6.857 Recitation 8
Bitcoin and TLS

TA: Leo de Castro

April 14, 2019

Today

- Bitcoin
  - Proof of Work & Consensus
- PKI
  - Man in the Middle Attacks
  - Chain of trust
- TLS
  - Protocol Steps
  - Attacks

1 Bitcoin

1.1 Blocks

The fundamental data-structure underneath Bitcoin is the block-chain, which implements a distributed append-only ledger. This ledger keeps track of all transactions in the network.

How does the block-chain work? The block-chain is made up of blocks. Each block consists of it's contents (i.e. a list of transactions) as well as some meta-data. The two most important pieces of meta-data that we're going to discuss is the previous block hash and the nonce.

The previous block hash is what connects the block to the chain. A *block hash* is simply the hash of all the data that makes up a block, both the contents and the meta-data. By including the previous block hash in a block, the blocks are chained together to make a block-chain.

In order for a block to be accepted on the chain, it's hash value must be *valid*. The hash of a block is valid if the hash value has d leading zeros, where d is the current difficulty of the chain. How can we have arbitrary block contents while still having a valid hash? Enter the nonce, the piece of meta-data whose sole responsibility is to make the block hash valid. When a new block is being created, essentially all of the work goes towards finding a good nonce.

This is what it means to "mine" a block. All miners have a set of transactions that they are trying to get on the chain, and they must work to find a nonce that makes the block that includes their transactions valid.

1.2 Integrity through Proof of Work

Why does Bitcoin bother with nonces and valid block hashes? The answer is at the heart of why Bitcoin works. When a miner broadcasts a new block to be part of the chain, they are essentially "vouching for" the validity of the transactions in the block's contents. When another miner sees this new block, they must make a decision. They can either continue to mine the block they are currently mining, or switch to mine this new block. Each block's meta-data contains the public key of the miner who produced it, and each miner who produced a block on the main chain gets 12.5 bitcoins plus the tips included in the transaction. Since the main chain consists only of the longest chain, the miners have incentive to only mine the block at the very end of the longest chain.

When a miner switches to mining a new block, they are also vouching for the validity of the blocks contents. But, what prevents a dishonest miner from vouching for invalid transactions? This gets to the key assumption of bitcoin, which is that more than 50% of the miners are honest. If 50% of the miners are honest, then they will be able to create and vouch for valid blocks faster than any malicious minority, so no incorrect transaction will ever be on the main chain.

This is why the proof of work is so fundamental to the block-chain's integrity. It forces the creation of each block to take up a real amount of computer time, making the cost of outpacing the hashing rate of all honest miners too high to be feasible.

2 Public Key Infrastructure

Whenever you deploy a crypto-system in the real-world, especially one where messages are sent over the open Internet, it must be secure against the strongest, most active adversaries.

In many of the security games we've looked at in the class, we often have made an implicit assumption that the honest participants know each others public keys or somehow know they are talking with the right person. In practice, this is not a reasonable assumption to make, as there are devastating "man-in-the-middle" attacks that can make even the strongest cryptoscheme useless by simply replacing the key exchange messages with public keys owned by the adversary (see last recitation notes for an example).

How can we defend against this attack? We need a root of trust, usually a verification key, that we can be reasonably confident is not owned by an adversary. Once we have this verification key, we can request signatures of public keys under this verification key from parties who we wish to communicate with.

For example, suppose we had a verification key $CAVK$ and we receive a public key GPK from a machine claiming to be Google. This machine can present a *certificate* of the form

$$\sigma = Sign\Big(CASK, m = \big(\text{google.com, } GPK\big)\Big)$$

where $CASK$ is the certificate authority's signing key. If we trust the certificate authority, we can be confident that this public key really did come from Google.