Ankita Zore

Wallet Creation

1. **Generate a new private/public key pair**:
    - Use a cryptographic library to generate a new private key.
    - Derive the public key from the private key.
    - Store the keys securely.
2. **Import an existing wallet**:
    - Allow users to input their private key.
    - Derive the public key from the provided private key.

Check Balance

1. **Connect to the Future (FTR) blockchain**:
    - Use an API or SDK provided by the Future (FTR) blockchain to connect to the network.
    - Retrieve the balance of the wallet using the public key.

Send Transactions

1. **Enable sending transactions**:
    - Create a transaction object with the recipient's address, amount, and other necessary details.
    - Sign the transaction with the private key.
    - Broadcast the transaction to the Future (FTR) network.

QR Code Scanner

1. **Integrate a QR code scanner**:
    - Use a QR code scanning library to scan wallet addresses and payment details.
    - Parse the scanned data to extract the address and amount.
    - Populate the transaction fields with the scanned data.

Bonus Features

1. **Implement basic encryption**:
    - Use encryption algorithms to secure private keys.
    - Encrypt the private key before storing it and decrypt it when needed.
2. **Display recent transaction history**:
    - Retrieve the transaction history from the Future (FTR) blockchain using the public key.
    - Display the recent transactions in the wallet interface.

3. **Support connection to testnet and mainnet**:
    - Allow users to switch between the Future (FTR) testnet and mainnet.
    - Use different API endpoints or configurations for testnet and mainnet connections.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <openssl/rsa.h>

#include <openssl/pem.h>

#include <openssl/err.h>


// Function prototypes

void createWallet();

void importWallet(const char *privateKey);

void checkBalance(const char *address);

void sendTransaction(const char *fromAddress, const char *toAddress, double amount);

void scanQRCode();

void encryptPrivateKey(const char *privateKey, const char *password);

void decryptPrivateKey(const char *encryptedKey, const char *password);

void getTransactionHistory(const char *address);

void switchNetwork(const char *network);


// Main function

int main() {

    int choice;

    char privateKey[256];

    char address[256];

    char toAddress[256];

    double amount;

    char password[256];
```

```
while (1) {
        printf("1. Create Wallet\n");
        printf("2. Import Wallet\n");
        printf("3. Check Balance\n");
        printf("4. Send Transaction\n");
        printf("5. Scan QR Code\n");
        printf("6. Get Transaction History\n");
        printf("7. Switch Network\n");
        printf("8. Exit\n");
        printf("Choose an option: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createWallet();
                break;
            case 2:
                printf("Enter private key: ");
                scanf("%s", privateKey);
                importWallet(privateKey);
                break;
            case 3:
                printf("Enter wallet address: ");
                scanf("%s", address);
                checkBalance(address);
                break;
            case 4:
                printf("Enter from address: ");
                scanf("%s", address);
                printf("Enter to address: ");
                scanf("%s", toAddress);
                printf("Enter amount: ");
                scanf("%lf", &amount);
            sendTransaction(address, toAddress, amount);
            break;
        case 5:
            scanQRCode();
            break;
        case 6:
            printf("Enter wallet address: ");
            scanf("%s", address);
            getTransactionHistory(address);
            break;
        case 7:
            printf("Enter network (mainnet/testnet): ");
            char network[10];
            scanf("%s", network);
            switchNetwork(network);
            break;
        case 8:
            exit(0);
        default:
            printf("Invalid option. Please try again.\n");
        }
    }
    return 0;
}

// Function implementations
void createWallet() {
    // Generate RSA keys (for demonstration purposes)
    RSA *rsa = RSA_generate_key(2048, RSA_F4, NULL, NULL);
    if (rsa == NULL) {
        fprintf(stderr, "Error generating RSA keys\n");
        return;
```

```
}

    // Save private key
    FILE *privateKeyFile = fopen("private_key.pem", "wb");
    PEM_write_RSAPrivateKey(privateKeyFile, rsa, NULL, NULL, 0, NULL, NULL);
    fclose(privateKeyFile);

    // Save public key
    FILE *publicKeyFile = fopen("public_key.pem", "wb");
    PEM_write_RSA_PUBKEY(publicKeyFile, rsa);
    fclose(publicKeyFile);

    RSA_free(rsa);
    printf("Wallet created with public/private keys.\n");
}

void importWallet(const char *privateKey) {
    // Import wallet logic (not implemented)
    printf("Wallet imported with private key: %s\n", privateKey);
}

void checkBalance(const char *address) {
    // Connect to FTR blockchain and check balance (not implemented)
    printf(" Balance checked for address: %s\n", address);
}

void sendTransaction(const char *fromAddress, const char *toAddress, double amount) {
    // Logic to send transaction (not implemented)
    printf("Transaction of %.2f FTR sent from %s to %s\n", amount, fromAddress, toAddress);
}
```

```
void scanQRCode() {
    // QR code scanning logic (not implemented)
    printf("QR Code scanned.\n");
}

void encryptPrivateKey(const char *privateKey, const char *password) {
    // Encryption logic (not implemented)
    printf("Private key encrypted.\n");
}

void decryptPrivateKey(const char *encryptedKey, const char *password) {
    // Decryption logic (not implemented)
    printf("Private key decrypted.\n");
}

void getTransactionHistory(const char *address) {
    // Fetch transaction history (not implemented)
    printf("Transaction history retrieved for address: %s\n", address);
}

void switchNetwork(const char *network) {
    // Logic to switch between mainnet and testnet (not implemented)
    printf("Switched to %s network.\n", network);
}
```