Wallet development

What are a wallet's functions?

- Key management
  - Identify owned transactions
  - Generating new addresses
  - Determining how to sign transactions
- Constructing and sending transactions
  - Parsing addresses and turning them into txOuts
  - Selecting UTXOs (coin selection)
  - Signing inputs
- Persistence
  - Storing keys
  - Storing UTXOs
  - Storing transaction history
  - Storing metadata (eg how far through the blockchain have I parsed?)

Glossary

- pubkey - a public key, used to verify signatures. A point on the secp256k1 curve. (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/pubkey.h#L30)
- privkey - a private key, kept secret and used to sign data. A scalar in the secp256k1 group. (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/key.h#L27)
- CKeyID - a key identifier, which is the RIPEMD160(SHA256(pubkey)). This is the hash used to create a P2PKH or P2WPKH address. (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/pubkey.h#L20)
- CTxDestination - a txout script template with a specific destination. Defined in (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/script/standard.h#L139) Stored as a variant variable. Can be a:
  - CNoDestination: no destination set
  - CKeyID: TX_PUBKEYHASH destination (P2PKH)
  - CScriptID: TX_SCRIPTHASH destination (P2SH)
  - WitnessV0ScriptHash: TX_WITNESS_V0_SCRIPTHASH destination (P2WSH)
  - WitnessV0KeyHash: TX_WITNESS_V0_KEYHASH destination (P2WPKH)

Initialization and Interfaces

- The wallet component is initialized through the WalletInitInterface: https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/walletinitinterface.h. This virtual interface is defined for all bitcoind builds.
- For builds with wallet, the interface is overridden in src/wallet/init: https://github.com/bitcoin/bitcoin/blob/master/src/wallet/init.cpp#L16
- For `--disable-wallet` builds, a dummy interface is defined in src/dummywallet.cpp: https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/dummywallet.cpp#L15
- Those initiation interface methods are called during node initialization, eg: https://github.com/bitcoin/bitcoin/blob/44d81723236114f9370f386f3b3310477a6dde43/src/init.cpp#L1288
- WalletInit::Construct() adds a client interface for the wallet.
- The node then tells the wallet to load/start/stop/etc through the ChainClient interface in src/interfaces/wallet.cpp https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/interfaces/wallet.cpp#L504
- Most of those methods in that interface call through to functions in src/wallet/load.cpp https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/wallet/load.cpp
- The node holds a WalletImpl interface to call functions on the wallet https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/interfaces/wallet.cpp#L122.
- The wallet holds a Chain interface, which is used by the wallet to call functions on the node https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/interfaces/chain.cpp#L244.
- The node notifies the wallet about new transactions and blocks through the Validation Interface (https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/validationinterface.h#L72 and https://github.com/bitcoin/bitcoin/blob/f792395d13aa99ce51887db14e4f77a746d910e3/src/interfaces/chain.cpp#L162)

Why all this indirection?

To fully separate the wallet from the node:

- Well defined interface is easier to reason about
- Individual components can be tested in isolation
- Separate wallet into a different process

- Potential for different wallet implementations

Code Management

- **coinselection.cpp|h** - Coin selection algorithm
- **crytper.cpp|h** - encrypting the wallet's private keys
- **[wallet]db.cpp|h** - interface to wallet's database for persistent storage
- **init.cpp** - initializing the wallet module
- **load.cpp|h** - loading/starting/stopping individual wallets
- **rpc*.cpp|h** - wallet's RPC interface
- **wallettool.cpp|h** - standalone wallet tool binary
- **wallet.cpp|h** - EVERYTHING ELSE

Key Management

Identify Owned Transactions

- When a transaction is added to the mempool, or a block is connected, the wallet is notified through the validation interface: https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1232 https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1251
- The wallet needs to know if the transaction belongs to it. That happens in SyncTransaction(), (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1222), which calls AddToWalletIfInvolvingMe() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1040)
- The magic happens in IsMine() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1061 https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/script/ismine.cpp#L175)
- This takes the scriptPubKey, interprets it as a Destination type, and then checks whether we have the key(s) to watch/spend the coin.
- This is overly complicated, inefficient due to pattern matching, not selective, and not scalable. (https://gist.github.com/sipa/125cfa1615946d0c3f3eec2ad7f250a2)

Generating Addresses

- The Bitcoin Core wallet was originally a collection of unrelated private keys
- If a new address was required, a new private key could be generated

- What are the problems with this?
- Giving an address out and then restoring from a backup loses funds!

Keypools

- Introduced by Satoshi in 2010 (https://github.com/bitcoin/bitcoin/commit/103849419a9c014a69c76b6f96e48b66cbc838ca)
- Cache (100) private keys before they're needed
- When a new public key is needed (either for address or change), draw it from the keypool and refresh the pool
- (Also allows an encrypted wallet to give out an address without unlocking)

HD Wallets

- A minimal HD wallet implementation was added to Bitcoin Core in 2016 (https://github.com/bitcoin/bitcoin/pull/8035)
- A new HD seed is set (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L1486) on first run or when upgrading the wallet (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L4094)
- Restoring old backups can no longer definitively lose funds (since all private keys can be rederived).
- However, if many addresses were used since the backup, then the wallet may not know how far ahead in the HD chain to look for its addresses.
- The keypool essentially became an address look-ahead pool. It is used to implement a 'gap limit'.

Generating Keys

(https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L190)

- For HD wallets, new keys are derived using the BIP32 HMAC derivation scheme (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L227)
- For non-HD wallets, strong randomness is used to generate a new key (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/key.cpp#L157)
- In both cases, we test the new key by signing a message (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/key.cpp#L232)
- We save the key to the DB before using it (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L221)

Constructing and Sending Transactions

Parsing addresses and Constructing Transactions

- Sending from the wallet happens through the RPC or GUI
    - sendtoaddress (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/rpcwallet.cpp#L345)
    - sendmany (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/rpcwallet.cpp#L800)
    - {create,fund,sign,send}rawtransaction
- The address is decoded into a CDestination (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/key_io.cpp#L73)
- Other parameters can be added for finer control (RBF, fees, etc)
- The wallet creates the transaction in CreateTransaction() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L2733)

Selecting UTXOs (coin selection)

- By default, coin selection is automatic
- The logic starts in CWallet:SelectCoins() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L2480)
- By preference, we choose coins with more confirmations (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L2550)
- The actual logic for selecting which UTXOs to use is in coinselection.cpp, which implements the branch and bound algorithm (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/coinselection.cpp#L21)
- If that fails, we fall back to using the old KnapsackSolver (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/coinselection.cpp#L216)
- Manual coin selection (Coin Control) is possible. See the CCoinControl structure (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/coincontrol.h#L16).

Signing inputs

- Signing is (almost) the last step in CreateTransaction() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L3055)
- The CWallet is an implementation of the SigningProvider interface (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/keystore.h#L19)
- The signing logic for the SigningProvider is all in src/script/sign.cpp (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/script/sign.cpp#L190)

Sending Transactions

- The wallet saves and broadcasts the wallet in CommitTransaction() (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.cpp#L3100)
- The transaction is added to the mempool over the submitToMemoryPool() interface method (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/interfaces/chain.cpp#L152) and relayed on the network in the relayTransaction() method (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/interfaces/chain.cpp#L293)

Persistence

- Bitcoin Core wallet uses berkeley db for storage (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/walletdb.h#L22)
- db.cpp|h is for the low-level interaction with bdb (eg setting up environment/opening/closing database, batch writes, etc) (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/db.cpp). walletdb.cpp|h is for higher-level database read/write/erase operations (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/walletdb.cpp).
- bdb is a key-value store. There's no database schema.
- Keys are a type (eg "tx") followed by an identifier (eg txid). The value is the serialized data (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/walletdb.cpp#L50).

- Object serialization code is in wallet.h (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/wallet.h#L206) and walletdb.h (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/walletdb.h#L74)
- Additional deserialization logic in walletdb.cpp (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/wallet/walletdb.cpp#L206)

Future Directions

- Descriptor-based wallets (https://github.com/bitcoin/bitcoin/pull/15764)
- Hardware wallet integration (https://github.com/bitcoin-core/HWI)
- Improve wallet<->node interface (https://github.com/bitcoin/bitcoin/blob/431d81b61ca968da2d7c25f0d56455a44cd46fed/src/interfaces/chain.h#L37)
- Separate wallet into separate process (https://github.com/bitcoin/bitcoin/pull/10102)
- Different backend storage?
- Re-implementation?