Create a HD Bitcoin Wallet

The objective of this assignment is to become familiar with BIP32, BIP39, BIP43, and BIP44 standards. Create a new directory and install the bitcoinjs-lib library by following the `README.md` at the linked repository to get started.

Creating public/private key pairs

Mnemonic seed phrase

Before BIP32, bitcoin wallets held a collection of key pairs that had no relationship; this kind of wallet is known as a JBOK ("Just a Bunch of Keys") wallet. Since most wallet implementations generate a new key pair after every transaction, JBOK wallets are considered highly secure because a compromised private key only permits an adversary to spend funds associated with that one address. However, the disjointed nature of JBOK wallets necessitates frequent backups of keys and makes wallet imports/exports cumbersome.

The BIP32 standard introduced the concept of a HD (hierarchical deterministic) wallet that can derive child key pairs from a master key pair. This way, only one key is needed to import/export, and only that key needs a backup.

The BIP39 standard introduced an even more convenient and secure method of wallet import/export and backups. This standard specifies how to use a *mnemonic seed phrase* to derive a master key pair for an HD wallet. This provides enhanced convenience because now a set of readable words can be used represent a master key, and it provides enhanced security because the seed phrase can be more faithfully transcribed for paper backups.

> *Task*: Create a mnemonic seed phrase. From this seed phrase, generate a master key.

Derivation path

BIP43 and BIP44 introduced a paradigm for denoting the path of a bitcoin address within a wallet based on the purpose, coin type, account number, whether it is a change address, and address index.

`m / purpose' / coin_type' / account' / change / address_index`

> *Task*: Using the master key and the derivation path, generate a child bitcoin address.

Fund the address

> *Task*: Using a bitcoin testnet faucet (i.e. https://bitcoinfaucet.uo1.net), send funds to your newly generated child address.

Create a raw transaction

Inputs and outputs

Bitcoin transactions are comprised of input and output segments. The inputs of a transaction are used to verify that the address intending to send bitcoin is indeed entitled to those funds. The outputs of the transaction are used specify who is to be the newly entitled owner of the funds. For a detailed description of these segments, see this article on dissecting transactions, by *@klmoney8*.

> *Task*: Gather, then serialize the necessary input data to form a raw unsigned transaction. Be sure to read the section on the UTXO model to learn how to create transactions using an arbitrary amount of bitcoin associated with your address.

Broadcast the raw transaction

Signing a transaction

The two fields of particular interest within a bitcoin transaction are the ScriptSig and ScriptPubKey. These two fields are responsible for the phenomenon of digital ownership that is at the heart of the bitcoin protocol. The ScriptPubKey is responsible for specifying the condition in which a UTXO can be spent, and the ScriptSig is the input that intends to satisfy the given condition. The most typical bitcoin transactions use a Pay-to-Public-Key-Hash (P2PKH) ScriptPubKey, which necessitates a user to prove access to the private key that corresponds to the public key hash the UTXO was sent to. This process of proving ownership of the funds being sent is considered *signing* the transaction.

> *Task*: Sign the transaction and serialize in preparation to broadcasted over the network.

Send the transaction over the network

Once you have a serialized signed transaction, you are ready to send it over the network to be verified and included in the blockchain! This is done by sending an RPC call to a node saying that you would like your transaction broadcasted. Instead of running a node ourselves, we can use a public API to interact with the blockchain, such as the Blockstream API.

> *Task*: Send the raw transaction hex to the blockchain via a public API. You should be able to confirm that the transaction has been sent by searching for its hash on a block explorer.