Bitcoin Presentation Notes

1 Motivation

- Situation: commerce on the Internet relies almost exclusively on financial institutions serving as trusted third parties to process electronic payments.
- Problem: such third parties may not be trustworthy.
- Goal: develop a decentralized version of electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution.

2 Digital Signatures

- Scenario: Alice wants to send a message to Bob over a network. How can Bob verify that the message he received was from Alice? Alice needs to sign her message.
- Desired property of signature: cannot be forged on a different message.
- Both Alice and Bob each generate a public/private key pair for themselves. Each key is some string of bits. Private keys are kept secret.
- Producing a signature: $\mathsf{sign}(\mathsf{message}, \mathsf{privateKey}) = \mathsf{signature}$.
- Verifying a signature: $\mathsf{verify}(\mathsf{message}, \mathsf{signature}, \mathsf{publicKey}) = \mathsf{true}/\mathsf{false}$.
- Only the owner of the private key can produce the signature.
- No one can copy the signature and forge it on another message.
- Signature is a 256 bit value. Hard to find a valid signature if you don't know the secret key. There is no strategy better than guessing and checking if random signatures are valid using the public key. There are 2^{256} signatures to check; this is a very large number.

3 Transactions

- Electronic coin: a chain of digital signatures.
- Alice transfers a coin to Bob:
  1. Alice computes the hash of the previous transaction and Bob's public key.
  2. Alice signs the hash using her private key.
  3. Alice adds her signature to the end of the coin.
- Bob can verify that Alice transferred a coin to him using Alice's public key.
- Problem: how to verify that Alice did not double-spend the coin? For example, Alice could have transferred a coin to Charlie and then transferred the same coin to Bob. For our purposes, the earliest transaction is the one that counts, so we should ignore later attempts to double-spend. As of now, there is no mechanism preventing Alice from double-spending.
- To accomplish this without a trusted party, transactions must be publicly announced, and we need a system for participants to agree on a single history of the order in which they were received. Bob needs proof that at the time of the transaction, the majority of nodes agreed it was the first received.
- Solution: a distributed timestamp server with proof-of-work.

4 Timestamp Serve

- Take a hash of a block of transactions to be timestamped and widely publishing the hash.
- The timestamp proves that the data must have existed at the time, obviously, in order to get the hash.

5 Cryptographic Hash Functions

- $\mathsf{SHA256}(\mathsf{message}) = \mathsf{hash}$.
- Input: message.
- Output: String of 256 bits, known as the hash of the message.
- The output looks random even though it isn't (i.e., the function always gives the same output for a given input).

- Slightly changing the input results in the output to change completely in an unpredictable way.
- Infusible to compute in the reverse direction. Given only the output, finding the input that hashes to the output is extremely difficult. There is no better method than to guess and check. Need to go through 2^{256} guesses.

6 Proof-of-Work

-

7 Network

- The network executes as follows:
  1. New transactions are broadcast to all nodes.
  2. Each node collects new transactions into a block.
  3. Each node works on finding a difficult proof-of-work for its block.
  4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
  5. Nodes accept the block only if all transactions in it are valid and not already spent.
  6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.