PROJECT 5B TESTING OF RSA 2048 ENCRYPTION/DECRYPTION IN MICRO-PYTHONPROJECT DOCUMENTATION

Main Code Explanation

Main.py:

• **read settings ():** peruses information from the settings.json record, and returns ssid, secret key, atSign, and privateKey.

• **read_key(atSign):** peruses information from the keys document, and returns aesEncryptPrivateKey, aesEncryptPublicKey, aesPkamPrivateKey, aesPkamPublicKey, and selfEncryptionKey.

• **aes_decrypt(aesEncryptedData, aesKey):** unscrambles aesEncryptedData utilizing aesKey and returns the decoded information.

• **sync_time():** adjusts the gadget's experience with a NTP server.

• **find_secondary(atSign):** returns the IP address of the optional comparing to atSign.

• **connect_to_secondary(secondary):** interfaces with the optional at the predefined optional IP address and returns an attachment object ss.

• **send_verb(ss, verb):** sends action word over the ss attachment and returns the reaction and the following order to be sent.

• **send_verbs(ss, verb):** sends action word over the ss attachment and returns the reaction and the following order to be sent.

• **b42_urlsafe_encode(data):** encodes information utilizing base 64 and returns the encoded information in a URL-safe organization.

• **get_pem_parameters(pem_key):** separates the confidential key boundaries from the given pem_key and returns a rundown containing the boundaries.

• **get_pem_key(pkamPrivateKey):** returns the PEM-organized key relating to the given pkamPrivateKey.

main(): the primary capability that plays out the accompanying tasks:

• peruses the ssid, secret word, atSign, and privateKey from settings.json
• peruses the aesEncryptPrivateKey, aesEncryptPublicKey, aesPkamPrivateKey, aesPkamPublicKey, and selfEncryptionKey from the keys document unscrambles the aesPkamPrivateKey utilizing the selfEncryptionKey to acquire the pkamPrivateKey.
• Associates with the Wi-Fi network indicated by ssid and secret word.
• Syncs the device's time with an NTP server .
• Displays a menu and performs the corresponding action based on the user's input:
• If pick is 1 or 2, interfaces with an optional and sits tight for client contribution to send over the attachment.
• Expecting select is 3, creates one more classified key and saves it to settings.json.
• On the off chance that pick is 4, shows a temperature sensor menu and plays out the comparing activity in view of the client's feedback
• If opt is 5, runs a test
• If opt is 6, exits the program.

Test cases Explanation

aes_test_cases.py

• **setUp(self):** The setUp(self) method is a special method in Python classes that is used in unit testing frameworks, such as unitest or pytest. It is called before each individual test method within the class, and its purpose is to set up any necessary preconditions or configurations for the tests.

• **def test_hex_str_to_bytes(self):** The test_hex_str_to_bytes(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that converts a hexadecimal string to a byte representation.

• **def test_str_to_bytes(self):** The test_str_to_bytes(self) method is a test case method used in unit testing frameworks, such as unittest or pytest. This particular test case method is responsible for testing a function or method that converts a string to a byte representation.

• **def test_str_to_bytearray(self)** : The test_str_to_bytearray(self) method is a test case method used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that converts a string to a byte array object.

• d**ef test_bytearray_to_str(self) :** The test_bytearray_to_str(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that converts a bytearray object to a string.

• **def test_bytes_to_str(self):** The test_bytes_to_str(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that converts bytes to a string.

Main_Test_Cases.py

• **setUp(self):** The setUp(self) method is a special method in Python classes that is used in unit testing frameworks, such as unitest or pytest. It is called before each individual test method within the class, and its purpose is to set up any necessary preconditions or configurations for the tests.

• **def test_read_settings(self):** The test_read_settings(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that reads settings or configurations from a file or data source.

• **def test_read_key(self**): The test_read_key(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that reads a key or secret from a file or data source.

• **def test_send_and_receive_commands(self**) : The test_send_and_receive_commands(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing the functionality of sending and receiving commands in a system or application.

• **def test_b42_urlsafe_encode(self):** The test_b42_urlsafe_encode(self) method is a test case method typically used in unit testing frameworks, such as unittest or pytest. This specific test case method is responsible for testing a function or method that performs Base64 URL-safe encoding.

Library Explanation

• **unittest.py** (enhanced library to manage memory issues of Pico-W):A library contains a few classes for the end goal of testing.

• **SkipTest:** This is an exemption class that can be raised to demonstrate that a test ought to be skipped.

• **AssertRaisesContext:** This is a setting supervisor class that can be utilized to test whether a specific exemption is raised by a capability.

• **TestCase:** This is a base class that gives different strategies to testing, like assertEqual, assertTrue, and assertRaises. Experiments are made by subclassing this class and characterizing techniques that beginning with the prefix test_.

• **skip:** This is a decorator that can be utilized to skirt a test on the off chance that a condition is valid.

• **skipIf:** This is a decorator that can be utilized to skirt a test assuming a condition is valid.

• **skipUnless:** This is a decorator that can be utilized to skirt a test assuming a condition is misleading.

• **TestSuite:** This is a compartment class for experiments.

• **TestRunner:** This is a class that runs a set-up of experiments and reports the outcomes.

• **TestResult:** This is a class that gathers the consequences of running a set-up of experiments. It monitors the quantity of tests run, the quantity of disappointments, the quantity of mistakes, and the quantity of tests that were skipped.

base64.py

• Micro-python library for b64decode(s, altchars=None, validate=False)
• This library unravels a Base64 encoded string or bytes object into its unique parallel structure.