PROJECT 5A
TESTING OF RSA 2048 ENCRYPTION/DECRYPTION IN MICRO-PYTHON

USER MANUAL AND GUIDE

Developed with MicroPython.

Usage

- Create One atSign and download its .atKeys file and store it locally. (Click on atSign to create one)
- Install the latest firmware.uf2 onto your Pico W from atsign-foundation/micropython Releases, as this is patched to enable AES CTR, which is used by atSigns.
- Connect the Pico-W to your system and put the downloaded firmware.uf2 file into the Pico-W. (Once you put this file into pico-w it will exit automatically)
- Clone the code from GitHub. (GitHub)
- Download Thonny IDE and place all the files of this repository in the Pico-w file system.
- Fill all the fields of the settings.json file (ssid/password of your Wi-Fi network and atSign).
- Place your .atKeys file in the ~/keys/ directory (if the folder doesn't exist, create it manually)
- Make sure pico-w is connected to internet
- Run main.py and select option 3 in the REPL ("Get privateKey for @[yourAtSign]")
- Re-launch the REPL (run main.py again)
- Now you can select option 2 in the REPL to automatically get authenticated in your DESS (If you get an error when attempting to find the secondary or when trying to connect to it, run again the REPL)
- To run the test cases for the main there are 2 ways.
  First way:
  - Re-launch the REPL (run main.py again)
  - Now you can select option 5 in the REPL to automatically run all the test cases related to main.

  Second way:
  - Open the Micropython_Test.py and click on the run button on the thonny ide.
- To run the test cases for the pkcs1.py
  - Open the testcase.py file in thonny (it is present under lib/third_party/rsa/testcase.py)
  - Click on the run button on the thonny ide.
- Enjoy!.

Test cases Explanation.

- Micropython_Test.py
  - test_read_settings(): This test case checks whether the read_settings() function in the main module returns a tuple with a length of 4.
  - test_read_key(): This test case checks whether the read_key() function in the main module returns a tuple with a length of 5.
  - test_find_secondary(): This test case checks whether the find_secondary() function in the main module returns a string.
  - test_connect_to_secondary(): This test case checks whether the connect_to_secondary() function in the main module returns a boolean value True if the connection to the secondary is successful.
  - test_send_verbs(): This test case checks whether the send_verbs() function in the main module returns a string for response and command.
- testcase.py
  - test_encrypt_decrypt(): tests the encryption and decryption functionality of a public-key encryption algorithm. It generates a pair of public and private keys, encrypts a message using the public key,

then decrypts the message using the private key, and finally checks that the decrypted message is the same as the original plaintext message.
- test_sign_verify(): tests the signature and verification functionality of a digital signature algorithm. It generates a pair of public and private keys, signs a message using the private key and a specified hash function, then verifies the signature using the public key.
- test_sign_hash(): tests the signature and hash functionality of a digital signature algorithm. It generates a pair of public and private keys, hashes a message using a specified hash function, signs the hash using the private key, then verifies the signature using the public key.
- test_decrypt_fail(): tests the exception handling of the decryption function. It generates a pair of public and private keys, encrypts a message using the public key, modifies the cipher text to cause the decryption to fail, then checks that an exception is raised when attempting to decrypt the modified cipher text using the private key.

Library Explanation

- unittest.py (optimized library to deal with memory issues of Pico-W)
  - It is a library that contains several classes for testing purposes.
  - SkipTest: This is an exception class that can be raised to indicate that a test should be skipped.
  - AssertRaisesContext: This is a context manager class that can be used to test whether a particular exception is raised by a function.
  - TestCase: This is a base class that provides various methods for testing, such as assertEqual, assertTrue, and assertRaises. Test cases are created by subclassing this class and defining methods that start with the prefix test_.
  - skip: This is a decorator that can be used to skip a test if a condition is true.
  - skipIf: This is a decorator that can be used to skip a test if a condition is true.
  - skipUnless: This is a decorator that can be used to skip a test if a condition is false.
  - TestSuite: This is a container class for test cases.
  - TestRunner: This is a class that runs a suite of test cases and reports the results.
  - TestResult: This is a class that collects the results of running a suite of test cases. It keeps track of the number of tests run, the number of failures, the number of errors, and the number of tests that were skipped.

- base64.py
  - Micro-python library for b64decode(s, altchars=None, validate=False)
  - This library decodes a Base64 encoded string or bytes object into its original binary form.