Encryption, Decryption & Digital Signature

Vathna.lay@cadt.edu.kh

User Class (Key Management)

- Write code that generates an RSA key pair. Create an instance of a User class that has a name and generates a key pair.

- Hint: Use the rsa.generate_private_key function with parameters for public_exponent and key_size.

User Class (Key Management)

- Add a method in the User class called sign_data that takes some data (like a string) and signs it using the user's private key.
- Hint: Use the sign() method with PSS padding and SHA256 hashing.

User Class (Key Management)

- Add a method called get_public_key_pem that serializes the public key into PEM format.
- Hint: Use the public_bytes() function for serialization, and ensure to specify the encoding and format.

User Class (Key Management)

Test the Class

- Activity: Write code that creates two instances of the User class (e.g., alice and bob). Print their public keys in PEM format to verify they are different.

Block Class (Representing Each Block)

Block Structure

- Define a Block class with attributes like block_number, data, previous_hash, creator_name, signature, and public_key_pem.

Block Class (Representing Each Block)

Calculating the Hash

- Implement the calculate_hash() method that computes the hash of the block using SHA-256.
- Hint: Use the hashlib.sha256() function and include all relevant attributes (block_number, data, etc.) to create a unique hash for each block.

Block Class (Representing Each Block)

Signature Verification

- Activity: Implement a verify_signature() method that uses the public key to verify the signature of the block.
- Hint: Load the public key from PEM format and use the verify() function.

Block Class (Representing Each Block)

Test the Block Class

- Activity: Create an instance of a Block using the previously created alice instance to provide the public key and a signature. Verify that the signature is correct using the verify_signature() method.

Blockchain Class (Managing the Chain)

Genesis Block

- Activity: Create a Blockchain class and initialize it with a chain containing a single Block object named "Genesis Block".

Blockchain Class (Managing the Chain)

Adding Blocks to the Blockchain

- Activity: Write a method called add_block() in the Blockchain class that takes data and a User object, creates a new block, verifies its signature, and appends it to the chain.

Blockchain Class (Managing the Chain)

Signature Validation for New Blocks

- Activity: Ensure that `add_block()` only adds the block if `verify_signature()` returns True.

Blockchain Class (Managing the Chain)

Testing the Blockchain

- Activity: add three blocks to the blockchain — e.g., "Block 1 by Alice", "Block 2 by Bob", and "Block 3 by Alice". Use add_block() and then print out the details of each block to verify that the blockchain is correctly formed.

Example Usage and Testing

Creating Users and Adding Blocks

- Activity: Create two users (alice and bob) and use them to add blocks to the blockchain.

Example Usage and Testing

Creating Users and Adding Blocks

- Activity 1: Create two users (alice and bob) and use them to add blocks to the blockchain.
- Activity 2: Write a print_chain() method in the Blockchain class to print all blocks in the chain, including their data, hashes, previous hashes, creator names, and signature verification results.