Thesis

Paul van Grol

June 1, 2018

Contents

Chapter 1

Introduction

This thesis has been inspired by a recent paper that introduced OmniLedger [10]: a scalable, decentralized ledger. In a world where more and more uses are found for blockchain technology, the promise of such a technology that scales in performance with the amount of users is incredibly interesting. Considering that current systems might actually perform worse when the amount of users increases, it is interesting to look at OmniLedger to see whether it can help in that regard.

In particular, this thesis focuses on multisignature algorithms that may be used to improve the performance of RandHound [18]. RandHound is used by OmniLedger to ensure that the system remains uncompromised. Evaluation has shown that RandHound takes up more than 70% of the total runtime in the bootstrap process of OmniLedger [10]. Since this process happens periodically, improving the performance of RandHound should lead to performance improvement of OmniLedger.

Therefore the research question of this thesis is as follows: " .. ". To answer this question, some background is needed, before comparisons can be made between various multisignature algorithms.

As such the second chapter, Mathematical Background, introduces algebraic varieties, leading up to an explanation of elliptic curves and some properties that are used to great effect in cryptography. The chapter concludes with an explanation of the Weil pairing on elliptic curves, which is used in one of the multisignature algorithms used.

The third chapter, Technical Background, then introduces the technological background of ledgers, blockchain and multisignatures among other terms and concepts used to clearly show the context of the research.

Chapter four, Related Work, discusses related works and contributions made by those.

In chapter five, BLS Multisignature Scheme and six, Schnorr Multisignature Scheme, two different multisignature algorithms are studied, which leads up to a comparison of the two algorithms and a conclusion regarding the most suitable choice for RandHound in chapter seven.

Chapter 2

Mathematical Background

[16, 8] [17, Page 10-15]

2.1 Algebraic Varieties

2.2 Algebraic Curves

2.3 Elliptic Curves

Chapter 3

Technical Background

This chapter provides the technical background to the various terms and concepts used in this thesis. If the reader is already familiar with anything explained in a section in relation to blockchain, skipping it should not hinder in further reading. If the reader is unfamiliar, or wishes to refresh his memory, reading this chapter is advised.

A bottom-up approach is used in this chapter, which means that sections may use or require terms and concepts explained in the previous section(s). Each section aims to give an intuitive understanding via an example and offers a detailed explanation. A visual representation of how the various sections are linked can be found in appendix A.

The chapter first leads up to explaining blockchain, followed by an introduction to signature schemes. To end the chapter, RandHound and finally OmniLedger are explained.

3.1 Concensus protocol

By definition consensus is an opinion that everyone in a group agrees with or accepts. An agreement, in other words. A protocol, in the technical definition, is an established method for connecting computers so that they can exchange information. A consensus protocol is therefore a method to reach agreement between connected computers.

In relation to blockchain technology, a consensus protocol [5] is a protocol used to reach agreement over input to add to the ledger, and the order in which it should be added. Blockchain technology recognises various such protocols, operating in various ways under different assumptions and requirements.

Below brief explanations will be given of the consensus protocols used by

Bitcoin and Stellar. The former uses a concept known as proof-of-work, the latter introduces a notion of trust.

3.1.1 Bitcoin

Conversely, the consensus protocol used by Bitcoin [13] does not require any trust in other participants. Instead it relies on a mechanism known as proof-of-work. In Bitcoin a reward goes to the participant that manages to create the next agreement. This agreement is a set of transactions, called a block, that fulfills the requirements set for the block. Once a participant has found such a block, he sends it to as many others as he can, so that his result will be on the chain, which makes the reward his. Other participant can easily verify that a block meets those requirements, after which they accept the result and forward it to others.

The protocol functions with the help of a one-way hash function. A one-way function is a function that makes it easy to compute $f(x)$ given x, but makes calculating x from $f(x)$ practically impossible. A one-way hash function has the added property of producing output of a fixed length.

What happens in the protocol used by Bitcoin is that each participant will create a block. This block then serves as input for the one-way hash function. The goal is to have a block that results in a hash with a certain amount of leading zero's. Given the block, others can easily verify that hashing the block results in a hash with a sufficient amount of leading zero's.

Producing a wrong block, therefore, is not in the best interest of the participant. Others will quickly refure the block, and all the effort put into creating the block will be wasted.

Therefore the notion of proof-of-work for the Bitcoin consensus protocol is apt. Arriving at a sufficient result requires the participant to try various input blocks, with no way to compute one from the desired result.

3.1.2 Stellar

To understand the Stellar Consensus Protocol [12] (SCP) some definitions are needed. SCP introduces a quorum, a set of participants that is sufficient to reach agreement, and a quorum slice, a subset of a quorum that can convince one participant of the agreement.

The basis of SCP is that participant trust others in their quorum slice to behave honestly, hence the notion of trust in SCP. Assuming all participant behave honestly and under the same set of input and rules, each participant should arrive at the same conclusion. After a participant reaches his conclusion, he publishes it to the others in his quorum slice. In publishing his conclusion he votes for it, as do all others in his slice. The final conclusion is the one most voted for.

As long as there are sufficient reputable participants, the network can never be corrupted. Behaving dishonestly is further disincentivised by the protocol, since those participants are not trusted to be part of a quorum slice, and the influence of their opinion will then quickly be reduced. Conversely, reputable participants will be part of multiple quorum slices, which gives them a large influence over the decisions made.

To ensure that the whole network is indeed connected and reaches network wide consensus, SCP requires that there are no disjunct quorums. Since disjunct quorum can reach their opinion separate from the network, a different agreement undermines the network wide consensus. Participants wishing to join the network have to make sure that they join a quorum in such a way that no disjunct quorums are created.

3.2 Distributed Ledger

By definition, a distributed ledger [19] (DL) would be shared records. This remains true in the context of blockchain technology, but that is not all that it is. Although in use it acts as if there is one ledger that everybody uses, so that everybody agrees what has happened, the technique behind it works differently.

The most important property of a DL in the context of blockchain is the irrefutable state of truth it represents. Anything on the distributed ledger has happened exactly as described there, and no-one can alter those existing records. A DL may therefore be of interest to any group of parties looking to abolish ledger conflicts, to be able to trust that anything in the ledger has indeed happened, nothing more, nothing less, nothing else.

In a DL system, each participant holds and keeps track of his own copy of the ledger. This happens independently, so ledger states are never directly compared. The mechanism used by the sytem ensures that each participants holds the exact same conclusion on his ledger, thereby assuring the irrefutable state of truth.

What happens in the system, is that each participant starts at the same basis: an empty ledger. Input is then presented to all participants, who will handle the input as the system describes. Having handled the input and produced the result, a consensus protocol is used to ensure network wide agreement. Once agreement has been reached, each participant adds input to his ledger as agreed upon, ensuring that each participant reaches the same next ledger state.

Since it may happen that participants join at a later time, mechanisms exist to ensure that the participant can catch-up to the current ledger. This is not done by presenting the ledger, but instead by presenting the input for each ledger state, and corresponding consensus. The new participant then uses that information to construct his ledger from the basis state up, eventually reaching the same ledger state as all other participants.

Blockchain is a form of a DL, using blocks of transactions to shape the ledger and various consensus protocols to agree upon blocks to be added. Since this thesis focusses on techniques used in blockchain, any references to a chain, ledger or anything similar will mean a blockchain, a specific form of a DL.

3.3 Sharding

Sharding is a concept mostly encountered in relation to databases. [9] In that context, it refers to splitting up a large set of data in smaller pieces, shards. These shards can then be spread across distinct containers, be it a table, scheme or even different physical databases. Sharding databases can drastically improve performance and is a well-proven technique for large data sets. Google, for example, uses it for their own globally distributed database, Spanner [6].

In a more generic sense sharding refers to a partitioning of workload. By dividing a system into various parts, that each handle a particular subset of the workload, more work can be done simultaniously, which improves throughput and consequently performance. At the same time, it also means that the amount of resources requires remains limited, since it is not the whole system that has to act, but just a part of it.

Looking at sharding in blockchain [10], it is important to note that in traditional blockchain technologies, each participant handles each input. That means that in a particularly large network, each participant will have to handle a large set of transactions, which uses time and resources.

Sharding in a blockchain means dividing your participants into shards [10]. Transactions will be handled by the shard or shards to which the participants in that particular transaction belong. A transaction that takes place between participants in different shards is known as a cross-shard transaction and requires a different method to handle than transactions happening whithin a shard. Assuming that there are not too many cross-shard transactions, sharding results in a higher throughput of transactions, since each group handles a subset of the transactions simultaniously.

Since blockchain requires that malicious participants cannot influence the result, is is important to note that a shard is somewhat of a mini blockchain, which means that sharding must happen in such a way that shards cannot be compromised.

3.4 Discrete Logarithm Problem

On the set of real numbers ($\mathbb{R}$) the $\log_b$ function has been defined as the solution to the following problem.

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 same problem can also be defined over modulo p, which is known as the discrete logarithm 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 $a = b^n \mod p$?

In a more generic sense, this problem can be defined 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 $a = b^n$?

In particular, since an elliptic curve E is a group, the problem holds over elliptic curves. Taking elliptic curve E as group G, it follows that.

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 no actual proof exists, the assumption is that the discrete logarithm problem in a well chosen group G is a hard problem. The group must be well chosen, for there are groups that have a structure that allows for an algorithm to solve the problem, but there is a sufficiently large group of groups left for which no such algorithm exists.

In the context of computer science, this means that there is no known efficient algorithm to solve the problem, other than trying various solutions, quite like the mechanic used in the Bitcoin consensus protocol (section 1). More specifically, the problem being hard means that the runtime of the solution finding algorithm grows linearly to the group size. Or in other words, exponentially in the amount of digits of the group size.

Problems that are hard to solve in this sense of the word, lead to applications in cryptography. Since we can ensure a group large enough that trying all solutions becomes infeasible, it allows for cryptographic security.

3.5 Random Oracle

The random oracle model [7] was introduced by Fiat and Shamir. It is a theoretical black box that, for each unique input, presents a random output from its output domain. It is important to note that the oracle is deterministic; each unique input will produce the same output every time.

3.6 Forking Lemma

The forking lemma [2] gives a relation between the chance of a fork and the chance of success for the attacker. Bellare and Neven state that.

$$frk \geq acc \cdot (\frac{acc}{q} - \frac{1}{h}) \tag{3.1}$$

Here frk represents the chance of obtaining two good forgeries, and acc the probability of success of an adversary on a random input. The lemma shows that frk is non-neglible if acc is non-neglible. In other words, if the underlying problem is hard, then no adversary can forge signatures.

3.7 Signature Scheme

A signature is used in all manner of situations. An artist might add his signature to his creation to show that he was in fact the one to create it. People sign documents to show that they have created them, or to signify that they agree with the content. Because signatures are unique to a person and therefore difficult to recreate by others, a signature gives truth, validity to the signed piece. It can also be used to verify that someone is who they say they are. If they can produce the same signature, they are likely the person in the flesh.

Digital signatures serve the exact same purpose. It is a proof of identity, genuineness. A signature scheme is a way to present and verify the authenticity of a digital message. Signature schemes work with a pair of a private and public. The private key is some piece of information that is only known to the signer, the public key is made public and used in signature verification. The signer uses this private key to sign a message, which produces a signature. This signature can be verified by others using the public key and the signare algorithm.

In further reading, a signature will refer to a digital signature produced by a signature scheme and any digital message with such a signature has been signed.

A classical signature is one that proves the identity of just one person. This is, however, not the only kind. Among others, there are group- and multisignatures, which are discussed below. A group signature scheme allows a signer to sign on behalf of a group, whereas a multisignature scheme allows a group to sign a document, with the result begin a joint signature that is more compact than all individual signatures separately.

3.7.1 Group Signature Scheme

As explained just before, a group signature scheme allows a user to sign a message on behalf of a group. Various implementations of group signature schemes each offer different behaviour. It is possible to have a group signature be created in such a way that the signer may be determined, but also so that the signer remains obfuscated. A scheme may offer restrictions on the signature, so that it may only be produced if sufficient signers participate in the signing. In any case, a group signature implies that the group, as a whole, agrees with the content of the message signed.

3.7.2 Multisignature Scheme

A multisignature scheme is somewhat of a generalisation of a group signature scheme. A multisignature scheme will produce a joint signature for the group of signers. It finds particular use in blockchain technology because it limits the size of the signature, and therefore the size of the input that has to be sent over the network.

3.7.3 Schnorr Group

A Schnorr group [14], proposed by Claus P. Schnorr, inventor of the Schorr Signature Scheme, is defined as follows.

Definition 3.5 (Schnorr Group)**.** Generate p, q, r such that $p = qr + 1$ with p, q prime. Then pick any $1 < h < p$ such that $h^r \not\equiv 1 \mod p$. Then $g = h^r \mod p$ is the generator of the Schnorr group, which is a subgroup of $\mathbb{Z}_p^*$ of order q

Proof 3.6 (Schnorr Group)**.** It is trivial to see that the Schnorr group is indeed a group. Note that the order of $\mathbb{Z}_p^*$ is $p - 1 = qr$. Because $\mathbb{Z}_p^*$ is cyclic, for each divisor d of qr there is one subgroup of order d, generated by $a^{n/d}$, with $a \in \mathbb{Z}_p^*$. As such there is a subgroup of order q generated by $a^{qr/q}$, which is precisely the h picked. As such the order of the Schnorr group is indeed q.

For puruposes in cryptography, p is typically 1024 to 3072 bits and q 160 to 256 bits, which means that the discrete logarithm problem is sufficiently hard to solve for both.

3.7.4 Schnorr Signature Scheme

The Schnorr signature scheme is considered a simple and efficient signature scheme that sees widespread use. The algorithm works as follows.

First all users agree on a group G of prime order q with generator g. This group is typically a Schnorr group, but it is not required. They also agree on a hash function $H : \{0,1\}^* \to \mathbb{Z}_q^*$.

With these prerequisites each user can choose a private key $x \in \mathbb{Z}_q^*$ which leads to the corresponding public key $y = g^x$.

Signing a message M goes as follows. $r = g^k, k \in \mathbb{Z}_q^*$. Then $e = H(r \parallel M)$, with $\parallel$ representing concatenation and $r, M \in \{0,1\}^*$, the bitstring representations. Finally $s = k - xe$ leading to the signature (s, e) with $(s, e) \in \mathbb{Z}_p$.

Verification of the message is done by determining $r_v = g^s y^e$ and $e_v = H(r_v \parallel M)$. The signature is genuine if $e_v = e$.

To prove that the signature is indeed genuine if $e_v = e$, consider that $s = k - xe$. It follows that $r_v = g^s y^e = g^{k-xe} g^{xe} = g^k = r$. As such $e_v = H(r_v \parallel M) = H(r \parallel M) = e$.

Because G, g, q, y, s, e, r are all public and k, x private, it is easy to see that anyone can verify, but only the owner of k and x can have signed the message.

It is important to note that the security of the signature scheme relies on the security of the hash algorithm used. Fiat and Shamir have argued [7] that the algorithm is secure if H is modeled as a random oracle.

Seuring went on to show [15] that proof of security with the Forking lemma is the best possible result for the Schnorr signature scheme (among others).

The actual securiy proof is not discussed in this thesis, for it would broaden the scope too much.

3.8 RandHound

RandHound [18] is a protocol to create public, verifiable and unbiasable randomness. The protocol is modeled as a client/server model, wherein the client requests a random value, that is generated by a set of RandHound servers. Each server in the set will generate his own secret, which is then shared with the client via a share that proves that the server has generated a secret according to the rules set, but that does not share the secret itself. After a certain time, the client commits to a certain set of secrets to use to create the random value. Having committed to the set, the client asks the servers to co-sign his choice. The servers then acknowledge the commitment, verify that it is genuine and respond with a Schnorr response. The client receives these responses and creates the aggregate Schnorr response, which he then presents to the servers in order to receive the actual secret. Each server then responds with his actual secret value, which the client finally combines to create the complete random value.

3.9 OmniLedger

OmniLedger [10] is presented as a scalable distributed ledger. It uses sharding to divide the workload over participants. OmniLedger uses RandHound to divide the participants over shards, so that the chance of any shard being compromised is negligible. Moreover, shards are periodically reformed, to further reduce the chance that they become compromised.

Chapter 4

Related Work

[**WeilShortSignature**, 1]

Chapter 5

Boneh-Lynn-Shacham Multisignature Scheme

5.1 Boneh-Lynn-Shacham Signature Scheme

The Boneh-Lynn-Shacham [4] (BLS) signature scheme, is a signature scheme utilising the Weil pairing to create signatures. The scheme uses the following.

- A bilineair pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ which is efficiently computable and non-degenerate. The three groups have prime order q and the generators of $\mathbb{G}_0$ and $\mathbb{G}_1$ are g_0 and g_1 respectively.
- A hash function $H_0 : M \to \mathbb{G}_0$ to hash the message M.

The scheme then has three phases. In the first phase the key is generated, in the second the message is signed and in the final phase the signature is verified.

- To generate the key, $\alpha \in \mathbb{Z}_q$ is picked randomly and then used to create $h = g_1^\alpha \in \mathbb{G}_1$. Here α is the private key, and h the public key.
- The message m is then signed by creating $\sigma = H_0(m)^\alpha \in \mathbb{G}_0$. The result is therefore a single group element.
- The signature is verified if $e(g_1, \sigma) = e(h, H_0(m))$ holds, otherwise the signature is a forgery.

To see that the verification holds, note that e is bilinear, and as such $e(g^x, g^y) = e(g, g)^{xy}$. Therefore it follows that $e(g_1, \sigma) = (g_1, H_0(m)^\alpha) = (g_1, H_0(m))^\alpha = e(h, H_0(m))$ and the signature is verified.

5.2 Algorithm

Boneh, Drijvers and Neven noted [3] that the intuitive multisignature algorithm using BLS, which aggregates signatures $\sigma_1, \ldots, \sigma_n$ by computing $\sigma = \sigma_1 \ldots \sigma_n$ can be easily attacked. An attacker can register his public key h_2 by using h_1 of another user, called Alice such that $h_2 = g_1^\beta \cdot h_1^{-1}$, where the attacker picks $\beta \in \mathbb{Z}_q$ himself.

The attacker can now present the signature $\sigma = H_0(m)^\beta$ for some message m and claim that it was signed by both him and Alice since it follows that $e(g_1, \sigma) = e(g_1, H_0(m)^\beta) = e(g_1^\beta, H_0(m)) = e(h_1 \cdot h_2, H_0(m))$, and that therefore the signature is genuine.

The BLS multisignature scheme that is resistant against this type of attack works slightly different, but uses the same principles. As added prerequisite, the BLS multisignature scheme requires a second hash function $H_1 : \mathbb{G}_1^n \to R^n$ with $R^n = \{2^0, \ldots, x^1 28\}$.

Key generation and message signing remain the same as in single use BLS, but the aggregation scheme works differently.

- The function H_1 is used to calculate $(t_1, \ldots, t_n) = H_1(h_1, \ldots, h_n) \in R^n$. These values are used to create the aggregate signature $\sigma = \sigma_1^{t_1} \ldots \sigma_n^{t_n} \in \mathbb{G}_0$.
- The signature σ is verified by computing $(t_1, \ldots, t_n) = H_1(h_1, \ldots, h_n) \in R^n$ and $h = h_1^{t_1} \ldots h_n^{t_n} \in \mathbb{G}_1$. The signature is genuine if $e(g_1, \sigma) = e(h, H_0(m))$.

5.3 Security

5.4 Performance

Chapter 6

Schnorr Multisignature Scheme

6.1 Algorithm

[11]

6.2 Security

6.3 Performance

Bibliography

[1] Alex Edward Aftuck. *The Weil Pairing on Elliptic Curves and Its Cryptographic Applications.* 2011. URL: `https://digitalcommons.unf.edu/etd/139/` (visited on 04/28/2018).

[2] Mihir Bellare and Gregory Neven. "Multi-signatures in the Plain public-Key Model and a General Forking Lemma". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security.* CCS '06. ACM, 2006, pp. 390–399.

[3] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. URL: `https://crypto.stanford.edu/~dabo/pubs/papers/BLSmultisig.html` (visited on 05/01/2018).

[4] 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.

[5] 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).

[6] James C. Corbett et al. "Spanner: Google's Globally-Distributed Database". In: *ACM Trans. Comput. Syst.* 31.3 (Aug. 2013), 8:1–8:22.

[7] A. Fiat and A. Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Advances in Cryptology: CRYPTO '86.* 1987, pp. 186–194.

[8] Misha Kapovich. *The Riemann-Roch Theorem.* 2007. URL: `https://www.math.ucdavis.edu/~kapovich/RS/RiemannRoch.pdf` (visited on 04/25/2018).

[9] Craig Kerstiens. *Database sharding explained in plain English.* 2018. URL: `https://www.citusdata.com/blog/2018/01/10/sharding-in-plain-english/` (visited on 05/16/2018).

[10] 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).

[11] Gregory Maxwell et al. *Simple Schnorr Multi-Signatures with Applications to Bitcoin.* 2018. URL: `https://eprint.iacr.org/2018/068` (visited on 04/30/2018).

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

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

[14] Claus P. Schnorr. *Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system.* 1989. URL: `https://patents.google.com/patent/US4995082` (visited on 05/02/2018).

[15] Yannick Seurin. *On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model.* 2012. URL: `https://eprint.iacr.org/2012/029` (visited on 05/02/2018).

[16] Joseph H. Silverman. *An Introduction to the Theory of Elliptic Curves.* 2006. URL: `https://www.math.brown.edu/~jhs/Presentations/WyomingEllipticCurve.pdf` (visited on 04/27/2018).

[17] Joseph H. Silverman. *The Arithmetic of Elliptic Curves.* Springer, New York, NY, 2009.

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

[19] 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).