Topic 13: Distributed Systems
Blockchains

Hilmar Gústafsson

January 6, 2020

1 Cryptographical Tools

A summary of the tools necessary for Blockchains.

1.1 Hash

A hash function H takes binary input of arbitrary length, and creates fixed-length output of it.

$$H : X = \{0,1\}^* \to \{0,1\}^L$$
$$\text{typically where } L \in \{128, 160, 256, 512\}$$

For security purposes, it is important that a small change in the input results in a large change in the output. Collisions exist, but they are hard to find with this property.

1.2 Signatures using Asymmetric Cryptography

Digital signatures are typically done using three algorithms, and a known hash H.

KeyGen An algorithm which returns two keys: a private signing key, and a public verification key.

Sign An algorithm which computes the signature of some input, given the private key. This input is typically a hash of some data to be sent.

Verify Decrypts the signature using the public key and compare the result with the hash of the received data.

1.3 Merkle Trees

A complete binary tree of hashes. It is built starting from an initial set of symbols. Also known as a hash tree. Explots a hash function H (SHA1, MD5). Leaves are H applied to the initial symbols. Internal nodes are H applied to the sons of a node.

2 Blockchain Basics

A blockchain is a digitized, decentralized, public ledger of all cryptocurrency transactions. Constantly growing as 'mined' blocks (the most recent transactions) are recorded and added to it in chronological order, it allows market participants to keep track of digital currency transactions without central recordkeeping [1]. The Blockchain in essence is a collection of Blocks. Each Block is composed of the following elements: data, a hash pointer, and a timestamp.

2.1 Hash Pointers

Pointer to where some info is stored. Cryptographic hash of the info. If we have a hash pointer, we can: (1) ask to get the info back, and (2) verify that it hasn't changed. Similar to how programs to download are often accompanied by a hash, to verify that you downloaded the correct file.

2.2 Consensus

The Blockchain network is in principle completely asynchronous and decentralized. In terms of cur-

[1] Merkle tree

rency, the problem to solve is double-spending, i.e. being able to spend the same money more than once.

Nakamoto Consensus Assuming that 1. most nodes are honest, and 2. adding a block is computationally expensive, we can ignore received blockchains which are shorter than ours, for two reasons: 1. it was created later than mine, and 2. it can be out of sync. This assumes that the honest nodes have more CPU power than attacker nodes.

2.3 Proof of Work

Provide a computational puzzle that is hard to solve when you want to add a valid block, but easy to solve when you want to verify that a block is valid. Specifically, the new block must be a combination of a nonce and a header. The nonce is accepted when $H(\texttt{header})$ starts with n zeros. n defines how hard it is to add a new block.

Example Say the hashing algorithm is `SHA256`, and $n = 36$. Each nonce has the probability 2^{-36} of success. The nonce must therefore be brute-forced, but only requires one hash to verify.

3 Blockchain Details

3.1 Transactions

A transaction contains the following data:

- A list of input transactions
- A list of tuples of the recipient public key, and the amount to send.
- Personal signature, signed with private key

In order to verify a transaction, one must:

1. Verify the signature using the public key of the sender
2. Verify the signatures of each of the input transactions
3. Ensure that the money has not been spent between the input transactions and the new transaction.

Since the transaction is signed using the private key, only the owner of the identity can transfer the money from that point. A single user can have an arbitrary amount of identities.

3.2 User Miners

A miner does the following:

1. Verify all the transactions by looking that input transactions are covered and properly signed
2. Compute the Merkle root hash for the transactions
3. Solve the puzzle on the previous block, for immutability
4. Broadcast the new header
5. Go on collecting new transactions for next block

Miners receive compensation for computing the next block. However, it is estimated that we waste $15 million / day on energy to power the miners.

3.3 Smart Contracts

Definition Computer protocols that facilitate, verify or enforce the negotiation or performance of a contract, or that make a contractual clause unnecessary. Define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations, i.e. code is law.

Ethereum Contracts are the main building blocks of Ethereum, the second most popular blockchain. The contract is a program which lives inside the distributed Ethereum network and has its own ether balance of Ether[2], memory and code. Every time you send a transaction to a contract, it executes its code, which can store data, send transactions and interact

[2] Ether is the cryptocurrency / cryptofuel of Ethereum

with other contracts. In order to run the contract, you create a transaction of Ether to the contract, optionally with some input information. The contract runs until it completes or runs out of Ether. The Ether paid is awarded to the winning miner. Each miner runs the smart contract, and produces the same output. The winning miner will publish the block to the rest of the network. Other miners validate that they get the same result. The contracts may be compiled online using the Solidity Compiler[3].

4 Summary

When to use blockchains:

1. Are there multiple parties in the ecosystem?
2. Is establishing trust between all the parties an issue?
3. Is it critical to have a tamper-proof and permanent record of transactions?
4. Are we securing the ownership or management of a finite resource?
5. Does this ecosystem benefit from improved transparency?

[3]https://remix.ethereum.org/