Table of Contents

Chapter 1: **Appendix A Software Development of Proof of Concept program**

1.1 Appendix A.2 Requirements and Analysis

The overall goal of the system is to provide primarily, a basic implementation of the PKCS signature scheme from which a user can interact with via a user interface to perform relevant actions. The program will also include the other considered schemes, to be integrated once the implementation for the PKCS scheme has been established. The core actions comprise, the generation of keys, creation of signatures and finally verification of previously created signatures. The program will form the foundation for the eventual delivery of the benchmarking program used to examine the discussed provable security overhead.

1.1.1 Description of Actors

Table 1.1: Description of Actors for the POC Digital Signature Program

Actor / Role Name	**Role Description and Objective**
User/Signer	Individual who wishes to digitally sign a piece of content. The signer generates key pairs, can input content, and create a digital signature using their private key. Their main goal is to ensure that the content they're signing is authenticated and its integrity is maintained, proving that it hasn't been tampered with.
Verifier	Entity that needs to validate the authenticity and integrity of a digitally signed piece of content. The verifier inputs signed content, a corresponding public key, and attempts to verify a specified digital signature. Their primary objective is to ascertain that the content hasn't been altered post-signing and to confirm the identity of the signer.

1.1.2 User Stories

Essential Requirements

1. Potential signer should be able to generate and retrieve a public-private key pair having provided a key size.
    - User should be presented with a text box to input the key size.
    - The system should handle any exceptions or errors during key generation, displaying to the user of any issues.
    - The system should notify the signer once the key generation process is successful.
    - Once the key is generated the user should have the option to save it to a file.
2. Having provided a message and private key, the signer should be able to retrieve the resulting computed digital signature.

- The signer should be presented with a text box to input the message intended for signing.
- The signer should be able to specify and input the private key using file selection via a browse option.
- The system should handle any exceptions or errors during signature generation, displaying to the signer of any issues.
- The system should notify the signer once the signing process is successful.
- Once the signature is generated the signer should have the option to copy the signature to the clipboard or save it to a file.

3. Having provided a message, its corresponding digital signature, and a public key, the verifier should be able to verify the authenticity of the signature.
   - The verifier should be presented with a text box or file browse option to input the message corresponding to the signature intended for verifying.
   - The verifier should be presented with a file browse option to input the signature intended for verifying.
   - The system should handle any exceptions or errors during verification process, displaying to the verifier of any issues.
   - The system should notify the verifier once the verification process is successful.
4. The signer should be able to sign messages using the PKCS#1 v1.5 Signature Scheme.
5. The verifier should be able to verify messages using the PKCS#1 v1.5 Signature Scheme.
6. The signer should be able to sign messages using the ANSI X9.31 rDSA Signature Scheme.
7. The verifier should be able to verify messages using the ANSI X9.31 rDSA Signature Scheme.
8. The signer should be able to sign messages with partial or full recovery using the ISO/IEC 9796-2:2010 Signature Scheme 1.
9. The verifier should be able to verify messages with partial or full recovery using the ISO/IEC 9796-2:2010 Signature Scheme 1.

Non Essential Requirements

Performance

7. User shoulder be able view a measurement of time taken for each of the signature related operations i.e., Key generation, Signature creation and verification.

Non Functional

8. The program should generate keys within a reasonable timeframe.
9. The program should create signatures within a reasonable timeframe.
10. The program should verify signatures within a reasonable timeframe.

1.1.3 UML Use Case

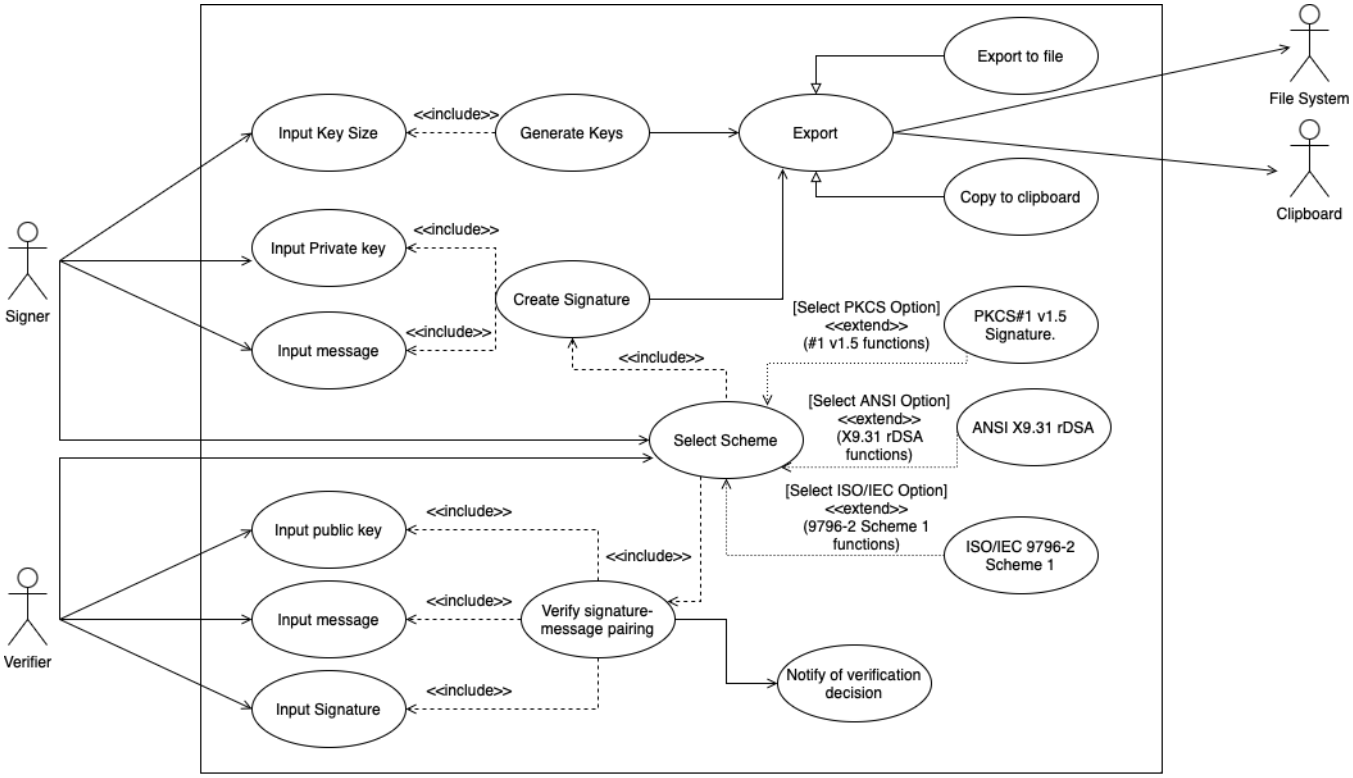

Figure 1.1: UML Use Case Diagram

Generate Keys Use Case

Flow of Events:

1. User selects "Generate Key" from the main menu options panel.
2. User is presented with an input box labeled "Input Key Size".
3. User inputs desired key size into the box.
4. System processes the request and generates the public-private key pair.
5. System displays a notification informing the user that the key generation process was successful.
6. User is presented with options "Export to file" and "Copy to clipboard" for the signature.
7. User selects desired option to either save the keys to a file or copy them to clipboard.

Alternative flows:

3a. User inputs an invalid key size.

3a1. System warns user about the invalid input and prompts them to enter a valid key size again.

5a. System encounters an error during key generation.

5a1. System displays an error message and prompts the user to try again.

Create Signature Use Case

Flow of Events:

1. User selects "Sign message" from the main menu options panel.
2. User is presented with text boxes labeled "Input Private Key" and "Input Message".
3. User inputs their private key and the message they wish to sign.
4. User selects the desired signature scheme from options like "PKCS#1 v1.5 Signature", "ANSI X9.31 rDSA", etc.
5. System processes the input and computes the digital signature.
6. System displays a notification informing the signer that the signing process was successful.
7. User is presented with options "Export to file" and "Copy to clipboard".
8. User selects desired option to either save the keys to a file or copy them to clipboard.

Alternative flows:

3a. User inputs an invalid or mismatched private key.

3a1. System warns user about the invalid input and prompts them to enter a valid key.

5a. System encounters an error during signature creation.

5a1. System displays an error message and prompts the user to try again.

7a. User selected an ISO/IEC 9796-2 scheme in step 4.

7a1. User is presented with options "Export to file" and "Copy to clipboard" for the computed signature and additionally if applicable. a computed non recoverable portion of their initially submitted message.

Verify Signature Use Case

Flow of Events:

1. User selects "Verify Signature" from the main menu options panel.
2. User selects the desired signature scheme from options like "PKCS#1 v1.5 Signature", "ANSI X9.31 rDSA", etc..
3. User is presented with options to input the message, its corresponding signature, and the public key
4. User provides all required inputs.
5. System processes the information and verifies the authenticity of the signature.

6. System displays a notification with the result, either confirming the authenticity or notifying of a mismatch.

Alternative flows:

3a. User selected the ISO/IEC 9796-2 scheme 1 with full message recovery option scheme in step 2.

3a1. System greys out box requiring message input so that user cannot input a message.

3b. User selected the ISO/IEC 9796-2 scheme 1 with partial message recovery option scheme in step 2.

3b1. System changes the displayed label for message input to non-recoverable message portion.

4a. User inputs mismatched or incorrect information.

4a1. System warns user about the incorrect input and suggests rechecking the inputs.

5a. System encounters an error during verification.

5a1. System displays an error message and prompts the user to try again.

6a. User selected an ISO/IEC 9796-2 scheme in step 2.

6a1. User is presented with options "Export to file" and "Copy to clipboard" for the computed signature and additionally if applicable a recovered portion of a message submitted some time in the past to the signature generation process.

1.1.4 Acceptance Tests

1. Key Pair Generation:

1. Open the application and locate the key generation section.
2. Input a valid key size into the provided text box and submit.
3. If key size is invalid no key is issued and the user is informed to try again.
4. Observe that no exceptions or errors are displayed during the key generation process.
5. If there are errors during key generation the user is informed to try again.
6. Confirm that a notification is presented to the user upon successful key generation.
7. Check if there is an option to save the generated key pair to a file and perform a successful save.

2. Digital Signature Generation:

1. Locate the signature generation section in the application.
2. Input a test message into the provided text box.
3. Use the browse option to provide a valid private key file.

4. If empty message or invalid file is provided, the user is informed to try again.
5. Ensure no errors or exceptions are displayed during the signing process.
6. Confirm that a notification is presented to the signer upon successful signature generation.
7. Check for options to either copy the signature to clipboard or save it to a file and verify both functionalities.

3. Digital Signature Verification:

1. Locate the signature verification section in the application.
2. Use the text box or file browse option to input the original test message.
3. Use the browse option to provide the generated signature file.
4. If empty message or invalid file is provided, the user is informed to try again.
5. Ensure no errors or exceptions are displayed during the verification process.
6. Confirm that a notification is presented to the verifier upon successful verification.

4. Signature and Verification with PKCS#1 v1.5:

1. Set the application to use the PKCS#1 v1.5 Signature Scheme.
2. Sign a test message and verify its signature using the previous steps. Ensure both processes succeed.

5. Signature and Verification with ANSI X9.31 rDSA:

1. Set the application to use the ANSI X9.31 rDSA Signature Scheme.
2. Sign a test message and verify its signature using the previous steps. Ensure both processes succeed.

6. Signature Generation with ISO/IEC 9796-2:2010 Scheme 1:

1. Set the application to use the ISO/IEC 9796-2:2010 Signature Scheme 1.
2. Locate the signature generation section in the application.
3. Input a test message into the provided text box.
4. Use the browse option to provide a valid private key file.
5. If empty message or invalid file is provided, the user is informed to try again.
6. Ensure no errors or exceptions are displayed during the signing process.
7. Confirm that a notification is presented to the signer upon successful signature generation.
8. Check and verify separate options (copying to clipboard and saving to a file) for the signature.

9. Check and verify separate options (copying to clipboard and saving to a file) for non message portion if if user submitted a sufficiently short message relative to modulus (if message is too short there is no non recoverable portion).

6. Signature Verification with ISO/IEC 9796-2:2010 Scheme 1

1. Set the application to use a variant of ISO/IEC 9796-2:2010 Signature Scheme 1.
2. Locate the signature verification section in the application.
3. Use the browse option to provide the generated signature file.
4. Use the text box or file browse option to input the original test message if user did received a non recoverable message from previous signature generation process
5. Ensure no errors or exceptions are displayed during the verification process.
6. Confirm that a notification is presented to the verifier upon successful verification.
7. Check and verify separate options (copying to clipboard and saving to a file) for recoverable message portion if user submitted a legitimate non recoverable message portion to the verification process in step 4.

1.2 Appendix A.3 Design

1.2.1 Program packages

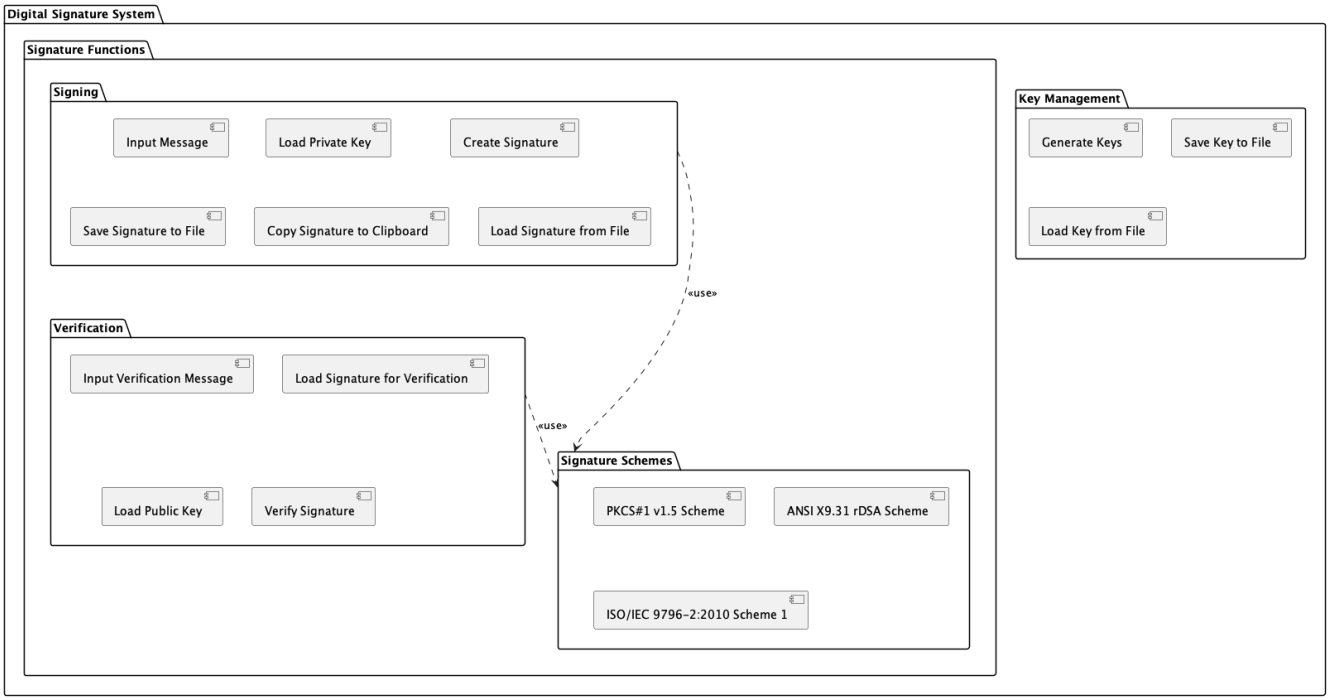

Figure 1.2: POC program Packages

Figure **??** depicts the core functionality of the POC program and is in direct alignment with previously elaborated on (see requirements) user activities of signing, creating keys, and verifying, using a specified scheme like PKCS#1-v1.5.

1.2.2 UML sequence diagram

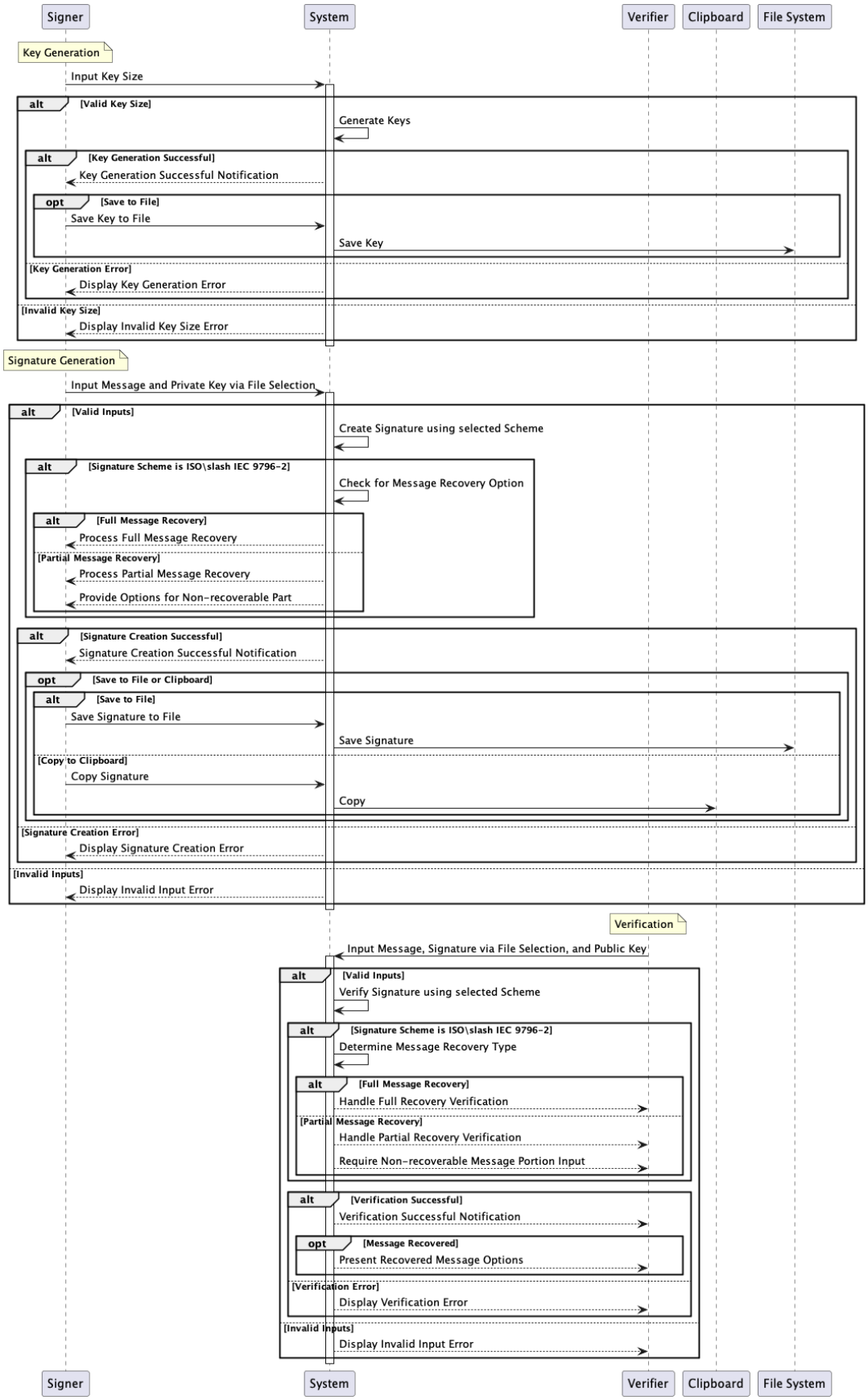

Figure 1.3: UML Sequence Diagram

The above diagram is mostly a high level view of the core behaviour that can be expected to performed by a user of the proof of concept of program. The point at which the diagram departs this is the specialised functionality to related to the message recovery signature schemes of the ISO standard. These schemes require special consideration because the related behaviour differs from the standard digital signature process. The ISO/IEC 9796-2 schemes incorporate message recovery features, where part or all of the original message can be reconstructed from the signature itself. This necessitates additional logic in both the signing and verifying processes. For partial message recovery, the signer needs to manage the non-recoverable portion of the message, ensuring that it is correctly returned alongside the signature. Subsequently they may then input the non-recoverable portion as part of their interaction with the verification process in attempt to recover the remaining portion of message. For full message recovery, the entire message is embedded in the signature, eliminating the need for a separate message input during verification but requiring careful handling to extract and validate the message from the signature. These nuances demand specialised user interfaces and system checks, making the ISO schemes distinct in their interaction and processing requirements within the application.

Initially, the user is prompted to generate cryptographic keys, providing a key size that, if valid, leads to the creation of a private and public key pair. The user can then opt to save these keys onto their file system. Once keys are in place, the user can sign a message. They input the message into the system and load their private key. If the inputs are correct, the system employs a chosen signature algorithm to create a digital signature, which the user can save or copy to their clipboard. In case of invalid input or an error during signature creation, the user is informed with an error message.

For verification, the user inputs a message, loads the digital signature and the corresponding public key. The system checks the signature against the message using the public key. If the signature is valid, a success notification is displayed; otherwise, the user is alerted to a verification error. Throughout this process, the system guides the user with notifications or error messages based on the success or failure of the operations performed.

1.3 System Testing

Table 1.2: Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
MainMenu-001	Application is launched and the user is presented with the main menu.	Click on the "[K] Generate Keys" button.	N/A	The application should navigate to the key generation page without errors.	
MainMenu-002	Application is launched and the user is presented with the main menu.	Click on the "[S] Sign Document" button.	N/A	The application should navigate to the signature creation page without errors.	
MainMenu-003	Application is launched and the user is presented with the main menu.	Click on the "[V] Verify Signature" button.	N/A	The application should navigate to the signature verification page without errors.	

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
KeyGen-001	Application is installed and operational; the user is on the Key Generation page.	Navigate to the "Generate Keys" section. Enter a valid bit size in the input field. Click the "Generate Keys" button.	2048	The system should generate a key pair using the specified bit sizes without errors.	
KeyGen-002	Application is installed and operational; the user is on the Key Generation page.	Navigate to the "Generate Keys" section. Enter a string of special characters in the input field. The system should not accept the input and display an error message indicating that only numerical bit sizes are valid.			

KeyGen-003	Application is installed and operational; the user is on the Key Generation page.	Navigate to the "Generate Keys" section. Enter an excessively long string of numbers in the input field. Click the "Generate Keys" button.	A string of numbers exceeding normal bit size lengths (e.g., 1000 digits).	The system should reject the input and display an error message indicating that the bit size is too long and not valid.	
KeyGen-004	Application is installed and operational; the user is on the Key Generation page.	Navigate to the "Generate Keys" section. Enter alphanumeric characters in the input field. Click the "Generate Keys" button.	abc123	The system should not accept the input and should display an error message that only numeric values are valid.	
KeyGen-005	Application is installed and operational; the user is on the Key Generation page.	Navigate to the "Generate Keys" section. Enter SQL injection code in the input field. Click the "Generate Keys" button.	' OR '1'='1	The system should sanitize the input, not execute the code, and display an error message about invalid input.	

KeyGen-201	Application is installed and operational; the user has successfully generated keys using the "Generate Keys" feature.	After key generation, click on the "Export Private Key" button. Wait for the application to perform the export operation automatically. Check the application's default save location or the location indicated by the application for the presence of the new signature file. Open the signature file with a text editor to verify that it contains the correct signature data.	N/A (The action uses the application's UI)	The signature file is automatically saved to the default location specified by the application. The file should contain the correct signature data, formatted as expected for a digital signature.	

Table 1.4: Signature Creation Test Cases

Test ID	**Prerequisites**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**
Sign-001	User is on the "Sign" page of the application.	Click the "Import Text..." button. Select a valid text file to import for signing. Verify that the text box is replaced with the name of the imported file and a green checkmark is displayed. Click the "Import Private Key" button and select a valid private key. Choose a signature scheme from the dropdown menu if available. Click "Create Signature".	Valid text file for import, valid private key file.	The application should display the name of the imported text file with a green checkmark, import the private key successfully, and upon clicking "Create Signature", generate a digital signature using the imported text and private key.	
Sign-002	User is on the "Sign" page of the application without any key or text pre-loaded.	Manually enter text into the "ENTER TEXT TO SIGN:" field. Click "Create Signature" without importing a private key.	"Example text to sign"	The application should prompt the user to import a private key before allowing the signature creation to proceed.	

Continued on the next page

Table ?? (continued): Signature Creation Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-003	User is on the "Sign" page of the application. A valid private key is already imported.	Click the "Import Text..." button. Select an invalid file format or a corrupted text file. Attempt to create a signature.	Invalid or corrupted text file.	The application should not replace the text box with the file name, should not show a green checkmark, and should display an error message indicating the file is not valid for import.	
Sign-004	User is on the "Sign" page of the application. A valid private key is already imported.	Import a valid text file. After the text file name and green checkmark are displayed, remove the private key by any means provided by the application (if possible). Attempt to create a signature.	Valid text file, then remove the private key.	The application should prevent signature creation and prompt the user to import a private key.	

Continued on the next page

Table ?? (continued): Signature Creation Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-005	User is on the "Sign" page of the application.	Import a text file with a very long content. Import a valid private key. Select a signature scheme if available. Click "Create Signature".	A text file with content that exceeds typical limits (if any are defined).	The application should either successfully create a signature for the long text file or display an error message if the content is too long for the selected signature scheme or exceeds the application's handling capacity.	

Table 1.5: Signature Export Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-Export-002	User has successfully created a digital signature on the "Sign" page.	After signature creation, if there is an option to copy the signature to the clipboard, click the "Copy to Clipboard" button or equivalent UI element. Open a text editor and paste the content from the clipboard.	N/A (The action uses the application's UI)	The digital signature that was copied to the clipboard should be pasted into the text editor, and it should match the signature displayed or generated in the application.	

Continued on the next page

Table ?? (continued): Signature Export Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-Export-003	User has attempted to create a digital signature but the process failed due to an invalid key or other errors.	Attempt to click on the "Export Signature" or "Copy to Clipboard" button after a failed signature creation attempt.	N/A (The action uses the application's UI)	The application should either disable the export/copy functionality or display an error message indicating that there is no signature to export or copy because the signature creation process was unsuccessful.	
Sign-Export-004	User has successfully created a digital signature on the "Sign" page.	After signature creation, click on the "Export Signature" button. Choose an invalid file system location or enter an unsupported file name to save the signature. Attempt to confirm the export operation.	Invalid path or file name.	The application should prevent the export operation and display an error message indicating that the file could not be saved, explaining the reason (e.g., invalid path, permission issues, unsupported characters in the file name).	

Continued on the next page

Table ?? (continued): Signature Export Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-Export-005	User has successfully created a digital signature on the "Sign" page.	After signature creation, check for any UI indication that the signature is ready to be exported (such as a confirmation message or an enabled "Export" button). If a confirmation message or similar indicator is part of the design, confirm its presence. Proceed with the export or copy operation as designed.	N/A (The action uses the application's UI)	Any UI indicators or messages that should appear post-signature creation to guide the user to export or copy the signature should be present and correct according to the application design.	

Continued on the next page

Table ?? (continued): Signature Export Test Cases

Test ID	Prerequisites	Test Steps	Test Data	Expected Result	Actual Result
Sign-Export-006	User has successfully created a digital signature on the "Sign" page.	After signature creation, click on the "Export Signature" button. Wait for the application to perform the export operation automatically. Check the application's default save location or the location indicated by the application for the presence of the new signature file. Open the signature file with a text editor to verify that it contains the correct signature data.	N/A (The action uses the application's UI)	The signature file is automatically saved to the default location specified by the application. The file should contain the correct signature data, formatted as expected for a digital signature.	

Table 1.6: Verification Test Cases

Test ID	**Prerequisites**	**Test Steps**	**Expected Result**	**Actual Result**
Verify-002	User is on the "Verify" page of the application with no files pre-loaded.	1. Manually input text into "ENTER TEXT TO VERIFY:". 2. Click "Import Public Key". 3. Select a valid public key file. 4. Manually input a signature into "ENTER SIGNATURE:". 5. Select the appropriate signature scheme, if applicable. 6. Click "Verify Signature".	The application should accept manual input and the imported public key, perform verification on "Verify Signature" click, and display the result.	
Verify-003	User is on the "Verify" page of the application.	1. Attempt to import an invalid or corrupted text file by clicking "Import Text...". 2. Attempt to verify the signature.	The application should not replace "ENTER TEXT TO VERIFY:" with the file name, should not show a green checkmark, and should display an error message indicating the file is not valid for import.	

Continued on the next page

Table ?? (continued): Verification Test Cases

Test ID	Prerequisites	Test Steps	Expected Result	Actual Result
Verify-004	User is on the "Verify" page with a valid text and signature imported.	1. Click "Import Public Key" and select an invalid or corrupted public key file. 2. Attempt to verify the signature.	The application should not replace "PUBLIC KEY:" with the file name, should not show a green checkmark, and should display an error message indicating the public key file is not valid for import.	
Verify-005	User is on the "Verify" page with valid text and public key imported.	1. Click "Import Signature..." and select an invalid or corrupted signature file. 2. Attempt to verify the signature.	The application should not replace "ENTER SIGNATURE:" with the file name, should not show a green checkmark, and should display an error message indicating the signature file is not valid for import.	