLE

```
frames.signing.SELECT_TARGET_PDF_TITLE = "Set target location and filename for signed PDF"
```

Title for the file dialog when selecting the output path for the signed PDF.

5.3.1.19 SIGN_BUTTON_INITIAL_TEXT

```
frames.signing.SIGN_BUTTON_INITIAL_TEXT = "Sign the PDF file"
```

Initial text for the button that initiates the PDF signing process.

5.3.1.20 SIGNING_ERROR_TEXT

```
frames.signing.SIGNING_ERROR_TEXT = "This PDF file may have already been signed or is unsuitable  
for signing. Please choose a different file."
```

Error message displayed if the PDF is already signed or unsuitable for signing.

5.3.1.21 SIGNING_SUCCESS_TEXT

```
frames.signing.SIGNING_SUCCESS_TEXT = "Successfully signed PDF and saved to the designated  
location."
```

Message displayed in the status label upon successful PDF signing.

5.3.1.22 SOURCE_PDF_LABEL_TEXT

```
frames.signing.SOURCE_PDF_LABEL_TEXT = "Selected PDF file to sign:"
```

Text for the label indicating the selected source PDF file path.

5.3.1.23 TARGET_PDF_LABEL_TEXT

```
frames.signing.TARGET_PDF_LABEL_TEXT = "Location where the signed PDF file will be saved:"
```

Text for the label indicating the selected target path for the signed PDF file.

5.3.1.24 UNEXPECTED_SIGNING_ERROR_TEXT

```
frames.signing.UNEXPECTED_SIGNING_ERROR_TEXT = "An unexpected error occurred during signing:
{error_type}. Please try again"
```

Error message template for unexpected errors during the signing process.

5.4 frames.start Namespace Reference

Classes

- class [StartFrame](#)
The [StartFrame](#) class provides user interface for the application.

Variables

- tuple [LARGE_FONT_CONFIG](#) = ("TkDefaultFont", 16)
Font configuration for large text elements.
- int [DEFAULT_WRAP_LENGTH](#) = 750
Default wrap length in pixels for text in labels.
- int [DEFAULT_PADDING_X](#) = 200
Default horizontal padding in pixels for UI elements.
- int [DEFAULT_PADDING_Y](#) = 10
Default vertical padding in pixels for general UI elements.
- int [BUTTON_PADDING_Y](#) = 20
Specific vertical padding in pixels for buttons.
- string [LABEL_TEXT](#) = "Please choose whether you want to sign a PDF or verify the signature of a PDF file."
Text content for the main instruction label on the [StartFrame](#).
- string [SIGN_BUTTON_TEXT](#) = "Sign a PDF file"
Text content for the button that initiates the PDF signing process.
- string [VERIFY_BUTTON_TEXT](#) = "Verify PDF file signature"
Text content for the button that initiates the PDF signature verification process.
- string [FOREGROUND_COLOR](#) = "#ffffff"
- string [BACKGROUND_COLOR](#) = "#1e1e1e"
- string [BACKGROUND2_COLOR](#) = "#2d2d2d"
- string [BLUE_BUTTON_COLOR](#) = "#007acc"
- string [ACTIVATE_BUTTON_COLOR](#) = "#005f99"

5.4.1 Variable Documentation

5.4.1.1 ACTIVATE_BUTTON_COLOR

```
string frames.start.ACTIVATE_BUTTON_COLOR = "#005f99"
```

5.4.1.2 BACKGROUND2_COLOR

```
string frames.start.BACKGROUND2_COLOR = "#2d2d2d"
```

5.4.1.3 BACKGROUND_COLOR

```
string frames.start.BACKGROUND_COLOR = "#1e1e1e"
```

5.4.1.4 BLUE_BUTTON_COLOR

```
string frames.start.BLUE_BUTTON_COLOR = "#007acc"
```

5.4.1.5 BUTTON_PADDING_Y

```
frames.start.BUTTON_PADDING_Y = 20
```

Specific vertical padding in pixels for buttons.

5.4.1.6 DEFAULT_PADDING_X

```
frames.start.DEFAULT_PADDING_X = 200
```

Default horizontal padding in pixels for UI elements.

5.4.1.7 DEFAULT_PADDING_Y

```
frames.start.DEFAULT_PADDING_Y = 10
```

Default vertical padding in pixels for general UI elements.

5.4.1.8 DEFAULT_WRAP_LENGTH

```
frames.start.DEFAULT_WRAP_LENGTH = 750
```

Default wrap length in pixels for text in labels.

Used to ensure text fits within the UI layout.

5.4.1.9 FOREGROUND_COLOR

```
string frames.start.FOREGROUND_COLOR = "#ffffff"
```

5.4.1.10 LABEL_TEXT

```
frames.start.LABEL_TEXT = "Please choose whether you want to sign a PDF or verify the signature  
of a PDF file."
```

Text content for the main instruction label on the [StartFrame](#).

5.4.1.11 LARGE_FONT_CONFIG

```
frames.start.LARGE_FONT_CONFIG = ("TkDefaultFont", 16)
```

Font configuration for large text elements.

Tuple specifying font family ("TkDefaultFont") and size (16).

5.4.1.12 SIGN_BUTTON_TEXT

```
frames.start.SIGN_BUTTON_TEXT = "Sign a PDF file"
```

Text content for the button that initiates the PDF signing process.

5.4.1.13 VERIFY_BUTTON_TEXT

```
frames.start.VERIFY_BUTTON_TEXT = "Verify PDF file signature"
```

Text content for the button that initiates the PDF signature verification process.

5.5 frames.usb_key_get Namespace Reference

Classes

- class [KeyFromUSBFrame](#)

The [KeyFromUSBFrame](#) class handles the UI for retrieving a private key from a USB drive.

Variables

- tuple `LARGE_FONT_CONFIG` = ("TkDefaultFont", 16)
Font configuration for large text elements.
- int `DEFAULT_WRAP_LENGTH` = 750
Default wrap length in pixels for text in labels.
- int `DEFAULT_PADDING_X` = 10
Default horizontal padding in pixels for UI elements.
- int `DEFAULT_PADDING_Y` = 10
Default vertical padding in pixels for general UI elements.
- int `INPUT_AREA_PADDING_Y` = 5
Vertical padding in pixels for the PIN input area.
- int `BUTTON_PADDING_Y` = 20
Specific vertical padding in pixels for buttons.
- string `INITIAL_INSTRUCTION_TEXT` = "To sign the PDF file, first the key has to be read from the USB drive and deciphered with your PIN."
Initial instruction text displayed to the user.
- string `PIN_LABEL_TEXT` = "Enter PIN:"
Text for the PIN entry label.
- string `ACTION_BUTTON_INITIAL_TEXT` = "Find and Read Key"
Initial text for the main action button.
- string `ACTION_BUTTON_RETRY_TEXT` = "Try Again"
Text for the action button when a retry is suggested.
- string `ACTION_BUTTON_EXIT_TEXT` = "Exit Program"
Text for the action button when exiting is the only option.
- string `PIN_REQUIRED_MSG` = "PIN is required. Please enter your 4-digit PIN."
Error message when the PIN is not entered.
- string `PIN_INVALID_FORMAT_MSG` = "The PIN must be 4 digits. Please try again."
Error message when the PIN format is incorrect.
- string `UNSUPPORTED_PLATFORM_MSG` = "Error: Current operating system is not supported for USB key retrieval."
Error message for unsupported operating systems.
- string `NO_USB_DRIVES_MSG` = "No USB drives found. Please insert the USB drive with the key and try again."
Error message when no USB drives are detected.
- string `NO_KEY_FILE_MSG` = "No key file found on any USB drive. Please ensure the key file is present and try again."
Error message when the key file is not found on USB drives.
- string `MULTIPLE_KEYS_MSG` = "Multiple key files found across different USB drives. Please ensure only one USB drive with the key file is connected and try again."
Error message when multiple key files are found.
- string `KEY_OR_PIN_INVALID_MSG` = "Invalid PIN or key file. Please verify your PIN and the key file, then try again."
Error message for an invalid PIN or key file.
- string `KEY_INVALID_MSG` = "The key file is invalid or corrupted. Please ensure you have the correct key file."
Error message when the key file itself is invalid or corrupted.
- string `FOREGROUND_COLOR` = "#ffffff"
- string `BACKGROUND_COLOR` = "#1e1e1e"
- string `BACKGROUND2_COLOR` = "#2d2d2d"
- string `BLUE_BUTTON_COLOR` = "#007acc"
- string `ACTIVATE_BUTTON_COLOR` = "#005f99"

5.5.1 Variable Documentation

5.5.1.1 ACTION_BUTTON_EXIT_TEXT

```
frames.usb_key_get.ACTION_BUTTON_EXIT_TEXT = "Exit Program"
```

Text for the action button when exiting is the only option.

5.5.1.2 ACTION_BUTTON_INITIAL_TEXT

```
frames.usb_key_get.ACTION_BUTTON_INITIAL_TEXT = "Find and Read Key"
```

Initial text for the main action button.

5.5.1.3 ACTION_BUTTON_RETRY_TEXT

```
frames.usb_key_get.ACTION_BUTTON_RETRY_TEXT = "Try Again"
```

Text for the action button when a retry is suggested.

5.5.1.4 ACTIVATE_BUTTON_COLOR

```
string frames.usb_key_get.ACTIVATE_BUTTON_COLOR = "#005f99"
```

5.5.1.5 BACKGROUND2_COLOR

```
string frames.usb_key_get.BACKGROUND2_COLOR = "#2d2d2d"
```

5.5.1.6 BACKGROUND_COLOR

```
string frames.usb_key_get.BACKGROUND_COLOR = "#1e1e1e"
```

5.5.1.7 BLUE_BUTTON_COLOR

```
string frames.usb_key_get.BLUE_BUTTON_COLOR = "#007acc"
```

5.5.1.8 BUTTON_PADDING_Y

```
frames.usb_key_get.BUTTON_PADDING_Y = 20
```

Specific vertical padding in pixels for buttons.

5.5.1.9 DEFAULT_PADDING_X

```
frames.usb_key_get.DEFAULT_PADDING_X = 10
```

Default horizontal padding in pixels for UI elements.

5.5.1.10 DEFAULT_PADDING_Y

```
frames.usb_key_get.DEFAULT_PADDING_Y = 10
```

Default vertical padding in pixels for general UI elements.

5.5.1.11 DEFAULT_WRAP_LENGTH

```
frames.usb_key_get.DEFAULT_WRAP_LENGTH = 750
```

Default wrap length in pixels for text in labels.

5.5.1.12 FOREGROUND_COLOR

```
string frames.usb_key_get.FOREGROUND_COLOR = "#ffffff"
```

5.5.1.13 INITIAL_INSTRUCTION_TEXT

```
frames.usb_key_get.INITIAL_INSTRUCTION_TEXT = "To sign the PDF file, first the key has to be read from the USB drive and deciphered with your PIN."
```

Initial instruction text displayed to the user.

5.5.1.14 INPUT_AREA_PADDING_Y

```
frames.usb_key_get.INPUT_AREA_PADDING_Y = 5
```

Vertical padding in pixels for the PIN input area.

5.5.1.15 KEY_INVALID_MSG

```
frames.usb_key_get.KEY_INVALID_MSG = "The key file is invalid or corrupted. Please ensure you have the correct key file."
```

Error message when the key file itself is invalid or corrupted.

5.5.1.16 KEY_OR_PIN_INVALID_MSG

```
frames.usb_key_get.KEY_OR_PIN_INVALID_MSG = "Invalid PIN or key file. Please verify your PIN and the key file, then try again."
```

Error message for an invalid PIN or key file.

5.5.1.17 LARGE_FONT_CONFIG

```
frames.usb_key_get.LARGE_FONT_CONFIG = ("TkDefaultFont", 16)
```

Font configuration for large text elements.

5.5.1.18 MULTIPLE_KEYS_MSG

```
frames.usb_key_get.MULTIPLE_KEYS_MSG = "Multiple key files found across different USB drives. Please ensure only one USB drive with the key file is connected and try again."
```

Error message when multiple key files are found.

5.5.1.19 NO_KEY_FILE_MSG

```
frames.usb_key_get.NO_KEY_FILE_MSG = "No key file found on any USB drive. Please ensure the key file is present and try again."
```

Error message when the key file is not found on USB drives.

5.5.1.20 NO_USB_DRIVES_MSG

```
frames.usb_key_get.NO_USB_DRIVES_MSG = "No USB drives found. Please insert the USB drive with the key and try again."
```

Error message when no USB drives are detected.

5.5.1.21 PIN_INVALID_FORMAT_MSG

```
frames.usb_key_get.PIN_INVALID_FORMAT_MSG = "The PIN must be 4 digits. Please try again."
```

Error message when the PIN format is incorrect.

5.5.1.22 PIN_LABEL_TEXT

```
frames.usb_key_get.PIN_LABEL_TEXT = "Enter PIN:"
```

Text for the PIN entry label.

5.5.1.23 PIN_REQUIRED_MSG

```
frames.usb_key_get.PIN_REQUIRED_MSG = "PIN is required. Please enter your 4-digit PIN."
```

Error message when the PIN is not entered.

5.5.1.24 UNSUPPORTED_PLATFORM_MSG

```
frames.usb_key_get.UNSUPPORTED_PLATFORM_MSG = "Error: Current operating system is not supported for USB key retrieval."
```

Error message for unsupported operating systems.

5.6 frames.verifying Namespace Reference

Classes

- class [VerifyingFrame](#)

The [VerifyingFrame](#) class provides the UI for PDF signature verification.

Variables

- tuple [LARGE_FONT_CONFIG](#) = ("TkDefaultFont", 16)
Font configuration for large text elements like status labels and buttons.
- int [DEFAULT_WRAP_LENGTH](#) = 750
Default wrap length in pixels for text in labels to ensure proper UI layout.
- list [PDF_FILE_TYPES](#) = [("PDF files", "*.pdf")]
File type filter for PDF file selection dialogs, showing only ".pdf" files.*
- list [PUBLIC_KEY_FILE_TYPES](#) = [("Public Key files", "*.pem *.key")]
File type filter for public key file selection dialogs, showing ".pem" and "*.key" files.*
- string [SELECT_PDF_TO_VERIFY_TITLE](#) = "Select the PDF file to verify"
Title for the file dialog when selecting the PDF file to be verified.
- string [SELECT_PUBLIC_KEY_TITLE](#) = "Select the public key file"
Title for the file dialog when selecting the public key file.
- int [DEFAULT_PADDING_X](#) = 10
Default horizontal padding in pixels for UI elements.
- int [DEFAULT_PADDING_Y](#) = 10
Default vertical padding in pixels for general UI elements.
- int [SECTION_SPACING_Y](#) = 20
Vertical spacing in pixels used between UI sections (e.g., between PDF selection and public key selection).
- int [BUTTON_PADDING_Y](#) = 20
Specific vertical padding in pixels for buttons.
- string [INITIAL_INSTRUCTION_TEXT](#) = "Please choose the PDF file to verify and the public key corresponding to the signature."
Initial instructional text displayed to the user in the status label.
- string [VERIFY_BUTTON_INITIAL_TEXT](#) = "Verify PDF Signature"
Initial text for the button that initiates the PDF signature verification process.
- string [VERIFY_BUTTON_RETRY_TEXT](#) = "Try Again"
Text for the main action button when a retry is suggested after an error.
- string [VERIFY_BUTTON_GO_BACK_TEXT](#) = "Go back to main menu"
Text for the main action button to navigate back to the main menu after verification (success or failure).
- string [PATHS_REQUIRED_MSG](#) = "Both PDF file and public key file locations are required. Please select them."
Error message displayed if either the PDF file or public key file path is not selected.
- string [PUBLIC_KEY_NOT_FOUND_MSG](#) = "Public key file not found at the specified location. Please check the path and try again."
Error message displayed if the specified public key file cannot be found.
- string [PUBLIC_KEY_INVALID_MSG](#) = "The selected file is not a valid public key or is corrupted. Please verify the key file."
Error message displayed if the selected public key file is invalid or corrupted.
- string [PDF_INVALID_MSG](#) = "The selected file is not a valid PDF file. Please verify the PDF and try again."
Error message displayed if the selected PDF file is invalid or cannot be read.
- string [NO_SIGNATURE_MSG](#) = "This PDF file does not appear to be signed. Please select a signed PDF."

Message displayed if the selected PDF file does not contain a digital signature.

- string `VERIFICATION_ERROR_MSG` = "An error occurred during verification. Please try again."

General error message displayed if an unexpected error occurs during verification.

- string `SIGNATURE_VALID_MSG` = "The signature is VALID."

Message displayed in the status label when the PDF signature is successfully validated.

- string `SIGNATURE_INVALID_MSG` = "The signature is INVALID."

Message displayed in the status label when the PDF signature is found to be invalid.

- string `FOREGROUND_COLOR` = "#ffffff"
- string `BACKGROUND_COLOR` = "#1e1e1e"
- string `BACKGROUND2_COLOR` = "#2d2d2d"
- string `BLUE_BUTTON_COLOR` = "#007acc"
- string `ACTIVATE_BUTTON_COLOR` = "#005f99"

5.6.1 Variable Documentation

5.6.1.1 ACTIVATE_BUTTON_COLOR

```
string frames.verify.ACTIVATE_BUTTON_COLOR = "#005f99"
```

5.6.1.2 BACKGROUND2_COLOR

```
string frames.verify.BACKGROUND2_COLOR = "#2d2d2d"
```

5.6.1.3 BACKGROUND_COLOR

```
string frames.verify.BACKGROUND_COLOR = "#1e1e1e"
```

5.6.1.4 BLUE_BUTTON_COLOR

```
string frames.verify.BLUE_BUTTON_COLOR = "#007acc"
```

5.6.1.5 BUTTON_PADDING_Y

```
frames.verify.BUTTON_PADDING_Y = 20
```

Specific vertical padding in pixels for buttons.

5.6.1.6 DEFAULT_PADDING_X

```
frames.verifying.DEFAULT_PADDING_X = 10
```

Default horizontal padding in pixels for UI elements.

5.6.1.7 DEFAULT_PADDING_Y

```
frames.verifying.DEFAULT_PADDING_Y = 10
```

Default vertical padding in pixels for general UI elements.

5.6.1.8 DEFAULT_WRAP_LENGTH

```
frames.verifying.DEFAULT_WRAP_LENGTH = 750
```

Default wrap length in pixels for text in labels to ensure proper UI layout.

5.6.1.9 FOREGROUND_COLOR

```
string frames.verifying.FOREGROUND_COLOR = "#ffffff"
```

5.6.1.10 INITIAL_INSTRUCTION_TEXT

```
frames.verifying.INITIAL_INSTRUCTION_TEXT = "Please choose the PDF file to verify and the  
public key corresponding to the signature."
```

Initial instructional text displayed to the user in the status label.

5.6.1.11 LARGE_FONT_CONFIG

```
frames.verifying.LARGE_FONT_CONFIG = ("TkDefaultFont", 16)
```

Font configuration for large text elements like status labels and buttons.

5.6.1.12 NO_SIGNATURE_MSG

```
frames.verifying.NO_SIGNATURE_MSG = "This PDF file does not appear to be signed. Please select a signed PDF."
```

Message displayed if the selected PDF file does not contain a digital signature.

5.6.1.13 PATHS_REQUIRED_MSG

```
frames.verifying.PATHS_REQUIRED_MSG = "Both PDF file and public key file locations are required. Please select them."
```

Error message displayed if either the PDF file or public key file path is not selected.

5.6.1.14 PDF_FILE_TYPES

```
frames.verifying.PDF_FILE_TYPES = [("PDF files", "*.pdf")]
```

File type filter for PDF file selection dialogs, showing only "*.pdf" files.

5.6.1.15 PDF_INVALID_MSG

```
frames.verifying.PDF_INVALID_MSG = "The selected file is not a valid PDF file. Please verify the PDF and try again."
```

Error message displayed if the selected PDF file is invalid or cannot be read.

5.6.1.16 PUBLIC_KEY_FILE_TYPES

```
frames.verifying.PUBLIC_KEY_FILE_TYPES = [("Public Key files", "*.pem *.key")]
```

File type filter for public key file selection dialogs, showing "*.pem" and "*.key" files.

5.6.1.17 PUBLIC_KEY_INVALID_MSG

```
frames.verifying.PUBLIC_KEY_INVALID_MSG = "The selected file is not a valid public key or is corrupted. Please verify the key file."
```

Error message displayed if the selected public key file is invalid or corrupted.

5.6.1.18 PUBLIC_KEY_NOT_FOUND_MSG

```
frames.verifying.PUBLIC_KEY_NOT_FOUND_MSG = "Public key file not found at the specified location.  
Please check the path and try again."
```

Error message displayed if the specified public key file cannot be found.

5.6.1.19 SECTION_SPACING_Y

```
frames.verifying.SECTION_SPACING_Y = 20
```

Vertical spacing in pixels used between UI sections (e.g., between PDF selection and public key selection).

5.6.1.20 SELECT_PDF_TO_VERIFY_TITLE

```
frames.verifying.SELECT_PDF_TO_VERIFY_TITLE = "Select the PDF file to verify"
```

Title for the file dialog when selecting the PDF file to be verified.

5.6.1.21 SELECT_PUBLIC_KEY_TITLE

```
frames.verifying.SELECT_PUBLIC_KEY_TITLE = "Select the public key file"
```

Title for the file dialog when selecting the public key file.

5.6.1.22 SIGNATURE_INVALID_MSG

```
frames.verifying.SIGNATURE_INVALID_MSG = "The signature is INVALID."
```

Message displayed in the status label when the PDF signature is found to be invalid.

5.6.1.23 SIGNATURE_VALID_MSG

```
frames.verifying.SIGNATURE_VALID_MSG = "The signature is VALID."
```

Message displayed in the status label when the PDF signature is successfully validated.

5.6.1.24 VERIFICATION_ERROR_MSG

```
frames.verifying.VERIFICATION_ERROR_MSG = "An error occurred during verification. Please try again."
```

General error message displayed if an unexpected error occurs during verification.

5.6.1.25 VERIFY_BUTTON_GO_BACK_TEXT

```
frames.verifying.VERIFY_BUTTON_GO_BACK_TEXT = "Go back to main menu"
```

Text for the main action button to navigate back to the main menu after verification (success or failure).

5.6.1.26 VERIFY_BUTTON_INITIAL_TEXT

```
frames.verifying.VERIFY_BUTTON_INITIAL_TEXT = "Verify PDF Signature"
```

Initial text for the button that initiates the PDF signature verification process.

5.6.1.27 VERIFY_BUTTON_RETRY_TEXT

```
frames.verifying.VERIFY_BUTTON_RETRY_TEXT = "Try Again"
```

Text for the main action button when a retry is suggested after an error.

5.7 key_generate Namespace Reference

Namespaces

- [AES_key_generator](#)
- [RSA_key_generator](#)

5.8 key_generate.AES_key_generator Namespace Reference

Functions

- def [hash_pin](#) (str pin)
Generates a 256-bit key from PIN code.
- bool [aes_encrypt_file](#) (str file_to_encrypt, str pin)
Encrypts a file using a 4-digit PIN code and AES encryption.
- (bool, bytes) [aes_decrypt_file](#) (str file_to_decrypt, str pin)
Decrypts a file using a 4-digit PIN code and AES decryption.

5.8.1 Function Documentation

5.8.1.1 `aes_decrypt_file()`

```
(bool, bytes) key_generate.AES_key_generator.aes_decrypt_file (  
    str file_to_decrypt,  
    str pin )
```

Decrypts a file using a 4-digit PIN code and AES decryption.

The function reads a file from the given path, changes the given 4-digit PIN code to a 256-bit key, and then decrypt the file with this key using AES decryption.

Parameters

<i>file_to_encrypt</i>	Path to the input file to decrypt
<i>pin</i>	4-digit PIN code

Returns

Tuple (True,data) if AES decryption was successful, where 'data' is the decryption content; Tuple (False, None) if the file was not found or decryption failed.

5.8.1.2 `aes_encrypt_file()`

```
bool key_generate.AES_key_generator.aes_encrypt_file (  
    str file_to_encrypt,  
    str pin )
```

Encrypts a file using a 4-digit PIN code and AES encryption.

The function reads a file from the given path, changes the given 4-digit PIN code to a 256-bit key, and then encrypt the file with this key using AES encryption.

Parameters

<i>file_to_encrypt</i>	Path to the input file to encrypt
<i>pin</i>	4-digit PIN code

Returns

True if AES encryption was successful; False if the file was not found or encryption failed.

5.8.1.3 hash_pin()

```
def key_generate.AES_key_generator.hash_pin (
    str pin )
```

Generates a 256-bit key from PIN code.

The function validates the given PIN code and generate a 256-bit code from given PIN code

Parameters

<i>pin</i>	A PIN code as string
------------	----------------------

Returns

The derived PIN code as a 256-bit key

5.9 key_generate.RSA_key_generator Namespace Reference

Functions

- bool [generate_keys](#) (str public_key_location, str private_key_location)
Generate public/private key pairs.

5.9.1 Function Documentation

5.9.1.1 generate_keys()

```
bool key_generate.RSA_key_generator.generate_keys (
    str public_key_location,
    str private_key_location )
```

Generate public/private key pairs.

The function reads locations for public and private keys, keys generated with RSA algorithm.

Parameters

<i>public_key_location</i>	Path to save generated a public key
<i>private_key_location</i>	Path to save generated a private key

Returns

True if RSA generation was successful; False if the RSA generation thrown exception.

5.10 key_getter Namespace Reference

Namespaces

- [AES_PIN_decryptor](#)
- [key_getter](#)
- [usb_finder_linux](#)
- [usb_finder_windows](#)

5.11 key_getter.AES_PIN_decryptor Namespace Reference

Functions

- bytes [hash_pin](#) (str pin)
Hashes a numeric PIN string using SHA256.
- bytes [aes_decrypt_file](#) (bytes encrypted_data, str pin)
Decrypts data encrypted using AES in EAX mode.

5.11.1 Function Documentation

5.11.1.1 aes_decrypt_file()

```
bytes key_getter.AES_PIN_decryptor.aes_decrypt_file (
    bytes encrypted_data,
    str pin )
```

Decrypts data encrypted using AES in EAX mode.

Parameters

<i>encrypted_data</i>	The encrypted data, which includes a 16-byte nonce, a 16-byte tag, and the ciphertext. @type encrypted_data bytes
<i>pin</i>	The numeric PIN string used to derive the decryption key. @type pin str

Returns

The decrypted data as bytes. @rtype bytes

Exceptions

<i>ValueError</i>	If the <code>encrypted_data</code> is less than 32 bytes (too short to contain nonce and tag).
-------------------	--

5.11.1.2 hash_pin()

```
bytes key_getter.AES_PIN_decryptor.hash_pin (
    str pin )
```

Hashes a numeric PIN string using SHA256.

Parameters

<i>pin</i>	The numeric PIN string to hash. @type pin str
------------	---

Returns

The SHA256 hash of the PIN as bytes. @rtype bytes

5.12 key_getter.key_getter Namespace Reference

Classes

- class [UnsupportedPlatformException](#)
Exception raised when the current operating system is not supported for USB key retrieval.
- class [NoUSBDrivesFoundException](#)
Exception raised when no USB drives are found connected to the system.
- class [NoKeyFoundException](#)
Exception raised when the key file is not found on any detected USB drives.
- class [MultipleKeysFoundException](#)
Exception raised when the key file is found on multiple USB drives, creating ambiguity.
- class [KeyOrPinInvalidException](#)
Exception raised when the provided PIN is incorrect or the key file is corrupted/cannot be decrypted with the PIN.
- class [KeyInvalidException](#)
Exception raised when the decrypted key data cannot be parsed as a valid RSA private key.

Functions

- `rsa.RSAPrivateKey` [get_key](#) (str pin)
Retrieves and decrypts the RSA private key from a USB drive using a PIN.
- `bytes` [_get_key_windows](#) ()
Internal function to retrieve the encrypted key data from USB drives on Windows.
- `bytes` [_get_key_linux](#) ()
Internal function to retrieve the encrypted key data from USB drives on Linux.
- `bytes` [_get_key_paths](#) (list[str] usb_paths)
Internal function to search for and read the key file from a list of USB paths.

Variables

- string [WINDOWS_PLATFORM_NAME](#) = "Windows"
String constant representing the Windows platform identifier.
- string [LINUX_PLATFORM_NAME](#) = "Linux"
String constant representing the Linux platform identifier.
- string [KEY_FILE_NAME](#) = "private_key.key"
The expected filename of the encrypted private key on the USB drive.

5.12.1 Function Documentation

5.12.1.1 `_get_key_linux()`

```
bytes key_getter.key_getter._get_key_linux ( ) [private]
```

Internal function to retrieve the encrypted key data from USB drives on Linux.

Calls `get_usb_mount_paths_linux` to find USB drives and then `_get_key_paths` to locate and read the key file.

Returns

The encrypted key data as bytes. @rtype bytes

Exceptions

<i>NoUSBDrivesFoundException</i>	If no USB drives are detected by <code>_get_key_paths</code> .
<i>NoKeyFoundException</i>	If the key file is not found on any detected USB drives by <code>_get_key_paths</code> .
<i>MultipleKeysFoundException</i>	If the key file is found on multiple USB drives by <code>_get_key_paths</code> .

5.12.1.2 `_get_key_paths()`

```
bytes key_getter.key_getter._get_key_paths (
    list[str] usb_paths ) [private]
```

Internal function to search for and read the key file from a list of USB paths.

Parameters

<i>usb_paths</i>	A list of file system paths where USB drives are mounted. @type usb_paths list[str]
------------------	---

Returns

The content of the key file as bytes. @rtype bytes

Exceptions

<i>NoUSBDrivesFoundException</i>	If the <code>usb_paths</code> list is empty.
<i>NoKeyFoundException</i>	If <code>KEY_FILE_NAME</code> is not found in any of the provided <code>usb_paths</code> .
<i>MultipleKeysFoundException</i>	If <code>KEY_FILE_NAME</code> is found in more than one path in <code>usb_paths</code> .

5.12.1.3 `_get_key_windows()`

```
bytes key_getter.key_getter._get_key_windows ( ) [private]
```

Internal function to retrieve the encrypted key data from USB drives on Windows.

Calls `get_usb_mount_paths_windows` to find USB drives and then `_get_key_paths` to locate and read the key file.

Returns

The encrypted key data as bytes. @rtype bytes

Exceptions

<i>NoUSBDrivesFoundException</i>	If no USB drives are detected by <code>_get_key_paths</code> .
<i>NoKeyFoundException</i>	If the key file is not found on any detected USB drives by <code>_get_key_paths</code> .
<i>MultipleKeysFoundException</i>	If the key file is found on multiple USB drives by <code>_get_key_paths</code> .

5.12.1.4 `get_key()`

```
rsa.RSAPrivateKey key_getter.key_getter.get_key (
    str pin )
```

Retrieves and decrypts the RSA private key from a USB drive using a PIN.

Parameters

<i>pin</i>	The PIN code to decrypt the private key. @type pin str
------------	--

Returns

The decrypted RSA private key. @rtype rsa.RSAPrivateKey

Exceptions

<i>UnsupportedPlatformException</i>	If the current operating system is not supported.
<i>NoUSBDrivesFoundException</i>	If no USB drives are detected.
<i>NoKeyFoundException</i>	If the key file is not found on any USB drive.
<i>MultipleKeysFoundException</i>	If the key file is found on more than one USB drive.
<i>KeyOrPinInvalidException</i>	If the PIN is incorrect or the key data is malformed leading to decryption failure.
<i>KeyInvalidException</i>	If the decrypted data cannot be loaded as a valid PEM-encoded private key.

5.12.2 Variable Documentation

5.12.2.1 KEY_FILE_NAME

```
key_getter.key_getter.KEY_FILE_NAME = "private_key.key"
```

The expected filename of the encrypted private key on the USB drive.

5.12.2.2 LINUX_PLATFORM_NAME

```
key_getter.key_getter.LINUX_PLATFORM_NAME = "Linux"
```

String constant representing the Linux platform identifier.

5.12.2.3 WINDOWS_PLATFORM_NAME

```
key_getter.key_getter.WINDOWS_PLATFORM_NAME = "Windows"
```

String constant representing the Windows platform identifier.

5.13 key_getter.usb_finder_linux Namespace Reference

Functions

- list[str] [_get_usb_devices_linux](#) ()
Identifies USB block devices connected to a Linux system.
- list[str] [get_usb_mount_paths_linux](#) ()
Retrieves the mount paths for all connected USB storage devices on a Linux system.

5.13.1 Function Documentation

5.13.1.1 `_get_usb_devices_linux()`

```
list[str] key_getter.usb_finder_linux._get_usb_devices_linux ( ) [private]
```

Identifies USB block devices connected to a Linux system.

This function filters `/sys/block` for entries corresponding to USB storage devices (typically `sd*`). It checks the device path for the presence of "usb" to confirm it's a USB device. Source: <https://stackoverflow.com/a/64000192>

Returns

A list of base names for USB block devices (e.g., ['sdb', 'sdc']). @rtype list[str]

5.13.1.2 `get_usb_mount_paths_linux()`

```
list[str] key_getter.usb_finder_linux.get_usb_mount_paths_linux ( )
```

Retrieves the mount paths for all connected USB storage devices on a Linux system.

This function first gets the list of USB block devices using `_get_usb_devices_linux`. Then, for each device, it uses the `lsblk` command to find its mount point(s). It parses the output of `lsblk` to extract and return a list of valid mount paths. Source: <https://stackoverflow.com/a/64000192>

Returns

A list of strings, where each string is an absolute mount path of a USB device. Returns an empty list if no USB devices are mounted or found. @rtype list[str]

5.14 key_getter.usb_finder_windows Namespace Reference

Functions

- list[str] [get_usb_mount_paths_windows\(\)](#)
Retrieves the drive letters for all connected USB storage devices on a Windows system.
- list[str] [_get_usb_devices_windows\(\)](#)
Identifies removable logical disks on a Windows system.

5.14.1 Function Documentation

5.14.1.1 `_get_usb_devices_windows()`

```
list[str] key_getter.usb_finder_windows._get_usb_devices_windows ( ) [private]
```

Identifies removable logical disks on a Windows system.

This function queries WMI for `Win32_LogicalDisk` instances and filters them based on `DriveType == 2`, which indicates a removable disk. It collects the `DeviceID` (drive letter, e.g., "E:") for each such disk. Source: <https://github.com/tjguk/wmi/blob/master/docs/cookbook.rst#find-drive-types>

Returns

A list of strings, where each string is the `DeviceID` (drive letter) of a removable disk (e.g., 'E:', 'F:'). @rtype list[str]

<https://github.com/tjguk/wmi/blob/master/docs/cookbook.rst#find-drive-types>

Function to get device ID from removable disk

5.14.1.2 `get_usb_mount_paths_windows()`

```
list[str] key_getter.usb_finder_windows.get_usb_mount_paths_windows ( )
```

Retrieves the drive letters for all connected USB storage devices on a Windows system.

This function first calls `get_usb_devices_windows()` to get a list of `DeviceIDs` (drive letters) for removable drives. It then formats these `DeviceIDs` into a list of strings representing paths (e.g., "C:", "D:").

Returns

A list of strings, where each string is a drive letter followed by a colon (e.g., 'E:', 'F:') representing detected USB drives. Returns an empty list if no removable drives are found. @rtype list[str]

Convert device ID to the path

5.15 main Namespace Reference

Classes

- class [App](#)
Main application class that inherits from `tkinter.Tk`.

Variables

- int [APP_WIDTH](#) = 800
The width of the application window in pixels.
- int [APP_HEIGHT](#) = 600
The height of the application window in pixels.
- string [APP_TITLE](#) = 'TEST APP'
The title of the application window.

5.15.1 Variable Documentation

5.15.1.1 APP_HEIGHT

```
int main.APP_HEIGHT = 600
```

The height of the application window in pixels.

5.15.1.2 APP_TITLE

```
string main.APP_TITLE = 'TEST APP'
```

The title of the application window.

5.15.1.3 APP_WIDTH

```
int main.APP_WIDTH = 800
```

The width of the application window in pixels.

5.16 pdf_signer Namespace Reference

Namespaces

- [signer](#)
- [verifier](#)

5.17 pdf_signer.signer Namespace Reference

Functions

- def [sign](#) (rsa.RSAPrivateKey private_key, str pdf_in_path, str pdf_out_path)
Signs a PDF document using a provided RSA private key.
- Tuple[asn1_x509.Certificate, asn1_keys.PrivateKeyInfo] [_generate_self_signed_cert](#) (rsa.RSAPrivateKey private_key)
Generates a self-signed X.509 certificate and private key information in ASN.1 format.

5.17.1 Function Documentation

5.17.1.1 _generate_self_signed_cert()

```
Tuple[asn1_x509.Certificate, asn1_keys.PrivateKeyInfo] pdf_signer.signer._generate_self_↵
signed_cert (
    rsa.RSAPrivateKey private_key ) [private]
```

Generates a self-signed X.509 certificate and private key information in ASN.1 format.

This internal helper function takes an RSA private key and creates a self-signed certificate suitable for use with pyhanko. The certificate has a common name "myPAdESCertificate" and is valid for 10 years.

Parameters

<i>private_key</i>	The RSA private key object from which to generate the public key for the certificate and to sign the certificate. @type private_key rsa.RSAPrivateKey
--------------------	---

Returns

A tuple containing:

- `asn1_cert`: The generated self-signed certificate in `asn1crypto.x509.Certificate` format.
- `asn1_private_key`: The private key information in `asn1crypto.keys.PrivateKeyInfo` format. @rtype Tuple[asn1_x509.Certificate, asn1_keys.PrivateKeyInfo]

5.17.1.2 sign()

```
def pdf_signer.signer.sign (
    rsa.RSAPrivateKey private_key,
    str pdf_in_path,
    str pdf_out_path )
```

Signs a PDF document using a provided RSA private key.

This function creates a self-signed certificate from the given private key and uses it to apply a digital signature to the input PDF. The signed PDF is saved to the specified output path. A signature field is added to the first page of the PDF. If an error occurs during signing, any partially created output file is removed.

Parameters

<i>private_key</i>	The RSA private key object to use for signing. @type private_key rsa.RSAPrivateKey
<i>pdf_in_path</i>	The file system path to the input PDF document that needs to be signed. @type pdf_in_path str
<i>pdf_out_path</i>	The file system path where the signed PDF document will be saved. @type pdf_out_path str

Exceptions

<i>FileNotFoundError</i>	When the input file doesn't exist
<i>PdfReadError</i>	When an error occurs during signature or while reading the input PDF file

5.18 pdf_signer.verifier Namespace Reference

Classes

- class [NoSignatureFound](#)

Exception raised when a PDF document does not contain any embedded digital signatures.

Functions

- bool [verify](#) (rsa.RSAPublicKey public_key, str pdf_path)

Verifies the digital signature found in a PDF document against a provided public key.

5.18.1 Function Documentation

5.18.1.1 [verify\(\)](#)

```
bool pdf_signer.verifier.verify (
    rsa.RSAPublicKey public_key,
    str pdf_path )
```

Verifies the digital signature found in a PDF document against a provided public key.

This function reads a PDF, extracts its first embedded signature, and performs two main checks:

1. It compares the public key embedded in the signature's certificate with the `public_key` argument. If they do not match, verification fails (returns `False`).
2. It validates the integrity of the signature itself using `pyhanko`'s validation mechanism. For self-signed certificates, it creates a `ValidationContext` trusting the embedded certificate itself to validate the signature.

Parameters

<i>public_key</i>	The RSA public key expected to correspond to the signature. @type public_key rsa.RSAPublicKey
<i>pdf_path</i>	The file system path to the PDF document whose signature is to be verified. @type pdf_path str

Returns

True if the embedded public key matches the provided `public_key` AND the signature is intact
Returns `False` otherwise. @rtype bool

Exceptions

<i>FileNotFoundError</i>	If the <code>pdf_path</code> does not exist.
<i>NoSignatureFound</i>	If the PDF document does not contain any embedded signatures.
<i>PdfReadError</i>	When an error occurs during verifying or while reading the PDF file

Chapter 6

Class Documentation

6.1 main.App Class Reference

Main application class that inherits from tkinter.Tk.

Inheritance diagram for main.App:

Collaboration diagram for main.App:

Public Member Functions

- def `__init__` (self)
Initializes the `App` class.
- def `start_signing` (self)
Switches the current frame to the `KeyFromUSBFrame`.
- def `start_verifying` (self)
Switches the current frame to the `VerifyingFrame`.
- def `get_key_from_usb_result` (self, rsa.RSAPrivateKey key)
Handles the result of the USB key retrieval and switches to the `SigningFrame`.
- def `main_menu` (self)
Switches the current frame back to the `StartFrame` (main menu).
- def `__init__` (self)

Public Attributes

- `current_frame`

Private Member Functions

- def `_change_frame` (self, tk.Frame frame)
Internal method to change the currently displayed frame.

6.1.1 Detailed Description

Main application class that inherits from `tkinter.Tk`.

Main application class for key generation GUI.

This class is responsible for initializing the main window and managing frame transitions.

This class inherits from `tk.Tk` and loads the main GUI frame responsible for generating RSA keys and encrypting them using a 4-digit PIN.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `__init__()` [1/2]

```
def main.App.__init__ (
    self )
```

Initializes the [App](#) class.

Sets up the window title, geometry, and resizability. It also initializes and displays the starting frame.

6.1.2.2 `__init__()` [2/2]

```
def main.App.__init__ (
    self )
```

6.1.3 Member Function Documentation

6.1.3.1 `_change_frame()`

```
def main.App._change_frame (
    self,
    tk.Frame frame ) [private]
```

Internal method to change the currently displayed frame.

Parameters

<i>frame</i>	The new <code>tkinter.Frame</code> to display. @type frame <code>tk.Frame</code>
--------------	--

Destroys the current frame and packs the new frame into the window.

6.1.3.2 get_key_from_usb_result()

```
def main.App.get_key_from_usb_result (
    self,
    rsa.RSAPrivateKey key )
```

Handles the result of the USB key retrieval and switches to the SigningFrame.

Parameters

key	The RSA private key retrieved from the USB device. @type key rsa.RSAPrivateKey
-----	--

6.1.3.3 main_menu()

```
def main.App.main_menu (
    self )
```

Switches the current frame back to the StartFrame (main menu).

6.1.3.4 start_signing()

```
def main.App.start_signing (
    self )
```

Switches the current frame to the KeyFromUSBFrame.

6.1.3.5 start_verifying()

```
def main.App.start_verifying (
    self )
```

Switches the current frame to the VerifyingFrame.

6.1.4 Member Data Documentation

6.1.4.1 current_frame

```
main.App.current_frame
```

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

- signing/[main.py](#)

6.2 frames.generate_window.GenerateKeys Class Reference

Inheritance diagram for frames.generate_window.GenerateKeys:

6.3 frames.usb_key_get.KeyFromUSBFrame Class Reference

The [KeyFromUSBFrame](#) class handles the UI for retrieving a private key from a USB drive.

Inheritance diagram for frames.usb_key_get.KeyFromUSBFrame:

Collaboration diagram for frames.usb_key_get.KeyFromUSBFrame:

Public Member Functions

- `def __init__ (self, tk.Tk parent, Callable[[rsa.RSAPrivateKey], None] on_key_retrieved_callback)`
Initializes the [KeyFromUSBFrame](#).

Public Attributes

- [on_key_retrieved_callback](#)
- [status_label](#)
- [pin_entry](#)
- [action_button](#)

Private Member Functions

- `def _setup_ui (self)`
Sets up the user interface elements for the [KeyFromUSBFrame](#).
- `def _update_feedback (self, str status_message, str button_text, Callable button_command=None)`
Updates the status label and action button with new messages and commands.
- `def _process_pin_and_get_key (self)`
Processes the entered PIN and attempts to retrieve the private key from a USB drive.

6.3.1 Detailed Description

The [KeyFromUSBFrame](#) class handles the UI for retrieving a private key from a USB drive.

This frame prompts the user for a PIN, attempts to read and decrypt the key, and then calls a callback function with the retrieved key or displays error messages. It inherits from `tk.Frame`.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
def frames.usb_key_get.KeyFromUSBFrame.__init__ (
    self,
    tk.Tk parent,
    Callable[[rsa.RSAPrivateKey], None] on_key_retrieved_callback )
```

Initializes the [KeyFromUSBFrame](#).

Parameters

<i>parent</i>	The parent tk.Tk window or tk.Frame that this frame will be placed in. @type parent tk.Tk
<i>on_key_retrieved_callback</i>	A function to be called when the private key is successfully retrieved. This callback should take one argument, the rsa.RSAPrivateKey, and return None. @type on_key_retrieved_callback Callable[[rsa.RSAPrivateKey], None]

6.3.3 Member Function Documentation

6.3.3.1 _process_pin_and_get_key()

```
def frames.usb_key_get.KeyFromUSBFrame._process_pin_and_get_key (
    self ) [private]
```

Processes the entered PIN and attempts to retrieve the private key from a USB drive.

This method is called when the action button is pressed. It validates the PIN format, then calls the `key_getter.get_key` service. Based on the outcome, it either calls the `on_key_retrieved_callback` with the key or updates the UI with an appropriate error message using `_update_feedback`. It handles various exceptions that can be raised during the key retrieval process.

6.3.3.2 _setup_ui()

```
def frames.usb_key_get.KeyFromUSBFrame._setup_ui (
    self ) [private]
```

Sets up the user interface elements for the [KeyFromUSBFrame](#).

This private method creates and arranges the instruction label, PIN entry field, and action button within the frame.

Creates and arranges UI elements within the frame.

6.3.3.3 _update_feedback()

```
def frames.usb_key_get.KeyFromUSBFrame._update_feedback (
    self,
    str status_message,
    str button_text,
    Callable button_command = None ) [private]
```

Updates the status label and action button with new messages and commands.

Parameters

<i>status_message</i>	The message to display in the status label. @type status_message str
<i>button_text</i>	The new text for the action button. @type button_text str
<i>button_command</i>	The new command to associate with the action button. Defaults to _process_pin_and_get_key. @type button_command Callable

This is a helper method to centralize UI updates for feedback.

6.3.4 Member Data Documentation

6.3.4.1 action_button

```
frames.usb_key_get.KeyFromUSBFrame.action_button
```

6.3.4.2 on_key_retrieved_callback

```
frames.usb_key_get.KeyFromUSBFrame.on_key_retrieved_callback
```

6.3.4.3 pin_entry

```
frames.usb_key_get.KeyFromUSBFrame.pin_entry
```

6.3.4.4 status_label

```
frames.usb_key_get.KeyFromUSBFrame.status_label
```

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

- [signing/frames/usb_key_get.py](#)

6.4 key_getter.key_getter.KeyInvalidException Class Reference

Exception raised when the decrypted key data cannot be parsed as a valid RSA private key.

Inheritance diagram for key_getter.key_getter.KeyInvalidException:

Collaboration diagram for key_getter.key_getter.KeyInvalidException:

6.4.1 Detailed Description

Exception raised when the decrypted key data cannot be parsed as a valid RSA private key.

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

- [signing/services/key_getter/key_getter.py](#)

6.5 `key_getter.key_getter.KeyOrPinInvalidException` Class Reference

Exception raised when the provided PIN is incorrect or the key file is corrupted/cannot be decrypted with the PIN.

Inheritance diagram for `key_getter.key_getter.KeyOrPinInvalidException`:

Collaboration diagram for `key_getter.key_getter.KeyOrPinInvalidException`:

6.5.1 Detailed Description

Exception raised when the provided PIN is incorrect or the key file is corrupted/cannot be decrypted with the PIN.

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

- [signing/services/key_getter/key_getter.py](#)

6.6 `key_getter.key_getter.MultipleKeysFoundException` Class Reference

Exception raised when the key file is found on multiple USB drives, creating ambiguity.

Inheritance diagram for `key_getter.key_getter.MultipleKeysFoundException`:

Collaboration diagram for `key_getter.key_getter.MultipleKeysFoundException`:

6.6.1 Detailed Description

Exception raised when the key file is found on multiple USB drives, creating ambiguity.

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

- [signing/services/key_getter/key_getter.py](#)

6.7 `key_getter.key_getter.NoKeyFoundException` Class Reference

Exception raised when the key file is not found on any detected USB drives.

Inheritance diagram for `key_getter.key_getter.NoKeyFoundException`:

Collaboration diagram for `key_getter.key_getter.NoKeyFoundException`:

6.7.1 Detailed Description

Exception raised when the key file is not found on any detected USB drives.

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

- [signing/services/key_getter/key_getter.py](#)

6.8 pdf_signer.verifier.NoSignatureFound Class Reference

Exception raised when a PDF document does not contain any embedded digital signatures.

Inheritance diagram for pdf_signer.verifier.NoSignatureFound:

Collaboration diagram for pdf_signer.verifier.NoSignatureFound:

6.8.1 Detailed Description

Exception raised when a PDF document does not contain any embedded digital signatures.

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

- [signing/services/pdf_signer/verifier.py](#)

6.9 key_getter.key_getter.NoUSBDrivesFoundException Class Reference

Exception raised when no USB drives are found connected to the system.

Inheritance diagram for key_getter.key_getter.NoUSBDrivesFoundException:

Collaboration diagram for key_getter.key_getter.NoUSBDrivesFoundException:

6.9.1 Detailed Description

Exception raised when no USB drives are found connected to the system.

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

- [signing/services/key_getter/key_getter.py](#)

6.10 frames.signing.SigningFrame Class Reference

The [SigningFrame](#) class provides the UI for the PDF signing process.

Inheritance diagram for frames.signing.SigningFrame:

Collaboration diagram for frames.signing.SigningFrame:

Public Member Functions

- `def __init__ (self, tk.Tk parent, rsa.RSAPrivateKey private_key, Callable[[], None] end_signing_callback)`
Initializes the [SigningFrame](#).

Public Attributes

- [private_key](#)
- [end_signing_callback](#)
- [source_pdf_path_var](#)
- [target_pdf_path_var](#)
- [status_label](#)
- [source_pdf_path_entry](#)
- [target_pdf_path_entry](#)
- [sign_button](#)

Private Member Functions

- `def _setup_ui (self)`
Sets up the user interface elements for the [SigningFrame](#).
- `def _select_source_pdf_file (self)`
Opens a file dialog to allow the user to select the source PDF file.
- `def _select_target_pdf_path (self)`
Opens a file dialog to allow the user to select the target path and filename for the signed PDF.
- `def _update_feedback (self, str status_message, str button_text, Callable button_command=None)`
Helper method to update the status label and the main action button's text and command.
- `def _sign_pdf_document (self)`
Handles the PDF signing process based on selected file paths and the provided private key.

6.10.1 Detailed Description

The [SigningFrame](#) class provides the UI for the PDF signing process.

This frame allows users to select a source PDF, specify a target location for the signed PDF, and initiate the signing operation. It handles user interactions and feedback. Inherits from tk.Frame.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__()

```
def frames.signing.SigningFrame.__init__ (
    self,
    tk.Tk parent,
    rsa.RSAPrivateKey private_key,
    Callable[[ ], None] end_signing_callback )
```

Initializes the [SigningFrame](#).

Parameters

<i>parent</i>	The parent tk.Tk window or tk.Frame that this frame will be placed in. @type parent tk.Tk
<i>private_key</i>	The RSA private key to be used for signing the PDF document. @type private_key rsa.RSAPrivateKey
<i>end_signing_callback</i>	A function to be called when the signing process is completed (either successfully or to go back). This callback should take no arguments and return None. @type end_signing_callback Callable[[], None]

6.10.3 Member Function Documentation

6.10.3.1 `_select_source_pdf_file()`

```
def frames.signing.SigningFrame._select_source_pdf_file (
    self ) [private]
```

Opens a file dialog to allow the user to select the source PDF file.

Updates the `source_pdf_path_var` and the corresponding entry field with the path of the file selected by the user.

6.10.3.2 `_select_target_pdf_path()`

```
def frames.signing.SigningFrame._select_target_pdf_path (
    self ) [private]
```

Opens a file dialog to allow the user to select the target path and filename for the signed PDF.

Updates the `target_pdf_path_var` and the corresponding entry field with the path chosen by the user. Suggests ".pdf" as the default extension.

6.10.3.3 `_setup_ui()`

```
def frames.signing.SigningFrame._setup_ui (
    self ) [private]
```

Sets up the user interface elements for the [SigningFrame](#).

This private method creates and arranges labels, entry fields for paths, and buttons for file selection and signing.

6.10.3.4 `_sign_pdf_document()`

```
def frames.signing.SigningFrame._sign_pdf_document (
    self ) [private]
```

Handles the PDF signing process based on selected file paths and the provided private key.

This method is called when the sign button is pressed. It validates that both source and target paths are provided and are not the same. It then attempts to sign the PDF using `pdf_signer.sign` and updates the UI with success or error messages via `_update_feedback`.

Handles the PDF signing process and UI feedback.

6.10.3.5 `_update_feedback()`

```
def frames.signing.SigningFrame._update_feedback (
    self,
    str status_message,
    str button_text,
    Callable button_command = None ) [private]
```

Helper method to update the status label and the main action button's text and command.

Parameters

<i>status_message</i>	The message to display in the status label. @type status_message str
<i>button_text</i>	The new text for the action. @type button_text str
<i>button_command</i>	The new command to associate with the action button. Defaults to <code>_sign_pdf_document</code> if None. @type button_command Callable

Helper to update status label and sign button.

6.10.4 Member Data Documentation

6.10.4.1 `end_signing_callback`

```
frames.signing.SigningFrame.end_signing_callback
```

6.10.4.2 `private_key`

```
frames.signing.SigningFrame.private_key
```

6.10.4.3 sign_button

`frames.signing.SigningFrame.sign_button`

6.10.4.4 source_pdf_path_entry

`frames.signing.SigningFrame.source_pdf_path_entry`

6.10.4.5 source_pdf_path_var

`frames.signing.SigningFrame.source_pdf_path_var`

6.10.4.6 status_label

`frames.signing.SigningFrame.status_label`

6.10.4.7 target_pdf_path_entry

`frames.signing.SigningFrame.target_pdf_path_entry`

6.10.4.8 target_pdf_path_var

`frames.signing.SigningFrame.target_pdf_path_var`

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

- [signing/frames/signing.py](#)

6.11 frames.start.StartFrame Class Reference

The [StartFrame](#) class provides user interface for the application.

Inheritance diagram for frames.start.StartFrame:

Collaboration diagram for frames.start.StartFrame:

Public Member Functions

- `def __init__ (self, tk.Tk parent, Callable[[], None] on_signing_chosen_callback, Callable[[], None] on_verifying_chosen_callback)`

Initializes the [StartFrame](#).

Public Attributes

- [on_signing_chosen_callback](#)
- [on_verifying_chosen_callback](#)

Private Member Functions

- `def _setup_ui (self)`

Sets up the user interface elements for the [StartFrame](#).

6.11.1 Detailed Description

The [StartFrame](#) class provides user interface for the application.

This frame displays options for the user to either start the PDF signing process or the PDF signature verification process. It inherits from `tk.Frame`.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 __init__()

```
def frames.start.StartFrame.__init__ (
    self,
    tk.Tk parent,
    Callable[[], None] on_signing_chosen_callback,
    Callable[[], None] on_verifying_chosen_callback )
```

Initializes the [StartFrame](#).

Parameters

<i>parent</i>	The parent <code>tk.Tk</code> window or <code>tk.Frame</code> that this frame will be placed in. @type parent <code>tk.Tk</code>
<i>on_signing_chosen_callback</i>	A function to be called when the user chooses to sign a PDF. This callback should take no arguments and return <code>None</code> . @type <code>on_signing_chosen_callback</code> <code>Callable[[], None]</code>
<i>on_verifying_chosen_callback</i>	A function to be called when the user chooses to verify a PDF signature. This callback should take no arguments and return <code>None</code> . @type <code>on_verifying_chosen_callback</code> <code>Callable[[], None]</code>

6.11.3 Member Function Documentation

6.11.3.1 `_setup_ui()`

```
def frames.start.StartFrame._setup_ui (
    self ) [private]
```

Sets up the user interface elements for the [StartFrame](#).

This private method creates and arranges the instruction label and the sign and verify buttons within the frame.

Creates and arranges UI elements within the frame.

6.11.4 Member Data Documentation

6.11.4.1 `on_signing_chosen_callback`

```
frames.start.StartFrame.on_signing_chosen_callback
```

6.11.4.2 `on_verifying_chosen_callback`

```
frames.start.StartFrame.on_verifying_chosen_callback
```

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

- [signing/frames/start.py](#)

6.12 `key_getter.key_getter.UnsupportedPlatformException` Class Reference

Exception raised when the current operating system is not supported for USB key retrieval.

Inheritance diagram for `key_getter.key_getter.UnsupportedPlatformException`:

Collaboration diagram for `key_getter.key_getter.UnsupportedPlatformException`:

6.12.1 Detailed Description

Exception raised when the current operating system is not supported for USB key retrieval.

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

- [signing/services/key_getter/key_getter.py](#)

6.13 frames.verifyng.VerifyingFrame Class Reference

The [VerifyingFrame](#) class provides the UI for PDF signature verification.

Inheritance diagram for frames.verifyng.VerifyingFrame:

Collaboration diagram for frames.verifyng.VerifyingFrame:

Public Member Functions

- `def __init__ (self, tk.Tk parent, Callable[[], None] end_verifying_callback)`
Initializes the [VerifyingFrame](#).

Public Attributes

- [end_verifying_callback](#)
- [pdf_to_verify_path_var](#)
- [public_key_path_var](#)
- [status_label](#)
- [pdf_to_verify_entry](#)
- [public_key_entry](#)
- [verify_button](#)

Private Member Functions

- `def _setup_ui (self)`
Sets up the user interface elements for the [VerifyingFrame](#).
- `def _select_file (self, str title, list file_types, tk.StringVar string_var)`
Generic helper method to open a file dialog and update a Tkinter StringVar with the selected path.
- `def _select_pdf_to_verify_file (self)`
Opens a file dialog for the user to select the PDF file to be verified.
- `def _select_public_key_file (self)`
Opens a file dialog for the user to select the public key file.
- `def _update_feedback (self, str status_message, str button_text, Callable button_command=None)`
Helper method to update the status label and the main action button.
- `rsa.RSAPublicKey|None _load_public_key (self, str path)`
Loads an RSA public key from a PEM-encoded file at the given path.
- `def _process_verification (self)`
Handles the PDF signature verification process.

6.13.1 Detailed Description

The [VerifyingFrame](#) class provides the UI for PDF signature verification.

This frame allows users to select a PDF file and a public key file. It then attempts to verify the PDF's signature using the provided key and displays the result (valid, invalid, or error messages). Inherits from tk.Frame.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `__init__()`

```
def frames.verifying.VerifyingFrame.__init__ (
    self,
    tk.Tk parent,
    Callable[[], None] end_verifying_callback )
```

Initializes the [VerifyingFrame](#).

Parameters

<i>parent</i>	The parent tk.Tk window or tk.Frame that this frame will be placed in. @type parent tk.Tk
<i>end_verifying_callback</i>	A function to be called when the verification process is completed. This callback should take no arguments and return None. @type end_verifying_callback Callable[[], None]

6.13.3 Member Function Documentation

6.13.3.1 `_load_public_key()`

```
rsa.RSAPublicKey | None frames.verifying.VerifyingFrame._load_public_key (
    self,
    str path ) [private]
```

Loads an RSA public key from a PEM-encoded file at the given path.

Parameters

<i>path</i>	The file system path to the public key file. @type path str
-------------	---

Returns

The loaded `rsa.RSAPublicKey` object if successful, otherwise `None`.

If the file is not found or an error occurs during parsing, it updates the UI with an error message via `_update_feedback` and returns `None`.

6.13.3.2 `_process_verification()`

```
def frames.verifyng.VerifyingFrame._process_verification (
    self ) [private]
```

Handles the PDF signature verification process.

This method is called when the verify button is pressed. It retrieves the PDF and public key paths from the UI. Validates that both paths are provided. Loads the public key using `_load_public_key`. Calls `pdf_signer.verify` to perform the signature verification. Updates the UI with the result (valid, invalid) or an error message

6.13.3.3 `_select_file()`

```
def frames.verifyng.VerifyingFrame._select_file (
    self,
    str title,
    list file_types,
    tk.StringVar string_var ) [private]
```

Generic helper method to open a file dialog and update a Tkinter `StringVar` with the selected path.

Parameters

<i>title</i>	The title for the file dialog window. @type title str
<i>file_types</i>	A list of tuples defining the acceptable file types for the dialog. @type file_types list
<i>string_var</i>	The <code>tk.StringVar</code> instance to update with the selected file path. @type string_var <code>tk.StringVar</code>

Helper to open a file dialog and update a `StringVar`.

6.13.3.4 `_select_pdf_to_verify_file()`

```
def frames.verifyng.VerifyingFrame._select_pdf_to_verify_file (
    self ) [private]
```

Opens a file dialog for the user to select the PDF file to be verified.

Calls `_select_file` with appropriate parameters for PDF selection and updates `self.pdf_to_verify_path_var`.

6.13.3.5 `_select_public_key_file()`

```
def frames.verifying.VerifyingFrame._select_public_key_file (
    self ) [private]
```

Opens a file dialog for the user to select the public key file.

Calls `_select_file` with appropriate parameters for public key selection and updates `self.public_key_path_var`.

6.13.3.6 `_setup_ui()`

```
def frames.verifying.VerifyingFrame._setup_ui (
    self ) [private]
```

Sets up the user interface elements for the [VerifyingFrame](#).

This private method creates and arranges labels, entry fields for file paths, and buttons for file selection and initiating verification.

6.13.3.7 `_update_feedback()`

```
def frames.verifying.VerifyingFrame._update_feedback (
    self,
    str status_message,
    str button_text,
    Callable button_command = None ) [private]
```

Helper method to update the status label and the main action button.

Parameters

<i>status_message</i>	The message to display in the status label. @type status_message str
<i>button_text</i>	The new text for the action button. @type button_text str
<i>button_command</i>	The new command to associate with the action button. Defaults to <code>_process_verification</code> if None. @type button_command Callable

6.13.4 Member Data Documentation

6.13.4.1 `end_verifying_callback`

```
frames.verifying.VerifyingFrame.end_verifying_callback
```

6.13.4.2 pdf_to_verify_entry

```
frames.verifyng.VerifyingFrame.pdf_to_verify_entry
```

6.13.4.3 pdf_to_verify_path_var

```
frames.verifyng.VerifyingFrame.pdf_to_verify_path_var
```

6.13.4.4 public_key_entry

```
frames.verifyng.VerifyingFrame.public_key_entry
```

6.13.4.5 public_key_path_var

```
frames.verifyng.VerifyingFrame.public_key_path_var
```

6.13.4.6 status_label

```
frames.verifyng.VerifyingFrame.status_label
```

6.13.4.7 verify_button

```
frames.verifyng.VerifyingFrame.verify_button
```

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

- signing/frames/[verifyng.py](#)

Chapter 7

File Documentation

7.1 generating/frames/generate_window.py File Reference

GUI window for generating RSA key pairs.

Classes

- class [frames.generate_window.GenerateKeys](#)

Namespaces

- [frames.generate_window](#)

Variables

- string [frames.generate_window.PRIVATE_KEY_NAME](#) = "private_key.key"
Default filename for the private key.
- string [frames.generate_window.PUBLIC_KEY_NAME](#) = "public_key.key"
Default filename for the public key.
- string [frames.generate_window.FOREGROUND_COLOR](#) = "#ffffff"
- string [frames.generate_window.BACKGROUND_COLOR](#) = "#1e1e1e"
- string [frames.generate_window.BACKGROUND2_COLOR](#) = "#2d2d2d"
- string [frames.generate_window.BLUE_BUTTON_COLOR](#) = "#007acc"
- string [frames.generate_window.ACTIVATE_BUTTON_COLOR](#) = "#005f99"

7.1.1 Detailed Description

GUI window for generating RSA key pairs.

Provides input fields for taking a path to saving public/private keys, setting 4-digit PIN and a progress bar to display the current status of the generating process.

7.2 generating/key_generate/AES_key_generator.py File Reference

Namespaces

- [key_generate.AES_key_generator](#)

Functions

- def [key_generate.AES_key_generator.hash_pin](#) (str pin)
Generates a 256-bit key from PIN code.
- bool [key_generate.AES_key_generator.aes_encrypt_file](#) (str file_to_encrypt, str pin)
Encrypts a file using a 4-digit PIN code and AES encryption.
- (bool, bytes) [key_generate.AES_key_generator.aes_decrypt_file](#) (str file_to_decrypt, str pin)
Decrypts a file using a 4-digit PIN code and AES decryption.

7.3 generating/key_generate/RSA_key_generator.py File Reference

Namespaces

- [key_generate.RSA_key_generator](#)

Functions

- bool [key_generate.RSA_key_generator.generate_keys](#) (str public_key_location, str private_key_location)
Generate public/private key pairs.

7.4 signing/frames/__init__.py File Reference

Namespaces

- [frames](#)

7.5 signing/services/key_getter/__init__.py File Reference

Namespaces

- [key_getter](#)

7.6 signing/services/pdf_signer/__init__.py File Reference

Namespaces

- [pdf_signer](#)

7.7 generating/frames/___init___py File Reference

Namespaces

- [frames](#)

7.8 generating/key_generate/___init___py File Reference

Namespaces

- [key_generate](#)

7.9 signing/frames/signing.py File Reference

A Tkinter Frame for selecting a PDF, choosing an output location, and signing the PDF.

Classes

- class [frames.signing.SigningFrame](#)

The [SigningFrame](#) class provides the UI for the PDF signing process.

Namespaces

- [frames.signing](#)

Variables

- tuple [frames.signing.LARGE_FONT_CONFIG](#) = ("TkDefaultFont", 16)
Font configuration for large text elements (e.g., status labels, buttons).
- int [frames.signing.DEFAULT_WRAP_LENGTH](#) = 750
Default wrap length in pixels for text in labels to ensure proper layout.
- list [frames.signing.PDF_FILE_TYPES](#) = [("PDF files", "*.pdf")]
File type filter used in file dialogs, restricting selection to PDF files.
- string [frames.signing.SELECT_SOURCE_PDF_TITLE](#) = "Select the PDF file to sign"
Title for the file dialog when selecting the source PDF to be signed.
- string [frames.signing.SELECT_TARGET_PDF_TITLE](#) = "Set target location and filename for signed PDF"
Title for the file dialog when selecting the output path for the signed PDF.
- string [frames.signing.INITIAL_STATUS_TEXT](#) = "Please choose the PDF file to sign and where the signed PDF file should be placed."
Initial instructional text displayed in the status label.
- string [frames.signing.SOURCE_PDF_LABEL_TEXT](#) = "Selected PDF file to sign:"
Text for the label indicating the selected source PDF file path.
- string [frames.signing.TARGET_PDF_LABEL_TEXT](#) = "Location where the signed PDF file will be saved:"
Text for the label indicating the selected target path for the signed PDF file.
- string [frames.signing.SELECT_SOURCE_BUTTON_TEXT](#) = "Select PDF file"

- Text for the button used to trigger source PDF file selection.*
- string `frames.signing.SELECT_TARGET_BUTTON_TEXT` = "Set target location"
- Text for the button used to trigger target PDF path selection.*
- string `frames.signing.SIGN_BUTTON_INITIAL_TEXT` = "Sign the PDF file"
- Initial text for the button that initiates the PDF signing process.*
- string `frames.signing.RETRY_BUTTON_TEXT` = "Try Again"
- Text for the main action button when a retry is suggested after an error.*
- string `frames.signing.GO_BACK_BUTTON_TEXT` = "Go back to main menu"
- Text for the main action button to navigate back to the main menu after success.*
- string `frames.signing.PATHS_REQUIRED_ERROR_TEXT` = "Both original and target PDF locations are required. Please select them."
- Error message displayed when either source or target PDF paths are not selected.*
- string `frames.signing.PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT` = "Original and target PDF locations cannot be the same. Please change one of them and try again."
- Error message displayed when source or target PDF paths are the same.*
- string `frames.signing.SIGNING_SUCCESS_TEXT` = "Successfully signed PDF and saved to the designated location."
- Message displayed in the status label upon successful PDF signing.*
- string `frames.signing.PDF_READ_ERROR_TEXT` = "Selected source file is not a valid PDF file. Please verify and try again."
- Error message displayed if the selected source file is not a valid PDF.*
- string `frames.signing.SIGNING_ERROR_TEXT` = "This PDF file may have already been signed or is unsuitable for signing. Please choose a different file."
- Error message displayed if the PDF is already signed or unsuitable for signing.*
- string `frames.signing.UNEXPECTED_SIGNING_ERROR_TEXT` = "An unexpected error occurred during signing: {error_type}. Please try again"
- Error message template for unexpected errors during the signing process.*
- string `frames.signing.FOREGROUND_COLOR` = "#ffffff"
- string `frames.signing.BACKGROUND_COLOR` = "#1e1e1e"
- string `frames.signing.BACKGROUND2_COLOR` = "#2d2d2d"
- string `frames.signing.BLUE_BUTTON_COLOR` = "#007acc"
- string `frames.signing.ACTIVATE_BUTTON_COLOR` = "#005f99"

7.9.1 Detailed Description

A Tkinter Frame for selecting a PDF, choosing an output location, and signing the PDF.

This frame guides the user through selecting an input PDF, specifying an output path for the signed PDF, and then performs the signing operation using a provided private key. It handles UI updates for status messages and error reporting.

7.10 signing/frames/start.py File Reference

A Tkinter Frame providing options to navigate to PDF signing or verification sections.

Classes

- class `frames.start.StartFrame`

The `StartFrame` class provides user interface for the application.

Namespaces

- [frames.start](#)

Variables

- tuple [frames.start.LARGE_FONT_CONFIG](#) = ("TkDefaultFont", 16)
Font configuration for large text elements.
- int [frames.start.DEFAULT_WRAP_LENGTH](#) = 750
Default wrap length in pixels for text in labels.
- int [frames.start.DEFAULT_PADDING_X](#) = 200
Default horizontal padding in pixels for UI elements.
- int [frames.start.DEFAULT_PADDING_Y](#) = 10
Default vertical padding in pixels for general UI elements.
- int [frames.start.BUTTON_PADDING_Y](#) = 20
Specific vertical padding in pixels for buttons.
- string [frames.start.LABEL_TEXT](#) = "Please choose whether you want to sign a PDF or verify the signature of a PDF file."
Text content for the main instruction label on the [StartFrame](#).
- string [frames.start.SIGN_BUTTON_TEXT](#) = "Sign a PDF file"
Text content for the button that initiates the PDF signing process.
- string [frames.start.VERIFY_BUTTON_TEXT](#) = "Verify PDF file signature"
Text content for the button that initiates the PDF signature verification process.
- string [frames.start.FOREGROUND_COLOR](#) = "#ffffff"
- string [frames.start.BACKGROUND_COLOR](#) = "#1e1e1e"
- string [frames.start.BACKGROUND2_COLOR](#) = "#2d2d2d"
- string [frames.start.BLUE_BUTTON_COLOR](#) = "#007acc"
- string [frames.start.ACTIVATE_BUTTON_COLOR](#) = "#005f99"

7.10.1 Detailed Description

A Tkinter Frame providing options to navigate to PDF signing or verification sections.

This frame serves as the initial screen, allowing users to choose between the signing workflow or the verification workflow.

7.11 signing/frames/usb_key_get.py File Reference

A Tkinter Frame for prompting the user for a PIN to read and decrypt a private key from a USB drive.

Classes

- class [frames.usb_key_get.KeyFromUSBFrame](#)
The [KeyFromUSBFrame](#) class handles the UI for retrieving a private key from a USB drive.

Namespaces

- [frames.usb_key_get](#)

Variables

- tuple `frames.usb_key_get.LARGE_FONT_CONFIG` = ("TkDefaultFont", 16)
Font configuration for large text elements.
- int `frames.usb_key_get.DEFAULT_WRAP_LENGTH` = 750
Default wrap length in pixels for text in labels.
- int `frames.usb_key_get.DEFAULT_PADDING_X` = 10
Default horizontal padding in pixels for UI elements.
- int `frames.usb_key_get.DEFAULT_PADDING_Y` = 10
Default vertical padding in pixels for general UI elements.
- int `frames.usb_key_get.INPUT_AREA_PADDING_Y` = 5
Vertical padding in pixels for the PIN input area.
- int `frames.usb_key_get.BUTTON_PADDING_Y` = 20
Specific vertical padding in pixels for buttons.
- string `frames.usb_key_get.INITIAL_INSTRUCTION_TEXT` = "To sign the PDF file, first the key has to be read from the USB drive and deciphered with your PIN."
Initial instruction text displayed to the user.
- string `frames.usb_key_get.PIN_LABEL_TEXT` = "Enter PIN:"
Text for the PIN entry label.
- string `frames.usb_key_get.ACTION_BUTTON_INITIAL_TEXT` = "Find and Read Key"
Initial text for the main action button.
- string `frames.usb_key_get.ACTION_BUTTON_RETRY_TEXT` = "Try Again"
Text for the action button when a retry is suggested.
- string `frames.usb_key_get.ACTION_BUTTON_EXIT_TEXT` = "Exit Program"
Text for the action button when exiting is the only option.
- string `frames.usb_key_get.PIN_REQUIRED_MSG` = "PIN is required. Please enter your 4-digit PIN."
Error message when the PIN is not entered.
- string `frames.usb_key_get.PIN_INVALID_FORMAT_MSG` = "The PIN must be 4 digits. Please try again."
Error message when the PIN format is incorrect.
- string `frames.usb_key_get.UNSUPPORTED_PLATFORM_MSG` = "Error: Current operating system is not supported for USB key retrieval."
Error message for unsupported operating systems.
- string `frames.usb_key_get.NO_USB_DRIVES_MSG` = "No USB drives found. Please insert the USB drive with the key and try again."
Error message when no USB drives are detected.
- string `frames.usb_key_get.NO_KEY_FILE_MSG` = "No key file found on any USB drive. Please ensure the key file is present and try again."
Error message when the key file is not found on USB drives.
- string `frames.usb_key_get.MULTIPLE_KEYS_MSG` = "Multiple key files found across different USB drives. Please ensure only one USB drive with the key file is connected and try again."
Error message when multiple key files are found.
- string `frames.usb_key_get.KEY_OR_PIN_INVALID_MSG` = "Invalid PIN or key file. Please verify your PIN and the key file, then try again."
Error message for an invalid PIN or key file.
- string `frames.usb_key_get.KEY_INVALID_MSG` = "The key file is invalid or corrupted. Please ensure you have the correct key file."
Error message when the key file itself is invalid or corrupted.
- string `frames.usb_key_get.FOREGROUND_COLOR` = "#ffffff"
- string `frames.usb_key_get.BACKGROUND_COLOR` = "#1e1e1e"
- string `frames.usb_key_get.BACKGROUND2_COLOR` = "#2d2d2d"
- string `frames.usb_key_get.BLUE_BUTTON_COLOR` = "#007acc"
- string `frames.usb_key_get.ACTIVATE_BUTTON_COLOR` = "#005f99"

7.11.1 Detailed Description

A Tkinter Frame for prompting the user for a PIN to read and decrypt a private key from a USB drive.

This frame handles user input for a PIN, interacts with the `key_getter` service to retrieve an RSA private key from a USB device, and provides feedback to the user regarding the success or failure of this operation.

7.12 signing/frames/verifying.py File Reference

A Tkinter Frame for selecting a PDF file and a public key to verify the PDF's digital signature.

Classes

- class `frames.verifying.VerifyingFrame`
The `VerifyingFrame` class provides the UI for PDF signature verification.

Namespaces

- `frames.verifying`

Variables

- tuple `frames.verifying.LARGE_FONT_CONFIG` = ("TkDefaultFont", 16)
Font configuration for large text elements like status labels and buttons.
- int `frames.verifying.DEFAULT_WRAP_LENGTH` = 750
Default wrap length in pixels for text in labels to ensure proper UI layout.
- list `frames.verifying.PDF_FILE_TYPES` = [("PDF files", "*.pdf")]
File type filter for PDF file selection dialogs, showing only "*.pdf" files.
- list `frames.verifying.PUBLIC_KEY_FILE_TYPES` = [("Public Key files", "*.pem *.key")]
File type filter for public key file selection dialogs, showing "*.pem" and "*.key" files.
- string `frames.verifying.SELECT_PDF_TO_VERIFY_TITLE` = "Select the PDF file to verify"
Title for the file dialog when selecting the PDF file to be verified.
- string `frames.verifying.SELECT_PUBLIC_KEY_TITLE` = "Select the public key file"
Title for the file dialog when selecting the public key file.
- int `frames.verifying.DEFAULT_PADDING_X` = 10
Default horizontal padding in pixels for UI elements.
- int `frames.verifying.DEFAULT_PADDING_Y` = 10
Default vertical padding in pixels for general UI elements.
- int `frames.verifying.SECTION_SPACING_Y` = 20
Vertical spacing in pixels used between UI sections (e.g., between PDF selection and public key selection).
- int `frames.verifying.BUTTON_PADDING_Y` = 20
Specific vertical padding in pixels for buttons.
- string `frames.verifying.INITIAL_INSTRUCTION_TEXT` = "Please choose the PDF file to verify and the public key corresponding to the signature."
Initial instructional text displayed to the user in the status label.
- string `frames.verifying.VERIFY_BUTTON_INITIAL_TEXT` = "Verify PDF Signature"
Initial text for the button that initiates the PDF signature verification process.

- string `frames.verify.VERIFY_BUTTON_RETRY_TEXT` = "Try Again"
Text for the main action button when a retry is suggested after an error.
- string `frames.verify.VERIFY_BUTTON_GO_BACK_TEXT` = "Go back to main menu"
Text for the main action button to navigate back to the main menu after verification (success or failure).
- string `frames.verify.PATHS_REQUIRED_MSG` = "Both PDF file and public key file locations are required. Please select them."
Error message displayed if either the PDF file or public key file path is not selected.
- string `frames.verify.PUBLIC_KEY_NOT_FOUND_MSG` = "Public key file not found at the specified location. Please check the path and try again."
Error message displayed if the specified public key file cannot be found.
- string `frames.verify.PUBLIC_KEY_INVALID_MSG` = "The selected file is not a valid public key or is corrupted. Please verify the key file."
Error message displayed if the selected public key file is invalid or corrupted.
- string `frames.verify.PDF_INVALID_MSG` = "The selected file is not a valid PDF file. Please verify the PDF and try again."
Error message displayed if the selected PDF file is invalid or cannot be read.
- string `frames.verify.NO_SIGNATURE_MSG` = "This PDF file does not appear to be signed. Please select a signed PDF."
Message displayed if the selected PDF file does not contain a digital signature.
- string `frames.verify.VERIFICATION_ERROR_MSG` = "An error occurred during verification. Please try again."
General error message displayed if an unexpected error occurs during verification.
- string `frames.verify.SIGNATURE_VALID_MSG` = "The signature is VALID."
Message displayed in the status label when the PDF signature is successfully validated.
- string `frames.verify.SIGNATURE_INVALID_MSG` = "The signature is INVALID."
Message displayed in the status label when the PDF signature is found to be invalid.
- string `frames.verify.FOREGROUND_COLOR` = "#ffffff"
- string `frames.verify.BACKGROUND_COLOR` = "#1e1e1e"
- string `frames.verify.BACKGROUND2_COLOR` = "#2d2d2d"
- string `frames.verify.BLUE_BUTTON_COLOR` = "#007acc"
- string `frames.verify.ACTIVATE_BUTTON_COLOR` = "#005f99"

7.12.1 Detailed Description

A Tkinter Frame for selecting a PDF file and a public key to verify the PDF's digital signature.

This frame allows the user to select a PDF document and a public key file to verify the integrity and authenticity of the PDF's digital signature. It interacts with the `pdf_signer` service for the verification logic.

7.13 signing/main.py File Reference

Entry point for the signing application.

Classes

- class `main.App`
Main application class that inherits from `tkinter.Tk`.

Namespaces

- [main](#)

Variables

- int [main.APP_WIDTH](#) = 800
The width of the application window in pixels.
- int [main.APP_HEIGHT](#) = 600
The height of the application window in pixels.
- string [main.APP_TITLE](#) = 'TEST APP'
The title of the application window.

7.13.1 Detailed Description

Entry point for the signing application.

Launches the signing application GUI settings (width, height, title)

7.14 generating/main.py File Reference

Entry point for the key generating application.

Classes

- class [main.App](#)
Main application class that inherits from tkinter.Tk.

Namespaces

- [main](#)

7.14.1 Detailed Description

Entry point for the key generating application.

Launches the key generation GUI with preset width, height and windows title.

7.15 signing/services/key_getter/AES_PIN_decryptor.py File Reference

Provides cryptographic utility functions for hashing and AES decryption.

Namespaces

- [key_getter.AES_PIN_decryptor](#)

Functions

- bytes [key_getter.AES_PIN_decryptor.hash_pin](#) (str pin)
Hashes a numeric PIN string using SHA256.
- bytes [key_getter.AES_PIN_decryptor.aes_decrypt_file](#) (bytes encrypted_data, str pin)
Decrypts data encrypted using AES in EAX mode.

7.15.1 Detailed Description

Provides cryptographic utility functions for hashing and AES decryption.

This module contains functions to hash a PIN using SHA256 and to decrypt data encrypted with AES in EAX mode.

7.16 signing/services/key_getter/key_getter.py File Reference

Retrieves and decrypts a private key from a USB drive.

Classes

- class [key_getter.key_getter.UnsupportedPlatformException](#)
Exception raised when the current operating system is not supported for USB key retrieval.
- class [key_getter.key_getter.NoUSBDrivesFoundException](#)
Exception raised when no USB drives are found connected to the system.
- class [key_getter.key_getter.NoKeyFoundException](#)
Exception raised when the key file is not found on any detected USB drives.
- class [key_getter.key_getter.MultipleKeysFoundException](#)
Exception raised when the key file is found on multiple USB drives, creating ambiguity.
- class [key_getter.key_getter.KeyOrPinInvalidException](#)
Exception raised when the provided PIN is incorrect or the key file is corrupted/cannot be decrypted with the PIN.
- class [key_getter.key_getter.KeyInvalidException](#)
Exception raised when the decrypted key data cannot be parsed as a valid RSA private key.

Namespaces

- [key_getter.key_getter](#)

Functions

- `rsa.RSAPrivateKey key_getter.key_getter.get_key (str pin)`
Retrieves and decrypts the RSA private key from a USB drive using a PIN.
- `bytes key_getter.key_getter._get_key_windows ()`
Internal function to retrieve the encrypted key data from USB drives on Windows.
- `bytes key_getter.key_getter._get_key_linux ()`
Internal function to retrieve the encrypted key data from USB drives on Linux.
- `bytes key_getter.key_getter._get_key_paths (list[str] usb_paths)`
Internal function to search for and read the key file from a list of USB paths.

Variables

- `string key_getter.key_getter.WINDOWS_PLATFORM_NAME = "Windows"`
String constant representing the Windows platform identifier.
- `string key_getter.key_getter.LINUX_PLATFORM_NAME = "Linux"`
String constant representing the Linux platform identifier.
- `string key_getter.key_getter.KEY_FILE_NAME = "private_key.key"`
The expected filename of the encrypted private key on the USB drive.

7.16.1 Detailed Description

Retrieves and decrypts a private key from a USB drive.

This module provides functionality to locate USB drives on Windows and Linux, find a specific key file (`private_key.key`), read its encrypted content, and decrypt it using a PIN to obtain an RSA private key. It defines several custom exceptions to handle various error conditions during this process.

7.17 signing/services/key_getter/usb_finder_linux.py File Reference

Provides functions to find USB device mount paths on Linux systems.

Namespaces

- `key_getter.usb_finder_linux`

Functions

- `list[str] key_getter.usb_finder_linux._get_usb_devices_linux ()`
Identifies USB block devices connected to a Linux system.
- `list[str] key_getter.usb_finder_linux.get_usb_mount_paths_linux ()`
Retrieves the mount paths for all connected USB storage devices on a Linux system.

7.17.1 Detailed Description

Provides functions to find USB device mount paths on Linux systems.

This module uses system utilities like `glob` and `lsblk` to identify connected USB storage devices and determine their mount points. The primary function `get_usb_mount_paths_linux` is intended for use by other modules needing to access files on USB drives.

7.18 signing/services/key_getter/usb_finder_windows.py File Reference

Provides functions to find USB drive letters on Windows systems.

Namespaces

- [key_getter.usb_finder_windows](#)

Functions

- list[str] [key_getter.usb_finder_windows.get_usb_mount_paths_windows](#) ()
Retrieves the drive letters for all connected USB storage devices on a Windows system.
- list[str] [key_getter.usb_finder_windows.get_usb_devices_windows](#) ()
Identifies removable logical disks on a Windows system.

7.18.1 Detailed Description

Provides functions to find USB drive letters on Windows systems.

This module uses the Windows Management Instrumentation (WMI) library to identify removable disk drives (typically USB drives) and retrieve their assigned drive letters.

7.19 signing/services/pdf_signer/signer.py File Reference

Provides functions for signing PDF documents using RSA private keys.

Namespaces

- [pdf_signer.signer](#)

Functions

- def [pdf_signer.signer.sign](#) (rsa.RSAPrivateKey private_key, str pdf_in_path, str pdf_out_path)
Signs a PDF document using a provided RSA private key.
- Tuple[asn1_x509.Certificate, asn1_keys.PrivateKeyInfo] [pdf_signer.signer.generate_self_signed_cert](#) (rsa.RSAPrivateKey private_key)
Generates a self-signed X.509 certificate and private key information in ASN.1 format.

7.19.1 Detailed Description

Provides functions for signing PDF documents using RSA private keys.

This module leverages the `pyhanko` library to perform PAdES digital signatures. It includes functionality to generate a self-signed certificate on-the-fly for the signing process.

7.20 signing/services/pdf_signer/verifier.py File Reference

Provides functionality to verify digital signatures in PDF documents.

Classes

- class `pdf_signer.verifier.NoSignatureFound`
Exception raised when a PDF document does not contain any embedded digital signatures.

Namespaces

- `pdf_signer.verifier`

Functions

- bool `pdf_signer.verifier.verify` (`rsa.RSAPublicKey` public_key, str pdf_path)
Verifies the digital signature found in a PDF document against a provided public key.

7.20.1 Detailed Description

Provides functionality to verify digital signatures in PDF documents.

This module uses `pyhanko` and `cryptography` libraries to validate the integrity of a PDF signature and compare the embedded public key with a provided public key.

Index

- `__init__`
 - `frames.signing.SigningFrame`, 51
 - `frames.start.StartFrame`, 55
 - `frames.usb_key_get.KeyFromUSBFrame`, 46
 - `frames.verifying.VerifyingFrame`, 58
 - `main.App`, 44
- `_change_frame`
 - `main.App`, 44
- `_generate_self_signed_cert`
 - `pdf_signer.signer`, 39
- `_get_key_linux`
 - `key_getter.key_getter`, 34
- `_get_key_paths`
 - `key_getter.key_getter`, 34
- `_get_key_windows`
 - `key_getter.key_getter`, 34
- `_get_usb_devices_linux`
 - `key_getter.usb_finder_linux`, 36
- `_get_usb_devices_windows`
 - `key_getter.usb_finder_windows`, 37
- `_load_public_key`
 - `frames.verifying.VerifyingFrame`, 58
- `_process_pin_and_get_key`
 - `frames.usb_key_get.KeyFromUSBFrame`, 47
- `_process_verification`
 - `frames.verifying.VerifyingFrame`, 59
- `_select_file`
 - `frames.verifying.VerifyingFrame`, 59
- `_select_pdf_to_verify_file`
 - `frames.verifying.VerifyingFrame`, 59
- `_select_public_key_file`
 - `frames.verifying.VerifyingFrame`, 59
- `_select_source_pdf_file`
 - `frames.signing.SigningFrame`, 52
- `_select_target_pdf_path`
 - `frames.signing.SigningFrame`, 52
- `_setup_ui`
 - `frames.signing.SigningFrame`, 52
 - `frames.start.StartFrame`, 56
 - `frames.usb_key_get.KeyFromUSBFrame`, 47
 - `frames.verifying.VerifyingFrame`, 60
- `_sign_pdf_document`
 - `frames.signing.SigningFrame`, 52
- `_update_feedback`
 - `frames.signing.SigningFrame`, 53
 - `frames.usb_key_get.KeyFromUSBFrame`, 47
 - `frames.verifying.VerifyingFrame`, 60
- `action_button`
 - `frames.usb_key_get.KeyFromUSBFrame`, 48
- `ACTION_BUTTON_EXIT_TEXT`
 - `frames.usb_key_get`, 20
- `ACTION_BUTTON_INITIAL_TEXT`
 - `frames.usb_key_get`, 20
- `ACTION_BUTTON_RETRY_TEXT`
 - `frames.usb_key_get`, 20
- `ACTIVATE_BUTTON_COLOR`
 - `frames.generate_window`, 9
 - `frames.signing`, 12
 - `frames.start`, 16
 - `frames.usb_key_get`, 20
 - `frames.verifying`, 25
- `aes_decrypt_file`
 - `key_generate.AES_key_generator`, 30
 - `key_getter.AES_PIN_decryptor`, 32
- `aes_encrypt_file`
 - `key_generate.AES_key_generator`, 30
- `APP_HEIGHT`
 - `main`, 39
- `APP_TITLE`
 - `main`, 39
- `APP_WIDTH`
 - `main`, 39
- `BACKGROUND2_COLOR`
 - `frames.generate_window`, 10
 - `frames.signing`, 12
 - `frames.start`, 16
 - `frames.usb_key_get`, 20
 - `frames.verifying`, 25
- `BACKGROUND_COLOR`
 - `frames.generate_window`, 10
 - `frames.signing`, 12
 - `frames.start`, 17
 - `frames.usb_key_get`, 20
 - `frames.verifying`, 25
- `BLUE_BUTTON_COLOR`
 - `frames.generate_window`, 10
 - `frames.signing`, 12
 - `frames.start`, 17
 - `frames.usb_key_get`, 20
 - `frames.verifying`, 25
- `BUTTON_PADDING_Y`
 - `frames.start`, 17
 - `frames.usb_key_get`, 21
 - `frames.verifying`, 25
- `current_frame`
 - `main.App`, 45

- DEFAULT_PADDING_X
 - frames.start, 17
 - frames.usb_key_get, 21
 - frames.verifying, 25
- DEFAULT_PADDING_Y
 - frames.start, 17
 - frames.usb_key_get, 21
 - frames.verifying, 26
- DEFAULT_WRAP_LENGTH
 - frames.signing, 12
 - frames.start, 17
 - frames.usb_key_get, 21
 - frames.verifying, 26
- end_signing_callback
 - frames.signing.SigningFrame, 53
- end_verifying_callback
 - frames.verifying.VerifyingFrame, 60
- FOREGROUND_COLOR
 - frames.generate_window, 10
 - frames.signing, 12
 - frames.start, 17
 - frames.usb_key_get, 21
 - frames.verifying, 26
- frames, 9
 - frames.generate_window, 9
 - ACTIVATE_BUTTON_COLOR, 9
 - BACKGROUND2_COLOR, 10
 - BACKGROUND_COLOR, 10
 - BLUE_BUTTON_COLOR, 10
 - FOREGROUND_COLOR, 10
 - PRIVATE_KEY_NAME, 10
 - PUBLIC_KEY_NAME, 10
 - frames.generate_window.GenerateKeys, 46
 - frames.signing, 11
 - ACTIVATE_BUTTON_COLOR, 12
 - BACKGROUND2_COLOR, 12
 - BACKGROUND_COLOR, 12
 - BLUE_BUTTON_COLOR, 12
 - DEFAULT_WRAP_LENGTH, 12
 - FOREGROUND_COLOR, 12
 - GO_BACK_BUTTON_TEXT, 13
 - INITIAL_STATUS_TEXT, 13
 - LARGE_FONT_CONFIG, 13
 - PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT, 13
 - PATHS_REQUIRED_ERROR_TEXT, 13
 - PDF_FILE_TYPES, 13
 - PDF_READ_ERROR_TEXT, 14
 - RETRY_BUTTON_TEXT, 14
 - SELECT_SOURCE_BUTTON_TEXT, 14
 - SELECT_SOURCE_PDF_TITLE, 14
 - SELECT_TARGET_BUTTON_TEXT, 14
 - SELECT_TARGET_PDF_TITLE, 14
 - SIGN_BUTTON_INITIAL_TEXT, 15
 - SIGNING_ERROR_TEXT, 15
 - SIGNING_SUCCESS_TEXT, 15
 - SOURCE_PDF_LABEL_TEXT, 15
 - TARGET_PDF_LABEL_TEXT, 15
 - UNEXPECTED_SIGNING_ERROR_TEXT, 15
 - frames.signing.SigningFrame, 51
 - __init__, 51
 - _select_source_pdf_file, 52
 - _select_target_pdf_path, 52
 - _setup_ui, 52
 - _sign_pdf_document, 52
 - _update_feedback, 53
 - end_signing_callback, 53
 - private_key, 53
 - sign_button, 53
 - source_pdf_path_entry, 54
 - source_pdf_path_var, 54
 - status_label, 54
 - target_pdf_path_entry, 54
 - target_pdf_path_var, 54
 - frames.start, 16
 - ACTIVATE_BUTTON_COLOR, 16
 - BACKGROUND2_COLOR, 16
 - BACKGROUND_COLOR, 17
 - BLUE_BUTTON_COLOR, 17
 - BUTTON_PADDING_Y, 17
 - DEFAULT_PADDING_X, 17
 - DEFAULT_PADDING_Y, 17
 - DEFAULT_WRAP_LENGTH, 17
 - FOREGROUND_COLOR, 17
 - LABEL_TEXT, 18
 - LARGE_FONT_CONFIG, 18
 - SIGN_BUTTON_TEXT, 18
 - VERIFY_BUTTON_TEXT, 18
 - frames.start.StartFrame, 54
 - __init__, 55
 - _setup_ui, 56
 - on_signing_chosen_callback, 56
 - on_verifying_chosen_callback, 56
 - frames.usb_key_get, 18
 - ACTION_BUTTON_EXIT_TEXT, 20
 - ACTION_BUTTON_INITIAL_TEXT, 20
 - ACTION_BUTTON_RETRY_TEXT, 20
 - ACTIVATE_BUTTON_COLOR, 20
 - BACKGROUND2_COLOR, 20
 - BACKGROUND_COLOR, 20
 - BLUE_BUTTON_COLOR, 20
 - BUTTON_PADDING_Y, 21
 - DEFAULT_PADDING_X, 21
 - DEFAULT_PADDING_Y, 21
 - DEFAULT_WRAP_LENGTH, 21
 - FOREGROUND_COLOR, 21
 - INITIAL_INSTRUCTION_TEXT, 21
 - INPUT_AREA_PADDING_Y, 22
 - KEY_INVALID_MSG, 22
 - KEY_OR_PIN_INVALID_MSG, 22
 - LARGE_FONT_CONFIG, 22
 - MULTIPLE_KEYS_MSG, 22
 - NO_KEY_FILE_MSG, 22
 - NO_USB_DRIVES_MSG, 23
 - PIN_INVALID_FORMAT_MSG, 23

- PIN_LABEL_TEXT, 23
- PIN_REQUIRED_MSG, 23
- UNSUPPORTED_PLATFORM_MSG, 23
- frames.usb_key_get.KeyFromUSBFrame, 46
 - __init__, 46
 - _process_pin_and_get_key, 47
 - _setup_ui, 47
 - _update_feedback, 47
 - action_button, 48
 - on_key_retrieved_callback, 48
 - pin_entry, 48
 - status_label, 48
- frames.verifying, 24
 - ACTIVATE_BUTTON_COLOR, 25
 - BACKGROUND2_COLOR, 25
 - BACKGROUND_COLOR, 25
 - BLUE_BUTTON_COLOR, 25
 - BUTTON_PADDING_Y, 25
 - DEFAULT_PADDING_X, 25
 - DEFAULT_PADDING_Y, 26
 - DEFAULT_WRAP_LENGTH, 26
 - FOREGROUND_COLOR, 26
 - INITIAL_INSTRUCTION_TEXT, 26
 - LARGE_FONT_CONFIG, 26
 - NO_SIGNATURE_MSG, 26
 - PATHS_REQUIRED_MSG, 27
 - PDF_FILE_TYPES, 27
 - PDF_INVALID_MSG, 27
 - PUBLIC_KEY_FILE_TYPES, 27
 - PUBLIC_KEY_INVALID_MSG, 27
 - PUBLIC_KEY_NOT_FOUND_MSG, 27
 - SECTION_SPACING_Y, 28
 - SELECT_PDF_TO_VERIFY_TITLE, 28
 - SELECT_PUBLIC_KEY_TITLE, 28
 - SIGNATURE_INVALID_MSG, 28
 - SIGNATURE_VALID_MSG, 28
 - VERIFICATION_ERROR_MSG, 28
 - VERIFY_BUTTON_GO_BACK_TEXT, 29
 - VERIFY_BUTTON_INITIAL_TEXT, 29
 - VERIFY_BUTTON_RETRY_TEXT, 29
- frames.verifying.VerifyingFrame, 57
 - __init__, 58
 - _load_public_key, 58
 - _process_verification, 59
 - _select_file, 59
 - _select_pdf_to_verify_file, 59
 - _select_public_key_file, 59
 - _setup_ui, 60
 - _update_feedback, 60
 - end_verifying_callback, 60
 - pdf_to_verify_entry, 60
 - pdf_to_verify_path_var, 61
 - public_key_entry, 61
 - public_key_path_var, 61
 - status_label, 61
 - verify_button, 61
- generate_keys
 - key_generate.RSA_key_generator, 31
- generating/frames/__init__.py, 65
- generating/frames/generate_window.py, 63
- generating/key_generate/__init__.py, 65
- generating/key_generate/AES_key_generator.py, 64
- generating/key_generate/RSA_key_generator.py, 64
- generating/main.py, 71
- get_key
 - key_getter.key_getter, 35
- get_key_from_usb_result
 - main.App, 44
- get_usb_mount_paths_linux
 - key_getter.usb_finder_linux, 37
- get_usb_mount_paths_windows
 - key_getter.usb_finder_windows, 38
- GO_BACK_BUTTON_TEXT
 - frames.signing, 13
- hash_pin
 - key_generate.AES_key_generator, 30
 - key_getter.AES_PIN_decryptor, 32
- INITIAL_INSTRUCTION_TEXT
 - frames.usb_key_get, 21
 - frames.verifying, 26
- INITIAL_STATUS_TEXT
 - frames.signing, 13
- INPUT_AREA_PADDING_Y
 - frames.usb_key_get, 22
- KEY_FILE_NAME
 - key_getter.key_getter, 36
- key_generate, 29
- key_generate.AES_key_generator, 29
 - aes_decrypt_file, 30
 - aes_encrypt_file, 30
 - hash_pin, 30
- key_generate.RSA_key_generator, 31
 - generate_keys, 31
- key_getter, 32
- key_getter.AES_PIN_decryptor, 32
 - aes_decrypt_file, 32
 - hash_pin, 32
- key_getter.key_getter, 33
 - _get_key_linux, 34
 - _get_key_paths, 34
 - _get_key_windows, 34
 - get_key, 35
 - KEY_FILE_NAME, 36
 - LINUX_PLATFORM_NAME, 36
 - WINDOWS_PLATFORM_NAME, 36
- key_getter.key_getter.KeyInvalidException, 48
- key_getter.key_getter.KeyOrPinInvalidException, 49
- key_getter.key_getter.MultipleKeysFoundException, 49
- key_getter.key_getter.NoKeyFoundException, 49
- key_getter.key_getter.NoUSBDrivesFoundException, 50
- key_getter.key_getter.UnsupportedPlatformException, 56
- key_getter.usb_finder_linux, 36
 - _get_usb_devices_linux, 36

- get_usb_mount_paths_linux, 37
- key_getter.usb_finder_windows, 37
 - _get_usb_devices_windows, 37
 - get_usb_mount_paths_windows, 38
- KEY_INVALID_MSG
 - frames.usb_key_get, 22
- KEY_OR_PIN_INVALID_MSG
 - frames.usb_key_get, 22
- LABEL_TEXT
 - frames.start, 18
- LARGE_FONT_CONFIG
 - frames.signing, 13
 - frames.start, 18
 - frames.usb_key_get, 22
 - frames.verifying, 26
- LINUX_PLATFORM_NAME
 - key_getter.key_getter, 36
- main, 38
 - APP_HEIGHT, 39
 - APP_TITLE, 39
 - APP_WIDTH, 39
- main.App, 43
 - __init__, 44
 - _change_frame, 44
 - current_frame, 45
 - get_key_from_usb_result, 44
 - main_menu, 45
 - start_signing, 45
 - start_verifying, 45
- main_menu
 - main.App, 45
- MULTIPLE_KEYS_MSG
 - frames.usb_key_get, 22
- NO_KEY_FILE_MSG
 - frames.usb_key_get, 22
- NO_SIGNATURE_MSG
 - frames.verifying, 26
- NO_USB_DRIVES_MSG
 - frames.usb_key_get, 23
- on_key_retrieved_callback
 - frames.usb_key_get.KeyFromUSBFrame, 48
- on_signing_chosen_callback
 - frames.start.StartFrame, 56
- on_verifying_chosen_callback
 - frames.start.StartFrame, 56
- PATHS_CANNOT_BE_THE_SAME_ERROR_TEXT
 - frames.signing, 13
- PATHS_REQUIRED_ERROR_TEXT
 - frames.signing, 13
- PATHS_REQUIRED_MSG
 - frames.verifying, 27
- PDF_FILE_TYPES
 - frames.signing, 13
 - frames.verifying, 27
- PDF_INVALID_MSG
 - frames.verifying, 27
- PDF_READ_ERROR_TEXT
 - frames.signing, 14
- pdf_signer, 39
 - pdf_signer.signer, 39
 - _generate_self_signed_cert, 39
 - sign, 40
 - pdf_signer.verifier, 40
 - verify, 41
 - pdf_signer.verifier.NoSignatureFound, 50
- pdf_to_verify_entry
 - frames.verifying.VerifyingFrame, 60
- pdf_to_verify_path_var
 - frames.verifying.VerifyingFrame, 61
- pin_entry
 - frames.usb_key_get.KeyFromUSBFrame, 48
- PIN_INVALID_FORMAT_MSG
 - frames.usb_key_get, 23
- PIN_LABEL_TEXT
 - frames.usb_key_get, 23
- PIN_REQUIRED_MSG
 - frames.usb_key_get, 23
- private_key
 - frames.signing.SigningFrame, 53
- PRIVATE_KEY_NAME
 - frames.generate_window, 10
- public_key_entry
 - frames.verifying.VerifyingFrame, 61
- PUBLIC_KEY_FILE_TYPES
 - frames.verifying, 27
- PUBLIC_KEY_INVALID_MSG
 - frames.verifying, 27
- PUBLIC_KEY_NAME
 - frames.generate_window, 10
- PUBLIC_KEY_NOT_FOUND_MSG
 - frames.verifying, 27
- public_key_path_var
 - frames.verifying.VerifyingFrame, 61
- RETRY_BUTTON_TEXT
 - frames.signing, 14
- SECTION_SPACING_Y
 - frames.verifying, 28
- SELECT_PDF_TO_VERIFY_TITLE
 - frames.verifying, 28
- SELECT_PUBLIC_KEY_TITLE
 - frames.verifying, 28
- SELECT_SOURCE_BUTTON_TEXT
 - frames.signing, 14
- SELECT_SOURCE_PDF_TITLE
 - frames.signing, 14
- SELECT_TARGET_BUTTON_TEXT
 - frames.signing, 14
- SELECT_TARGET_PDF_TITLE
 - frames.signing, 14
- sign
 - pdf_signer.signer, 40

- sign_button
 - frames.signing.SigningFrame, [53](#)
- SIGN_BUTTON_INITIAL_TEXT
 - frames.signing, [15](#)
- SIGN_BUTTON_TEXT
 - frames.start, [18](#)
- SIGNATURE_INVALID_MSG
 - frames.verifying, [28](#)
- SIGNATURE_VALID_MSG
 - frames.verifying, [28](#)
- signing/frames/__init__.py, [64](#)
- signing/frames/signing.py, [65](#)
- signing/frames/start.py, [66](#)
- signing/frames/usb_key_get.py, [67](#)
- signing/frames/verifying.py, [69](#)
- signing/main.py, [70](#)
- signing/services/key_getter/__init__.py, [64](#)
- signing/services/key_getter/AES_PIN_decryptor.py, [71](#)
- signing/services/key_getter/key_getter.py, [72](#)
- signing/services/key_getter/usb_finder_linux.py, [73](#)
- signing/services/key_getter/usb_finder_windows.py, [74](#)
- signing/services/pdf_signer/__init__.py, [64](#)
- signing/services/pdf_signer/signer.py, [74](#)
- signing/services/pdf_signer/verifier.py, [75](#)
- SIGNING_ERROR_TEXT
 - frames.signing, [15](#)
- SIGNING_SUCCESS_TEXT
 - frames.signing, [15](#)
- SOURCE_PDF_LABEL_TEXT
 - frames.signing, [15](#)
- source_pdf_path_entry
 - frames.signing.SigningFrame, [54](#)
- source_pdf_path_var
 - frames.signing.SigningFrame, [54](#)
- start_signing
 - main.App, [45](#)
- start_verifying
 - main.App, [45](#)
- status_label
 - frames.signing.SigningFrame, [54](#)
 - frames.usb_key_get.KeyFromUSBFrame, [48](#)
 - frames.verifying.VerifyingFrame, [61](#)
- TARGET_PDF_LABEL_TEXT
 - frames.signing, [15](#)
- target_pdf_path_entry
 - frames.signing.SigningFrame, [54](#)
- target_pdf_path_var
 - frames.signing.SigningFrame, [54](#)
- UNEXPECTED_SIGNING_ERROR_TEXT
 - frames.signing, [15](#)
- UNSUPPORTED_PLATFORM_MSG
 - frames.usb_key_get, [23](#)
- VERIFICATION_ERROR_MSG
 - frames.verifying, [28](#)
- verify
 - pdf_signer.verifier, [41](#)
- verify_button
 - frames.verifying.VerifyingFrame, [61](#)
- VERIFY_BUTTON_GO_BACK_TEXT
 - frames.verifying, [29](#)
- VERIFY_BUTTON_INITIAL_TEXT
 - frames.verifying, [29](#)
- VERIFY_BUTTON_RETRY_TEXT
 - frames.verifying, [29](#)
- VERIFY_BUTTON_TEXT
 - frames.start, [18](#)
- WINDOWS_PLATFORM_NAME
 - key_getter.key_getter, [36](#)