What is a UTXO, and how does it work for a blockchain ledger?

Today I'd like to introduce the basics of how a blockchain works, and how it keeps track of money in a secure manner. I will be covering the UTXO model, as it is used by Bitcoin and Qtum. There is another way of managing funds on the blockchain called the account model, but it will not be covered here.

First I'd like to give some definitions in case you do not know anything about Bitcoin.

- One-way hash (or just "hash") - A cryptographic algorithm which converts an arbtritrary amount of data into a fixed-length "digest". The algorithm does this in a way that given just the digest it is impossible to determine what the input data was, and furthermore it is impossible to predict what the digest is from the given input data. The most common example is SHA256 which is used extensively in Bitcoin, but there are many others including SHA3, RIPEMD160, scrypt, and many others.
- Public key cryptography - A cryptographic mechanism by which a "private" secret key can be converted into a "public" key and used to prove ownership of the private key without giving away the secret. Additionally it is possible to encrypt data using the public key so that only the person holding the private key can decrypt it. In Bitcoin this is commonly used to sign transactions. It is possible to prove that the creator of the transaction owns the secret private key by using only the signature data and the public key.
- Merkle root - A tree data structure that uses one-way hashing to hold multiple pieces of data making it so that any data in the input of the tree can not be modified without changing the final value of the merkle root hash.
- UTXO - Unspent Transaction Output, an unspent vout from a transaction
- Block - The smallest verifiable and unforgeable unit on the blockchain. It contains various data to prove it's consensus as well as transactions

So, let's talk about how transactions work in this. Transactions in Bitcoin resemble a cashier's check in some ways. When you want to spend an "output" of a transaction you must spend the entire thing. It's similar to how you can't walk into the bank and say "I want to cash half of this check". However, in this model there is no equivalent of cash or bank accounts. So in order to send money anywhere you must "cash" a check written out to you, and "output" from that cashing process a check to your intended destination, and another check back to yourself.

This "cashing process" is actually a transaction in Bitcoin. In a transaction you spend 1 or more "checks" (actually known as UTXOs) and create 1 or more UTXOs to new destinations from those spent funds. The UTXOs you spend in a transaction are called "vins", and the new UTXOs you create are called "vouts". Once a UTXO is spent by a transaction it can be considered gone and destroyed. You can see it's history in the blockchain, but there is nothing that can done with it.

So, one problem in our system so far is that checks are normally written out to names, such as "Jordan Earls". Anyone of course can say they are any name on the internet. This is where we introduce public key cryptography and programming into UTXOs. In Bitcoin UTXOs contain a script, or a computer program, which are only spendable if you can make that script end by saying "True". Let's look at the most simple script possible that does something useful:

```
[pubKey] OP_CHECKSIG
```

This is referred to as a "pay-to-pubkey" script. It was the first standard Bitcoin transaction type. The first item is `[pubKey]`. This is the data for a public key. Remember that for each public key there is a private key which is kept secret by it's owner. It is safe to publish the public key, but not the private key. The Bitcoin "Script" language is stack based. So imagine you have a stack of papers. You write the public key on a piece of paper and then place it on the stack. The next piece of this script is OP_CHECKSIG. This specific operation will take 2 things off of the top of the stack. The first thing it takes off is the public key. Then, the second thing it takes off is the cryptographic signature.

This is confusing now though. OP_CHECKSIG takes 2 values from the stack (also known as arguments), but our script appears to only have 1 value, `pubKey`. This is where the `vin` portion becomes important. You can imagine the vout script as the "pay to" field on a check, and the vin script as the place you sign on the back, proving that you are indeed the intended party from the "pay to" field. In Bitcoin, a script is not executed until it is spent. And when it is spent, it first executes the vin script, and then places the resulting data from the vin stack on to the vout stack. So in actual execution, the script might look rather like:

```
[signature from vin] [pubKey] OP_CHECKSIG
```

One could consider the vout script as a challenge, and the vin as the answer to give the vout to satisfy it. Anyway, now that we have a vin providing the signature and attempting to spend these funds, we can actually execute the script. If the signature and public key is valid, then OP_CHECKSIG will push "true" on the stack, resulting in the UTXO being succesfully spent.

So in a transaction, each vin specifies a previous UTXO, and provides an answer that causes the UTXO's script to return "true". If an invalid signature or similar is used, then the scripts will return "false" and the transaction will not be valid. It is not possible to partially spend a UTXO. It must be completely spent or left untouched. This means that if you have a UTXO worth 10 tokens, and you want to send 7 tokens to Bob, then you must make a transaction that spends this 10 token UTXO, and creates 2 outputs. One output to Bob (using his public key), and one output back to yourself (ensuring that you can provide an "answer" to the vout to spend it successfully). This second output back to yourself is called a "change address".

Finally, we have a reasonable way of exchanging tokens using transactions and scripts. However, we face a problem. When someone sends you a transaction output, how can you be sure that their vins for that transaction only use unspent outputs. This is where the concept of the blockchain becomes important.

A block in Bitcoin has a header. The header contains the following:

- Version
- Previous block header hash
- Merkle root hash of all transactions in the block
- Time of creation
- Difficulty
- Nonce

The body of the block is complete transactions (and eventually witnesses as well, but that's another topic).

Because each block includes a reference to the previous block, it is impossible to modify a previous block sereptitiously. To modify a previous block would change the block hash, and thus break the "chain" made of block hashes.

Bitcoin uses the Proof of Work (PoW) consensus system. This will be explained more in a later article, but basically it is a system which requires participants (miners) in the block creation process to put in a certain amount of computational work to solve a difficult puzzle. The first miner to solve the puzzle gets a reward and their created block is added to the network's blockchain. How much work must be done is controlled by the "difficulty" specified in the block.

In PoW, only the block header is actually used for the consensus mechanism. The merkle root hash ensures that despite this, it is possible to validate every transaction in the body of the block, as well as ensure that every transaction has been received.

Once a block has been created, it's transactions can be mostly considered permanent. The only way to "double spend" a UTXO is to replace the block in which the spending transaction took place. This can happen naturally in some cases (known as orphan blocks), but as more blocks are built on top of the transaction containing block, the likelyhood of this becomes exponentially less likely, and furthermore, would require exponentially more work to maliciously attack and replace.

This is why many services that accept Bitcoin wait for 3 or 6 confirmations (blocks placed on top of the transaction containing block). It becomes incredibly unlikely that the blockchain could be broken and those funds spent by another transaction.

We have only one remaining problem. Where do the tokens initially come from? They come from the mining process. As part of mining, the miner adds a special transaction called a "coinbase" transaction. This transaction has no inputs, and is allowed to have outputs worth a set amount (currently 12 in Bitcoin). This coinbase transaction is where all of the tokens in circulation actually come from. Without tokens there would be no transactions to create, and thus nothing to be done.

Now we have a functioning blockchain that is capable of holding it's value securely, ensuring that double spends are extremely difficult to execute (and increasing in difficulty with more confirmations). You should now know enough to understand how Bitcoin, Qtum, and other UTXO cryptocurrencies really work at the protocol level and can begin to look into more advanced topics on the blockchain.

References:

1. https://en.bitcoin.it/wiki/Script#Obsolete_pay-to-pubkey_transaction

Tags: blockchain-edu qtum utxo blockchain Posted: 7/27/2017 6:20:44 PM Tweet

Comments

Posting comments is currently disabled(probably due to spam)