CMSC 398F
Week #4
Transactions
...

Announcements

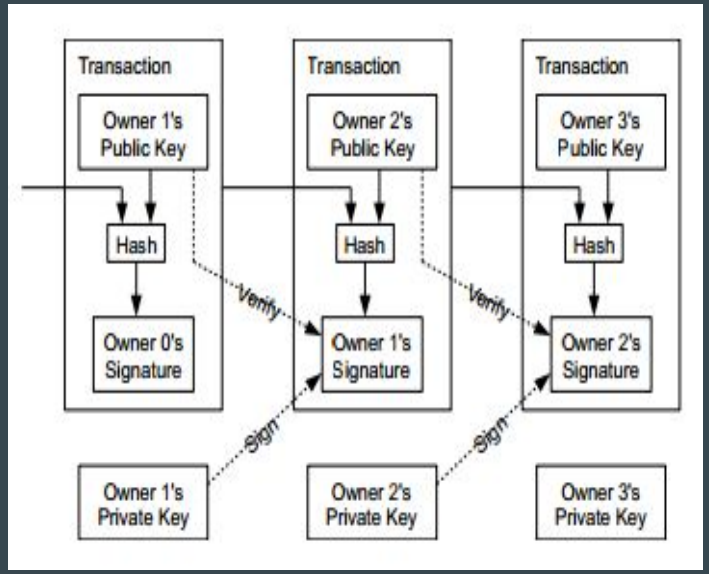
- Quiz 3 was due this morning, grades will be out soon
- Quiz 4 will be released this weekend
- Project #1 will be released soon
- Join the class Piazza!
    - piazza.com/umd/fall2022/cmsc398f

Faucets

- A crypto faucet lets users earn small crypto rewards by completing simple tasks.
- The earliest crypto faucet may be a bitcoin faucet created in 2010 by the then-lead developer of the Bitcoin network named Gavin Andresen. It gave 5 BTC for free to each user who completed a simple captcha.
- Crypto faucets are designed to provide users with free cryptocurrency to start learning about digital assets and eventually use them.
- Examples: Coinbase, Bitcoinker

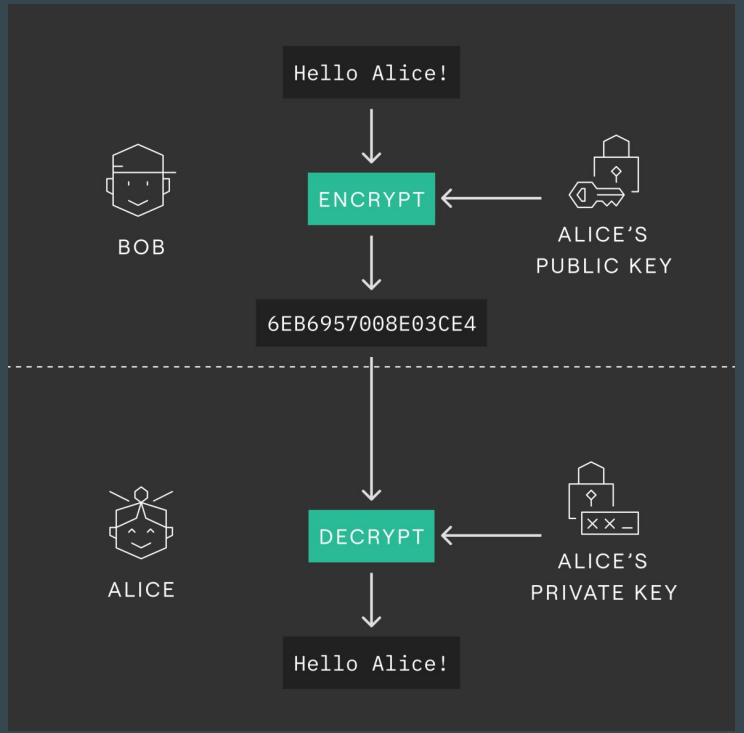
Transactions: Bitcoin

- In order to own bitcoin, you must own a bitcoin address
- A Bitcoin address is a string of letters and numbers that represents a destination on the Bitcoin network, nothing more.
- Private keys are important because private keys are what allows you to send bitcoin from your address to another
- Your private key effectively works as a password
- Your bitcoin address is a hash of your public key, which itself is part of an ECDSA (more on it later) pair with your private key
- Each person can create their own private keys in whatever method they choose



Transactions: Public and Private Keys

- Your private and public keys are a 256-bit integers, and your address is a 160-bit hash of your public key
- Private key (256 bits) → Public key (256 bits) → Hash of public key (256 bits) → Address (160 bits)
- Public keys are derived from private keys via Elliptic Curve Digital Signature Algorithm (ECDSA)
- Cannot derive a private key given just a public key
- Public keys are then hashed using SHA-256, and then run through the RIPEMD-160 hashing algorithm to make a 160-bit address
- Private keys allow you to create a digital signature that proves that you want to pay your bitcoin to someone else

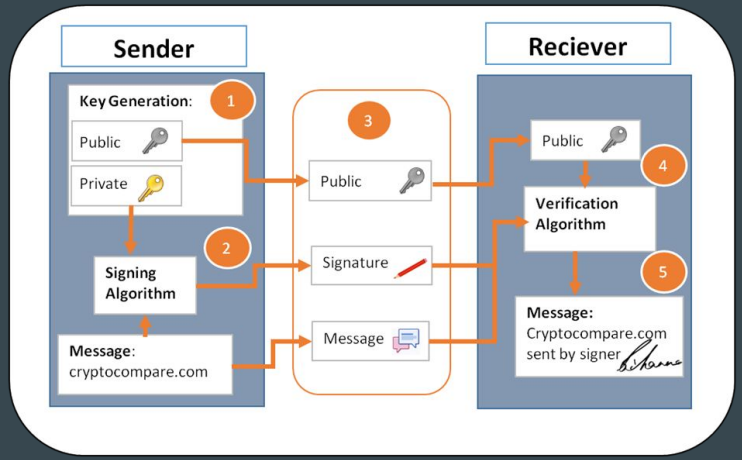


Transactions: Digital Signatures

- A signature is a hash comprised of several inputs
- Normally, a (simplified) function signature is called like:
- signature = sig(private_key, message, public_key)
- A signed message m would look something like:
- (m, sig(private_key, m, public_key))
- Verifying a signature requires the message, the signature, and the public key of the key that signed the message
- is_valid = verify(m, sig, public_key)
- This signed message is broadcast to the network when someone wants to spend bitcoin from one address to another

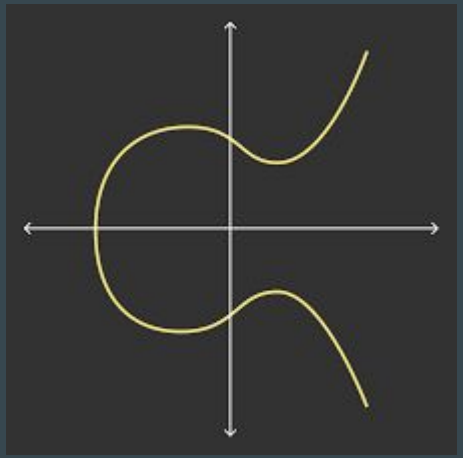
Validating Transactions

- If Alice wants to send Bob 1 bitcoin, she must sign a transaction spending 1 bitcoin of inputs with her private key and send it to nodes on the network. The miners, who know her public key, will then check the conditions of the transaction and validate the signature.



ECDSA

- Bitcoin's current signature scheme is known as the Elliptic Curve Digital Signature Algorithm (ECDSA).
- An elliptic curve is a finite group of points on a curve where some operations are easy to perform in one direction but difficult in the other direction.
- The ECDSA algorithm relies on this to generate signatures that are difficult to forge and easy to verify.
- More about ECDSA in CMSC456!!



Why have a digital signature?

- Owners of Bitcoin can create a transaction and ‘sign’ it using their private key
- This signature can then be verified mathematically by using the owner’s public key via ECDSA
- It keeps transactions secure as an owner can only sign transactions with their private key

How is bitcoin transferred?

- When transactions are created, they are posted to the network
- Nodes have to verify the transaction by checking several things
  - The signature over the transaction input has to be valid
  - The amount of bitcoin being sent must be less than or equal to the amount of bitcoin the user has
  - The bitcoin being spent has not already been spent in another transaction
- Nodes pass verified transactions on to miners so that they may be included in the next valid block
- In general, transactions are not considered to be final until they have a certain number of confirmations
  - Usually somewhere between 2 and 6

What exactly goes into a bitcoin transaction?

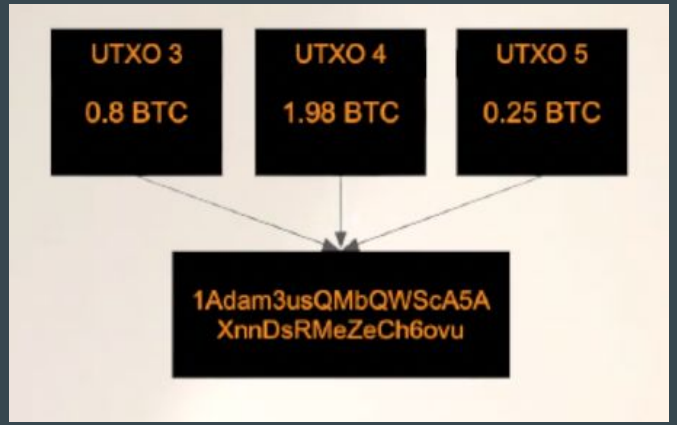
- Transactions are comprised of inputs and outputs
- Recall that transactions are just chains of digital signatures
  - This means that in order to claim bitcoin, you must point to a previous transaction that has the digital signature that sent the bitcoin to you
  - What does this claim look like?
  - In terms of inputs and outputs, the input to your transaction (which is the bitcoin you are trying to spend) must be the output of a previous transaction
  - If you've been sent bitcoin, but have not spent it yet, that output is called an **unspent transaction output (UTXO)**

Transactions

- We know how banks keep track of accounts and the identities of the people who own these accounts
- How do we do this in Bitcoin and other cryptocurrencies? How do we know who owns what?
- Also, how are transactions between users actually handled in Bitcoin?

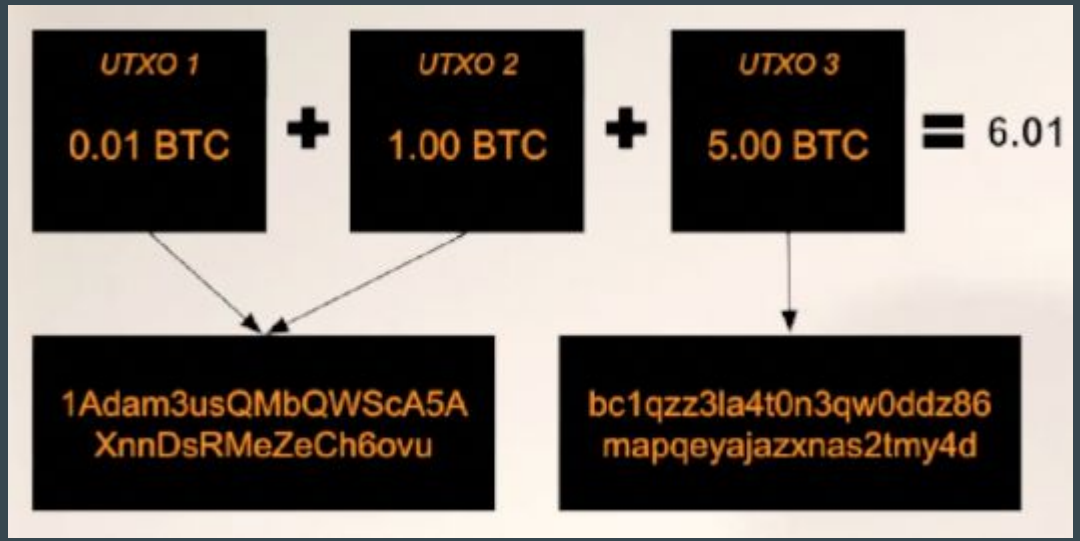
UTXOs

- Contrary to popular belief, Bitcoin is not a system of accounts and balances
- Instead, the network keeps track of these things called UTXOs
  - Stands for ‘Unspent Transaction Output’
- It is simply an **amount of bitcoin that is assigned to a Bitcoin address**
  - Can be any amount of BTC, as long as it is 1 Satoshi
- Ensure that users are not spending more than they can



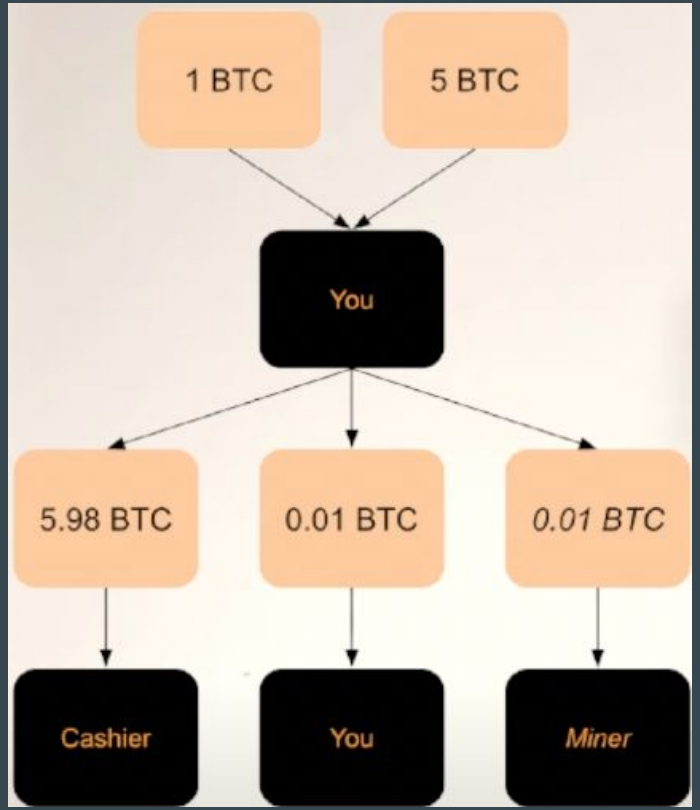
An Example

- Say you have 6.01 BTC in your wallet, that is represented as 3 UTXOs.
  - UTXO 1 is .01 BTC
  - UTXO 2 is 1 BTC
  - UTXO 3 is 5 BTC



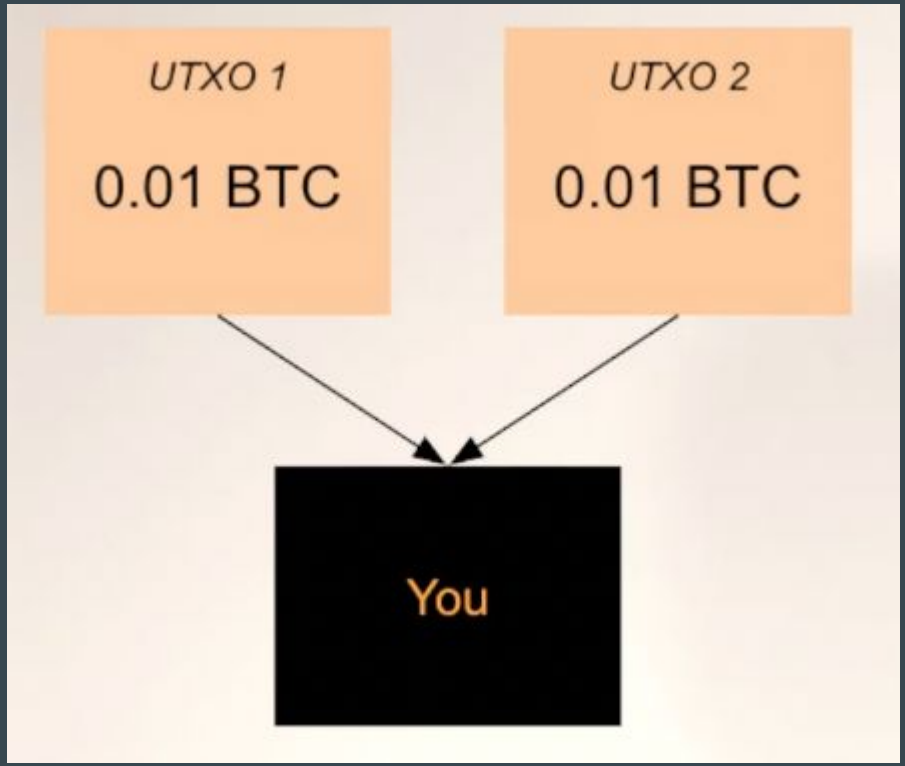
An Example

- Say you want to pay your friend 5.98 BTC using your 1 BTC UTXO and 5 BTC UTXO
  - These are known as inputs to a transaction
- Your friend will receive a new UTXO that has 5.98 BTC
- Another UTXO worth .01 BTC is created, and sent back to yourself
- Another UTXO worth .01 BTC is given to the miner (mining fee)
- At this point, the two UTXOs used as inputs are deleted and cannot be used again
- The new UTXOs created as outputs are unspent, and will be used in future transactions



An Example

- We are left with two UTXOs
  - One UTXO was given back to us as ‘change’. (Output of previous transaction)
  - The other UTXO was ours to begin with (was never an input)



UTXOs

- Think of UTXO's as Piggy Banks
  - Every time a transaction is made to us, we put all that money into a UTXO, or piggy bank.
  - When we want to spend money, we break open that piggy bank (UTXO), spend whatever we want to (and send it as a UTXO), and then put the rest of the money (change) into another piggy bank (UTXO) and give it back to ourselves.
- Complexity for transaction verification drastically improves
  - Question changes from "Is this user trying to spend more money than they can?" to "Does this UTXO have enough funds for the current transaction?"

UTXOs: Recap

1. UTXO: amount of Bitcoin assigned to an address
2. Your balance is the sum of all of your UTXOs
3. UTXOs are inputs to transactions
4. New UTXOs with new values are created as outputs
5. UTXO can be any size bigger than 100 millionth of a BTC

UTXOs

- By adding together the entire BTCs UTXO set, we can calculate the amount of Bitcoin currently in circulation
- In other words, we can independently verify and audit Bitcoins money supply.

```
umbrel@umbrel:~ $ ~/umbrel/bin/bitcoin-cli gettxoutsetinfo
{
  "height": 716884,
  "bestblock": "00000000000000000000ad8639af353cdcb5958fd150ebd3ba9afcd24371362fe",
  "txouts": 78075591,
  "bogosize": 5838694906,
  "hash_serialized_2": "6146964c6f3aee7b694b148909328586989183b8d06a14d96d957de4e52b3469",
  "total_amount": 18917825.04143106,
  "transactions": 47204957,
  "disk_size": 4750590682
}
```

What happens once a transaction is included in the block?

- Usually merchants wait for confirmation before releasing goods/services
- Recall the concept of UTXOs
  - Once a new block is created, the UTXOs that were claimed will now be spent
- All UTXOs are kept track of by nodes in the **UTXO pool**
  - When blocks are published, nodes update their UTXO pools to remove the inputs that are being spent and add the outputs that were created by the block
  - This will maintain the accuracy of the total amount of bitcoin a user currently has

Other commonly used signature schemes in blockchain

- BLS Signatures (ETH2)
  - The current Ethereum chain also uses ECDSA
- Schnorr Signatures (Bitcoin)
  - Much more elegant than ECDSA
- EdDSA