Chapter 1

Introduction

Chapter 2

Mathematical Background

In this chapter we will introduce the mathematical background needed for a clear understanding of the subject we will be discussing. In particular, this means that we will be leading up to introducing elliptic curves. If you are already familiar with elliptic curve, skipping this section should not hinder you in further reading. Those who are unfamiliar, or wish to refresh their memory may read any of the sections below.

Chapter 3

Technical Background

In this chapter we will introduce the technical background needed for a clear understanding of the subject we will be discussing. If you are already familiar with the various terms and concepts, skipping this section should not hinder you in further reading. Those who are unfamiliar, or wish to refresh their memory may read any of the sections below.

The chapter is ordered using a bottom-up approach, starting with an explanation of a consensus protocol, and ending with an explanation of OmniLedger. For each concept discussed, we will try to make it intuitive with an example, either at the beginning or the end of the explanation.

3.1 Consensus protocol

A consensus protocol is any protocol that may be used to reach network wide agreement, consensus, over a decision to add input to the ledger.[2] To clarify the idea of a consensus protocol, we will take a brief look at two different implementations, the protocols used by Stellar and Bitcoin.

3.1.1 Stellar

The Stellar Consensus Protocol (SCP) introduces the notion of a quorum, a set of participants that is sufficient to reach agreement. They further introduce a quorum slice, a subset of a quorum that can convince one participant of the agreement. This requires that the participants trust the others in their quorum slice to behave honestly. In that sense, one might think of the

consensus protocol as one requiring trust.

To ensure that consensus is reached over the whole network, SCP requires that there are no disjunct quorums. Disjunct quorums can reach a different agreement and therefore undermine network wide consensus. Participants wanting to join will have to make sure that they join a quorum in such a way that they do not violate create a disjunct quorum.

Finally we will note that in SCP, each participant wants to be trusted. As such, acting dishonestly is disincentivised, since the participant will be trusted by fewer others.[5]

3.1.2 Bitcoin

The consensus protocol used by Bitcoin completely negates any notion of trust, and instead relies on what is commonly referred to as proof-of-work. In this protocol each participant wishes to create the next set of transactions, which we will refer to as a block, to be added to the ledger, since it holds a reward for the participant that managed to create this set. To be able to add a block, a participant needs to create a hash for the block that has a low enough value. Or, more technical, the hash requires a certain number of leading zeros.

Because a hash function is a one-way function, there is no way to go from a desired output to an input, and as such a participant will have to try various inputs in order to get a desired hash.

Acting dishonestly in this system is disincentivised by the wasted effort. Since the result is easily verified, other participants will reject a false result. The first to create a proper block will reap the reward, the others will be left with nothing but wasted effort.[6]

3.2 Distributed Ledger

A good way to think of a distributed ledger, is to imagine one central ledger, to which any participating party adds every transaction that they created, in a chronological manner.

A distributed ledger may be of interest to any group of parties looking to have no ledger conflicts by having one irrefutable state of truth. In order to understand how a distributed ledger helps with this, we will take a look at the implementation and how it ensures this irrefutable state of truth.

Contrary to the example given, a distributed ledger is not actually one central ledger, but rather a system in which every participant holds and keeps track of his own copy of the ledger. What ensures the irrefutability of the system, is that each participant uses the same set of input and rules to arrive at a conclusion, that must be shared with and agreed upon with the other participants. Upon agreement over an input, each participant then adds this input to his ledger copy, ensuring that each participant holds the same end result.

What is interesting about this technology, is that new participants may join at any time, and will still arrive at the same ledger state. A participant looking to join the network will build the same ledger using the historical set of input and consensus, which ensures that he will arrive at the same ledger state.[8]

As a final note, we will state that a blockchain is an example of a distributed ledger. Since this thesis considers signature schemes in the context of blockchain, we will further reference to it as such. We will also reference to the input as transactions, since the input in a blockchain system is a set of transactions.

3.3 Sharding

In relation to databases, sharding refers to the partitioning of data: dividing the database in distinct, independent parts called shards. Each shard can then be held on a different server and requests for data will be handled by the shard that contains this data. Google uses this principle for their own globally distributed database, Spanner.[3]

In a more generic sense, sharding refers to the partitioning of workload. By dividing a system in various shards, that each handle a subset of the workload, it is no longer needed for the whole system to handle each request, which saves time and resources.

Of course, we are most interested in sharding in the context of blockchain. In a classical blockchain, every participant handles every transaction, even those that have no effect on them. To apply sharding to a blockchain, shards are formed by groups of participants. A transaction will be handled by the shard that includes the participants in the transaction. Since the other shards need not be included in this process, it leaves those resources free to handle other transactions. As such sharding increases the throughput of a system,

assuming transactions remain neatly in one shard.

If a transaction occurs between participants that are in different shards, then a cross-shard transaction has to take place, which requires extra resources. It is important, therefor, that shards be created in such a way that the amount of cross-shard transaction remains limited, otherwise sharding offers no improvement over a general blockchain system.

Furthermore, shards will also have to be picked in such a way that malicious participants are unable to take over shards and corrupt the system by allowing false transactions.[4]

3.4 Discrete Logarithm Problem

On the set of real numbers ($\mathbb{R}$) there exists the $\log_b$ function, which solves the following:

Definition 3.1. Given $a, b, n \in \mathbb{R}$, base b and power a of b, what is n such that $a = b^n$?

This problem also exists modulo p, and we call this the discrete logaritm problem over $\mathbb{Z}/p\mathbb{Z}$.

Definition 3.2 (Discrete Logarithm Problem over $\mathbb{Z}/p\mathbb{Z}$)**.** Given $a, b \in \mathbb{Z}/p\mathbb{Z}$, base b and power a of b, what is n such that $a = b^n \mod p$?

We can even define this problem more generally over a group G.

Definition 3.3 (Discrete Logarithm Problem over group G)**.** Given $a, b \in G$, base b and power a of b, what is n such that $a = b^n$?

And since we can define this problem over a group, we can, in particular, take an elliptic curve E as the group G, which leads us to the following.

Definition 3.4 (Elliptic Curve Discrete Logarithm Problem)**.** Given elliptic curve E and points $P, Q \in E$, what is n such that $nP = Q$?

Although there is currently no proof, the discrete logarithm problem is considered hard, given that the group G was properly chosen, since there are some groups in which this problem is easy to solve.

The problem being hard to solve, means that there exists no efficient algorithm to solve the problem. More specifically, the runtime of the algorithm grows linearly to the group size. Or in other words, exponentially in the amount of digits of the group size.

The hardness of the problem leads us to applications in cryptography. Since the runtime of a solving algorithm grows exponentially, we need simply pick a sufficiently large group so that any attack on the encryption is infeasible.

3.5 Elliptic Curve Cryptography

3.6 Signature Scheme

A signature, in any use, serves to specify that a person holds a responsibility over a document. That is to say, the person has produced the document or agrees with the contents. They then add a signature to the document, so that anybody else can verify that it is the genuine article.

A signature scheme, then, is a way to present and verify the authenticity of digital messages, analogous to a signed document. For this reason, we will simply call the result of the scheme a signature and we can say that a digital message has a signature. It is important to note that a signature scheme includes an easy way to verify the autenticity of a signature, such that forgeries will be detected. In general a signer will sign a message using a scheme and some information only known to him, which we will refer to as the private key. Anyone who does not own the private key of a person, can not act as that person in signing a message.

3.6.1 Multi-signature Scheme

Aside a regular signature scheme, which may be used by a single person to sign a message, we introduce the notion of a multi-signature scheme, which allows for a group of people to sign a message. The principles of a signature scheme apply here as well: the signature is easily verified, and can only created when all private keys are owned.

It is important to note that, although it technically is a multi-signature, we do not consider a list of signatures as such for this thesis. Although easily verifiable, the list will grow linearly with the amount of participants, which

is undesired behaviour. Therefore a multi-signature scheme refers to any scheme that allows to create for signatures of a fixed length, independent of the input size.

3.6.2 Schnorr Group

Before introducing the Schnorr signature scheme, which is a well known (multi-)signature scheme, we will look at the Schnorr group introduced with the scheme.

3.6.3 Schnorr Signature Scheme

3.6.4 CoSi

3.7 RandHound

[7]

3.8 OmniLedger

[4]

Chapter 4

Considered Signature Schemes

[1]

Bibliography

[1] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. 2001, pp. 514–532.

[2] Christian Cachin. *Resilient Consensus Protocols for Blockchains*. 2017. URL: `https://www.ibm.com/blogs/research/2017/10/resilient-consensus-protocols-blockchains/` (visited on 05/09/2018).

[3] James C. Corbett et al. *Spanner: Google's Globally-Distributed Database*. 3. New York, NY, USA, Aug. 2013, 8:1–8:22. (Visited on 04/20/2018).

[4] Eleftherios Kokoris-Kogias et al. *OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding*. 2017. URL: `https://eprint.iacr.org/2017/406` (visited on 04/20/2018).

[5] David Mazières. *On Worldwide Consensus*. 2015. URL: `https://www.stellar.org/papers/stellar-consensus-protocol.pdf` (visited on 05/16/2018).

[6] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: `https://bitcoin.org/bitcoin.pdf` (visited on 05/16/2018).

[7] Ewa Syta et al. *Scalable Bias-Resistant Distributed Randomness*. 2016. URL: `https://eprint.iacr.org/2016/1067` (visited on 04/20/2018).

[8] Mark Walport. *Distributed Ledger Technology: beyond block chain*. 2016. URL: `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf` (visited on 05/02/2019).