Research Plan – Proof of Proximity Using a Verifiable Delay Function

Jani Anttonen

March 30, 2020

1 Introduction

Verifiable delay functions are the main focus of this thesis. By definition, a VDF is an algorithm that requires a specified number of sequential steps to evaluate, but produces a unique output that can be efficiently and publicly verified.[1] Verifiable delay functions are a relatively new subject, with first use of the term being in 2018. How they have come to be, I will explain in a few paragraphs next.

Peer-to-peer networking means that a client in a network also functions as a server. These kinds of networks require dynamic ways of finding other peers, since peers are required to connect to each other, often with a single hop with no relayed or tunneled connections. A single peer keeps information of the peers it has connected with, and shares this info with other peers on the network. Since P2P networks are just a definition for a specific kind of distributed network, they benefit from research that's done in distributed networks and computing. A significant driver for research lately has been blockchains.

The term blockchain was first coined by a person or a group under the name of Satoshi Nakamoto in 2008 in the Bitcoin whitepaper, which described a security model for a digital currency based on the data structure of a distributed sequential ledger.[2] In addition to requiring a way for peers to connect to each other without prior knowledge, the whitepaper describes a way to append new blocks to the chain by "mining", repeatedly trying to guess a correct answer to a hash puzzle. The network incentivizes the "miners" to do this guessing game as fast as they can, which has resulted in a race for computing resources to win the most Bitcoin. This race, and the consensus algorithm itself, Proof of Work, has shown to be a highly inefficient use of power and resources, which has spawned efforts to create a more economic way of reaching consensus in public blockchains.

Private or permissioned blockchains don't require as much power, since they rely on trust between nodes and their operators to act on good terms. This does not mean that they can work without a consensus algorithm at all, rather the consensus algorithm in this case is more focused on getting things in order and to make sure all participants have the latest info at all times. Public blockchains need to make attacks harder to perform, thus creating more pressure on the algorithm.

The most well-known alternative to Proof of Work is Proof of Stake. Proof of Stake was first introduced in the cryptocurrency project Peercoin in 2012, but there have been many efforts since to make Proof of Stake better by introducing beneficial safety properties, like byzantine fault tolerance and better validator elections. Ethereum is currently the second largest cryptocurrency by market cap, and has been doing open research towards a goal of extending its blockchain capabilities by converting from a Proof of Work to a Proof of Stake based system. Proof of Stake is different from Proof of Work in the way that it does not have mining, but a single block creator at once with a number of block validators that vote for the correctness of the blocks that the block creator has created. In order to create blocks, the peer needs to lock away a part of their funds in the network, risking losing a part of it via a mechanism called slashing in case the voters deem any blocks the block creator creates false. When a vote passes, the block will soon achieve finality, meaning it will reside in the blockchain for foreseeable future if the state is appended and no so-called forks happen in the network that continue with a block prior to the one in question.

To attack a Proof of Stake system, all the attacker needs to do is to have a way of influencing the election that decides the voters. Since it's randomized, there needs to be a pre-image resistant source of randomness. One way to create pre-image resistance is by repeated hashing, similarly to how a blockchain is constructed, the next block always being dependent on the information of the ones before it. One way to create this source of randomness is a verifiable delay function. Verifiable delay functions can be used for anything that requires a waiting period. I'm testing out a new use case in this thesis.

2 Goals

My goal with this thesis is to introduce a better way of constructing distributed hash tables for the routing of P2P networks with a novel algorithm constructed using prior knowledge from VDF research and open source code. I aim to achieve this task by constructing an algorithm that measures latency and difference in computational resources between two peers. This algorithm is meant to be hard to compute but fast and easy to publicly verify, borrowing these two traits from VDFs in general.

Overarching research questions regarding this research are if I can construct a valid mathematical proof of latency and computation resources, and if it can be applied to P2P network routing that results in a more performant network with less network hops between continents and unrelated peers. Another goal is to inspire ideas for further research of verifiable delay functions and my algorithm, finding new applications for it other than my proposed one.

3 Materials and methods

I have already gathered a stack of material on the subject, with the most important reference points being the individual whitepapers of blockchain networks, and the original two plus subsequent research papers by Wesolowski and Pietrzak describing verifiable delay functions. Material is mostly going to be in the form of research papers and articles. There are a few good explainers on verifiable delay functions in the form of blog posts that I'm going to be referring to. Unfortunately some new material in the field of blockchains is not peer reviewed and can resemble an advertisement more than a research paper, but on the other hand, verifiable delay functions have seen real academic research.

Most of the material used is public and with free access, and this also holds true with the programming tools used. All the code that I'm going to use along the way will be open-source, licenced with non-restrictive licences like MIT. The research I'm doing for this thesis requires no non-disclosure agreements to be signed, or any other contracts or permissions to be requested.

During the research I may consult Simon Peffers and Kelly Olson from Supranational, or any other VDF research related people on my progress, as they were present in the process of narrowing down the research question of this thesis. This said, I'm going to be doing the research and programming myself without external help on the execution.

Later on in the process I might want to try out formal methods, like TLA+, to prove that my proof is sound mathematically.

4 Timeline

Work on this thesis started in February 2020. I've spent the month of March studying verifiable delay functions and gathering existing research.

During next month, I aim to formalize and program a working prototype of my algorithm in Rust. To balance my workload I'm also planning to write some introductory parts of my thesis during April as well, mostly focusing on existing work.

In the following months, May and June, the plan is to make some code improvements to the algorithm, and write the thesis' parts considering my own research and finalize any introductory parts that need editing or additions.

In July, I plan to make revisions of the work suitable for review, and make any corrections needed to the thesis. If there's any considerable time left, I might study the subject further will it fit my thesis or not.

I plan to be finished by the end of summer to early fall.

References

[1] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. Technical Report 601, 2018.

[2] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. page 9.