UNIVERSITY OF CAMBRIDGE

Decentralised computer systems

Mansoor Anwar Ahmed

University of Cambridge
Department of Computer Science and Technology
Queens' College

January 2021

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Decentralised computer systems

Mansoor Anwar Ahmed

Summary

The architecture of the Web was designed to enable decentralised exchange of information. Early architects envisioned an egalitarian yet organic society thriving in cyberspace. The reality of the Web today, unfortunately, does not bear out these visions: information networks have repeatedly shown a tendency towards consolidation and centralisation with the current Web split between a handful of large corporations.

The advent of Bitcoin and successor blockchain networks re-ignited interest in developing alternatives to the centralised Web and paving a way back to the earlier architectural visions for the Web. This has led to immense hype around these technologies with the cryptocurrency market valued at several hundred billions of dollars at the time of writing. With great hype, apparently, come great scams. I start off by analysing the use of Bitcoin as an enabler for crime and then present both technical solutions as well as policy recommendations to mitigate the harm these crimes cause.

These policy recommendations then lead us on to look more closely at cryptocurrency's tamer cousin: permissioned blockchains. These systems, while less revolutionary in their premise, nevertheless aim to provide sweeping improvements in the efficiency and transparency of existing enterprise systems. To see whether they work in practice, I present the results of my work in delivering a production permissioned blockchain system to real users. This involves comparing several permissioned blockchain systems, exploring their deficiencies and developing solutions for the most egregious of those.

Lastly, I do a deep dive into one of the most persistent technical issues with permissioned blockchains, and decentralised networks in general: the lack of scalability in their consensus mechanisms. I present two novel consensus algorithms that aim to improve upon the state of the art in several ways. The first is designed to enable existing permissioned blockchain networks to scale to thousands of nodes. The second presents an entirely new way of building decentralised consensus systems utilising a trie-based data structure at its core as opposed to the usual linear ledgers used in current systems.

"The roots below the earth claim no rewards for making the branches fruitful."

—Rabindranath Tagore, Stray Birds

Acknowledgements

A lot of people have directly and indirectly contributed to the completion of this thesis. I would like to start by thanking my supervisor, Ross Anderson, for giving me free reign over the research agenda and for pushing me to look beyond the traditional confines of computer science and see the broader context in which computer systems operate. I would also like to thank my examiners, Rainer Böhme and Jon Crowcroft, for their helpful feedback and proactive engagement with this thesis. I would like to acknowledge nCipher Security and TODAQ for generously funding portions of my PhD.

Thanks are due to Ilia Shumailov, my fellow PhD and frequent co-author, for helping me convert chats around the coffee table into full-fledged papers. I would also like to thank Alastair Beresford and the Digital Technology Group for supporting me throughout the PhD and for helping me find an academic community I could feel a part of. I am grateful to the founding engineering team at Jitsuin (Robin Bryce, Jon Geater, John Hartley and Waldemar Parzonka) for trusting a naïve academic like me with building a production system and helping me understand real-world software engineering. I owe a lot to Dann Toliver who has not only been a co-author but also a mentor and a friend; our weekly calls were the highlight of my work over these three years.

Outside of the lab, there are more people to thank than the word limit allows me to mention. Starting off with the fantastic people I met at Queens' college: I am fortunate to consider Andrea Wessendorf a friend; she has been a patient listener and a source of benevolent peer pressure throughout this thesis. Another gem of a person that I've had the pleasure of knowing is Jasper Ma who has the distinction of not just being the best man but also the "minister" at my wedding. Elizabeth Weir has helped more than she knows, sharing many a warm cup over many a dreary day. I would be remiss if I didn't mention Apolline Blandin, Dante McGrath, Francesca van Tartwijk, Joe Stallard and Jolly Dusabe who have, in their own way, brought me joy.

Next on the list are my fellow writers from the Creative Writing Sessions. Rhian Holvey, Maria Bolson, Grace Rosemin, Matt Cooper, Emily Sandford, Sophia Cameron-Christie and the rest of the group have been a constant source of camaraderie and support. No matter how difficult a given week might have been, our writing sessions and meandering conversations never failed to bring back some light.

My parents, Asma and Anwar Ahmed, have always pushed me towards academic excellence and deciding to pursue this thesis is no doubt a direct consequence of that. My sister, Sarah Ahmed, has always been the invisible support structure that kept me safe even when I didn't realise it. Lastly, words cannot express my gratitude to whatever vagaries of the universe led to me meeting my wife, Emma. She is my family, my inspiration and my home. This thesis is dedicated to her and my sister.

Contents

"Isn't it nice to think that tomorrow is a new day with no mistakes in it yet?"

—Lucy Maud Montgomery, Anne of Green Gables

1 Preface

"Decentralised by design" has been a guiding principle for the design of numerous popular systems. FreeNet, BitTorrent and, indeed, the World Wide Web were all designed to enable the formation of a federated network of collaborating parties.

Yet, despite these efforts, it appears that centralised systems have won decisively. The vast majority of Internet traffic flows through a select few data centres. For a substantial part of humanity, their entire computing experience comprises of centralised services hosted by technology giants with market caps larger than most countries' GDPs.

The advent of Bitcoin in 2008 with its demonstrated ability to create a trustworthy system out of strangers on the Internet led some to think of Bitcoin and subsequent blockchain systems as a viable means to re-decentralise the Web. We will consider both the visions of the early architects of the Web as well as the hope for blockchain systems in chapter 2 which sets the context for my research.

A lot has changed in Bitcoin-land since 2008, however. Bitcoin has gone from being a cypherpunk hobby to a mainstream investment avenue with the network valued at several hundred billion dollars. This has also led to a creeping centralisation: the entire network is in practice run by a handful of so-called mining pools. Moreover, the killer app for Bitcoin seems not to be as a legitimate currency but rather speculative investment and money laundering. We look at the current state of the Bitcoin world and analyse the key properties of Bitcoin that enable some forms of crime in chapter 3. Then, we turn to legal precedent to find a way to rein in the harm caused by these Bitcoin-enabled crimes. We propose and build a system for tracing stolen and otherwise "tainted" bitcoins in a way that is compatible with the Common Law.

The publication of this system got us in touch with victims of said crimes and, in

turn, with regulators seeking to control the damage. This sets the stage for a discussion about the interplay between existing legal systems and cryptocurrency networks. We present several recommendations for regulators, including the suggestion that the true potential of blockchain networks may lie not in cryptocurrencies but rather in permissioned blockchains.

After that glimpse into the hypothetical, we turn our attention to real-world engineering when we talk about permissioned blockchains in chapter 4. To look past the immense hype around the potential of these technologies and understand the realities of using them, I pursued an industrial collaboration to develop a production permissioned blockchain system. I present the lessons learned during the process here. This includes a survey of existing permissioned blockchain frameworks with an analysis of their deficiencies.

I then highlight difficulties I've faced when working with permissioned blockchains in a real deployment. This leads to a couple of novel techniques that mitigate storage requirements of blockchains and help with compliance with laws such as GDPR that require redaction of data. The most pernicious problem of all in my experience is the limited scalability of the consensus algorithms used in existing frameworks.

I discuss these consensus algorithms in chapter 5 and present *Robust Round Robin*, a novel blockchain consensus algorithm that works well in both permissioned and permissionless settings and aims to achieve better scalability (in terms of both network size and throughput) than currently deployed permissioned systems.

However, pursuing scalability further, I realised that the core limitation to building large networks is the nature of the blockchain itself. Transmitting all data to all the nodes is inefficient and introduces inescapable latency. So, in chapter 6, I present *Cambium*, a consensus algorithm based not on a blockchain but upon a novel trie-based data structure. It promises even better scalability characteristics compared to Robust Round Robin though its transaction payload is less functional than a traditional blockchain transaction.

Finally, in chapter 7, I summarise the contributions presented in the thesis.

1.1 Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text.

It is not substantially the same as any work that has already been submitted before for any degree or other qualification except as declared in the preface and specified in the text.

It does not exceed the prescribed word limit for the Computer Science Degree Committee.

1.2 Contributions

During the course of this PhD, I was able to work on the following projects. I state my exact contribution to an individual project if its contents are used in this thesis.

1.2.1 Textbook

I was approached by publishers after my talk at the Open Source Summit Europe (§ 1.2.5) to see if there was a market for a textbook in the domain of permissioned blockchains. I believed there was and wrote a proposal which was reviewed by 16 anonymous reviewers. *Decentralised Enterprise Applications using DLTs* (working title) is the result of two years of work that followed. Introductory portions of the textbook are used in chapter 4. The textbook is scheduled to be out in mid-2021.

1.2.2 Academic papers

1. **Making Bitcoin Legal**, Ross Anderson, Ilia Shumailov and Mansoor Ahmed, Security Protocols Workshop (SPW) 2018 [17]

2. **Bitcoin Redux**, Ross Anderson, Ilia Shumailov, Mansoor Ahmed and Alessandro Rietmann, Workshop on Economics of Information Security (WEIS) 2018 [19]

3. **Tendrils of Crime: Visualizing the Diffusion of Stolen Bitcoins**, Mansoor Ahmed, Ross Anderson and Ilia Shumailov, Graphical Models For Security (GraMSec) at the Federated Logic Conference (FLoC) 2018 [2]

4. **Snitches Get Stitches: On the Difficulty of Whistleblowing**, Mansoor Ahmed-Rengers[1], Ross Anderson, Darija Halatova and Ilia Shumailov, Security Protocols Workshop (SPW) 2019 (Best Presentation Award) [8]

5. **Short Term Firm-Specific Stock Forecasting with BDI Framework**, Mansoor Ahmed-Rengers, Sanjay Singh and Anirudh Sriram, Journal of Computational Economics 2019 [3]

6. **FrameProv: Towards End-to-End Video Provenance**, Mansoor Ahmed-Rengers, New Security Paradigms Workshop (NSPW) 2019 [5]

7. **Don't Mine, Wait in Line: Fair and Efficient Blockchain Consensus with Robust Round Robin**, Mansoor Ahmed-Rengers and Kari Kostiainen, preparing for submission [9]

8. **Cambium: Growing Consensus Logarithmically**, Mansoor Ahmed-Rengers and D. R. Toliver, preparing for submission

[1]Since August 2019, I've used **Mansoor Ahmed-Rengers** as my name professionally. Unfortunately, antiquated bureaucratic traditions force me to use **Mansoor Ahmed** in certain settings, which is why some of my publications have a different name (including this thesis).

9. **Towards Integrity Checks in the Smart Home with Physical Home Endorsers**, Kaushal Kafle, Kirti Jagtap, Mansoor Ahmed-Rengers, Adwait Nadkarni, and Trent Jaeger, under review

10. **CoverDrop: Whistleblowing in a Listening World**, Mansoor Ahmed-Rengers, Diana Vasile, Daniel Hugenroth, Alastair Beresford and Ross Anderson, under review

11. **Democracy on the Margins of the Market: A Critical Look Into the Privatisation of Cyber Norm Formation**, Emma Ahmed-Rengers and Mansoor Ahmed-Rengers, The Hague Program for Cyber Norms, The Hague, 2020 [4]

Of the above, papers 1, 2 and 3 form the basis for chapter 3. The contributions for those papers were as follows: Ross Anderson came up with the initial idea of applying the *nemo dat* principle to cryptocurrencies and took the lead on the first two papers. Ilia Shumailov wrote the Rust parser used in papers 1 and 2; he also played a crucial role in our discussions that led to paper 3 and contributed to the writing of papers 1 and 2. I wrote the visualisation tool which forms the core of paper 3, did the legal analysis of the EU's directives which formed two sections in paper 2 and provided the technical expertise for applying the legal principles to both Bitcoin and Ethereum used in all three papers. Lastly, I took the lead on paper 3 and its presentation at FLoC.

Papers 7 and 8 form the basis for chapters 5 and 6. Both the protocols were designed collaboratively. The protocol presented in paper 7 was initiated by me and then was designed over several meetings with Kari Kostiainen. Kari also contributed three sections in the paper; I wrote the remainder of the paper and did the literature review, security analysis and performance analysis. Paper 8 has not been published yet. The protocol described there was designed collaboratively with Dann Toliver over the period of two years. Dann wrote the visualisation code and wrote the introductory sections in the paper. I performed the performance analysis and the security analysis. I also wrote the code for permutation used in the protocol and the rest of the paper.

The rest of the papers were side projects and aren't used in this thesis.

1.2.3 Patents

1. **GB1916295.7 - Data structure storage optimisation**, Mansoor Ahmed-Rengers and Jon Geater

2. **GB1916291.6 - Data block modification**, Jon Geater and Mansoor Ahmed-Rengers

Both of these patents appear in chapter 4. Patent 1 was designed by me solely; Jon Geater subsequently validated the design and assisted with the filing process. Patent 2 was designed collaboratively by Jon Geater and me. It was his initial idea to find a way to have modifiable blocks; the patent is a result of our joint exploration for a solution.

1.2.4 Invited talks

In addition to the talks given while presenting the aforementioned papers, I have given the following invited talks during my PhD:

1. FrameProv: Embedding trust in videos, Cambridge Blockchain Prize, 2020; winner of the competition.
2. DMapp: Decentralised mapping solutions, Future of Blockchain competition, 2019; winner of Zilliqa challenge.
3. Privacy in a mass surveillance world, University of Amsterdam, 2018
4. On the future of cryptocurrencies, University of Amsterdam, 2018
5. Future trends for blockchains, Inauguration of Cambridge Blockchain Society, 2018
6. Identity management in Hyperledger, Hyperledger session at Open Source Summit (OSS) Europe, Edinburgh, 2018
7. Access control in Hyperledger projects, Hyperledger Europe Meetup, Cambridge, 2018

1.2.5 Outreach and open source

In addition to these talks, I have tried to engage with the broader community by writing a popular science article about FrameProv for OpenDemocracy [7] and presenting posters at the Cambridge University Science and Policy Exchange 2019 and the Thales Academic Showcase 2019. The "Making Bitcoin Legal" and "Tendrils of Crime" papers also received a fair bit of media coverage and were covered by outlets such as WIRED [108] and MIT Technology Review [86], among others.

I also tried to keep in touch with the industrial side of my research to get a good sense of viability of technologies. To do this, I have worked part-time throughout the duration of my thesis. For the first few months, I worked as a security researcher at nCipher where I explored various blockchain application scenarios. In the last two years of my PhD, I worked at Jitsuin, a startup focused on providing traceability for IoT devices, as a security engineer. Here, I was the first employee and helped design their blockchain back-end system from the ground up. These experiences played a crucial part in the development of this thesis, and are at the core of chapter 4.

Lastly, I have striven to open source all the code that came out of the above projects [6]. This includes the visualisation tool for Tendrils of Crime, the index randomisation module for Cambium (with Intel SGX and without), all of the Robust Round Robin consensus code (written in collaboration with Robin Bryce) and the entire codebase for CoverDrop (written in collaboration with Daniel Hugenroth). In addition, I have open sourced all the code used in my textbook.

"A thousand flowers still bloom on this global network, but all of them rely on, and return spoils to, a handful of nodes..."

—Ian Bogost, The Constant Risk of a Consolidated Internet

2 Background

Discussions around decentralisation[1] have found fresh vigour in the wake of the cryptocurrency boom following the rise of Bitcoin. However, the debate around how best to structure computer networks is far older than that. In fact, as we will see in this chapter, the World Wide Web itself was expressly designed to enable decentralisation; its architecture arose from a philosophical stance that centralised, hierarchical structures are inimical to a healthy society.

After looking at these early design motivations, we shall see how the World Wide Web tended towards centralisation despite its creator's best efforts and why that is worrying. Next, we will talk about the rise of Decentralisation 2.0 with Bitcoin and subsequent research into blockchain networks. This will lead us to a few of the open problems that I've tried to address in my thesis. These problems, as we shall see, centre around the lawful use of cryptocurrencies, the decentralisation of enterprise applications and the scalability of decentralised networks.

2.1 The Promised Land

Tim Berners-Lee, the inventor of the World Wide Web, is quite explicit about his feelings towards centralised systems. Speaking of the increasing centralisation of the Web in 2018, he called it "a large-scale emergent phenomenon which is anti-human" [50]. So, if the current state of the Web is not what he wished for, what was his intention?

[1] A note about terminology: I use the word *decentralised* to refer to systems that have diffuse power structures and *distributed* to refer to any system with many computers working together. A Google server farm is a centralised distributed system.

Exactly 20 years prior, in 1998, Berners-Lee set out what he saw were the philosophical underpinnings of the World Wide Web in an essay where he finds parallels between the design of the Web and his religious beliefs [39].

Speaking of the guiding principles in Web design, the very first one he lists is decentralisation and says: "There is very little structure. There is the idea that society can run without a hierarchical bureaucratic government being involved at every step, if only we can hit on the right set of rules for peer-peer interaction." This aversion to a hierarchical system design is echoed by another Web pioneer, Brewster Kahle, the founder of the Internet Archive. Kahle speaks of centralised systems as "a dystopian world of closed, segmented, siloed, corporately-owned little pieces of property. I'd much rather see an open, next-generation web succeed" [212].

Support for decentralisation was prevalent at an organisational level as well. Take the IETF for example. In RFC1958, released in 1996, it lists the architectural principles of the Internet. Its support for decentralisation can be seen in Section 2.4: "Fortunately, nobody owns the Internet, there is no centralised control, and nobody can turn it off. Its evolution depends on rough consensus about technical proposals, and on running code" [121]. The instantiation of these principles can be seen in the development of technologies such as Universal Resource Identifiers (URIs) that were designed to be extensible and federated.

The 80s and 90s also saw a few bold proclamations by Web pioneers that seem almost naively optimistic in hindsight. John Perry Barlow's famous 1996 manifesto, *A Declaration of the Independence of Cyberspace*, starts off with the following bold proclamation:

> Governments of the Industrial World, you weary giants of flesh and steel, I come from Cyberspace, the new home of Mind. On behalf of the future, I ask you of the past to leave us alone. You are not welcome among us. You have no sovereignty where we gather[2].

The following fifteen paragraphs are similarly stirring in their tone and their rejection of hierarchical structures of what Barlow hoped would be the past. Another similar declaration from the 90's, *The Cluetrain Manifesto*, envisioned the end of traditional corporations with proclamations such as "Hyperlinks subvert hierarchy" and (speaking of networked users) "We are immune to advertising. Just forget it." [3]. These proclamations seems jarring especially considering how recently these statements were made.

These beliefs weren't just limited to declarations and manifestos; they were instantiated in the process of building the Internet. Volunteering for the Internet Engineering

[2]This anarchist view of the early Web proponents proclaiming cyberspace as being somehow separate to the physical world has been rejected by legal scholars. [164, Pg. 5]

[3]In some ways, the Cluetrain Manifesto's theses have been superficially adapted by modern corporations. The Manifesto urged companies to be part of the conversations that affect communities; today, almost every large corporation has a Twitter profile. It is, however, hard to imagine that this level of engagement is what the authors envisioned.

Task Force (IETF) is open to all with no fees or dues. The idea of rough consensus is put into action perhaps most literally in the "humming votes" that take place during IETF Working Groups: instead of voting by a show of hands or ballots, participants hum their approval of proposals with the chair deciding when the humming is loud enough [168]. Lastly, the IETF is very explicit in terms of its belief system when it states:

> The Internet isn't value-neutral, and neither is the IETF. We want the Internet to be useful for communities that share our commitment to openness and fairness. We embrace technical concepts such as decentralized control, edge-user empowerment and sharing of resources, because those concepts resonate with the core values of the IETF community. [13]

This mission statement, in my opinion, crystallises the beliefs of the early Web community and their wishes for the Web they were building.

2.1.1 2020 hindsight

So, the early architects of the Web valued decentralisation and intended its evolution to be free of traditional hierarchies and at least somewhat democratic. However, the Web as it stands today in 2020 hardly lives up to those principles. In many areas of the world, the Web is synonymous with Facebook which serves as the "free" gateway to its closed version of the Internet known as Internet.org [128]. According to a global study by Sandvine, Google, Netflix and Facebook account for 36.08% of all Internet traffic [188]; this number rises to a startling 65.42% when looking at mobile traffic alone [189]. If we turn our attention from traffic to time spent by users, the landscape appears even more consolidated: four of the top five apps by usage time globally are owned by Facebook [23].

Exactly how centralised the Web is is hard to quantify: share of traffic, extent of gatekeeping and control of widely used applications are all quantifiable metrics but they do not cover all aspects of centralisation; and indeed, centralisation is a multi-dimensional phenomenon so a single statistic cannot completely convey its extent. These statistics do, however, serve as reliable weather vanes and the signs they are giving are clear: the Web is more centralised now than in the early 90's and trending towards even greater consolidation. To use the parlance in distributed systems where we talk about "eventual consensus", it appears to me that Internet services demonstrate a tendency for "eventual centralisation"; and we are nearing the end-stages of that process[4].

This centralisation wasn't wholly unexpected. In fact, it was predicted long before the Web came into existence. Writing in 1967, Paul Baran noted how future computer utilities are likely to be centralised as communication networks tend to be "natural

[4]Further confirmation of this claim can be found in the changing attitudes of start-up founders and venture capitalists: building for sale is increasingly being preferred over building for scale. The eventual acquisition by a tech giant is seen as the most favourable outcome since competing with them is no longer considered feasible. [204]

monopolies" due to the cost of building the communication infrastructure [36]. Furthermore, he reasoned that information flows would tend towards centralisation due to the convenience of managing data from multiple users in one place and the profitability that comes with tight vertical integration, even if it comes at the cost of privacy. In these predictions, Baran has proven to be prophetic.

Of course, not all of Baran's predictions came true. For one, he envisioned heavy-handed government regulation as being one of the enablers of centralisation. In fact, it has been claimed that it is the lack of enforcement and modernisation of a certain kind of regulation—anti-trust regulations—that has allowed these modern oligopolies to flourish [173, 140, 171]. Another factor missing from Baran's predictions was the role of so-called *network effects* at the application layer, the economic phenomenon whereby a utility gains value with an increase in the number of users. Social networks and platforms such as Facebook are valuable not necessarily because of any individual standout feature but rather because *everyone is using them* [225].

Another explanatory factor for the centralisation of the Web is the difficulty of finding a sustainable economic model for online services. Advertising supported models seem to be the most viable option, and ad networks become more profitable the more data aggregation they do. This has led to the current *surveillance capitalism* economic model of the Web [226]. It is the pursuit for viable alternatives to surveillance capitalism (among other things) that has encouraged researchers to look into the new decentralisation movement, spearheaded by the blockchain community, as we shall see soon. First, let's try to understand the issues that arise out of a centralised Web.

2.1.2 Problems with centralisation

We've established that the Web is trending towards centralisation and is already quite consolidated, but why might that be problematic? Indeed some would argue the opposite: they would argue that centralisation is good because of the economy of scale it affords and the business models it enables. Peter Thiel, co-founder of Paypal and Palantir, has even argued that "Monopoly is (...) not a pathology or an exception. Monopoly is the necessary condition of every successful business" [207]. Vint Cerf, co-inventor of TCP/IP, has argued that the ad-driven economic model of Google search (his employer) is more egalitarian than any other existing economic model[5]. He further makes the case that instead of focusing on inventing decentralised monetisation models, the focus should be on rethinking the incentive structure for executives in modern corporations [54].

While Cerf's proposal of changing incentives for executives could solve some of the issues that have concerned other Web pioneers, it doesn't address all of them. According to Berners-Lee, the main challenges facing the Web are: "1. We've lost control of our personal data; 2. It's too easy for misinformation to spread on the web and; 3. Political

[5]Cerf reasons that this model allows anyone, regardless of their economic status, to make as many queries as anyone else. This, in his opinion, is more egalitarian than a system which would charge users by their usage.

advertising online needs transparency and understanding" [40]. One possible solution to these issues according to Berners-Lee is *Solid*, a decentralisation project that aims to provide "true data ownership as well as improved privacy" [175].

Although I concur with Berners-Lee's concerns, I find his criticisms of the centralised Web strangely narrow in scope. While these criticisms have been validated by countless headlines in the aftermath of the Cambridge Analytica scandal, they target a very specific kind of abuse on the Web. Consequently, his solution for the centralised Web, Solid, is targeted only at the problem of data accumulation and thus fails to address other issues brought about by centralisation.

This lack of breadth in criticising the Centralised Web is not peculiar to Berners-Lee alone. Opinion pieces written in the wake of Cambridge Analytica echo the same limited set of concerns. I do not intend to downplay the severity of the data centralisation, surveillance and political manipulation concerns; I want to highlight that those issues all belong to just one corner of Pandora's Box—the corner that's merely more visible due to the spotlight of recent media attention. So, what else is in the Box?

At its core, I venture that centralisation can be problematic because of two reasons:

1. It skews *power dynamics* in favour of the consolidated entity thus corroding any negotiating power and agency for the counter-party (and, indeed, entire sectors of the economy).

2. It creates a *single point of failure*, where failures may or may not be technical in nature.

Let us look at a few illustrative examples for each of those two reasons, starting with the latter.

2.1.2.1 Single point of failure

As we talked about earlier, Facebook offers a limited walled garden "Internet" for free in some parts of the world, especially in the so-called third world. This free access combined with an already large user base, gave Facebook an overwhelming share in messaging and social networking in several developing countries. The selection of which digital services are allowed within Facebook's walled garden is done by Facebook which gives it great power in shaping the reality perceived by millions. This situation was exploited with alarming results by the Myanmar armed forces during the ongoing Rohingya genocide [64].

Another illustrative case can be seen in India. Recently, Facebook India's head of public policy, Ankhi Das, was caught saying the quiet part loud when she opposed taking down posts inciting violence by leaders of India's ruling party[6] because doing so "would damage the company's business prospects in the country." [64]. To make matters worse,

[6]One of these leaders, T. Raja Singh, stated in Facebook posts that Rohingya Muslim immigrants should be shot, called Muslims traitors and threatened to raze mosques [176].

it was revealed that Das had previously shared Islamophobic posts on Facebook calling Muslims in India a "degenerate community" [191]. Anti-Muslim posts such as these have been linked to several cases of murder of Muslims inflicted by mob violence [85].

Now, hate speech and the direction of majoritarian violence towards a persecuted minority are not new evils. However, the centralisation of media in the hands of one company means that one person's prejudices or political preferences[7], or one company's corporate strategy can have devastating widespread impact on entire nations. This would have been more difficult to achieve in a world with a more fragmented and diverse media ecosystem. These are thus examples of societal single points of failure created by centralisation.

Examples of technical single points of failure being exploited are plentiful as well. Centralised systems make for tempting targets for hackers due to the potentially large payoff. One recent example of such a hack is the compromise of Twitter profiles of several prominent figures via a spear-phishing attack on one of Twitter's employees [205]. The hackers, allegedly a group of teenagers, didn't do much damage since they tried to use the hack to pull off an ill-advised Bitcoin scam but the damage a more malicious adversary could have done with a similar hack is far greater.

An example of a hack that proved to be more consequential is the Equifax hack in 2017 which exposed the personal details, including the social security numbers, of 147.7 million Americans [166] as well as personal data pertaining to 15.2 million UK customers [151]. This resulted in Equifax having to pay more than 575 million USD in fines [90]. Within a year of this data leak another was discovered, this time affecting Google. Google's social network, Google+, was found to contain a bug that exposed the data pertaining to 52.5 million accounts [165]. Google+ was shut down following these revelations [217].

Apart from making for interesting targets for hackers, centralisation also has the potential to make censorship easier. Brewster Kahle opined upon this issue after the Internet Archive was blocked in China and India thus: "(to) keep data safe; you make copies" [224]. This idea led Kahle to promote creating many versions of the Archive [224]. Of course, there are countless other examples of services being censored in different countries thus making it difficult for citizens of those countries to access information. Ross Anderson pointed out this worrying aspect of digital centralisation when he compared information dissemination on the Internet and traditional printing presses and noted how much easier it could be for a powerful adversary to remove all copies in the former compared to the latter; he illustrated this comparison by highlighting the relative ease with the Church of Scientology could get its secret books taken off the Internet versus the difficulty the Catholic Church had with doing the same with printed copies of Tyndale's translation nearly 500 years before [20, Pg. 679-709]. In this scenario at least, it seems we've taken a step backwards in making information more censorship resilient.

[7]Das had previously been in the news for publicly declared her support for the BJP stating (of Narendra Modi's 2014 win), "We lit a fire to his social media campaign and the rest is of course history". [192]

2.1.2.2 Power dynamics

Let us now turn our attention to the other problematic aspect of centralisation: the corrosion of negotiating power and agency. Digital services almost always come with so-called "Terms of Use" which represent the legal contract between the user and the service provider. One of the legal justifications given in defence of legal contracts (digital or otherwise) is that they are a tool to empower people and are an instantiation of the moral ideal of "equal respect" between persons; this moral ideal is "why contract law can produce genuine legal obligations and is not just a system of coercion" [134]. This dynamic of two equal parties coming together "with a sense of justice and interpersonal obligation" is seen by contractualists as the core justification for why contracts are defensible [134].

Terms of Use[8] agreements quite clearly do not live up to this ideal. These agreements are called "contracts of adhesion" or "wrap contracts" by legal experts because "they impose take-it-or-leave-it conditions on users that stick to them whether they like it or not" [226, Pg. 48]. Researchers have long pointed out how these contracts are intentionally excessively long and obtusely written to dissuade readers from reading them. The combination of these factors has called into question the validity of these agreements since users cannot meaningfully consent to them [147]. Unfortunately, despite these grave concerns, courts have by and large upheld the legitimacy of these contracts leading to their proliferation [226, Pg. 49].

Aside from being take-it-or-leave-it, another aspect of these agreements that make them pernicious is the ability for the service provider to unilaterally change the terms at any time without any user consent or knowledge. Nancy Kim cites Google's Terms of Use which states: "We may modify these terms or any additional terms that apply to a Service" and calls these unilateral modification clauses "Now you see it, now you don't" clauses since one can not be sure if the terms are still the same they agreed to. She characterises these clauses as "unrealistic and maybe even sadistic" [136, Pg. 65-66]. All of these factors combined make it clear that the agreements that users are forced to agree to are crafted for the benefit of the service provider at the expense of the users.

This power-imbalance-by-design is concerning because for an increasing number of people, not using these services is no longer a realistic option. Not only do people rely on digital services for their livelihoods, these services increasingly play an important role in people's social and political lives. Zuboff envisions that without fundamental changes, only those who are "wealthy or stubborn enough to lead effective lives" without the use of these services will be able to "escape the worst excesses of rendition" as "decision rights and self-determination become privileges of the wealthy" [226, Pg. 257] . In a world filled with centralised corporate digital services guarded by take-it-or-leave-it contracts, it is a luxury of the wealthy to be able to leave-it.

[8]The concerns mentioned here for Terms of Use agreements apply equally to Privacy Policy agreements.

It is worth noting that these concerns are no longer limited to the digital realm. What's doubly concerning isn't just the fact that the Web is becoming more centralised, it is also that the centralised Web is making the world more centralised. Unionised taxi drivers are being supplanted by Uber, independent stores by Amazon, travel agents by Booking.com, and so on. This consolidation results in an extractive economy that takes "more and more value from participants while continuing to enjoy the veneer of a disruptive, socially minded enterprise" [56]. This extraction leads to further consolidation of power and resources leading to further avenues for extraction in a vicious cycle.

Frustrated by this cycle of ever-increasing consolidation, many researchers have been looking for both a viable alternative economic model to surveillance capitalism as well as technical means to design systems that are more decentralised [74]. Both of these trains of research seemed to converge in the mid-2010s at an unlikely place: the tumultuous world of cryptocurrencies and blockchains.

2.2 A new hope

The cryptocurrency industry was kick-started by Bitcoin and its pseudonymous creator(s) Satoshi Nakamoto. Bitcoin didn't start off trying to be a substitute to surveillance capitalism. Its crosshairs were set on a very different kind of monolith: the banking sector. Let us take a quick look at the motivation of its creators and how they sought to fulfil them.[9]

2.2.1 Bitcoin primer

With Bitcoin, Nakamoto sought to create "an electronic payment system based on cryptographic proof instead of trust" [161]. Their motivation stemmed from a dislike of the traditional banking system which they saw as inefficient and open to fraud. This motivation is reiterated in the very first block of the Bitcoin blockchain which contains the headline: "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" indicating their dismay at the banking system [76]. So how did Nakamoto attempt to achieve this lofty goal? By creating a decentralised peer-to-peer network based on *blockchain* and *proof of work* consensus.

Blockchain is a data structure in which the data is organised into discrete *blocks*. These blocks contain application-specific data and are produced sequentially, with each block including a hash of the previous block. This inclusion of the previous block's hash is what turns this collection of blocks into a block*chain*. This chaining ensures that any content that has been included in the blockchain cannot be altered: if one were to modify the contents of a preceding block then its hash would change. Thus the recorded

[9] I assume that the reader is familiar with Bitcoin and Ethereum. I only provide a succinct introduction to them here. Interested readers should read the original Bitcoin paper [161] and the Ethereum yellow paper [219].

hash in the next block would no longer match the calculated hash and the chain would be considered broken.

Proof of work (PoW) is the consensus mechanism used in Bitcoin. In order to create a system not based on trusted third parties, Bitcoin needed a way to authenticate transactions and prevent double spending in a distributed manner. Proof of work accomplishes this as long as no single entity controls a majority of the computing power in the network. The algorithm for proof of work consensus is quite simple: all nodes compete to find x such that $H(x||r) < target$ where $H()$ denotes the SHA-256 hash of the parameters, r is a fixed string that includes the previous block's hash and $target$ is a numeric value that is periodically adjusted. Whichever node finds x first becomes the "miner". The miner collects transactions from clients, packages them into a block along with the proof of work x and sends it to the network via a gossip protocol. In return, the miner gets bitcoins as a reward.

In this elegant manner, Bitcoin accomplished consensus in a truly permissionless setting (i.e. a setting with no gatekeepers). Miners are incentivised to keep mining (doing computational work) using Bitcoin's own currency while allowing anyone to submit transactions to the network to be included. The cypherpunk[10] vision was that this would truly democratise finance and "bank the unbanked" [200]; and as Bitcoin began to rise in prominence some started to realise that Bitcoin pointed the way to something broader than just a cryptocurrency, it could be a *global trusted computer*.

2.2.2 Ethereum and altcoins

The term *smart contract* was defined by Nick Szabo in 1996 as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises." [202]. Szabo's argument was that a lot contracts that are written on paper and mediated by lawyers can be formalised in a way that can be processed by computers, thus increasing efficiency. The challenge with this idea was to find the right kind of trusted infrastructure that could host and execute these contracts.

Vitalik Buterin and Gavin Wood, the creators of Ethereum, sought to use a blockchain as that trusted infrastructure. They adopted the blockchain and PoW from Bitcoin and added a virtual machine (EVM) on top that was Turing complete. This meant that now the "global trusted computer" could perform arbitrary operations. Ethereum included a scripting language—Solidity—that made it easy to script smart contracts and deploy them to the network, where they would be stored on all nodes. Then, when one wished to interact with a smart contract, say, to update the state of a variable, you would send a *transaction* to the smart contract and all nodes would update their state in accordance with the EVM specification. In this way, Wood and Buterin believed, they had arrived at the trusted infrastructure required for smart contracts without introducing any trusted

[10]The cypherpunk movement advocates the use of cryptography as a means to enact social and political change [115]. Bitcoin seems to have emerged from this movement.

third parties.

Ethereum was introduced in 2014 and immediately caught public attention; at the time of this writing, it is still the second-most popular cryptocurrency after Bitcoin. Many proofs of concept were built seeking to revolutionise industries ranging from fisheries to military equipment manufacturing. Proponents claimed that smart contracts would make existing systems more efficient, more robust and more transparent.

Many researchers were also hopeful about blockchains. New cryptocurrencies, some of which started off as academic projects, ended up with market capitalisation of billions of dollars [222, 11]. The idea of cryptocurrency micro transactions (mechanisms by which users could pay fractions of pennies for services they use) began to take hold and developers rushed to incorporate mechanisms to enable them in their respective blockchains. These developments led some to proclaim blockchains as the long sought-after alternative economic model to advertising [74]. Soon, mainstream news got wind of this technological hype and Bitcoin's price increased ten-fold in the span of six months towards the end of 2017.

As with all bubbles, it burst, and cryptocurrency prices tumbled in the following year. Scepticism began to replace optimism.

2.3 Next steps for the blockchain

While enthusiasm for blockchain adoption remained quite high among technologists for many years after the introduction of Bitcoin and Ethereum, some challenges for widespread use of blockchains were apparent from the onset even before the price crash. On the pure cryptocurrency side these primarily revolved around usability, the privacy of transactions, throughput (transactions per second), efficiency of proof of work[11], and the potential for misuse of cryptocurrencies (and the legal grey areas that this entails) [47, 152, 95]. New threats emerged as Bitcoin became more popular: the network became increasingly centralised as only a few mining farms ended up controlling almost all of the computational power leading to concerns of censorship; new kinds of malware, known as ransomware, spread using cryptocurrencies as a monetisation strategy.

At the same time, while smart contracts generated a lot of interest, enterprises were reluctant to perform sensitive operations on public blockchains. The concerns weren't limited to just the sensitivity of any data being put on the blockchain but also metadata such as how often two companies interacted with each other, which companies were on a given value chain, how many employees were pushing updates, etc. All of these were hitherto trade secrets and no one wanted to give up their competitive advantage unilaterally.

This overwhelming scepticism in the wake of the market crash led Gartner to proclaim that blockchains were now entering the dreaded "trough of disillusionment" [100].

[11] This includes both the energy cost of proof of work as well as the embodied cost of specialist mining equipment used in large "mining farms".

2.3.1 Out of the trough

Not everyone believed that the blockchain experiment was finished and instead kept working to get blockchains out of the trough. Concerns about privacy of transactions led to the development of privacy-preserving coins such as Zcash and Monero. Concerns about mining farm concentration led to the development of ASIC-resistant PoW algorithms. Concerns about the legal grey areas led to a raft of new legislation [211, 89] as well as to the rise of cyrptocurrency compliance companies. Concerns about the scalability and energy use of PoW led to the invention of tens of new consensus algorithms [35].

On the institutional side of things, developers sought to alleviate enterprise reluctance by introducing *permissioned blockchains* – blockchain networks that are maintained not by random unknown miners but rather by nodes identified by some gatekeeping mechanism. These permissioned blockchains were supported by institutions such as JP Morgan—who developed their own platform, Quorum—that saw potential in the concept as an inter-organisation collaboration tool. Permissioned blockchains got a shot in the arm when the Linux Foundation started the Hyperledger Foundation which serves as an umbrella organisation for a host of permissioned blockchain frameworks that has proven to be extremely popular among technology companies [180].

2.3.2 My contributions

I do believe that striving for a more decentralised Web is a worthwhile goal, even if it seems quite impossible at times. I hope that the work presented in this thesis nudges us—even if ever so slightly—in that direction. I have focused on three main issues plaguing current decentralisation efforts:

1. Legal grey areas and misuse of cryptocurrencies
2. Difficulties with production deployment of enterprise blockchains
3. Creating scalable (in throughput and number of nodes) networks in an energy efficient manner

I discuss the first issue in chapter 3 when I talk about the problem of cryptocurrency-enabled crime and possible remedies. The second issue is discussed in chapter 4 where I talk about the state of permissioned blockchains and the steps I've taken in getting a real production system out to customers. In chapters 5 and 6, I tackle the third issue by presenting two novel consensus algorithms, one that operates on a blockchain and one that operates on a new kind of distributed data structure. Finally, I conclude with a summary of my findings.

"This is a song in defence of the fence. A little sing-along, an anthem to ambivalence."

— Tim Minchin, The Fence

3

Mitigating cryptocurrency-enabled crime

Bitcoin attempted to create a virtual currency outside of the control of governments—and indeed, of all institutional actors—using a decentralised peer-to-peer network. This was enabled by the clever adoption of proof of work in order to prevent sybil attacks [84] as well as to provide a unified view of the network. Lastly, the use of a blockchain ensured a high level of integrity for transactions.

This ethos of decentralisation as a Good was driven by a desire to escape traditional banking institutions that the author(s) viewed as being corrupt and fragile. However, as Bitcoin came to gain widespread adoption, especially in criminal circles, many began to doubt the effectiveness of such a decentralised network, both in keeping a stable value and in hindering crime. It increasingly began to be argued that Bitcoin throws the baby out with the bathwater; that while there are issues with the traditional banking system, it performs critical functions that cannot be disregarded. These functions include things like recovering stolen funds, tracking the proceeds of crime and preventing capital flight. Let us take a look at how Bitcoin fares in these regards.

3.1 Bitcoin and crime

The extent of cryptocurrency-enabled crime is hard to quantify due to varying definitions of what constitutes a crime and due to the pseudonymous identities used on the blockchain. That said, there have been several studies using different heuristics to try and gauge the scale of the problem.

According to a recent study by Chainalysis (a company that sells anti-money laun-

dering services for cryptocurrencies), the vast majority of criminal transactions take place on the Bitcoin blockchain. Therefore, they focus on Bitcoin and report some interesting statistics: "illicit" transactions (according to their definition of illicit) made up only 1.1% of the total transaction volume in 2019 [60]. This observation is closely corroborated by another analysis firm, Elliptic, who report that this number was "less than one percent of all transactions" [88] between 2013 and 2016.

Diving further into the numbers, Chainalysis reports that scams make up the largest share of these illicit transactions accounting for $4.9 billion in 2019, more than three times that in 2018 [60]. In addition, hacks of cryptocurrency exchanges accounted for $282.6 million in 2019 and a total of $1.8 billion over the last ten years. Overall, Chainalysis concludes that criminal activity on the Bitcoin network is on the rise, an observation that is mirrored by SWIFT and BAE Systems as well who note that cryptocurrencies are likely to be increasingly attractive to criminals [32]. [1]

This use of cryptocurrencies by criminals as well as the investment bubble in late 2017 led the Bank for International Settlements to label Bitcoin "a combination of a bubble, a Ponzi scheme and an environmental disaster" [58]. One of the major concerns with Bitcoin is the fact that if one were to fall victim to a scam or had bitcoins stolen, there is no recourse. The irreversibility of transactions was an explicit design goal for Nakamoto [161] but it turns out that when their money gets stolen, people want to get it back. Also, while the amount of cryptocurrency-enabled crime is relatively low currently, the trend is clearly upwards and having truly irreversible transactions makes dealing with crime very difficult.

In this chapter I discuss work done by me in collaboration with Ross Anderson and Ilia Shumailov across three papers[2] to address the issue of regulating Bitcoin and other cryptocurrencies. First, we discuss how the law might actually regulate bitcoin and other cryptocurrencies so as to provide the benefits, ranging from low-cost international money transfers and decentralised resilient operation, through to competitive innovation; while mitigating the harms – specifically the use of cryptocurrencies in extortion, money laundering and other crimes, and the difficulty that crime victims experience in getting redress. We show that where the relevant case law is understood, it becomes much easier to track stolen (or otherwise "tainted") bitcoins than previously thought, and we describe a prototype system for doing so.

Second, we use this system to find interesting patterns on the Bitcoin blockchain. To do this, we had to find ways to mitigate the problem that all Bitcoin tracing algorithms

[1] While these numbers are indeed large, it is worth putting them in context. The FinCEN file leaks of 2020 revealed that traditional banks were involved in laundering "suspicious transactions" worth more that $2 trillion [194]. Deutsche Bank alone was responsible for $1.3 trillion of that figure which dwarfs all crime facilitated by all cryptocurrencies by orders of magnitude. The nature of these suspicious transactions is also very grim, as exposed by these leaks: "Terror networks, drug cartels, organized crime rings, and rapacious kleptocrats have all benefited, using the US financial system to wash clean their illicit profits." [143]

[2] The individual contributions to these papers are as listed in § 1.2. In this chapter, I have collated the information from these three papers and brought them up to date with recent data.

face: they yield an enormous amount of data of which very few data points are relevant or interesting to investigators, let alone ordinary bitcoin owners interested in the provenance of the bitcoins they hold. To accomplish this we came up with a graphical model to represent the stolen coins and then implement this using two different visualisation techniques.

Third, we report our findings after talking to real-world victims who got in touch with us after the publication of our first paper on the topic. This led us to revise our initial assumptions about the cryptocurrency ecosystem.

Fourth, enlightened by the experiences of the victims, we look at laws passed to regulate Bitcoin in several jurisdictions and point out several issues with them. We also point out concerns with more recent technological developments in the cryptocurrency world, such as payment channels and privacy coins, and difficulties with their lawful usage. These concerns have since been borne out by surveys of cryptocurrency-enabled cybercrime. Finally, we present our recommendations for policymakers.

3.2 What the law says

Nemo dat quod non habet roughly translates to "No-one can give what they don't own" and is an established principle of many systems of law. If Alice steals Bob's horse and sells it to Charlie, Charlie doesn't end up owning it. When Bob sees him riding it, he can simply demand it back. This is natural justice; the horse wasn't Alice's to sell. However, it does leave a shadow of doubt over ownership in general. How can you buy something without constantly living in fear that a rightful owner will turn up and ask for it back?

In medieval times there arose a specific exception for a 'market overt' [186]: if Alice steals Bob's horse and then takes it to the local public market, where she sells it openly between dawn and dusk to Charlie, then Charlie does indeed now own the horse. Bob can still seek damages from Alice, or seek to have her transported to the colonies or even hanged; but the horse is now Charlie's. This incentivises people to buy and sell at markets (which the king can regulate and tax), and also encourages crime victims to go to the local market to check whether their property is on sale there, which in turn may deter crime.

Britain abolished the 'market overt' exception to the "*nemo dat* rule", as lawyers call it, in 1994 following abuse by thieves selling stolen antiques [187]. But two exceptions remain that are of possible relevance to some cryptocurrencies: for money and for bills of exchange. You can get good title to stolen money in two main cases:

1. You got the money in good faith for value. For example, you bought a microwave oven at a high street store and got a £10 note in your change. That note is now yours even if it was stolen in a bank robbery last year.

2. You got the money from a regulated institution, such as from an ATM. Then even if it was stolen in a robbery last year, that is now the bank's problem, not yours.

The *nemo dat* rule and its exceptions are discussed in the case of bitcoin by Fox [93], whose analysis we draw on and extend here. See also his book on the law of money for further details [94]. Now, the USA has designated bitcoin a commodity, but there is a lot of lobbying pressure to treat some of it, or at least some cryptocurrencies, as money; Japan has gone as far as designating it 'virtual money' while other countries treat it as money for some purposes [104][3]. In the UK, the tax authorities treat it as foreign currency for the purposes of value-added tax but as a commodity for income tax. A survey of cryptocurrency status conducted by Freshfields in 2018 stated that there appears to be nowhere that treats bitcoin simply as money [96]. This observation was corroborated by a study by the Cambridge Centre for Alternative Finance conducted in 2019 in which they compared the regulatory stances of governments across 23 jurisdictions [45].

In what immediately follows, we will assume that bitcoin is a commodity. We will explore what the consequences might be if it comes to be treated as money, or as a bill of exchange, in § 3.7. For present purposes, all we need to know is that someone who receives money or a bill of exchange in good faith and for value can get good title to it. Unless cryptocurrencies acquire this privileged status, there is no general exception to the *nemo dat* rule. As they have not achieved this status (except, apparently, in El Salvador), a theft victim can pursue and retrieve her stolen cryptocurrency.

The second important insight from the law is Clayton's Case [80]. In English law, there is a long-standing legal precedent on tracing stolen funds. It was established in 1816, when a court had to tackle the problem of mixing after a bank went bust and its obligations relating to one customer account depended on what sums had been deposited and withdrawn in what order before the insolvency. Clayton's case sets a simple rule of first-in-first-out (FIFO): withdrawals from an account are deemed to be drawn against the deposits first made to it [167]. The legacy of the British Empire and Commonwealth ensured that this principle has become embedded in the law of many other countries too [178].

Armed with this legal guidance, we can say that not only is it possible for the victim of Bitcoin theft to take back her coins (irrespective of where they ended up), but also that the right way to trace which of the bitcoins were the victim's is by using FIFO tracing. Now, we'll first see how tracing can be, and has been, done on a purely technical basis and then see how the situation changes when we apply the legal guidance.

3.3 Bitcoin tracing

Every bitcoin consists of its entire history since it was mined. What a wallet stores as a bitcoin is just a pointer to the relevant unspent transaction output (UTXO) and the signing key needed to assign the value therein to someone else. However the value

[3]After the submission of this thesis, the government of El Salvador announced that it would treat Bitcoin as legal currency becoming the first country to do so [38]. The consequences of this decision remain to be seen but it has already faced backlash from organisations such as the World Bank [220]

derives from a series of pointers to previous transactions in the blockchain, each of which has inputs and outputs, going all the way back to where the bitcoin's constitutive components were originally mined. So it is fairly straightforward to trace a transaction's history, at least in principle. How might it work in practice?

There has been significant work already on tracing transactions and analysing their patterns in the blockchain. For convenience, bitcoin operators use multiple wallets and pass money between them using automated scripts; change wallets are used to break up large amounts and give change, while peeling chains are used to pay multiple recipients out of a single wallet and multisource transactions are used to consolidate small sums into larger ones [4]. Clustering analysis can link up the different wallet addresses used by a single principal; Meiklejohn et al identified over half a million addresses used by Mt. Gox, then the second-largest bitcoin exchange [153]. Commercial blockchain analysis firms do this at scale. Their customers are typically law enforcement agencies and those exchanges that wish to do due diligence on payments to and from third parties.

There is also research by academics trying to understand and map out the ecosystem. Seminal papers were by Ron and Shamir who traced a significant number of Silk Road bitcoin that the FBI had missed [184], and two papers by Möser, Böhme and Breuker. In 2013, they used test transactions to analyse the operation of Bitcoin Fog, BitLaundry and other anonymisation services [158]; in the second, they present a detailed analysis of how taint tracking might work through multiple transactions [159]. Their focus was on two algorithms for dealing with multisource transactions of which one input was tainted: these were 'poison' (whereby the whole output is tainted) and 'haircut' (where the output is tainted by the percentage of input value tainted).

Commercial blockchain analysis firms are cagey about their methods – their terms of service typically require customers not to reverse engineer their algorithms. They seem to employ staff to make multiple small payments into and out of both exchanges and the underground merchants using bitcoin, use clustering analysis to link together the wallets each actor uses, and then track the flows between them; the focus is at the application layer of payer and payee intent rather than at the level of the blockchain. Whatever the details, coin checking appears to be accepted good practice.

3.3.1 Bitcoin mixing

One might wonder that if tracing algorithms such as haircut and poison exist, why do we need another one? The answer lies in how Bitcoin transactions are structured and the use of Bitcoin *mixes*.

First, it is impossible to subdivide a UTXO, so if Bob wants to pay Alice 0.5 bitcoins but his savings are in the form of a single UTXO worth 50 bitcoins, then he has to make a transaction with two outputs: one to Alice (for 0.5 bitcoins), and one to a change address owned by himself (for 49.5 bitcoins). This indivisibility leads us to classify

[4] If this is unfamiliar, the book by Narayanan et al. [162] describes Bitcoin mechanics in detail.

bitcoin transactions into the following types:

1-to-1 transactions

Transactions where a single UTXO is sent to a single output. These are quite rare although we've seen them used as building blocks in more complex payment schemes (perhaps as a naïve attempt to anonymise transactions).

Many-to-2 transactions

The workhorse of bitcoin transactions; as discussed, these are a natural consequence of the indivisibility of UTXOs, and most legitimate transactions belong in this category.

1-to-many transaction

These are quite rare since normal payments to multiple entities are executed by most wallets as a chain of many-to-2 transactions. 1-to-many transactions are sometimes used in technically simplistic mixes to split crime proceeds proceeds into many wallets in order to make tracing difficult.

Many-to-many transactions

These are like 1-to-many transactions except that they have multiple input UTXOs. They are the second kind of mixing strategy; they shuffle cryptocurrency between different keys, mostly controlled by the same people.

The default transaction type being a many-to-2 transaction rather than a simple account to account transfer as in traditional banking complicates things. It means that even if no one was trying to cover their tracks, tracing becomes convoluted. To illustrate, suppose Alice had 29.5 bitcoins before Bob sent her the 0.5 bitcoins; now, suppose it turns out that Bob is a cryptocurrency exchange hacker and therefore the 0.5 bitcoins are tainted. If we used poison then all of Alice's 30 bitcoins are also marked as tainted whereas if we used haircut all the 30 bitcoins would be marked as $\frac{1}{60}$ tainted. In either case, the initial taint from Bob would spread rapidly through the network putting more and more bitcoins in a grey area.

Things get further complicated when we bring mixes, and consequently the latter two types of transactions, into the picture. Cryptographers have long worked on remailers or mixes. Mixes were proposed in 1981 by Chaum to enable email and other message traffic to be sent and received anonymously [61]. If Alice wants to send an anonymous email to Bob, she can send it first to Charlie and ask him to forward it to Bob. Chaum proposed that, to frustrate naïve traffic analysis, Charlie would accumulate a number of encrypted messages and mix them up before relaying them. If Alice doesn't want Charlie to read her message, she can first encrypt it with Bob's public key. If she doesn't want to let her ISP (or a police wiretap) know she's communicating with Bob, she can take the message that's already encrypted with Bob's public key, and now encrypt it also with Charlie's public key, so that all the police see is a message to Charlie. If she wants Bob to be

able to reply to her, she can include a cryptographic reply coupon. As we think of more and more possible threats, such systems become ever more complex. The most common anonymity system, Tor, sends worldwide web traffic through three nodes between your Tor browser and the server you wish to visit, so that your anonymity is protected against one or two of them being compromised. There is now a very substantial literature on anonymity systems, with several sophisticated attacks on them and complex trade-offs between performance and security.

Cryptographic 'mixmaster' remailers were a significant part of the cypherpunk culture from which bitcoin emerged, and so it is unsurprising that various people started offering mixing services for bitcoin, with evocative names such as bitcoinfog, coinjoin and tumblebit. A newer cryptocurrency, Zcash, has a kind of Aladdin's laundry: it lets users put their coins back in the mine and get out new coins that are indistinguishable from other freshly mined coins. Some of these 'schemes', as cryptographers tend to call them, use clever tricks such as ring signatures and smart contracts. Others are simpler; Möser et al. reported that one bitcoin laundry turned out to be just a single fat wallet, and if a customer paid in some bitcoin on a Monday, the operator would return a slightly smaller sum on Tuesday [158]. But whatever the quality of the mixing (in a technical sense), the underlying idea is that if you put one black coin into a sack with nine white ones and shake them hard enough, the output will be ten white coins, or at least coins that are such a light grey that in practice they will be treated as white.

However, the perspectives of cryptographers and lawyers are sharply divergent. As noted above, even if cryptocurrency becomes money, you have to get coins in good faith in order to acquire good title; this is discussed extensively by Fox [93]. As all bitcoin transactions ever made are in plain sight on the blockchain, the act of passing a bitcoin through a laundry should put all its subsequent owners on notice that something may very well be wrong. Coin checking has been discussed since at least 2013, coin checking services exist, and bitcoin exchanges claim to do it. If coin checking is now a reasonable expectation, the likely outcome of feeding one black coin and nine white coins into a bitcoin laundry isn't ten white coins, but ten black ones. When matters come to court, any laundries that are clearly identifiable as such are likely to have exactly the opposite effect from that asserted by their designers and operators. In short, people designing money laundering mechanisms have been using the wrong metrics of quality from a legal point of view.

3.3.2 TaintChain: practical FIFO tracing

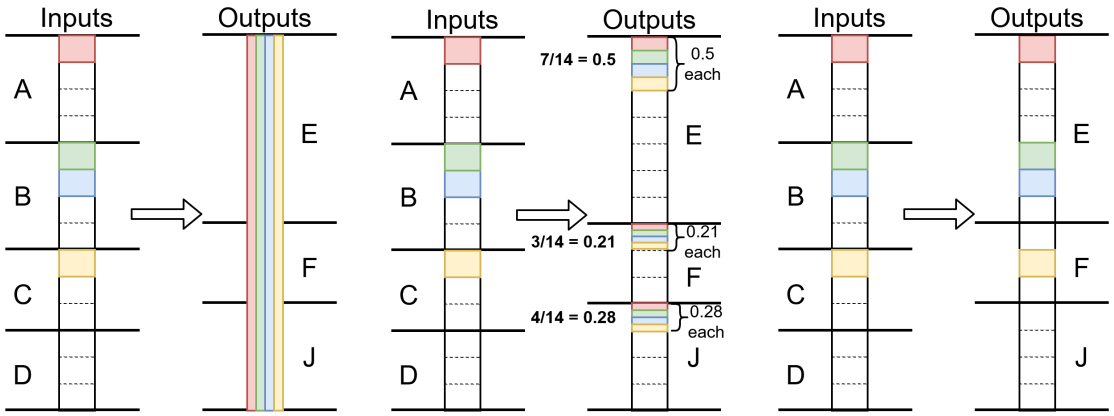
To see what a system that takes the legal perspective into account would look like, we implemented FIFO tracing and built it into a system we call the *Taintchain*. This starts off from a set of reported thefts or other crimes and propagates the taint backwards or forwards throughout the entire blockchain. If working forwards, it starts from all tainted

transaction outputs and marks all the affected satoshis[5] as tainted until it reaches the end of the blockchain. If working backwards, it traces each UTXO of interest backwards and if at any point it encounters a taint, it returns taint for the affected satoshis. We have made the system publicly available [203].

To test the system, we performed a FIFO taint trace starting from a few well-publicised coin thefts[6], and ran it from the genesis block to 2016. We found that it concentrated the taint more than haircut or poison tainting strategies.

For example, the 2012 theft of 46,653 bitcoin from Linode now taints 16,855,619 addresses, or just over 93% of the total, if we use the haircut (or poison) algorithm; with FIFO, it's 245,120 or just over 1.35%. More recent hacks spread the taint even less; for example, the 2014 Flexcoin hack (where "the world's first bitcoin bank" closed after all their coins were stolen) now taints only 15,265 accounts if we use FIFO, but 10,421,112 (or over 57% of all addresses) if we use haircut.

The reasons for this higher concentration with FIFO should be clear from the graphics below. Imagine that the red bitcoin inputs to the transaction are stolen satoshis, the green ones are blacklisted as they're from Iran, the blue ones have been marked by an anti-money-laundering screening program as the output of a bitcoin laundry, and the yellow ones are the proceeds of drug sales on an underground forum. The question for someone interested in enforcing the law is: which of the outputs of each transaction is tainted, and to what extent?



(a) Poison tainting showing a complete diffusion of taint

(b) Haircut tainting showing proportional diffusion of taint

(c) FIFO tainting showing no diffusion of taint

Figure 3.1: Comparison of taint tracing techniques using a many-to-many transaction. The input and output values are identical across all three examples (F with 7 BTC, A, J and B with 4 BTC and, C, D and E with 3 BTC).

In poison, if you have inputs with four different kinds of taint then all the outputs are tainted with everything. This leads to rapid taint contagion. Figure 3.1a illustrates poison tainting. If we were to use poison tainting for asset recovery then we would soon

[5] A satoshi is the lowest denomination of bitcoin possible. 1 bitcoin = 10^8 satoshis.

[6] Data from `https://bitcointalk.org/index.php?topic=576337.msg6289796#msg6289796`

end up having to confiscate almost all of the coins in the network.

Haircut is only slightly different. Here, taint is not binary but fractional. So, instead of saying that all the outputs are tainted with the four kinds of taint, we associate a fractional value to the taint. If half of the input was tainted red then all the outputs are half red-tainted. Taint diffuses quickly through the network as in poison, but the result is rapid taint diffusion and dilution, rather than contagion. Figure 3.1b illustrates haircut tainting. The taint diffuses so widely that the effect of aggressive asset recovery might be more akin to a tax on all users.

With the FIFO algorithm, the taint does not go across in percentages, but to individual components (indeed, individual Satoshis) of each output. As the taint does not spread or diffuse, the transaction processes it in a lossless way. This means that we can trace a bitcoin's heritage backwards as well as tracing taint forwards, and we can do tracing efficiently once the appropriate index tables have been built.

This served as a good demonstration of the effectiveness of FIFO in delivering more usable results than those provided by the existing state of the art. However, in order to truly utilise this tool to spot suspicious activity on the blockchain, we needed to find a way to find interesting patterns in the taint spread.

3.4 Finding patterns in the noise

When we started analysing the taintchain, we ran into a number of issues. First is the size of the datasets generated: with just 56 kinds of taint[7], we ended up with a dataset of about 450 GB. The second problem is that the things we're looking for—side effects of crime—are not always amenable to algorithmic analysis. Different criminals use different strategies to launder their money; and mixes are designed to be difficult to deal with.

We surmised that a good visual representation of the data might help us to spot patterns as well as to make the dataset size issue go away. Moreover, it could possibly make the taintchain more usable – you could just enter your txhash and follow the taint.

3.4.1 Preliminary model

Our first prototype used a simple graphical model for our taintchain data. We represented each transaction as a vertex and each hop as an edge. By hop, we refer to the output of a transaction that has been used as an input somewhere else. Then we looked to represent our graph sensibly on-screen.

We decided to retain the chronological order and represent blocks as columns of transactions. Each transaction is a coloured rectangle where the colour reflects the kind of taint, and the size of rectangle reflects the number of satoshis tainted. Lastly, we decided to ignore clean satoshis as the data were sparse and required too much scrolling.

[7]If we want to retain provenance information about why a satoshi ended up being tainted, we would have to mark each individual instance of a criminal act as one kind of taint.

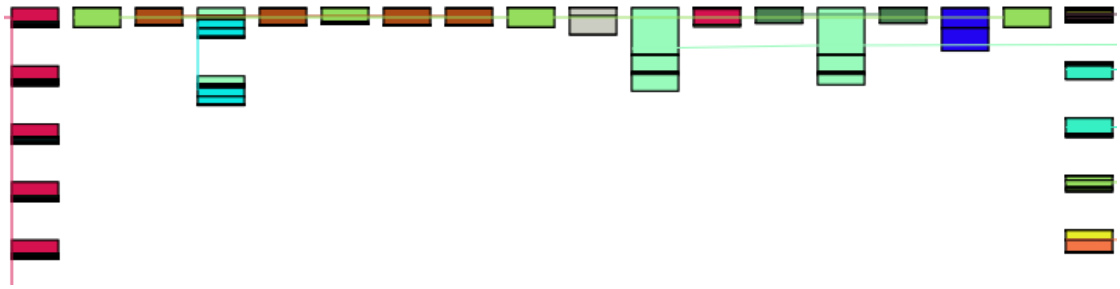


Figure 3.2: An illustrative image from our preliminary visualisation showing multitaint movement.

We displayed this model as a static SVG graphic with click-to-reveal txhashes. Figure 3.2 shows an example.

To our surprise, even this rudimentary model gave us good results. We were able to spot quite a few interesting patterns via the visualisation that we would not have been able to see otherwise. For example, Figure 3.3 shows someone collecting crime proceeds, that they had initially split to many addresses, into a single address. We call this a collection pattern and we observed similar patterns many times; in some of the instances, we were able to connect the collection address to illegal gambling sites.

Figure 3.4 shows the converse of a collection pattern: a splitting pattern. These may occur close to the time of a crime as criminals try to cover their tracks by feeding their loot into systems that divide their winnings into hundreds of tiny outputs.

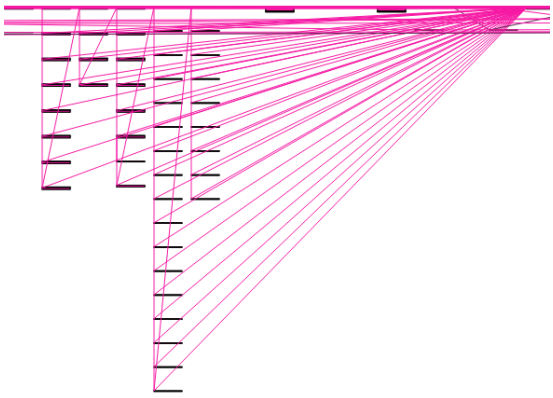


Figure 3.3: A collection pattern

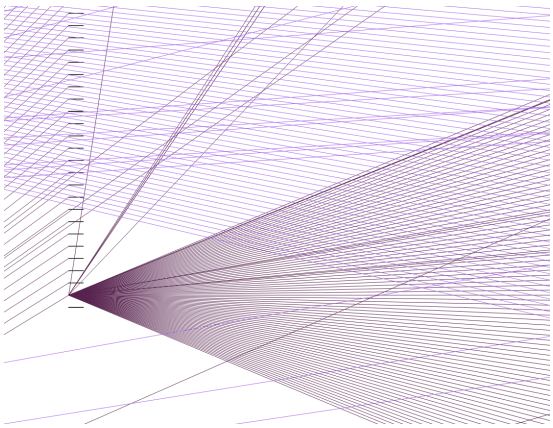


Figure 3.4: A splitting pattern

3.4.1.1 Limitations of the preliminary model

One of the main problems we faced was sheer data density. In Figure 3.5 we are displaying only four kinds of taint and yet it is strenuous to follow the many lines. Increased spacing is not a solution here as that would result in an unmanageable amount of vertical scrolling.

Another problem we faced was that taint tends to overlap, as shown in Figure 3.6.

In that case, do we retain just one colour? Or do we create a new colour to represent the combination? The answer isn't obvious.

Figure 3.5: Transaction density. The sheer number of tainted transactions renders some sections of the taintgraph uninterpretable.

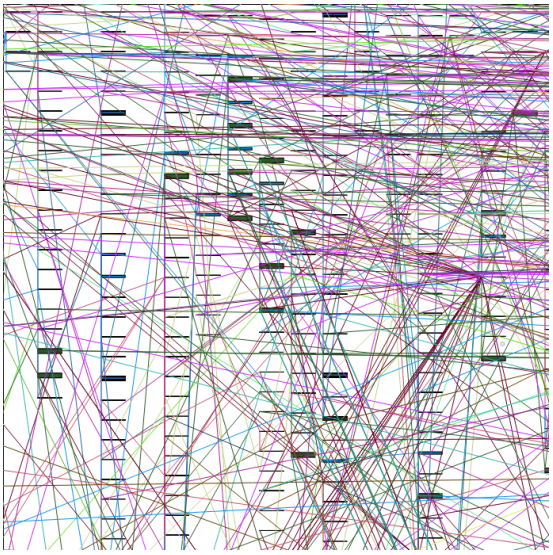


Figure 3.6: Complex collection pattern. We can see here the attempts by various actors to collect funds. However, this is difficult to spot due to the high degree of co-location of transactions.

3.4.2 Interactive visualisation

We therefore decided to rethink our approach. The second prototype makes the graph interactive so the user can choose which information is relevant to her on the fly. Secondly, we decided to make the edges more meaningful. Rather than just show a connection between nodes (and the associated taint colour), we incorporated the proportion of satoshis transferred in each hop into the edges. Lastly, we decided to abandon displaying the blocks as columns of transactions; instead we now focused solely on the transaction flows and included the block information as a hint box displayed on mouse hover. Thus, the depth of a vertex now does not relate to its absolute chronological position in the blockchain.

One of the problems that immediately vanished by the move to interactive representation was that of taint overlap. In our new system, we simply included a drop-down menu where the user can choose the taint type of interest and the graph adjusts its edges accordingly. Figure 3.7 shows this in action.

Making the graph interactive came at a cost, though, since now we want to store as much of the taintchain in RAM instead of on disk for greater responsiveness. Second, since the graph expands on click, random exploration could lead to many uninteresting paths being followed.

Nevertheless, we discovered some interesting patterns using this visualisation. We

Figure 3.7: These screenshots illustrate how the graph dynamically changes based on the taint type currently selected.

were able to find multiple instances of *peeling chains*, as shown in Figure 3.8. These are often used by exchanges or gambling sites – in this case a notorious criminal exchange. Its operators would pool their money into a single wallet and then they would pay their customers in turn, each time sending most of it to themselves at a change address. In this case, we can also see that this criminal exchange tried to hide their identity by shuffling their keys four times.

This interactive model has its drawbacks too, however. A fundamental issue seems to be the large outdegree of some transactions. A transaction can have an (effectively) unbounded number of outputs, which makes visualisations difficult. Figure 3.9 illustrates this difficulty. One possible solution is to have a filter for transactions: collapse all the outputs below a certain threshold. This would give a cleaner display image, but might hamper investigations.

Figure 3.8: A peeling chain used by a criminal exchange, discovered by following the larger branch at each vertex. Notice the sequence of four 1-to-1 transactions here: an attempt to cover up the exchange's identity by shuffling keys.

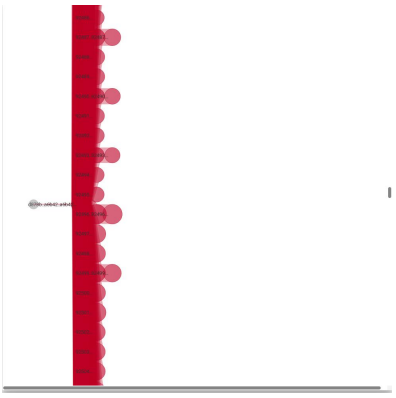


Figure 3.9: Exhaustive vertical scrolling due to high outdegrees of transactions. Notice the scroll bar on the right.

3.4.3 Other visualisation tools

A number of attempts have been made to visualise the Bitcoin network, with most of them focusing on some specific task. Early attempts were concerned with simple property representations e.g. Reid and Harrigan featured loglog plots of graph centrality measurements, graph representations with sizes of nodes showing the amounts of money transferred, geographical activity acquired through IP address mappings from Bitcoin Faucet, and graph representation of poison tainting [181].

Later came systems like BitIodine with graph-like outputs to support commonly available graph representation tools [196]. Graph approaches to transaction visualisation were also adopted for educational purposes by systems like CoinVis [1], while bitcoin-tx-graph-visualiser used alluvial diagrams to show Bitcoin movement [146].

A more mature system was BitConeView, presented by Battista and Donato in 2015 [37]. This was among the first to provide a GUI to inspect how a particular UTXO propagated through the network. In order to explain what it means for money to move, the authors came up with 'purity' – basically a version of haircut tainting. They evaluated the usability of their system informally, and came to the conclusion that more improvements were necessary to the way purity was presented to the user.

Our focus was on data representation of taint propagation when a taint graph becomes too massive for humans to comprehend. Unlike BitConduit and similar systems, we did not do any actor characterisation [138]. The generation of graph colours (and their implications) is exogenous, relying on external theft reports or on software that analyses patterns of mixes, ransomware and other undesirable activity.

3.4.4 Future work for taint visualisation

The tool we created led us to find some interesting patterns such as the peeling chain as well as splitting-collection cycles. Some companies, such as Chainalysis, have seemingly taken inspiration from our tool (or independently arrived at a similar solution). In their most recent report (released two years after our paper), Chainalysis also reported finding splitting-collection patterns and noted how they use these patterns in the provenance of a bitcoin to detect suspicious transactions [60, Pg. 49]. We are glad to see these insights being put to use in real systems.

That said, the visualisation tool still suffers from a number of shortcomings that invite further work. One avenue for research would be to explore different heuristics to portray the data more concisely. One might aim at a system that presents a global, zoomed-out view of the data and successively introduces more information as the user explores a particular pattern on the blockchain. Another direction would be to highlight suspicious patterns of transactions automatically, for example, by marking coins that have recently emerged from a flurry of splits and merges. There are many other plausible heuristics to explore, a lot of data to analyse, and real social problems to tackle that may prove fruitful for HCI researchers and criminologists alike.

3.5 Understanding the theft reporting ecosystem

While our FIFO tracking system gave us interesting insights, we wanted to have real-world impact and help victims of cybercrime to the greatest extent possible. To that end, we first looked into the practices of commercial cryptocurrency due-diligence companies. We found a set of companies to look at via recommendations from industry insiders (many of whom were attendees at Financial Cryptography 2019) and by looking at which firms were being used by popular exchanges (if any). Then, from this set we filtered down to those that allowed individuals to purchase due diligence reports and used our personal funds to get reports on well-known tainted addresses.

3.5.1 Incentives of the taint tracking ecosystem

Existing taint-tracking services appear to have two principal types of customers: the first consists of law-enforcement and intelligence agencies, who typically focus on serious crimes such as underground drug markets and multi-million-dollar hacks of exchanges[8]. The second consists of exchanges and financial institutions who want to be able to demonstrate that they exercised due diligence when acquiring cryptocurrency assets.

The second set of customers are purchasing due diligence, which is well known to suffer from perverse incentives [24]. Lobbying pressure from financial institutions leads to risk management morphing into standardised due diligence procedures that can be

[8]The leading service, Chainalysis, was set up in an attempt to recover bitcoins stolen from Mt. Gox in the first major heist in 2013.

applied mechanically – of which the standard requirement that new bank customers show a passport and two utility bills is a good example.

We therefore made a number of test purchases of AML reports on specific UTXOs which we identified as suspect. In one case, a 'Standard AML/KYC Risk Report' assessed a tainted coin as 'medium risk', noting 'illicit activity risk' (but giving two risk levels of 64% and 11% with no explanation), and unquantified 'Danger detected' for 'transactions impeding track of funds' and 'transactions with distinctive patterns'. Other reported categories for which danger was detected included cybercrime risk, industry risk and connected parties. Yet this coin contained a significant component that had been publicly reported as stolen, and the report was oblivious to the fact. In a second case, a checking firm returned 'scam alert: none' to one of the main Cryptolocker ransomware addresses and also to the main Sheep Marketplace theft laundry address. In a third case, a checking service gave the all-clear to an address being used by cryptomining malware distributors on an underground forum scraped by colleagues at the Cambridge Cybercrime Centre.

When we asked one firm why they stopped publishing negative recommendations and removed old ones from their websites, they said they "wouldn't match risk appetite of every user thus we can only provide risk assessment and leave the decision to the user." In short, the due-diligence market is not just a market for lemons, but one in which many customers show symptoms of information avoidance [106].

The incentives facing firms who supply blockchain intelligence to law enforcement are better. If hundreds of online test purchases of drugs provide evidence of drug dealers laundering their proceeds through an unregulated exchange such as BTC-e, this may provide probable cause for a warrant. And indeed the sales pitches of such firms (e.g. Bitfury [44]) target major crime.

But there are still shortcomings. The leading police and intelligence agencies tend to focus more on big busts, rather than on protecting ordinary consumers. This is already a problem in frauds using normal banking and payment systems; despite the fact that most property crimes in developed countries are now frauds rather than burglary or car theft, the resources devoted by most police forces to 'cybercrime' are tiny and they push crime victims to complain to their bank when they can, or even blame the victim for the crime [15]. Given the common police view that bitcoin users tend to acquire cryptocurrency with a view to buying drugs online, it is even less likely that they will bestir themselves to help ordinary bitcoin crime victims, and we have come across no sign of such enforcement action. If ordinary people are going to use cryptocurrencies at all, how can they protect themselves?

This is why we decided to make TaintChain public. We hoped to facilitate the emergence of an open crime-tracking community, first, as a resource for innocent bitcoin users to check out coins they're offered in payment; second, as a resource for small law-enforcement agencies who don't have the budget to buy in specialist services; third, as

a platform for academics studying cybercrime; and fourth, as a means of mitigating the lemons market in due diligence. After we wrote the first technical paper with some early results [17], we publicised it with a Computerphile video [16], and waited for some theft reports to roll in with the hopes of getting more on-the-ground data points.

3.5.2 Theft reporting in practice

We didn't have to wait for long. We were contacted by several victims of theft as well as by companies interested in refining their tracing systems. Talking to real victims and looking at real theft cases led us to radically amend our view of the cryptocurrency world. With one exception, the victims we talked to were all using hosted wallets[9]. So rather than downloading wallet software and running it on their own machine, they had gone to an online service—typically a firm that was also an exchange—and exchanged their dollars, euros or pounds for Bitcoin. When they logged on, a balance was displayed to them, and they could spend it by entering a payee and an amount, just like at a conventional bank website.

In one case (one of the thefts from Mt. Gox) the theft was apparently by an insider. Our complainant reported a bitcoin balance that amounted to thousands of dollars at the time had simply gone to zero, with an attacker presumably having intercepted the password or bypassed the password-checking mechanism. The outgoing transactions for that day include a set of four equal transactions, closely spaced in time, equal to the missing amount. That is the extent of the traceability we can offer by looking at the blockchain. The liquidators of Mt. Gox have shown little interest in such small cases.

Other cases are similar although it is generally less clear whether the compromise resulted from a customer's credentials being guessed, or stolen by malware, or whether there was inside collusion. In no case could we find any clear documentation of the actual ownership of the missing cryptocurrency. On inspection, this observation opens up a number of cans of worms starting with the nature of ownership of Bitcoins in the current ecosystem.

3.6 How the market really works now

In the traditional self-hosted model, each user would hold a *wallet*. This is a software program that stores and utilises private keys that correspond to addresses with unspent UTXOs. Thus, a Bitcoin users 'bank account' is her wallet which gives her access to all of her bitcoins in the form of unspent UTXOs.

One would assume that the hosted wallet of an exchange customer behaves in a similar fashion. However, even in early exchanges, a well known security measure was used which made hosted wallets behave differently: namely, the use of 'cold' and 'hot'

[9] At least at the time of the theft; one had BTC 42 in a desktop wallet, and after he transferred it to a hosted wallet, it was stolen.

wallets. Exchanges would keep most of their customers' bitcoins in offline machines (cold wallets) and transfer to and from them periodically to online machines (hot wallets) used for actual trading. This meant that the hot wallets would have enough coins to transact but not so much as to pose a catastrophic theft risk.

If that were the only optimisation introduced by the exchanges then it would matter little for coin tracing. If the bitcoin I bought from, or deposited at, an exchange were kept faithfully for me and made available for me to spend when I wished, then a stolen coin I received would still be traceable through my hands when I spent it later. This may have been the case at the time of Mt. Gox, but it does not appear to be generally the case now.

3.6.1 Who owns the bitcoin stock anyway?

There are two basic models for an institution to hold value on behalf of a customer. The first is the gold merchant. If I pay £44,000 for a 1Kg bar of gold and paid the merchant to store it for me in their vault, the merchant would place a sticker on that bar in his vault with my name on it[10]. If the merchant went bust, I could turn up at the vault with my paperwork and collect the gold from the administrators; it was my gold after all, and the company was merely keeping it for me.

The second model is the bank. If I had placed my £44,000 at HSBC, then the bank does not stick my name on 2,200 £20 notes; it merely owes me the sum of £44,000. If it goes bust, I have to stand in line with all the other creditors to get my share.

Similarly, there are basically three ways you can buy and hold cryptocurrency.

1. You buy it from an exchange and get them to transfer it to your own wallet which is resident on your computing device (or dedicated hardware wallets) and that contains your private key(s). This is the equivalent of collecting your gold from the bullion dealer.

2. You buy it from an exchange and keep it there in a hosted wallet where the exchange holds the private key(s) on your behalf but the cryptocurrency actually resides in that wallet, in the sense that the keys are available to no other customer. Here the exchange actually has control over your keys and executes transactions on your behalf. This is the equivalent of the gold merchant who keeps identifiable and marked gold bars on behalf of customers. You can buy, hold and sell gold without physically taking possession of it, and you can even order it to be transferred to the account of a different customer of that merchant, but it is identifiably and legally yours. We will call this '**the gold merchant model**'.

3. You buy it from the exchange and keep it in an account where you have a claim against a certain amount of cryptocurrency that the exchange is holding in its

[10]Nowadays the bars have QR codes

own wallet on behalf of all its customers. In other words, your balance is off-blockchain and intermediated by the exchange. The exchange simply runs an account for customers which is backed by the exchange's assets. The exchange might not actually possess assets that correspond exactly to its liabilities to its customers; it might lend cryptocurrency to other exchanges, trade in futures and options, and so on. The exchange may also offer transaction services whereby they will remit various cryptocurrency amounts, at your mandate, to the internal or external accounts of other parties. In other words, the exchange is operating as a bank. We call this '**the bank model**'.

In order to understand which model of ownership is being used in popular exchanges, we looked at the accounts filed by the leading UK exchange, Coinbase. It consists of two companies, CB Payments Ltd., which holds customers' fiat money balances and is now regulated under the E-money Regulations (see § 3.6.3), and Coinbase UK Ltd. which handles digital currency and is not regulated. According to accounts filed at Companies House [68, 69], the first of these companies shows a net profit of £481,000 in the year to December 2018 (the latest, at the time of writing) and net current assets of £6,935,000. The second company is more substantial with a net profit of £6,568,000 and net current assets of £8,156,000. Such accounts have been filed for several years and contain no record of the exact amount of cryptocurrencies held by either company.

Of course, the UK Coinbase companies are part of a larger group, so perhaps all the digital currency assets are kept by the US parent. A recent press profile of Coinbase emphasises its commitment to compliance and notes that it has $20bn in assets under management [97]. Nonetheless such a small balance sheet would be considered odd in a UK bank with an overseas parent. If the total market cap of Bitcoin is £300bn, and the UK's share of that is in line with its 5% share of world GDP, and Coinbase has a third of the UK market, then we'd expect to see a balance sheet of £5bn, not £15m. Alternatively if the UK is 20% of the size of the US market and Coinbase has the same share in both, we'd expect to see $4bn. In short, we're out by two orders of magnitude. Looking for a hint, we note that Coinbase claims that all customer funds are kept in its cold wallet, with only 1% of the total being in its hot wallets for trading at any one time, and that this 1% consists of its own reserves [97].

It is curious that we see no trace of customers' pooled assets on the Coinbase balance sheet, which does not look anything like that of a bank. Perhaps the assets appear on the balance sheet of a different group company, or perhaps Coinbase has transitioned from being like a gold merchant to being like a bank in the months since the last accounts were filed. Certainly Coinbase goes out of its way to present itself as the good guy in the Wild West of cryptocurrency and we are not imputing any impropriety whatsoever. But if even the best actors fall short of the standard of transparency normal in legacy banking, this raises further questions, to which we will return in § 3.7.

3.6.2 Off-chain transactions

So, in practice, the transfer of bitcoin from person to person appears to be more like this: Alice goes to a bitcoin exchange and pays it (say) £2000. The exchange gives her BTC 0.07 and displays this balance as being available to her to spend. If Alice now orders a payment of BTC 0.05 to Bob, then the exchange looks to see whether Bob is also a customer. If so, then the transfer is just a ledger entry; the balance seen by Alice reduces to BTC 0.02 while Bob's increases by BTC 0.05. This is known in the trade as an 'off-blockchain' or 'off-chain' transaction. These appear to have become the default over the period 2016–20.

The idea that off-chain transactions might become the norm was in fact first mooted by Bitcoin pioneer Hal Finney: " Bitcoin itself cannot scale to have every single financial transaction in the world be broadcast to everyone and included in the block chain... Most Bitcoin transactions will occur between banks, to settle net transfers. Bitcoin transactions by private individuals will be as rare as... well, as Bitcoin based purchases are today." [79]

Getting hard data on the scale of off-chain transactions is hard. Demeester reports that Western exchanges do $80m in off-chain transactions per day [79]; while charts by Cryptovoices show trading volumes per on-chain transaction taking off from early 2017 and showing peaks in the range of 6 to 14 times [73]. There have been various attempts to create off-chain payment mechanisms between exchanges but it appears, talking to industry insiders, that the great bulk of off-chain payments (at least for bitcoin) are between customers at the same exchange. One of the drivers appears to have been the massive congestion in the blockchain in late 2016, when transactions could be pending for a day before being mined into the blockchain and transaction fees hit $50; now many blocks are partly empty and mining fees are near zero. All such figures need to be treated with caution: Ribes investigated various bitcoin exchanges via test transactions and concluded that the largest exchange at the time was faking 93% of its trading volume [183].

In effect, crypto-currencies have morphed into an unregulated shadow banking system. While this may have initially been driven by congestion, it has a secondary effect of consolidation: network effects appear to be pushing particular communities to consolidate around specific exchanges. Many Bitcoin users in the USA and the UK use Coinbase, while Chinese speakers are more likely to use Binance, Japanese use bitFlyer and South Africans use Luno. It's convenient to use the same exchange as your counterparties: transactions are instant and fees are much lower.

Another recent development that is bound to make blockchain analysis more opaque is the development of off-chain *payment channels*. Payment channels allow Bitcoin users to only commit a very small subset (usually two: the first transaction is to put a "stake" or collateral into the payment channel and the second is to cash out the collateral plus/minus any transfers to/from the channel) of their total transactions to the

blockchain. These do not rely on trusted third parties like exchanges but rely on collateral put in by all parties as an economic incentive for good behaviour with the blockchain only used in case of disputes among the parties. The actual "Alice to Bob" transfers within a payment channel happen completely off-chain and payment channel systems can contain many entities. We refer interested readers to the systematisation-of-knowledge paper by Gudgeon et al. for an introduction to the field [109].

The benefits of these off-chain mechanisms are clear: they reduce congestion on the network, they have lower latency and lower transaction fees. The concern is that they further exacerbate the opacity of the Bitcoin network. If payment channels become the norm, one can expect to see even fewer transactions appearing on the blockchain at all. This is even worse (from a transparency standpoint) than the off-chain transactions mediated by cryptocurrency exchanges because in this case there is no exchange to serve a warrant on when the need for investigation arises. The lack of any such regulated entity also makes it difficult (if not impossible) for researchers to get a grasp on the popularity of payment channels as well as their usage in cybercrime. We will return to this issue of payment channels when we discuss privacy-preserving cryptocurrencies in § 3.7.

3.6.3 The E-Money Directive

The fact that substantial transaction volumes are now handled off-blockchain raises the issue of whether financial regulators in Europe should require exchanges to comply with the E-money Directive of 2009 [83]. According to this, "electronic money" means "electronically stored monetary value as represented by a claim on the electronic money issuer which is issued on receipt of funds for the purpose of making payment transactions; is accepted by a person other than the electronic money issuer; and is not excluded by regulation".

This regulation seeks to ensure, inter alia, that an issuer of prepaid debit cards has and maintains enough assets to back the credit balances on the cards that it currently has on issue. Exactly the same problem arises with bitcoin exchanges: what is to stop an exchange taking my money and displaying to me a credit of bitcoin (or other cryptocurrency assets) that it does not actually have? What is to stop an exchange selling $200m worth of bitcoin but buying only $100m in actual bitcoin, taking out the other $100m as dividends for its shareholders, and hoping to get away with it for a while? The rate at which exchanges have gone bust should warn regulators that this is a real risk.

The text of the E-Money Directive appears to describe an exchange's transaction processing business well. So do financial regulators make exchanges comply with this Directive, via the regulations that implement it in each Member State? The answer appears to be no. In the UK it is up to the Financial Conduct Authority to instruct the Payment Services Regulator to apply the E-Money Regulations (2011) to particular

payment systems; the Regulator told us in 2017 that as the FCA has not instructed her to regulate cryptocurrencies, she only applies the Regulations to the conventional currency balances kept at UK bitcoin exchanges. We will return to the FCA's position in § 3.6.5. Meanwhile, their reluctance to regulate anything other than the fiat money component of a transaction is exploited by the exchanges. Coinbase's terms and conditions [66], for example, make a clear distinction between 'E-money services' which relate to customer sterling balances, are regulated, and are provided by CB Payments Ltd., and 'digital money services' are provided by the separate company Coinbase UK, Ltd. We are warned "*You should be aware that the risk of loss in trading or holding Digital Currencies can be substantial ... Digital Currency Services and Additional Services are not currently regulated by the Financial Conduct Authority, the Central Bank of Ireland, or any other regulator in the UK or in Ireland.*"

The situation in Germany is similar, but with different details. The regulator, BaFin, has held back from imposing e-money regulation on virtual currencies (the term used in the EU) with the argument that they do not represent any claims on an issuer; as there is no issuer, they are not e-money within the meaning of the German Payment Services Supervision Act (Zahlungsdiensteaufsichtsgesetz). Bitcoins are however financial instruments, units of account like foreign exchange with the difference that they do not refer to a legal tender[11] [31]. BaFin does note that "Those buying and selling VCs commercially in their own name for the account of others carry out principal broking services which are subject to authorisation" and remarks in passing that "In practice, VC undertakings often did not offer detailed explanations as to how they work at all, or did so in a vague manner. In many cases, no general terms and conditions were provided." And there has been enforcement action: BaFin has issued cease-and-desist notices to ban the promotion of the 'OneCoin' trading system in Germany [30] and an unlicensed broker, Crypto.exchange GmbH [28].

The OneCoin case is particularly interesting because the cease-and-desist order related to the company's not having an e-money license in respect of Euro remittances made within Germany to acquire Onecoins [29]. In that case, players in the system were 'merely adjusting balances' to transfer funds. In any case, an institution providing off-blockchain transactions at scale would appear to fall under §1.1.5 of the German Payment Services Supervision Act as they are "enterprises that provide payment services either commercially or on a scale that requires a commercially equipped business operation" [112].

In short, in both the UK and Germany, the law empowers the regulator to require that digital currency operators who settle payments by means of off-blockchain transactions to register under the E-Money Directive, yet they have so far neglected to do so. Perhaps the cryptocurrency scene is simply moving too fast for them or perhaps the scale of the cryptocurrency enabled crime is not large enough yet. Once they catch up –

[11] This could change if some state were to declare a virtual currency to be legal tender, as El Salvador has recently done.

perhaps being forced to act by some scandal – the tools already exist. The UK E-money Regulations, for example, provide two years in prison for operating an e-money service without a license[12].

Once we realised that regulators were failing to apply applicable law to tackle the risks around off-blockchain transactions, we made a submission to the UK Parliament's Treasury Committee describing these risks and recommending that the E-money Regulations be applied to exchanges' digital currency services as well as to their customer balances in fiat currency [18]. We amplify that recommendation below, along with others on which our thinking has developed since our submission to Parliament.

3.6.4 Directive PE CONS 72/17

On May 12th 2018, the European Union published Directive PE CONS 72/17 [211], with the snappy title of *'Directive of the European Parliament and the Council amending Directive 2015/849 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing, and amending Directives 2009/138/EC and 2013/36/EU'*. This was agreed quietly between the European Parliament and the Council (the Member States) in April 2018, and it somewhat changes the regulatory landscape. Although it is justified as an anti-terrorism measure, it will have implications for consumer protection.

In December 2017, the Commission had signalled that regulation would be extended from exchanges to wallet hosting services. The new Directive does this but in a way that leaves a significant loophole. The new Directive has, in article 2 (d), a definition of a 'custodian wallet provider' which is just about services that hold cryptographic keys. Recall that in § 3.6.1 we described two models of exchange wallet operation: the gold merchant case where the wallet provided by the exchange to its customer contains merely the cryptographic keys needed to sign transactions with the customer's own cryptocurrency assets, and the bank case where the customer merely has a claim on the exchange's asset pool. This definition covers the gold merchant case but fails on the bank case.

The Directive says at recital 10 that virtual currencies (as the EU calls cryptocurrencies) should not be confused with electronic money, since although they can be used for payment, they can be used for other things too. This text does not exclude the application of the E-money Directive to off-blockchain transactions but may be used to confuse matters and argue that exchanges should continue to have a regulated business for fiat e-money balances and an unregulated one for digital currencies.

The Directive clarifies that the definition of electronic money is that given in Directive 2009/110/EC: "electronically, including magnetically, stored monetary value as represented by a claim on the issuer which is issued on receipt of funds for the purpose of

[12]There are a few surveys of the regulatory status of cryptocurrencies in various countries that interested readers would find useful. The latest is from the Cambridge Centre for Alternative Finance [45] which compares the regulatory attitudes of 23 jurisdictions

making payment transactions as defined in point 5 of Article 4 of Directive 2007/64/EC, and which is accepted by a natural or legal person other than the electronic money issuer". That seems to cover off-blockchain payments fair and square, and to our mind on-chain payments too. There is also a definition of "virtual currency" as "a digital representation of value that is not issued or guaranteed by a central bank or a public authority, is not necessarily attached to a legally established currency and does not possess a legal status of currency or money, but is accepted by natural or legal persons as a means of exchange and which can be transferred, stored and traded electronically."

However most of the substance of the new Directive consists of detailed amendments to the Fourth Anti-Money-Laundering Directive which can only be understood by painstaking cross-reference to the original. Some of the intentions are clear enough, such that there should be centralised systems recording the relationship between addresses and identified holders, which can be queried automatically by investigators on the trail of money laundering or terrorist financing (recital 21). Of real importance may be section 6: " Member States shall prohibit their credit institutions and financial institutions from keeping anonymous accounts, anonymous passbooks or anonymous safe-deposit boxes". The Directive also requires better public disclosure of the ultimate owners or beneficiaries of companies and trusts.

The lawgiver has in this case been contemplating only the money-laundering aspects of bitcoin exchanges, and not the fact that one can open an exchange and sell more bitcoin than they have. In addition to this consumer-protection risk there may also be a prudential risk: as some Member States (notably Malta but also Estonia and the UK) try to market themselves as natural homes for cryptocurrency innovation, there will be a temptation to race to the bottom at the cost of decreased transparency.

3.6.5 Positions of UK stakeholders

The UK parliament's Treasury Select Committee called an inquiry into digital currencies to which many interested parties made submissions in April 2018. Following oral hearings and written submissions, the formal report was published in September 2018. The submissions make for interesting reading.

We already noted that although off-chain transactions appear to fall squarely under the EU E-Money Directive and the UK E-Money Regulations, the Payment Services Regulator can't apply them as the Financial Conduct Authority (FCA) has not asked her to. The FCA explains its position in its Treasury submission [89]. It follows the definition in EU Directive PE CONS 72/17 in that it sees wallets as storing keys; there is no recognition or mention of off-chain transactions in the set of operations around cryptocurrencies that may or may not be regulated, and like the European Commission sees wallets as simply storing the customer's cryptographic key. It does not use the word 'currency', or even the EU term 'virtual currency', preferring its own term 'crypto-assets' – which further helps ignore off-chain transactions. It claims 'Where crypto-assets

form part of regulated services, regulated firms can take steps to mitigate the money laundering risks'. This may be somewhat optimistic given that Coinbase has separate firms for fiat money and crypto and carefully states in its terms and conditions that only the former is regulated, but the FCA is not too worried: unlike the EU, it sees the money-laundering risk as mostly in 'non-crypto-asset typologies'. This position brings to mind the literature on information avoidance [106]. The FCA appears to be shying away from a problem it should fix but which would complicate its mission. If it wants 'crypto-assets' to be treated exactly the same way as shares in Tesco, then it should forbid regulated exchanges from providing any service that allows one customer to transfer them to another directly as a means of payment, but it does not.

The FCA is not the only institution that just doesn't want to know. The UK Financial Reporting Council, in its submission [102], discusses the difficulty of valuing crypto-assets. They should be valued at market if they are financial assets, but they don't meet the definition; so they have to be valued at cost as commodities, unless we change the rules to treat them like gold. However, this is not on the agenda of the International Accounting Standards Board.

3.7 Policy recommendations

So regulators are just not managing to keep up, and policy perspectives have changed hugely in a few years. The 2015 survey of bitcoin economics, technology and governance by Böhme et al. now seems to come from a different century [46]. The number and scale of the scams together with the environmental harm caused by mining have led to an increase in concern among governments with central bankers pushing them in favour of regulation [45], but so long as this is based on an outdated view of the problem it's not likely to be optimal. In this section, I discuss the recommendations we made in 2018 which we were invited to present at a number of law and economics venues, and note how the ecosystem has changed in the intervening months.

3.7.1 Regulated exchanges

The main recommendation we made in our 2018 analysis was that governments should regulate exchanges based in the EU, or that do business with EU citizens, and which offer off-blockchain payments or consolidate cryptocurrency assets rather than merely holding crypto keys on behalf of customers, in respect of all these cryptocurrency assets under the E-Money Directive. Off-chain transactions, at the very least, fall within the definition of e-money and are vulnerable to exactly the kinds of scams and payment service failures that the E-Money Directive was established to prevent.

If regulators continue to believe that cryptocurrency exchanges fall outside the definition of e-money as per the E-Money Directive, then we will need a similar directive to tackle the same problems. However, that seems like a waste of time and resources. The

EU has a workable piece of legislation; it and its Member States just need to enforce it.

3.7.2 Consumer protection

A crime victim who asks an exchange for a refund of stolen bitcoins that were taken from an account there can expect to be told that as digital currency is unregulated, they are out of luck.

But this is nothing new. In fiat banking, a customer who complains of phantom withdrawals from her account used to get into an argument with her bank who would stonewall her with something like 'Our systems are secure so you must have been negligent or collusive.' Yet the law eventually caught up in most countries. In the USA, early court cases paved the way for Regulation E and Regulation Z which provide much of the consumer protection on which bank customers rely in card transactions [65]. In the EU, the Payment Services Directive requires that the contract terms governing the use of the payment instrument must be 'objective, non-discriminatory and proportionate' (article 69), and where a transaction is disputed, 'it is for the payment service provider to prove that the payment transaction was authenticated, accurately recorded, entered in the accounts and not affected by a technical breakdown or some other deficiency' (Article 71) [82]. Crucially, 'the use of a payment instrument recorded by the payment service provider, including the payment initiation service provider as appropriate, shall in itself not necessarily be sufficient to prove either that the payment transaction was authorised by the payer or that the payer acted fraudulently or failed with intent or gross negligence to fulfil one or more of the obligations' (Article 72). European law not only agrees that payment records are not constitutive of title to money; it also imposes reasonable constraints on what may be expected of users. Simply saying 'you should have chosen a better password' won't do; neither will 'the blockchain now says that your money belongs to Fred'.

At this point the provider's terms of service may say 'you can't sue us' while consumer-protection law holds such contracts to be unfair. Again, the Payment Services Directive comes into play, and there are other laws too around unfair contract and product liability. These can give some clarity if policy degenerates into a tussle over the burden of proof.

So our **second recommendation** was that the relationship between an exchange and its customer should be covered by the second Payment Services Directive.

3.7.3 Unregistered exchanges

Unregistered and downright criminal exchanges are an issue. Suppose that you were hit by the Wannacry ransomware, had paid a ransom, and wanted to get your money back. According to the US government, Wannacry was the work of North Korean government agents, but this isn't much help. You note from the BitFury report that almost all of the bitcoin collected by Wannacry was laundered through the HitBTC exchange, so you

want to serve a court order on them (whether for compensation, or merely to see the passport presented by whoever cashed those coins). You then find that their website does not contain a physical address for service, contrary to the E-commerce Directive, Article 5.1(b) of which requires " the geographic address at which the service provider is established" to be provided. A simple search reveals that others, including disappointed customers, have sought this information repeatedly. HitBTC does claim to abide by FATF rules, so where is it registered as a money service business? The Directive requires at 5.1(e) that it publishes "where the activity is subject to an authorisation scheme, the particulars of the relevant supervisory authority" yet there is no sign. It should perhaps surprise no-one that HitBTC is on Ribes' list of exchanges that appear to significantly overstate their trading volume; he uses the word 'fraud' [183].

HitBTC is believed by some in the industry to be run by criminals in Russia. If it turns out that HitBTC is in a non-compliant jurisdiction, so it can't be raided and shut down, then conversations need to turn to sanctions, and whether regulated exchanges should be permitted to transact with such operators at all.

The concern around exchanges based in non-compliant jurisdictions has taken an increased importance in the 18 months since we initially published our recommendations. The recent Chainalysis study of crypto crime suggests that 52.2% of all illicit bitcoins in 2020 went through just two exchanges: Huobi and Binance [60, Pg. 10] taking the place of HitBTC as the primary crime havens. Binance moved to Malta, which is infamous in cryptocurrency circles for its lack of enforcement, after being banned in China [148]. Huobi seems to be have subsidiaries in many jurisdictions with the Hong Kong office serving as headquarters although the exact structure of the Huobi group is quite difficult to decipher from their released documents.

The guidance for dealing with such exchanges exists in the aforementioned Directive 2015/849 discussed earlier, which imposes a duty in respect of transactions involving high-risk third countries, which must be presumed to apply to HitBTC and the like. Article 11 requires EU institutions to implement a number of enhanced due-diligence measures on such transactions including getting more information on the customer, the beneficial owner, the nature of the business relationship, the source of funds and the reasons for the intended transactions. Moreover, the EU institution doing such a transaction must have it approved by senior management. It is hard to see how a UK exchange could discharge these duties in respect of a transaction to or from HitBTC.

Again, this is nothing new. Cryptocurrencies do not solve the underlying problems that made bank regulation necessary, and we can expect that many of the familiar second-order problems will also reappear in due course.

Our **third recommendation** was that regulators should prohibit the cryptocurrency exchanges they regulate from clearing and settling transactions with unregulated exchanges.

3.7.4 Innovation and the role of central bank cryptocurrency

Debate continues on whether bitcoin and cryptocurrencies have actually achieved anything other than emitting carbon dioxide and facilitating crime. Stinchcombe argues that ten years into its development, nobody has found a legal killer app for bitcoin yet: 'Each purported use case ... amounts to a set of contortions to add a distributed, encrypted, anonymous ledger where none was needed. What if there isn't actually a use case for the blockchain at all?' [199].

But the markets still believe otherwise, with Bitcoin's valuation reaching new heights at the time of writing. This surge in valuation can also be seen for Ethereum, a system similar to Bitcoin but with a more expressive scripting language that allows the creation of smart contracts (see § 2.2.2). Whether these can be legally valid contracts has been an issue of some debate.

Once again, we look towards precedent. As Raskin notes, 'innovative technology does not necessitate innovative jurisprudence' [179]. In fact, a decent starting point is the existing law on vending machines and on the starter interruptors used to enforce some motor vehicle credit agreements. But although smart contracts are nothing especially new, regulatory intervention may be needed in egregious cases. Attempts to hide contracts behind machines have failed in the past: an early vending machine was invented by a 17th-century book publisher, Richard Carlile, who did not want to be jailed for selling books considered blasphemous. He argued that the purchaser's contract was with the machine, not with him; the court didn't buy this argument, and sent him to jail. The fact that he flaunted his attempts to evade prosecution made the case an easy one for the court [179]. We can expect courts to be similarly unimpressed by contracts that are unfair, unconscionable or illegal; that are made using the visible proceeds of crime; or that are clearly contrary to public policy.

Both regulators and entrepreneurs should consider common-mode failure risks. People have noted for some time that bitcoin is not as decentralised as some of its promoters claim. Gervais et al. raised this issue in 2014 [103], and Narayanan et al. expanded on it in their book [162], noting that a number of players – from the Bitcoin Core developers through the mining cartels to the exchanges – have outsized power in the system. Vorick gave a fascinating account of an attempt to set up a mining equipment vendor, which revealed that Bitmain has a near-monopoly in the mining equipment market [214]; it apparently earned $4bn in 2018 [114, 215].

Indeed, as Narayanan and his coauthors noted, the amazing and noteworthy thing about Bitcoin is that it continues to operate as a (sort-of) global trusted computer despite having various parts of its kill chain controlled by vendors, miners, developers and exchanges. However many people expect a denouement sooner or later, and this is one of the reasons that central banks might consider a properly-engineered cryptocurrency to be worthwhile.

A quite different approach to Bitcoin is that being pursued by the Enterprise Ethereum

Alliance, who have adapted blockchain technology to work in closed groups. These *permissioned blockchains* seem to be gaining traction in enterprise settings and offer tangible benefits to companies (as we shall see in the next chapter). Nawaz, for example, describes a project at JP Morgan to use enterprise Ethereum to automate the clearing and settlement of financial assets – which would enable the financial institutions who are members of an exchange to manage the asset register collectively. This enables the common-mode failure risks, the risks of transacting with criminal counterparties, and the more traditional solvency and liquidity risks, to be managed transparently [131]. [13]

So how might central bankers help? Bitcoin promoters have hoped for some years that bitcoin would become fungible, in the way that coins are – one coin is as good as any other. One way of promoting fungibility was by providing mixes and other money-laundering facilities, but, as we have discussed, such facilities do not work very well and are counterproductive as they simply taint the laundered coins as being crime proceeds.

Another approach has been to argue that bitcoin should be money. If it is, then there are two exceptions to the *nemo dat quod non habet* rule: money, and bills of exchange. The simplest way for a cryptocurrency to become money would be for a central bank to issue it. If the Bank of England were to provide cryptocoins saying, as banknotes do, *'I promise to pay the bearer on demand the sum of £20'*, then anyone who holds such a coin would be able to rely on it[14].

A 'LegitCoin', for want of a working name, would thus have powerful advantages over competitors[15]: certainty of title, trust in it as a platform, and predictable value. The E-Money Directive would apply immediately and directly, as such a coin would have a defined value.

So why should a central bank issue cryptocurrency? The best reason, as we see it, is to support innovation by providing a platform for smart contracts whose tokens can be converted into real money at par. Firms promoting businesses based on smart contracts should not have to contend with a wildly fluctuating exchange rate between ether and sterling, nor with the uncertainty that comes from dealing with coins that may previously have been crime proceeds. Another reason for central banks to consider cryptocurrencies is to enable micro transactions by issuing coins directly to users: the potential for these to disrupt existing economic models is not diminished by having the coins issued by a bank versus having them issued by a mining farm.

One of the pieces of existing infrastructure that central banks might consider for smart contract functionality can be found in the Hyperledger project, a Linux Foundation hosted project that aims to provide a multitude of permissioned blockchain systems

[13]The proposal would also make the assets programmable, so that participants could offer futures, options and other derivatives of arbitrary complexity – which may raise other regulatory issues, but they are not our concern here.

[14]The general exemption from the *nemo dat* rule is bills of exchange, which include cheques, bills of lading, and indeed banknotes. We've kept the discussion to banknotes for simplicity. However if we end up with central banks issuing cryptocurrencies that support smart contracts for supply chain management, other bills of exchange will surely be constructed using them

[15]such as Facebook's Diem (previously known as Libra)

depending on the application. Other popular permissioned blockchain frameworks include Corda by R3, MultiChain by Coin Sciences and Quorum (created by JP Morgan and recently taken over by Consensys). We will look at these frameworks in greater depth in chapter 4.

Our **fourth recommendation** was that central banks consider issuing a cryptocurrency that supports smart contracts, has the legal status of a bill of exchange and is redeemable at par for fiat money. The use of permissioned blockchains could provide for a convenient mechanism for the dissemination of this cryptocurrency to institutions in a transparent manner.

This recommendation seems to be seeing real traction at the time of writing. Several countries are experimenting with so-called Central Bank-issued Digital Currencies (CBDCs). A recent (August 2020) large scale study by the Bank of International Settlements [27] shows that 36 countries have so far published work on CBDCs with several of them conducting pilot deployments. The authors also note the change in attitudes of central banks towards digital currencies: "In 2017 and 2018, many [central bank governors] had a negative or dismissive stance, particularly toward retail CBDCs. Since late 2018, the number of positive mentions of retail and wholesale CBDCs in speeches has risen, and in fact there have now been more speeches with a positive than a negative stance" [27, Pg. 8]. The authors suggest that the turn in opinion is due to the announcement of Facebook's Libra which has led to the public sector accepting the need for a sovereign digital currency in an increasingly cash-free world. In another survey of central bank attitudes [216], primary motivators for CBDC development appear to be the desire to modernise inter-bank settlements and to improve cross border transaction systems; permissioned blockchains appear poised to play a major role in both applications.

3.7.5 Nature of ownership

As we've seen, a serious issue with existing exchanges is that it is unclear whether the bitcoins in the exchange's cold wallet are owned by the customer (as with a gold merchant) or by the exchange (as with a bank). The regulator should force exchanges to make that clear in their terms and conditions. As we noted, exchanges used to act sort-of like gold merchants (in the days of Mt. Gox) and appear to act sort-of like banks now. The lack of clarity goes back at least to Mt. Gox. According to their 2012 terms and conditions, *'it (MtGox) will hold all monetary sums and all Bitcoins deposited by each Member in its Account, in that Member's name as registered in their Account details, and on such Member's behalf.'* [107] The comment of one of the victims to us was: "It does not state that customers were signing up to a fractionally reserved exchange, and so customers had the understanding that MtGox (albeit in separate cold storage) actually possessed the bitcoins which customers saw in their balances when they logged in."

Indeed, at present the fungibility of bitcoin seems to flow from the lack of clarity around ownership; although theft victims can trace stolen assets, they cannot establish

whether they actually owned these assets, and so cannot sue to get them back. Clarity will enable the victims to sue either the exchange of which they were a customer when the theft occurred, or the exchange in whose custody the bitcoins now rest.

A separate policy issue is the nature of ownership of a digital asset. Some assets exist by virtue of registration, patents being an example. With most assets, the *nemo dat* rule makes the situation more complicated. Cryptographers assumed that owning the private key associated with a bitcoin's address was constitutive of ownership, but the law does not accept this at all. If registration is to constitute ownership (as with patents) there had better be a law to say so; but, as we noted above, the EU Payment Services Directive says no such thing.

Legislation that made cryptography constitutive of ownership would violate a number of established rights and principles, as we discussed. It would complicate legal reasoning about intent, agency, liability and other issues that have already been discussed in the context of the law on digital signatures. Probably the most that might reasonably be done is to treat the signature as a rebuttable presumption of ownership, following the Electronic Signature Directive [81]. However that had such adverse effects on liability that qualified electronic signatures found only very limited use. Here, we merely flag up such issues as needing clarification, perhaps in the course of implementing the central bank study project we recommend above.

In any case, our **fifth recommendation** was that regulators compel exchanges to make clear in their contracts with their customers whether they are custodians of cryptocurrency assets that the customers own, or whether the assets are owned by the exchange with the customers simply having a claim on the asset pool.

It is natural for exchanges to try to avoid stating publicly whether they are trustees, banks or both, as either choice brings responsibilities. It is time for regulators to force them to choose.

3.7.6 Dark market currencies

A further policy issue is how to deal with cryptocurrencies that are explicitly designed to provide more substantial transaction anonymity or even unlinkability, such as Zcash and Monero, and also to identifiable persons promoting anonymity services on bitcoin and other public and address-identifiable blockchains. In the case of Zcash, the system works like bitcoin except that coin holders can have their coins re-mined, so that they become indistinguishable from other recently mined coins. The analysis in this chapter would suggest that when a tainted coin is treated in this way, all the coins then mined become tainted, and the victim would have a cause for action against any of their holders.

Similar concerns hold for payment channels although there exists an out: one could simply apply the FIFO tracing to the collateral and cash-out transactions used to establish the payment channel. This might result in some unfair repossessions since it may be possible that the victim's proceeds end up with someone who never even directly

interacted with the thief. Still, a strict reading of Clayton's case would lead us down that path.

Perhaps the victim, in both the Zcash and payment channel cases, could also sue the operators or promoters of such a system for negligence – in that they knew that some wallets would be stolen and yet designed a system that would make it impossible to get the money back. It's not obvious that the liability stemming from this negligence in fulfilling their duty of care would be extinguished by a legal precedent that declared ordinary, traceable, bitcoins to be money.

There is also the criminal matter of obstruction of justice, which might be used by prosecutors along with more specific offences relating to money-laundering and (in the case of organisations such as the Izz ad-Din al-Qassam Brigades [60, Pg. 73-78]) terrorist financing. This might perhaps be used against the promoters of systems such as Monero that provide unlinkability by default and that are widely used by mining malware. At the very least, the developers and promoters of such systems must expect to be held to a higher degree of accountability, and it would be beneficial for all if policy could be clarified.

A related policy issue is what the law should consider to constitute behaviour 'in good faith'. We have argued here that bitcoin mixes are certainly bad faith, and the use of systems like Monero might be held to count as such. This could also hold for payment channels though arguments could be made that the primary incentive for someone to use a payment channel isn't in hiding their transaction history but in the reduction of transaction processing time and cost; without a clear legal precedent, this is a grey area.

However the new anti-money laundering regulations may settle the matter. As noted above, article 6 requires that 'Member States shall prohibit their credit institutions and financial institutions from keeping anonymous accounts, anonymous passbooks or anonymous safe-deposit boxes'. A sensible transposition of the directive would discountenance anonymous instruments such as Zcash and Monero at least, if not payment channels as well.

Our **sixth recommendation** was therefore that regulators should prohibit exchanges from buying and selling cryptocurrencies that are explicitly designed to evade money-laundering and terrorist financing controls. Perhaps anonymity should be restricted to cryptocoins issued by central banks, so that controls can be ramped up later if the need arises or be made contingent on transaction amounts. We note that Coinbase won't touch Monero (though bizarrely, it still supports Zcash). Coincheck seems to have seen the danger in supporting these currencies and discontinued its support for Zcash in 2018 [172]. So, although the market might abandon these anonymous coins eventually, it might take too much time without regulatory nudges.

3.7.7 Capital requirements

If the only thing that could go wrong with a bitcoin was that it had been stolen, and all thefts were promptly and dependably reported, then a technically competent exchange can write scripts to fragment all incoming coins into clean layers and stolen layers. The payer could get value for the clean money, while the victims of theft get their money back and the drug money can go into the local asset-forfeiture pot. We call this *satoshi sorting*.

Satoshi sorting is not really a practical solution, though, for at least three reasons. First, there are issues other than theft, such as whether drug money or flight capital is to be considered tainted – and some of these questions vary by jurisdiction. Second, crimes are not always discovered and reported immediately; a big drug bust may result in the tainting of coins in transactions from months or even years ago. Third is the complexity of evidence. A victim of bitcoin theft may take time to establish that fact and a theft report might only get to the taintchain after years of litigation.

Thus valid claims against an exchange's cryptocurrency assets can arise for months to years after these assets are received. This risk cannot be managed by a clearing period and it follows that, if exchanges are responsible under the E-Money Directive, or equivalently under securities law, for ensuring that the bitcoin balances they sell to their customers are backed by cryptocurrency assets that are sufficient in quantity and quality, then they will have to keep a significant level of reserves.

In order to set appropriate standards for reserves, proper accounting standards are also needed. We noted that Coinbase – a leading exchange, which claims to be one of the good guys – has published accounts that do not reflect the assets under its control. In an ideal world, if Coinbase operates like a bank, we'd like to see its balance sheet look like a bank's balance sheet, and we'd like to have international standards for capitalisation and reserves.

Our **seventh recommendation** was therefore that regulators should require regulated exchanges to be adequately capitalised – and develop proper accounting standards to support this.

3.7.8 Mitigating environmental harm

Our final policy issue is serious and controversial: the "environmental disaster", as the Bank for International Settlements describes bitcoin mining. A detailed analysis by De Vries in 2018 put cryptocurrency mining energy use at between 3 and 8 GW, that is, between the energy use by Ireland and by Austria; he noted that the current economics would drive usage towards the latter figure [215]. He was right: the Cambridge Bitcoin Electricity Consumption Index (CBECI) reported in late 2020 that Bitcoin's annual energy consumption now exceeds that of Austria [57].

Given the role of CO_2 in anthropogenic climate change and the relevant international agreements including the Paris Agreement, regulators should seek to mitigate the envi-

ronmental damage done by miners, for example by moving from proof of work systems to Byzantine fault tolerance or to proof of something else. Asking bank regulators to make technology choices might not be ideal, so perhaps the appropriate policy instrument here would be a carbon tax on mined coins.

Various policy mechanisms might be used to get from here to there including issuing central-bank cryptocurrencies or monetising existing crytocurrencies, but only where regulated entities such as exchanges, miners and wallet hosting firms pay their carbon taxes. The market could then decide whether to go for moving to proof-of-stake coins, or even (if they're properly capitalised) letting the exchanges run a ledger directly.

Our **eighth recommendation** was therefore that regulators decide how to levy a carbon tax on cryptocurrency mined using proof of work methods, and that the very minimum acceptable should be the EUR 33 per tonne floor of the Emissions Trading Scheme. From a technological point of view this would mean transitioning to more efficient consensus algorithms, such as the one I present in chapter 5.

3.8 Conclusion

In this chapter we analysed the treatment of tainted bitcoins from legal, economic and engineering perspectives, focusing on stolen bitcoins. Technologists claimed that taint tracking was hard, as they assumed that taint would mix and dilute when coins are joined; yet the relevant case law specifies first-in-first-out tracking, which turns out to be technically easy. Technologists also assumed that bitcoin mixing made coins derived from innocent and stolen inputs innocuous, whereas the legal effect of attempts to conceal the source of funds is to taint the output.

We first described how to make it practical to trace stolen coins on the blockchain, at least in the theoretical world described in academic research. The same applies to other kinds of tainted coins such as those acquired via other crimes from ransomware to drug trafficking.

We then built a visualisation tool to study the spread of taint on the blockchain. This led us to discover some interesting patterns that could serve as useful heuristics for picking out suspicious bitcoins. We published these tools and received communication from many victims of bitcoin theft.

This led us to explore the limitations around the use of taint-tracking in practice, at least by individual crime victims, and went on to describe how many bitcoin exchanges have started working since early 2017, with off-blockchain transactions and the ownership of the underlying bitcoins often being obscure.

We then took a close look at the measures taken by many governments to tackle the most urgent serious-crime threats including large-scale money laundering and underground drug markets, notably by forcing exchanges to register and perform basic due diligence on their customers. These have culminated in the EU's amending the fourth

anti-money-laundering Directive to bring wallet hosting service providers as well, with effect from November 2018. However, this still only tackles the problems of four years ago: we described how regulation has failed to keep up. While regulators have tackled the access and egress points where real money is transferred into digital currency and vice versa, they have failed to notice that the growing volume of off-blockchain transactions has created an unlicensed shadow banking system. This will have to be regulated, just as the real banking system is, and for precisely the same reasons.

Finally, when we did this analysis in 2018, we made eight recommendations as a guide for regulatory efforts which I gather here for convenience.

1. The E-Money Directive should apply to exchanges doing business with EU citizens which offer off-blockchain payments or consolidate cryptocurrency assets rather than merely holding cryptographic keys on behalf of customers, in respect of all these payments and assets.

2. The relationship between an exchange and its customer should be covered by the second Payment Services Directive.

3. Governments should prohibit the cryptocurrency exchanges they regulate from clearing and settling transactions with unregulated exchanges.

4. Central banks should consider issuing a cryptocurrency using a permissioned system that supports smart contracts and micro transactions, has the legal status of a bill of exchange and is redeemable at par for fiat money.

5. Regulators should compel exchanges to make clear in their terms and conditions whether they are custodians of cryptocurrency assets that the customers own, or whether the assets are owned by the exchange with the customers simply having a claim on the asset pool.

6. Regulators should prohibit exchanges from buying and selling cryptocurrencies that are explicitly designed to evade money-laundering and terrorist financing controls. Regulators also need to carefully consider the issue of off-chain payment mechanisms such as payment channels and what restrictions should be placed on their usage.

7. Regulators should require regulated exchanges to be adequately capitalised, and develop proper accounting standards to support this.

8. Regulators should decide how to levy a carbon tax on cryptocurrency mined using proof of work methods; the minimum acceptable should be the EUR 33 per tonne floor of the Emissions Trading Scheme.

We believe that existing laws can be used to tame the cryptocurrency jungle and make it safer both for private users and for innovation. An important step is to enforce

the EU's E-Money Directive in respect of digital currency assets held by EU exchanges on their customers' behalf, as well as for balances of Euros and other fiat money.

Settling the legal status of digital currencies should be used as an opportunity to move operators from the proof of work systems that now emit more CO_2 than Austria [57], to alternative systems that do not do as much environmental damage, by means of a carbon tax.

An interesting question is whether this would need new legislation, or even a trade treaty (as might be needed, for example, to impose a tax on the embedded carbon content of imported machines). If existing regulations can perhaps be used to implement our other seven recommendations, perhaps they can be used to enforce a carbon tax as well, by making it a condition of cryptocurrencies being traded on regulated exchanges.

At the time of writing, unfortunately, this carbon tax still has not been implemented and regulators have generally continued with their hands-off policy when it comes to PoW emissions. I hope that this changes in the near future since the popularity of cryptocurrencies is on the rise again, most probably due to the pandemic and consequent quantitative easing measures worldwide [34]. Cryptocurrencies seem to be here to stay, we ought to hurry and make them less harmful to the environment.

A bright point to end this chapter on is the apparent utility of a central bank issued cryptocurrency as well as of smart contracts to facilitate interactions between institutions. Here, our optimism seems to have been validated with many companies now adopting *permissioned blockchains* in a variety of contexts as well as several central banks making strides towards issuing their own cryptocurrencies. We will discuss this new avenue for research and engineering in the next chapter where we talk about the challenges that come with getting a permissioned blockchain system out to customers as well as the ways in which I have tried to address some of those issues.

"Many a calm river begins as a turbulent waterfall, yet none hurtles and foams all the way to the sea."

—Mikhail Lermontov

4

Taming the blockchain

Permissionless networks[1] such as Bitcoin have an uphill task: they are trying to create a trusted system out of a set of completely untrustworthy entities while also giving them worthwhile economic incentives to participate in the system. In enterprise settings, there is no need (nor is it usually feasible) to motivate actors to participate in this manner as they can simply be instructed to do so. Moreover, enterprise systems usually have defences against certain threats such as sybil attacks [84] via their identity management systems. Thus, efficiencies ought to be possible in this more constrained setting. This reasoning has led to the recent development of *permissioned blockchains*, also known as Distributed Ledger Technologies (DLTs).

Following on from our own 2018 recommendations presented in the previous chapter, where we urged governments and institutional actors to explore permissioned blockchains, I decided to take a closer look at this emerging technology and see whether it is more than just a solution looking for a problem.

To get hands-on experience with getting a real system out to paying customers, I joined a startup, Jitsuin, and designed their blockchain back-end from the ground up over the course of two years. The system we built is now being used by several large enterprises.

The Jitsuin collaboration led me to discover some issues with existing frameworks as well as mitigations to those problems. The very first of these issues was with GDPR

[1] A note on terminology: I use the word *permissionless* to denote systems where anyone is allowed to participate in the consensus mechanism, and the word *permissioned* to denote systems where there is gatekeeping on who can do so. This is completely orthogonal to a network being *public* or *private* which denote who can read and write data to the system. It is perfectly plausible to have a *permissioned* yet *public* blockchain as is the case for many self-sovereign identity projects such as Hyperledger Indy [118].

compliance and the need to edit historical data; this led us to file a patent, which I present here. The second issue plaguing blockchain systems in production is the ever-increasing cost of readily available storage. I designed a mitigation for it, a patent for which has also been filed.

As detailed in § 1.2, the patent on modifiable transactions was done collaboratively with Jon Geater whereas the patent for storage scaling was invented solely by me and later validated by Jon Geater.

I first present a succinct comparison of permissioned blockchain frameworks. I performed this initially in 2018 when we were deciding on which framework to use for Jitsuin; here, I have significantly shortened that focusing on attributes relevant to this thesis and brought it up to date.

4.1 Why permissioned?

As we talked about in the chapter 2, the renewed impetus in studying decentralised computation has been spurred on by the introduction of Ethereum and its smart contracts. Ethereum has been pitched as a global trusted computer; this led to conjectures that it could be used to disintermediate existing trusted third parties resulting in higher efficiencies and greater transparency.

4.1.1 Issues with permissionless blockchains

There are several issues with the permissionless way of doing things when applied to enterprise settings. These issues can broadly be categorised under one of four headings:

1. Cost of consensus
2. Transaction and metadata privacy
3. Throughput and latency limitations
4. Poor access control

I will discuss each in turn.

Cost of consensus. We have discussed the unsustainable toll that PoW has on the environment in the previous chapter; the associated high energy cost means that spinning up a new public chain that is resilient to attacks is prohibitively expensive[2]. Moreover, unless the originating enterprise has some very specific requirements for a blockchain, it makes little sense for them to spin up a new variant of Bitcoin or Ethereum.

[2]There is now a name for attacks that target small public blockchains: *altcoin infanticide*. Large miners of Bitcoin target smaller cryptocurrencies and trivially perform 51% attacks on them. This is done to snuff out any potential competition.[47]

This means that enterprises considering public blockchains would probably use a pre-existing one like Ethereum. This does not solve the cost of consensus, but merely obfuscates it. The computation cost of maintaining the network is then paid by the miners. In the early days of Bitcoin, the mining rewards were sufficient to motivate miners to include transactions in the block and continue mining. However, miners increasingly expect a transaction fee for them to consider including a transaction in a block. This transaction fee depends on congestion and can show wild price fluctuations – not a good basis for a reliable enterprise application.

Transaction and metadata privacy. By default, all transactions on the Ethereum blockchain are public. This means that smart contracts deployed on the blockchain, the data stored within them and all interactions with these contracts are visible globally. Even if we use some encryption scheme so the transaction payloads aren't visible to everyone, the patterns of interaction with the contract remain visible by all. This metadata leakage is unacceptable in many enterprise applications. An additional side effect of Ethereum's broadcast nature is that organisations need to bear the cost of storing its whole blockchain, including all the irrelevant data posted by parties in which they have no interest.

There is also a tension between accountability and privacy in a broadcast system: the more obfuscation we use, the harder it is for an auditor to verify the validity of actions. For example, if you used frequently changing pseudonyms to hide interaction metadata, it would be harder for a nosy competitor to glean your business secrets but it would also be harder for an auditor to sign off your accounts, or for your internal auditor to look for fraud.

Throughput and latency limitations. Another consequence of using proof of work is poor performance. On average, Bitcoin can process about seven transactions per second and each transaction takes tens of minutes to get confirmed. Compare that with Visa which can easily process ≈2000 transactions per second and confirmation happens within seconds [193].

While it may be possible to increase the throughput of a cryptocurrency system – and there is active research in this domain – the latency limits will be harder to budge. If transactions need to be communicated to an unbounded number of nodes, we need to set the round trip times quite high, or not all nodes will receive messages in time and frequent forks will occur. High latency appears inescapable for global permissionless blockchains.

Poor access control. Closely linked to the issue of transaction privacy is the issue of access control.[3] While it is possible to have write access control at the smart contract layer, managing such systems with existing permissionless blockchains is tricky.

[3]We can think of transaction privacy as read access control; here I refer to write access control.

Ethereum, and most other permissionless systems, use an *address-based* control which means that the contracts can be programmed to check if a particular transaction was signed using the right key. This is a very simplistic model for governing access to assets. The task of designing an access control system resembling the complicated, hierarchical systems that enterprise applications demand isn't trivial. Address-based access control also makes it difficult to use modern authentication methods such as Web access tokens and instead compels developers to use security-sensitive keys directly. Finally, from an operational standpoint, regulated businesses need to be able to exclude transactions emanating from states under sanctions (e.g. North Korea); this is difficult to ensure in a permissionless system.

4.1.2 Permissioned blockchain primer

Faced with these issues, many enterprises started to look for ways to gain the transparency and efficiency promised by blockchain networks without incurring the drawbacks of permissionless networks. This is perhaps best demonstrated by JP Morgan: at about the same time as its CEO, Jamie Dimon, was calling Bitcoin "a fraud" and claiming he would "fire in a second" any JP Morgan trader trading bitcoins [122], it was launching one of the first permissioned blockchain frameworks, Quorum [133]. What was seen as a dissonance in the company at the time was actually a different take on the blockchain model, one where not everyone was invited nor was everyone within the network trusted. It served as a middle ground between the absolute centralisation of the trusted third party model and the lawless world of the permissionless systems.

Of course JP Morgan wasn't the only player interested in developing permissioned blockchain frameworks. In December 2015, the Linux Foundation announced the creation of the Hyperledger Project [206]. This was intended to serve as an incubator hosting several different frameworks with promises of some interoperability between them. The initial projects, Hyperledger Fabric and Hyperledger Sawtooth, were developed by IBM and Intel respectively and subsequently entrusted to the Linux Foundation. Since then, Hyperledger has grown to include six different frameworks, four libraries and several other open-source tools.

So what distinguishes these frameworks from the permissionless networks that preceded them? The most obvious difference lies in the consensus layer. Instead of relying on PoW or PoS, these frameworks use distributed consensus mechanisms, usually based on some form of leader election protocol. These mechanisms are typically designed to be energy efficient and provide quick consensus; however, this comes at the cost of much lower scalability. Some of the common consensus algorithms used in frameworks are Raft [169] and Tendermint [53].

Another crucial change that permissioned blockchain frameworks incorporate is a more targeted network layer. Unlike permissionless blockchains where the goal is to gossip transactions to as many nodes as possible, in a permissioned setting we seek to

route transactions to specific nodes. Moreover, some frameworks incorporate the notion of *private channels* which are subsets of nodes communicating over data that the rest of the consensus network is not privy to.

4.1.3 What is the point?

What are companies that are investing in this technology hoping to gain? A recent survey of 160 organisations by Rauchs et al. [180] suggests that the main motivator for exploring permissioned blockchain systems is the potential for cost reduction stemming from efficiency gains. These efficiency gains are found in one of three ways, as the authors elaborate:

> "(1) reduce costs by removing avoidable reconciliation steps between company ledgers, (2) generate revenues through the provision of new services enabled by the access to shared data, and (3) create new market models and types uniquely enabled by the shared network that did not previously exist."[180, Pg. 34]

The first of these is the primary motivator for most networks in their survey. This suggests that disintermediation of trusted third parties and thus allowing direct collaboration is the focus of the push for permissioned blockchains.

Turning our attention to application domains, the authors report that the most common application scenario is supply-chain tracking. Modern supply chains have many independent actors with various degrees of collaboration between them leading to many interfaces mediated by trusted third parties[4]. This strong correlation between supply-chain tracking and permissioned blockchains has been corroborated by several other studies recently [92, 101].

Jitsuin's goal is to create a shared lifecycle assurance system for industrial IoT devices. This means incorporating not just the supply-chain relationships involved in the production of the devices, but also keeping track of maintenance commitments over the device's lifetime. I joined Jitsuin right after its founding in late 2018 (before any development had taken place) and was entrusted with designing the blockchain back-end underpinning this assurance system. The remainder of this chapter catalogues my efforts.

4.2 Comparison of permissioned blockchain frameworks

Table 4.1 illustrates a summary of the comparison between blockchain frameworks. The initial survey was performed in early 2019 and updated in mid-2020 with the introduction

[4]Some supply chain experts [12] have suggested that blockchains are being used as an excuse to finally move past cumbersome paper-based systems and that this is one of the main reasons for adoption as well as the efficiency gains seen. The survey by Rauchs et al. does not indicate this, however.

Characteristics	HL Fabric	HL Sawtooth	HL Burrow	HL Iroha	HL Besu	JPM Quorum	R3 Corda
Pluggable consensus	•	•	×	×	•	•	•
Flat hierarchy	×	•	•	×	•	•	×
Byzantine tolerance	×	•	•	○	•	×	○
EVM compatibility	○	○	•	×	•	•	×
Private channels	•	○	×	×	•	•	•
Production ready	•	•	×	•	•	•	•

Table 4.1: Summarised comparison of permissioned blockchain frameworks. *Legend:* characteristics are one of fully-exhibited (•), exhibited with caveats (○), not exhibited (×), not applicable (NA), Certificates (Cert). HL stands for Hyperledger and JPM for JP Morgan.

of Hyperledger Besu. I have only included frameworks with active repositories (at least one commit per month for the last year) to weed out defunct ones.

The characteristics I have focused on here are:

- **Pluggable consensus:** the ability to switch consensus algorithms in the framework. This is desirable so that one can choose the appropriate algorithm for a given application.
- **Flat hierarchy:** whether nodes perform homogeneous functions or if there are "special" nodes; it is important to check for this since some frameworks effectively centralise the network by introducing special nodes (without making it clear in their documentation that this is what they are doing).
- **Byzantine tolerance:** whether the framework comes with a Byzantine fault tolerant consensus algorithm. Where a framework supports multiple consensus algorithms, I rate it on the basis of the most resilient algorithm supported.
- **EVM compatibility:** whether the framework supports the Ethereum Virtual Machine and Solidity contracts; I pay special attention to this since by far the most amount of work around smart contracts has been done in Solidity resulting in extensive tooling and contract support.
- **Private channels:** whether the framework has support for privacy preserving channels to send data to subsets of nodes.
- **Production ready:** whether the framework is marked as ready (denoted by a 1.0 public release) or if it is experimental.

4.2.1 Hyperledger Fabric

Fabric [22] is one of the founding projects within Hyperledger and the most widely used framework in Rauchs' survey [180, Pg. 37]. Fabric foregoes the usual Ethereum Virtual

Machine (EVM) and Solidity combination used in most other permissioned frameworks and instead has its own Go based contract layer (which is called *chaincode* in Fabric parlance).

Fabric supports pluggable consensus as well as private channels. Private channels in Fabric protect against metadata leakage better than most: private data is never sent to nodes that are not party to the private channel effectively creating a distinct sub-network.

The main drawback of Fabric (apart from lack of EVM support) is its use of special nodes called *orderers* that are responsible for ordering transactions sent to the network. These orderers were built to reduce the overhead for reaching consensus by serialising transactions and efficiently preventing double spending. However, they represent a single point of failure in the system. While transactions are signed by ordinary *peer* nodes, if the sole orderer for a channel goes offline, the channel cannot make any progress.

It is possible to distribute the ordering service into many orderers and have them coordinate using a consensus algorithm; however, at that point we lose many of the efficiency gains that made the hierarchy of nodes worthwhile. The effects of this trade-off are apparent in the survey by Rauchs et al. where they found that most of the projects using Fabric weren't decentralised *at all* but rather were using a single orderer and sometimes even a single peer [180]. Fabric's documentation is also not explicit enough about the issues with using a single orderer, possibly leading to engineers not realising the centralised nature of their system.

A final issue with the idea of trying to decentralise the ordering service is that Fabric currently doesn't offer any byzantine fault tolerant consensus algorithm for coordinating orderers; the only production-ready option is Raft which is only crash fault tolerant [169]. This renders the usage of Fabric in adversarial settings insecure.

4.2.2 Hyperledger Sawtooth

Sawtooth differs from other frameworks by having a Trusted Execution Environment (TEE) based consensus algorithm called Proof of Elapsed Time (PoET) [120, 174]. Since Sawtooth is an Intel project, the TEE used for PoET is Intel SGX. PoET unfortunately suffers from fundamental issues: for one, it gives every node an incentive to break into their own SGX enclave and second, SGX has been shown to be susceptible to a range of attacks [62, 48, 190]. We will discuss PoET in greater detail in chapter 5.

To get past these vulnerabilities of PoET and the restriction of being tied to Intel hardware, Sawtooth engineers implemented a variant of Practical Byzantine Fault Tolerance (PBFT) [59]. PBFT is an early byzantine fault tolerant consensus algorithm with a flat hierarchy but poor scalability properties (networks with more than a dozen or so nodes are impractical).

Sawtooth is an oddity on the smart contract side of things since it doesn't come with a fully fledged smart contract layer. Instead, it comes with primitives that allow appli-

cation developers to define their own *transaction processors* and thus create quasi-smart contracts. Sawtooth introduced a transaction processor called Seth (Sawtooth-Ethereum integration project) that was supposed to make it possible to run EVM contracts on Sawtooth. While Seth works for simple contracts, severe deficiencies remain in its implementation meaning that Solidity contracts rarely work as intended on Seth. At the time of writing, it also appears that Seth has been abandoned, having seen no new releases in almost two years despite Solidity having gone through major revisions in that timeframe.

4.2.3 Hyperledger Burrow

Burrow is a simple permissioned blockchain for running Solidity contracts with very few add-on features. In fact, the architects proudly proclaim in their documentation: "Blockchains are too exciting. Burrow wants to be boring ... The kind of boring that lets you sleep well at night." [70].

This "boring" way of doing things leads to a very simple deployment model: Burrow comes with a flat, byzantine fault tolerant consensus algorithm (Tendermint [53]) that works with very little parameter tuning. This means that making simple applications in a small permissioned network is quite straightforward on Burrow compared to the other frameworks here. Tendermint also demonstrates better scalability than PBFT, though it too is limited to about a hundred nodes for reasonable latency requirements (say, 10 minutes for transaction confirmation).

Compounding this, Burrow doesn't have pluggable consensus so developers can't swap out Tendermint for another consensus algorithm. There is also no support for private transactions and private channels, rendering many enterprise applications difficult to implement. Another drawback is that while the project homepage says that Burrow is ready for production usage [117], the developers still haven't released a production 1.0 build.

4.2.4 Hyperledger Iroha

Iroha [119], like Burrow, is a framework focused on simplicity and ease of deployment. Consequently, it shares some of the same drawbacks: no private channels, no private transactions and no pluggable consensus.

There are a few key differences between the two, however. For one, Iroha doesn't have native EVM compatibility; instead, Iroha allows clients to send so-called *commands* to alter the state of assets. This is reminiscent of the transaction processor approach in Sawtooth. Just as in Sawtooth, creating complex contracts is more involved with Iroha as compared to EVM-compatible frameworks.

Another key difference is that Iroha's consensus algorithm—Yet Another Consensus (YAC) [160]—claims to provide Byzantine fault tolerance, but a closer reading of the protocol shows some issues. For one, the authors make an assumption that the adversary

cannot arbitrarily send *reject* packets without providing any justification. This means that an adversary can cause a livelock by sending these packets. Secondly, YAC has an ordering service performing the same functions as the orderers in Fabric; thus inheriting the same centralisation issues we saw there.

4.2.5 Hyperledger Besu

Besu [116] is the latest entry into Hyperledger and was developed in conjunction with the Enterprise Ethereum Alliance. It is fully compatible with the EVM and sticks to the upstream Ethereum specification. Another strong point for Besu is pluggable consensus: it supports three different consensus algorithm, one of which (IBFT2.0) is byzantine fault tolerant. IBFT2.0 demonstrates similar scalability to PBFT.

Private channels in Besu are managed by a *transaction manager* which keeps track of the state of private contracts. The transaction manager, called Orion [170], is interesting because it provides multitenancy: multiple entities can associate their private data with a single consensus node without data leakage. This is an important feature for network sizes that exceed the limits of the consensus algorithm. With Orion, it is possible for each entity to then be relegated to an Orion account while delegating consensus responsibility to a subset of nodes[5].

4.2.6 JP Morgan Quorum

Quorum [133] was initially developed by JP Morgan and has recently been acquired by Consensys. It resembles Besu in a lot of ways: both have native EVM support, pluggable consensus, a flat hierarchy, private channels and are production ready. Quorum falls behind Besu in two domains: byzantine fault tolerance and lack of multitenancy. While Quorum does support pluggable consensus, only one of its implemented algorithms—Istanbul BFT [130]—is supposed to be Byzantine fault tolerant.

Unfortunately, the design of Istanbul BFT has a critical bug that violates its liveness guarantees i.e. it is susceptible to deadlocks [129]. Thus, Quorum, despite having pluggable consensus, currently doesn't provide Byzantine fault tolerance.

4.2.7 R3 Corda

Corda [71] is the most popular non-Hyperledger platform in enterprises today [180, Pg. 37]. It has a heterogeneous composition of nodes in the consensus layer: where Fabric had orderers, Corda has *notaries* and they perform a similar function. As in Fabric, the default is for these notaries to be centralised for better performance but they can optionally be distributed. Unlike Fabric, Corda's notaries do come with a consensus

[5]This is obviously not as good as if the networks could scale to larger than a few dozen. But, in the absence of such consensus algorithms available in production networks, multitenancy serves as a good stop-gap measure.

algorithm (BFT-SMaRT [195]) that is supposed to be byzantine fault tolerant. Unfortunately, this algorithm is susceptible to replay attacks [182] which calls into question its fault tolerance.

4.2.8 Other permissioned models

So far we have only talked about pure permissionless and pure permissioned systems; in practice, there are a couple of additional models that we may encounter.

Hybrid model. A fairly common model [180, Pg. 62] for combining permissioned and permissionless systems is using *periodic anchoring*. Anchoring, in general, refers to the commitment of a hash to the blockchain instead of the actual payload; this is often used in pure permissioned and permissionless systems as an easy way to hide data and reduce storage costs. Periodic anchoring means that a permissioned system runs in parallel with a permissionless system; all the operations are performed on the permissioned system but the state of the permissioned blockchain is periodically anchored to a permissionless system. This gives the system additional tamper resilience (even if a majority of the nodes get compromised, they would not be able to revert transactions beyond the latest anchor point). Transfer of semantic information (i.e. beyond just opaque hashes) between different blockchain networks is an active area of research [177].

Not-a-network. It is an unfortunate reality brought about by the tremendous hype around blockchains that a lot of blockchain projects are in fact running on a single computer or under the control of a single entity. Rauchs et al. call these projects "blockchain memes"[6] and categorise a shocking 77% of live enterprise networks as belonging to this category.

Such deceptive models contribute to the suspect reputation of enterprise blockchains. I had a suspicion that some of the blame for this could be assigned to frameworks with non-flat hierarchies being opaque about their implicit centralisation. I contacted the authors of the survey and was able to get access to their dataset[7]. There appears to be some correlation here: when it comes to Corda, all but two projects implemented were de facto centralised. The story with Fabric isn't as clear cut but still worse than average: 83% of projects implemented with Fabric were centralised. This might suggest that the default deployment layout of a framework may have an impact on the architecture of the eventual live network. Further research is needed to see how many of these blockchain meme projects are being wilfully deceptive and how many were merely over-optimistic about their deployment.

[6] Perhaps a better name for such projects is "scams".

[7] Thanks to Apolline Blandin for graciously arranging this.

4.3 Blockchain archival

In the previous section we analysed existing permissioned blockchain systems and discussed their relative pros and cons. In this section and the next, I will highlight a couple of common issues faced by permissioned blockchain engineers irrespective of the framework used, and present solutions.

Blockchains impose certain infrastructure requirements that make their adoption into enterprise systems difficult. One of these requirements is an endless supply of storage. Furthermore, to verify the integrity of a blockchain, one needs to have access to the entire blockchain – from the first block (known as the *genesis block*) to the current one. This means that if one of the participants crashes and later wants to join the network (or if a new participant wants to join), it must contact some other member and download every missing block. This is unacceptable for systems that are expected to run for many years since this operation would take a prohibitive amount of time. We would like to be able to join the network—with full integrity guarantees—and only get historical blocks when we need them for some application-driven need.

Removing the requirement of having all the blocks readily available would also help nodes utilise cheaper, but slower, storage media for keeping historical blocks thus reducing the costs associated with blockchain storage. Where we have assurances that applications will not need access to raw historical data, we may even discard those blocks avoiding the ever-increasing storage requirements.

Known techniques exist that aim to mitigate the storage problem by reducing the storage space required for the blocks. This is usually done by either squashing empty blocks together (only retaining their hashes) or by using some form of compression to reduce the storage space requirement for all blocks in general. While these solutions help to a certain extent, they only delay the inevitable. Eventually, unless you discard old blocks, the same storage issues arise; in fact, because of the additional work of decompression, the time requirement for new nodes joining the network may be higher than uncompressed blocks.

In order to get around these storage, retrieval and verification requirements I created a scheme that allows network participants to prune away old data while preserving the following desirable properties:

- The current chain remains verifiable
- The points of time at which pruning occurred are clearly identifiable by all participants
- The participants in pruning the chain are clearly identified
- The participants in pruning the chain are in the same—or higher—security context as the participants in the consensus mechanism

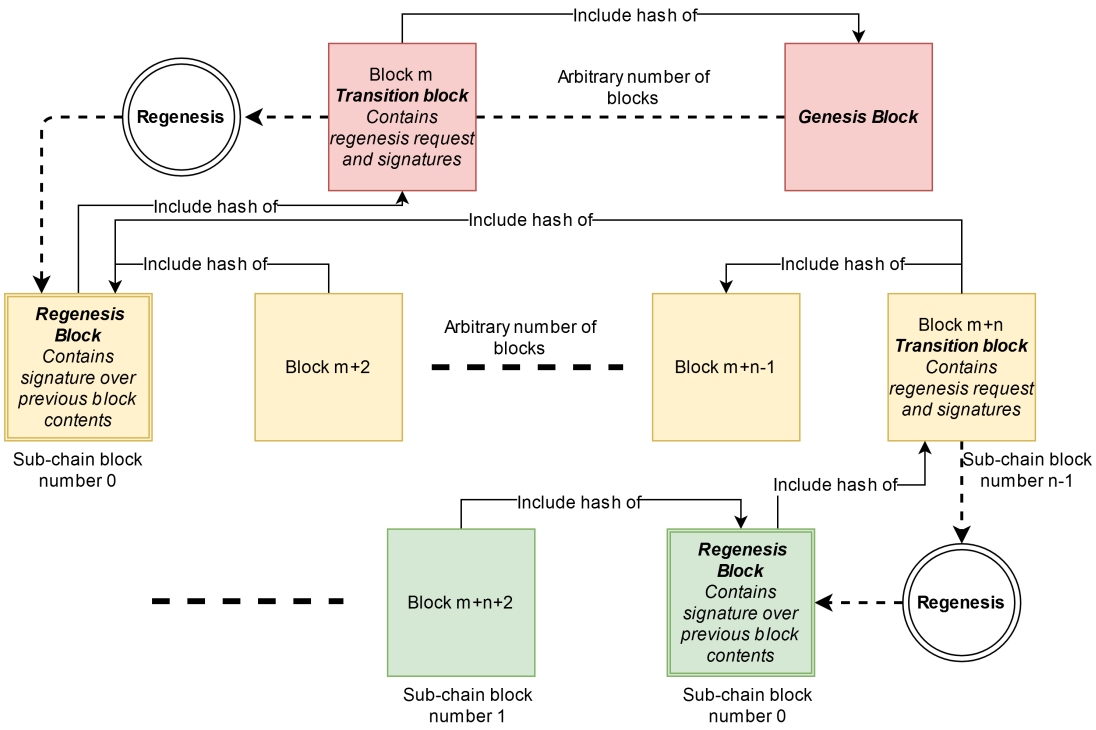


Figure 4.1: Overview of the regenesis system showing the creation of two sub-chains.

This scheme has since been filed as a patent (GB1916295.7); it has cleared the search phase with no objections from the IPO and is now in the pending state. Here, I present an abridged description, pseudocode can be found in Appendix A.1.

4.3.1 Regenesis overview

At the core of the scheme is the idea of *regenesis* which takes place periodically. This means that at arbitrary intervals, the network arrives at unanimous consensus over the current state of the blockchain and effectively commits a checkpoint. After this checkpoint, we say that a new *sub-chain* has been initiated. The blockchain validation process then only requires the validator to traverse blocks up to the last checkpoint and then traverse only the checkpoints until the genesis block. Figure 4.1 shows an overview of what the regenesis process looks like.

There are four main steps in the operation of this scheme:

1. Creation of the regenesis request
2. Creation of the transition block
3. Creation of the regenesis block
4. Validation of the chain

Let's look at each of these in turn.

4.3.2 Regenesis request

The *regenesis request* data structure is at the heart of this scheme, and contains the following:

- **Regenesis number:** A monotonically increasing counter of how many regenesis operations have taken place
- **Timestamp:** Wall-clock time at which the regenesis request was initiated
- **Sub-chain block number:** Block number within this sub-chain at which the regenesis request was initiated
- **Chain block number:** Block number within the entire chain at which the regenesis request was initiated
- **Public list of signatories:** Complete list of authorised signatories that are allowed to initiate and/or confirm regenesis requests. This may or may not be the same as the set of nodes doing consensus – we leave this intentionally configurable.
- **Signatures:** Signatures of the authorised signatories over the previous contents

A point of clarification about the last entry: we require that k of n authorised signatories sign the contents of the regenesis request, where $k > n/2$, in order to prevent so-called partitioning attacks [149] where some of the authorised entities are unaware of a fork in the blockchain. In cases where such partitioning attacks are unlikely, we can relax the signing requirements accordingly.

The creation of the regenesis request is fairly straightforward: when the authorised signatories decide that it is time to create a new sub-chain (e.g. by having a set number of blocks in every sub-chain or via off-chain messaging), they each create the regenesis request packet (all the content except the signatures). Then they each sign the packet and send their signatures to all the other nodes. The signatures are ordered in a deterministic manner (say, lexicographically) and then appended to the packet. The creation of the regenesis request is now complete.

4.3.3 Transition block

Once the regenesis request has been created it needs to be packaged into a format that can be easily parsed by the verification script. To do this we package it into a transition block. The transition block consists of the hash of the previous block, the regenesis request, the hash of the regenesis block of the current sub-chain and the consensus signatures (or whatever proof is associated with the consensus algorithm in use) as part of the usual block forming procedure. If, for some reason, the regenesis request is not included in the intended block then it must be regenerated as described above. If the set of authorised signatories has changed during the operation of the current sub-chain,

then the proof of addition/removal of those signatories must also be included in the transition block.

4.3.4 Regenesis block

Once the transition block has been accepted into the network, the nodes independently create the regenesis block of the successor sub-chain. The regenesis block consists of the hash of the previous block and the regenesis request. The consensus over this block requires the signing of the block by a quorum of the authorised signatories. The same caveat about partitioning attacks as in the creation of the regenesis request applies here.

Once the regenesis block has been accepted into the network, the blockchain can continue its operation as normal. The next block is a regular block that includes the hash of the regenesis block. The only modification from the usual operation of blockchains is that each block now contains two block numbers: one from the last regenesis and one from the original genesis block.

4.3.5 Validation

The above alterations do not change the linear, immutable nature of the blockchain – the sub-chains only serve as labelled portions of the total chain. Due to the creation of these sub-chains, the number of blocks one needs to read to validate a chain decreases drastically. In Figure 4.1, validating this chain from the current block (block number $m+n+2$) only requires us to read a total of 6 blocks as opposed to $m+n+2$ blocks. This is because we do not need to validate the regular blocks in any non-current sub-chain. All we need is the transition block and regenesis block for that sub-chain: validation is then simply checking if the right signatories had authorised the regeneration and whether the hash included in the transition block matches the hash of the regenesis block. Thus, we can skip the validation of most of the chain without compromising security.

4.3.6 Dealing with state

The scheme presented above works for all permissioned blockchains for the purposes of reducing the networking and computational cost associated with joining the network. However, if we also want to discard old blocks and actually limit the total storage cost then there are some caveats. If the framework uses an associative data store for keeping track of all the state variables on chain, then there should not be any issues with simply discarding old blocks. If the framework has components that use, say, a UTXO-based model (as in Bitcoin, see § 3.3), where the current state of variables may be stored in historical blocks with no forward references, then we need to checkpoint this state too. This involves creating special blocks that carry forward all UTXOs at the time of regenesis. However, depending on the number of such transactions, this may prove to

be infeasible. Schemes for efficiently snapshotting the state of the chain would make for an interesting avenue for future work.

4.4 Modifiable storage

The immutability of the blockchain is a double-edged sword: while it gives participants assurance of stored data, it presents a serious problem for enterprise use of the technology. Businesses are required to comply with many regulations in regard to information management (such as the EU's General Data Protection Regulation) which include requirements in some cases to delete information that they hold, or to correct information they have previously held or published. Companies face fines and reputational damage if they accidentally (or otherwise) publish sensitive data on a blockchain and are then unable to remove it [77].

This is just one of many legitimate reasons why it may be necessary to alter the contents of a ledger. In a traditional blockchain implementation there are only 3 options:

1. Leave the data there and face the consequences;
2. Change the data and break the chain;
3. Go back to the transaction you want to change and manipulate the consensus mechanism to create and force a fork.

To the best of my knowledge, there has been one other proposal to remedy this situation which is the "Redactable Blockchain" system by Ateniese et al. [26]. Their system replaces the standard hash functions used in blockchains (e.g. SHA-3) with chameleon hash functions. A chameleon hash function [139] is a hash function with a trapdoor: without knowledge of the trapdoor, it is hard to find collisions (as in a standard cryptographic hash function) but with knowledge of the trapdoor one can generate collision efficiently. So, by using chameleon hash functions, the authors propose a system whereby privileged actors (with knowledge of the trapdoor) can change arbitrary data in any block without breaking the hash chain.

While this proposal works in the sense that data can be changed, and the chain can be rebuilt, it suffers from a grave problem: nobody knows whether a change was made. If any party were to make claims based on the redacted data, a court would be unable to tell whether those claims were true or false if all the evidence available to them were the chain itself. In short, the evidential value of the chain is destroyed.

To prevent surreptitious edits, we might want a system that enables transactions to be modified (changed or redacted) with the following properties:

- The chain remains verifiable;
- The changed transactions are easily, and verifiably, identifiable;

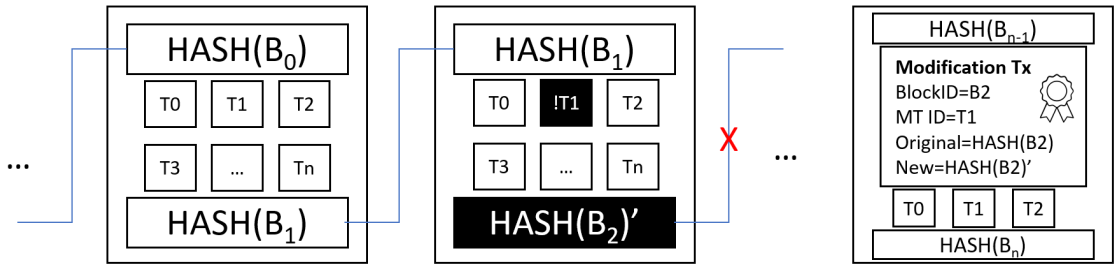


Figure 4.2: Overview of the modifiable transaction scheme. Here transaction $T1$ of block B_2 was modified changing the hash of B_2 breaking the hash chain. This break is justified by the MT included in block B_n.

- The participants who made the change are clearly, and verifiably, identifiable;
- The participants who made the change are in the same security context as the participants in the consensus mechanism.

Jon Geater and I created a scheme that provides these properties. It has been filed as a patent (GB1916291.6), has cleared the search phase and is now in the pending state. I will now present an abridged description, pseudocode can be found in Appendix A.2.

4.4.1 Modifiable transactions overview

At the core of the patent are *modification transactions (MTs)*. MTs are pointers to transactions that have been modified along with the necessary signatures. As with regenesis, only authorised signatories can issue MTs and these may or may not be the same as the consensus nodes. In our scheme, when a transaction needs to be modified, we simply instruct all nodes to do so. This breaks the hash chain. To remedy this, we include MTs at the end of the chain which effectively provide a justification for why the hash chain was broken at that point. Any entity validating the chain can then keep track of all the points at which the hash chain was broken and see if all of them have corresponding MTs with the requisite signatures. If they do, the chain is valid; if they don't, it is invalid. Figure 4.2 shows an overview of this scheme.

4.4.2 Modification request

A modification request initiates the modification process. It is generated by authorised signatories and consists of:

- The ID of the block being modified;
- The ID of the transaction being modified;
- The intended modification of the transaction. This could be a simple redact, or specify a specific pattern of bytes to overwrite. The pattern of bytes may be used to indicate why the transaction was modified (e.g. privacy redaction, error correction);

- Signatures of authorised signatories.

The modification request is sent to the network like any other transaction but is processed differently, as I explain below.

4.4.3 Processing a modification transaction

The modification request is understood as a special governance operation by the consensus nodes and causes the following to happen:

1. The identified block is retrieved, and the identified transaction is overwritten in the way specified in the transaction request. At this point, all nodes still retain a copy of the old data;
2. A modification transaction is created and is added to a block;
3. Consensus is attempted over this block;
4. After consensus is reached and the modification is committed to the chain, the specified transaction is overwritten and all the nodes store the new modified block, discarding the old data.

The committed modification transaction consists of:

- The ID of the block that was modified;
- The ID of the transaction within that block that was modified;
- The way in which the transaction was modified;
- The original hash of the block;
- The new hash of the block;
- Signatures of authorised signatories.

At this point the modification of the transaction is complete. We call the block in which modification transaction is present a *modification block*. Let us now look at how to validate a modified chain.

4.4.4 Validating a chain with modified transactions

The validation of blocks that contain no modified transactions is done as usual. Starting from the genesis block, calculate the hash of the current block and check whether it matches the hash stored in the next block; continue working forwards until you reach the most recent block.[8]

[8]Traditionally, validation is described as working backwards from the most recent block to genesis. Here, working forwards makes validating modified transactions easier to explain.

If there is a modified transaction present at say, block B_m, the hash stored in B_{m+1} will not match. Store block B_m and the hash of B_m stored in B_{m+1} in a list of questionable blocks. Now, when you encounter a modification block, execute the following steps:

1. Verify the signatures on the modification transaction and that the requisite signatories have all signed it.

2. Read the contents and look in the list of questionable blocks for a block ID matching the one in the modification transaction. If it is not found, then verification of the ledger fails immediately.

3. Verify that the original hash and the new hash of the questionable block match the corresponding entries in the modification transaction.

4. If the above verification steps all succeed, remove the questionable block from the list.

When the end of the chain is reached, if there are any questionable blocks left in the list, validation of the chain has failed; otherwise, it has succeeded.

Overall, this scheme resembles that employed by newspaper archives where redactions would carry a note along the lines of "This issue was redacted following the following libel judgement in the High Court". Here, the orders to redact are served collectively by the network.

4.5 Conclusion

In this chapter, we discussed permissioned blockchains. We started off by looking at the deficiencies of the permissionless model. Then we looked at the existing technologies and compared some of the most popular permissioned blockchain frameworks. Following from that comparison, we tackled a couple of common problems. First was the issue of storage requirements and the cost associated with joining a long-running blockchain system. I presented a solution to both these problems by introducing the concept of regenesis. The second issue was a business need to edit historical data, which is impossible in a traditional blockchain. I devised a way to support edits while leaving verifiable evidence of all edits for auditing purposes.

Perhaps the most difficult issue to solve in this domain is that of consensus: only three of the existing frameworks provide byzantine fault tolerance, and even those do so with poor scalability (none scale beyond about a hundred nodes). In the next chapter, I present a novel blockchain consensus algorithm that is byzantine fault tolerant and can scale to thousands of nodes.

"No one can whistle a symphony. It takes a whole orchestra to play it."

—H.E. Luccock

5

Scaling blockchain consensus

Consensus algorithms play a critical role in the behaviour of distributed systems. There has been a surge in interest in consensus research since the introduction of Bitcoin. These follow in the footsteps of earlier research such as PBFT [59] which were designed for use in small networks of up to a dozen or so nodes. Bitcoin, by leveraging economic incentives and proof of work (PoW), managed to scale its network far beyond those network sizes. It proved that it is indeed possible to realise a massively distributed system, but it has its share of drawbacks: its throughput is low (approximately 7 transactions per second), its latency is high (approximately 60 minutes) and most importantly, it wastes huge amounts of energy. As we've discussed previously, the estimated energy consumption of all Bitcoin miners is comparable to a medium-size country [57]. This has led a number of researchers to ask whether we can find better ways to get consensus, without the CO_2 emissions of bitcoin mining.

In this chapter, I present *Robust Round Robin*, an incentivised consensus algorithm that operates on a blockchain. It was designed in collaboration with Kari Kostiainen, as detailed in § 1.2. Robust Round Robin can be used in both permissioned and permissionless settings depending on the gatekeeping mechanism utilised. In permissioned settings, the incentivisation provided by the Robust Round Robin is unnecessary, thus mirroring the vestigial nature of "gas" in existing permissioned frameworks derived from Ethereum [87].

5.1 Motivation

Any decentralised asset tracking service requires a consensus mechanism to prevent double spending. Owing to the drawbacks of PoW mentioned above, several alternative permissionless blockchain consensus schemes have been proposed. Proof of stake (PoS) is arguably the most prominent approach that avoids the above energy waste. The basic idea in most PoS systems is to *randomly choose*, in each round, one of the system participants as a consensus leader that extends the chain with a new block. Selection is performed such that the probability of being chosen as the leader is proportional to the owned stake, such as coins.

5.1.1 Naive random selection

The first PoS proposals [67] suggested a simple technique where the hash of the previous block functions as a "random" seed for leader selection on the next round. However, this approach is vulnerable to *grinding attacks*, where the leader of the previous rounds tries different block candidates (e.g. by sampling from the pool of pending transactions) and picks the block that gives him an advantage in leader selection on the next round. By iterating through many candidate blocks he can pick one that makes him the leader on the next round as well.

Another simple approach is to run a bias-resistant *random beacon* protocol among all participants with stake. Random beacon is a distributed protocol that generates a new random value periodically. The main drawback of this approach is that such protocols traditionally have high communication and computation cost (e.g., $O(n^3)$ for a widely used protocol by Cachin [55]).

5.1.2 Sophisticated random selection

Recent research has suggested more efficient random beacons, both as standalone protocols and as part of PoS blockchain systems.

RandHerd [201] is a standalone random beacon that leverages publicly verifiable secret sharing (PVSS) and collective signing (CoSi) to produce unbiased and unpredictable random values among a large set of participants. RandHerd divides all participants into smaller committees of size c. A required threshold of participants from each committee contributes to the output random value. The per-round complexity of RandHerd is reduced to $O(c^2 log(n))$. The main problem for permissionless blockchains is an expensive initialisation routine where participants are divided into groups. This operation takes several minutes for networks with more than a few dozen nodes. This slow reconfiguration must be repeated when new participants join or leave the system.

Ouroboros [135] is a PoS system with a built-in random beacon. Ouroboros randomly samples a committee that runs a PVSS-based protocol with complexity of $O(n^3)$. Since this protocol is executed infrequently, the high cost is amortised over several rounds.

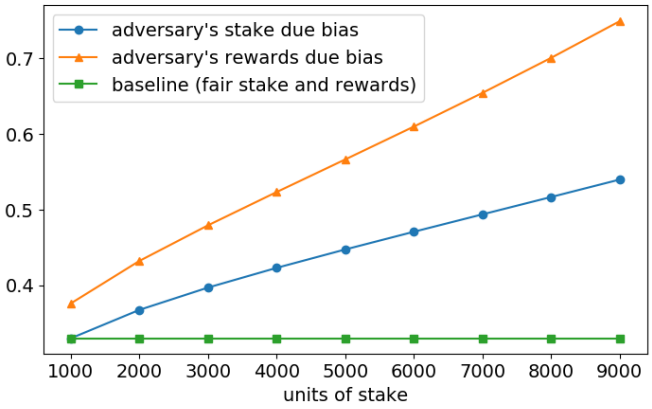


Figure 5.1: Example of the cumulative effect of minor selection bias in a system where block creation is rewarded with new stake. Triangles show the adversary's block creation rate and rewards increase. Circles show the adversary's stake increase. Squares represent the baseline ($\alpha = 0.33$) of fair rewards and stake without bias.

However, the main drawback of this solution is that it requires committees with an honest majority; this may mean that committees of *thousands of participants* must be used to gain reasonable confidence of an honest majority across the entire system lifetime which makes the protocol very expensive. Moreover, Ouroboros requires synchronous communication which is difficult to achieve in large peer-to-peer networks.

In Algorand [154], random values are derived using verifiable random functions (VRFs) [155][1]. In each round, the chosen leader computes the next random value using a VRF and the previous random value. A publicly verifiable proof π of this computation is added to the block. VRF-based selection is efficient, but the main problem is that the chosen leader may bias the protocol output, e.g. by skipping his turn.

To illustrate the effect of selection bias, consider an example system where multiple leader candidates with priorities are chosen. This approach is used in most PoS systems, because choosing only one leader prevents the system from proceeding if the leader is offline or otherwise unable to communicate. Assume an adversary that controls a fraction $\alpha = 0.33$ of stake. On average, every ninth round the leader candidate with the first and second priority both belong to the adversary; every 27 rounds this is the case for the top three priority candidates, and so on. The adversary can now choose which one of these leader candidates to use and pick the one that gives the most advantageous value for the next selection. While this bias can be relatively small, its effect will accumulate when system participation is incentivised by providing rewards such as new stake to the chosen leader, as is common practice in blockchain systems. Figure 5.1 shows an example starting from 1,000 units of stake. Due to the above bias, the adversary creates blocks at slightly higher rate (≈ 0.36) and thus his share of stake increases. By the

[1]VRFs are public-keyed hash functions. Only the owner of the private key can compute hashes but anyone with the public key can verify the hash.

time the system has 10,000 units of stake, the adversary controls most of the stake and creates over 70% of the blocks, at which point the system is theirs.

Ouroboros Praos [75] is another PoS scheme that uses VRFs for leader selection. As with Algorand, selection can be biased by the adversary. DFINITY [111] and RapidChain [221] are further examples of recent PoS schemes that perform unbiased leader selection with significant communication cost. We review them and their limitations in § 5.8.

5.1.3 Selection using TEEs

In PoET [120], the consensus participants are attested SGX enclaves that wait for random periods of time. The enclave that finishes first is chosen as the leader. If SGX ensures code integrity, this enables secure leader selection.

However, participants now have an incentive to break one of their own SGX processors in order to win the leadership as often as they want. SGX was designed to protect enclaves against malicious software and but not against physical attacks. Additionally, recent research has demonstrated that software-only attacks like Foreshadow [213, 127] based on the Meltdown vulnerability [144] can extract attestation keys from SGX processors. Developing schemes that detect processors that win statistically "too often" is possible, but eliminating all bias is difficult.

Requirements. Given these limitations of previous proposals, our goal was to design a blockchain consensus scheme that meets the following requirements.

- *Fairness.* Our design should ensure leader selection fairness. As explained above, even a relatively small bias in leader selection can have severe cumulative effects when combined with block creation rewards. Therefore, the block creation rate of each participant should be proportional to its stake in the system (we explain later how stake is defined in our system).
- *Efficiency.* Our system should be energy efficient. We do not want to run a system with continuous consumptions of large amounts of energy as in PoW.
- *Simplicity.* We want to avoid complicated leader selection protocols that have high communication costs, are slow to run, difficult to implement, or challenging to deploy.
- *Tolerance to physical TEE compromise.* If TEEs are used, the adversary should not gain any advantage by compromising protection mechanisms on their *own* processors.

To the best of our knowledge, none of the existing schemes meet all of the above requirements.

5.1.4 A deterministic approach to consensus

To tackle the leader selection problem, we propose a new permissionless blockchain consensus scheme called *Robust Round Robin* (RRR). We establish reliable and long-term identities and record the enrolment of each identity to the ledger. The number of identities controlled by each participant in the system is limited to their stake. We propose two concrete ways to establish identities, and thus two notions of stake.

Our first identity-creation mechanism is to bootstrap from existing infrastructures. As an example, we use Intel's SGX processors and attestation service (IAS) where the stake of each participant is the number of SGX processors they control. (Our approach is not limited to SGX, but similar identities could be bootstrapped also from other trust structures such as mobile subscriptions or credit cards.) Our second identity-creation mechanism is "mining" the identities starting from an initial fair distribution. In this approach, the identities themselves function as stake. The first approach applies to a partially decentralised setting where the consensus is fully decentralised but infrastructure providers can to be trusted for initial attestation. The second approach applies to a fully decentralised setting similar to Bitcoin.

Our solution performs *deterministic* leader candidate selection. We assign an age to each identity and place them into a queue in the order of decreasing age. Our notion of age is either the number of rounds since the enrolment of the identity or its previous successful block creation – whichever is later. Once a chosen leader candidate creates a block successfully, its age becomes zero and it moves to the end of the queue again, essentially achieving *round-robin* candidate selection.

Because such simple round-robin selection is vulnerable to attacks (e.g., deep forks) and provides poor liveness, we complement it with a lightweight endorsement mechanism. In each round, we sample a small subset of other identities as endorsers. Each deterministically chosen leader candidate runs a simple protocol with the endorsers and the candidate that first receives the required quorum of confirmations becomes the leader in the creation of a new block. In rare cases, more than one candidate may be chosen, or more than one block created by the same leader, but the probability of such events on multiple successive rounds reduces exponentially, so forks remain shallow. The adversary may bias endorser selection, but that does not enable attacks like double spending or increase his rewards.

The main benefits of our solution compared to other PoS systems are fairness and efficiency. As highlighted in Figure 5.2, solutions like Algorand [154] and Ouroboros Praos [75] suffer from selection bias which can have large cumulative effect. In our solution, leader selection is based on a deterministic schedule and thus hard to bias during system operation. Solutions like Ouroborous [135] and DFINITY [111] require expensive protocols to establish unbiased randomness periodically. Our lightweight endorsement protocol is simple and efficient. In contrast to previous TEE solutions like PoET [120], participants gain no advantage by compromising their own platforms and in this regard

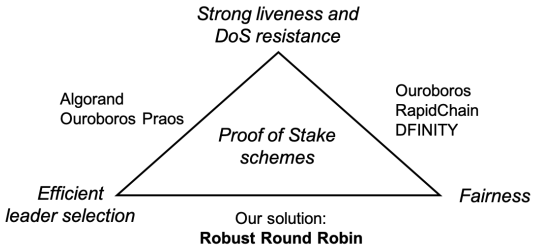


Figure 5.2: The state-of-the-art PoS systems choose consensus leaders randomly which enables good liveness and DoS resistance. Some such schemes use distributed randomness generation that is efficient but can be biased and therefore such solutions do not provide fairness. Other schemes use unbiased distributed randomness generation with high communication cost. Our solution, *Robust Round Robin*, is fair and efficient, but provides weaker denial-of-service resistance.

our solution is resilient to TEE compromise. Another advantage of our scheme is that since the rewards are distributed deterministically, there is no incentive for pooled mining. This is an important benefit in permissionless settings where these mining pools effectively centralise the network.

Deterministic leader selection has some drawbacks in contrast to randomised selection, as shown in Figure 5.2. Because the selection schedule is predictable, our solution can be more susceptible to denial-of-service (DoS) attacks that target the next leader. Another concern is an adversary that owns several old identities and therefore controls block creation on several successive rounds. Such an adversary could prevent transaction processing from targeted users temporarily. Although such DoS attacks cannot be prevented fully, we outline ways to make them difficult to deploy in practice.

The performance and scalability of our solution is comparable to recent PoS schemes. Users can consider transactions safely confirmed once they are extended by a small number of blocks (e.g. $d = 6$ or 12). Since our endorsement protocol is simple, rounds can be set short (e.g. 5 seconds in our experiments) which gives one or half a minute transaction latency and throughput of 1500 tps. The per-round communication and computation complexity is constant and small (e.g. approximately 100 messages per round).

The rest of the chapter is organised as follows. § 5.2 presents an overview of our system. § 5.3 details identity creation and § 5.4 system operation. § 5.5 provides security analysis and § 5.6 performance evaluation. § 5.7 presents a discussion, § 5.8 reviews related work and § 5.9 concludes the description of this algorithm.

5.2 Robust Round Robin overview

In this section, we provide an overview of our proposal, *Robust Round Robin*. We start by listing our assumptions. After that, we explain our main ideas, discuss challenges, and finally provide an overview of the protocol.

5.2.1 Assumptions

We consider two *trust models*. The first is a *partially decentralised* setting, where the blockchain consensus is maintained by a permissionless set of participants, but we rely on the integrity of an existing infrastructure (e.g. that Intel manufactures processors and runs the SGX attestation service correctly) for sybil attack prevention. Our second trust model is a *fully decentralised* setting with no trusted entities, in the spirit of Bitcoin and most other permissionless blockchains.

We consider an adversary that controls a significant fraction α of stake (e.g. $\alpha = 0.33$). Our adversary model is *non-adaptive* in the sense that the adversary cannot arbitrarily choose for every time window which computing platforms or users' key pairs he controls.[2] If TEEs are used, we assume that the adversary can extract secret keys, such as attestation keys, and modify attested enclave code on all of the processors to which they have physical access.

We assume that the participants communicate over a peer-to-peer network. Within each time window t, each participant is able to communicate with all other participants except a small fraction β (e.g. $\beta = 0.05$). This model is motivated by previous studies on the Bitcoin network where most, but not all, nodes receive broadcast messages within a delay that can be easily estimated [42, 78].

Finally, we assume that participants have loosely synchronised clocks.

5.2.2 Identity creation

We create long-term and reliable *identities* and record the enrolment of each identity on the blockchain. By the term "reliable" we mean identities that the adversary cannot create without restrictions (i.e. resilient to *sybil attacks* [84]). This approach can be used with different types of stake; we describe two concrete ways to establish identities, and thus two notions of stake.

The first is bootstrapping identities from an existing infrastructure. We use Intel SGX processors as an example because its attestation service provides a clean interface to implement our protocol. However, we emphasise that our proposal is not limited to SGX and identities could be bootstrapped from other infrastructures like mobile subscriptions, credit cards, passports and other TEEs [218]. We discuss this further in § 5.7. When SGX is used, the stake of each participant is the number of enrolled SGX processors she controls. This approach works in the partially decentralised setting where trust in Intel is required. We call this 'partially decentralised' because the infrastructure provider plays no role in the operation of the consensus algorithm, just in identity creation.

[2] We note that some previous works like Algorand and Ouroboros consider a stronger *fully adaptive* adversary [154, 135] that can freely choose controlled participants for each time window. Our take is that such a fully adaptive adversary is academically interesting but not realistic. In practice, platform compromise is hard to detect and repair. Furthermore, a compromise of one computing platform does not mean that another recovers control from the adversary. For these reasons, we focus on non-adaptive adversaries in this chapter.

Our second way to create identities is to "mine" them starting from an initial fair distribution. Each successful block creation is rewarded with new identities. In this approach, the controlled identities themselves function as the stake. This approach works in the fully decentralised setting. The starting point is an initial fair distribution which can be created using PoW or an initial permissioned phase.

5.2.3 Starting point: deterministic selection

The starting point of Robust Round Robin is *deterministic selection*. We assign an *age* to each identity such that the age refers to the number of rounds since its recorded enrolment or previous block creation event, and we place all enrolled identities to a virtual *queue* that is sorted in the decreasing order of age.[3] Once an identity creates a block, its age is reset to zero and it moves back to the end of the queue (hence 'round-robin'). Because this selection schedule is deterministic, the adversary cannot bias it.

5.2.3.1 Security and liveness challenges

Perhaps the simplest variant would be to select the oldest identity as the leader in each round and specify that a chain cannot skip rounds (i.e. each valid block must refer to a valid block on the previous round, created by the single eligible leader on that round).

Such a rule would be impractical because the system would stall when the designated leader is offline (i.e. very poor *liveness*). Thus, we need to consider variants where multiple senior leader candidates, ranked in order of age, are selected and a chain is allowed to skip rounds. This enables us to produce a new block in each round with high probability.

We then need to define which chain branch is valid when more than one leader candidate creates a block (i.e. the chain forks). One option would be to favour the fork mined by the oldest leader candidate. However, we then have to consider *history re-writing attacks* where the adversary intentionally stalls when they are the oldest candidate, but, after a long time, publishes a block that creates a fork deep in the chain.

To avoid such attacks, we adopt the common "longest chain" policy where the branch with the most valid blocks is valid. Given this definition, we have two remaining design challenges to consider. The first challenge, *deep forks*, is about security. Since enrolment of new identities is open (permissionless), the adversary could enrol, its share (say, 50) of identities successively regardless of the gatekeeping mechanism. Once these identities become the oldest, they would be chosen as the leader on 50 successive rounds. In each round, the adversary could extend one chain branch with a new block that he broadcasts to the network immediately and another block on a separate branch which will be published later causing a fork that is 50 blocks deep. Both branches would be

[3] A similar round-robin selection could be realised through other means such as selecting identities in lexicographic order.

equally long and thus valid. This attack is an instance of the *nothing-at-stake* problem[4] that is a common challenge in PoS systems.

The second challenge, *inactive identities*, is about liveness. We propose that participants establish long-lived identities, but it would be unrealistic to assume that all participants stay active forever. Once a blockchain has been running for ten years, a significant fraction of identities established nine years ago might have become inactive either temporarily or permanently (e.g. lost private key). This could cause extended periods where the system is unable to produce blocks if none of the eligible leader candidates are active.

5.2.4 Final solution: Robust Round Robin

To solve the above two problems (i.e. deep forks and inactive identities), we complement the simple and deterministic round-robin selection with a *lightweight leader endorsement* mechanism.

In each round, a small set of oldest identities are chosen as leader candidates. Additionally, we *randomly* sample a subset of recently active identities to serve as endorsers. The sampling is based on a random seed that is updated for each new block using verifiable random functions (VRFs) similar to some recent systems [154, 75]. Each candidate proposes a block and the endorsers confirm the block from the oldest candidate they observe. The leader candidate that receives the required quorum of q confirmations from the endorsers, is chosen as the leader to extend the chain with a new block.

The endorsers act as *witnesses* and vouch that (1) the candidate they confirmed was the oldest active on that round and (2) the candidate committed to extend the chain with a specific block. Such endorsement guarantees that—with a very high probability—only one block from one leader is produced in each round and that adversaries cannot go back in time to re-write history.

Could sampling be biased by the adversary? In Robust Round Robin, it brings no cumulative advantage, such as increased rewards or possibility of double spending (see § 5.5 for the security analysis).

Besides preventing deep forks, the secondary purpose of the endorsement mechanism is to track active identities. The leader that receives the required quorum of confirmations includes the received confirmations in the new block. By parsing the chain, it becomes possible to verify which identities are active, so inactive identities can be excluded from leader candidate and endorser selection.

Next, we describe our solution in more detail. § 5.3 explains identity creation and § 5.4 details system operation.

[4] As the name suggests, the core issue here is that it does not cost the adversary anything to maintain two different forks unlike PoW where they would have to do twice the work.

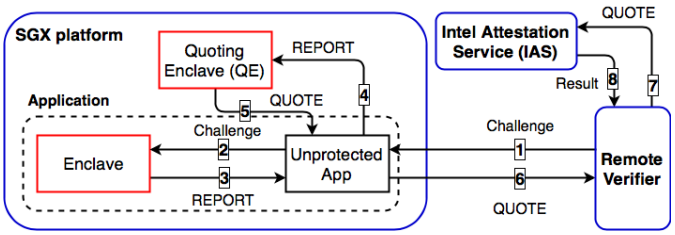


Figure 5.3: SGX remote attestation protocol involving three parties: (i) the remote verifier, (ii) the attested SGX platform, and (iii) Intel's IAS online service.

5.3 Identity creation

In this section we describe two ways to establish identities for our solution: bootstrapping from existing infrastructures and mining from an initial fair distribution.

5.3.1 Bootstrapping from existing infrastructures

Trusted Execution Environments (TEEs) like Intel's SGX [126] enable execution of *enclaves* in isolation from any untrusted software. For robust round robin, the most relevant part of SGX is its attestation protocol which enables a remote entity to verify that specific enclave code is running on a genuine SGX processor. The attested processor signs a statement over the enclave measurement, which was recorded during its initialisation. The verifier forwards the signed statement to the Intel Attestation Service (IAS), an online service run by Intel, that sends back signed attestation evidence. Let us look at this sequence in a bit more detail.

5.3.1.1 SGX attestations

The enclave initialisation actions performed by the operating system are recorded securely by the CPU, creating a *measurement* that captures the enclave's code configuration. Remote attestation is a protocol where an external verifier can verify that an enclave with the expected measurement was correctly initialised in a genuine SGX processor. The attestation protocol involves three parties: (i) the remote verifier, (ii) the attested SGX platform, and (iii) the IAS online service operated by Intel. This process is illustrated in Figure 5.3.

The protocol proceeds as follows: (1) the remote verifier sends a random challenge to an unprotected application on the attested platform that (2) forwards it to the enclave that (3) returns a REPORT data structure encrypted for the Quoting Enclave containing the enclave's measurement. The REPORT data structure includes a USERDATA field, where the attested enclave can include application-specific attestation information, such as the hash of its public key. (4) The application forwards REPORT to the Quoting Enclave that (5) verifies it and returns a QUOTE structure signed by a processor-specific attestation key. (6) The application sends QUOTE to the remote verifier that (7) for-

wards it to the IAS online service that (8) verifies the QUOTE signature, checks that the attestation key has not been revoked, and in case of a successful attestation returns the QUOTE structure signed by the IAS.

The attestation key is a part of a group signature scheme called Enhanced Privacy ID (EPID) [123] that supports two signature modes. The default mode is privacy-preserving. Another, *linkable mode*, allows the IAS to verify if the currently attested CPU is the same as a previously attested CPU. Usage of SGX attestation requires registration with Intel. Upon registration, each service provider receives a credential that they use to authenticate to IAS. If linkable mode of attestation is used, IAS reports the same *pseudonym* every time the same service provider requests attestation of the same CPU [125].

5.3.1.2 Bootstrapping from SGX

We leverage the linkable attestation mode for bootstrapping identities. For network identities, we use the public keys of key pairs that are generated inside enclaves. We bind these keys to the SGX attestation protocol and save the attestation evidence, signed by IAS, to the blockchain. Given such evidence, anyone can verify that the same processor is enrolled *at most once*. This is crucial in defeating sybil attacks.

Importantly, our solution *does not* require enclave data confidentiality or execution integrity. We use sealing to protect the IAS access credential, but its secrecy is not relevant for consensus and is mostly a convenience for developers. As our system tolerates adversaries that can compromise their own processors, the adversary has no incentive to compromise their own platform.

5.3.1.3 Initialisation

A new blockchain is initialised by an entity that we call the chain creator. The creator registers with Intel and obtains an access credential c_a for IAS. At registration, the creator specifies that linkable mode of attestation is used.

The creator chooses n_0 platforms as the initial system members. Each platform installs enclave code that creates an asymmetric key pair, seals the private key sk_i, and exports the public key pk_i. The creator performs a remote attestation on each platform. During attestation, it supplies a hash of pk_i as the USERDATA to be included as part of the QUOTE structure Q_i. If the attestation is successful, IAS signs Q_i that includes a pseudonym p_i for the platform. The attested enclaves send their public keys pk_i to the creator.

The creator checks that the public keys match the respective hashes reported in each QUOTE structure Q_i and that all attested platforms are separate, i.e. each Q_i has a different pseudonym p_i. The chosen n_0 platforms run a distributed random number generation protocol (e.g. RandHound [201]) to establish an initial $seed_0$ that is used to bootstrap seed generation for the following rounds. The platforms also produce a joint

proof π_0 that the seed was generated correctly (e.g. the seed signed by all participants). The creator constructs a genesis block as:

$$\mathsf{Block}_0 = (pk_1, Q_1, pk_2, Q_2, ..., h_e, seed_0, \pi_0, id)$$

that includes public keys pk_i and the signed quote structures Q_i for each initial member, a hash of the enclave code h_e, the initial seed $seed_0$ and proof π_0, and a hash id over all elements that serves as the chain identifier. The creator publishes the block and sends the IAS access credential c_a to the attested enclaves, which seal it.

5.3.1.4 Enrolment

After initialisation, the system proceeds in rounds that are explained in § 5.4. New participants can request enrolment to the system at any round. The joining platform installs the enclave code defined by h_e, creates a key pair, seals the private part sk_n, exports the public part pk_n and contacts one of the current members, e.g. by broadcasting to the peer-to-peer network.

The current member performs remote attestation on the new platform using h_e as the reference. During attestation, the enclave of the new platform supplies a hash $h(id||pk_n||r||h_b)$ as its $\mathsf{USERDATA}$, where id is the chain identifier, r the round number and h_b the hash of the latest block (to bind the enrolment to a specific branch). If the attestation is successful, the existing member obtains a signed QUOTE structure Q_n from IAS, including an attestation pseudonym p_n. It verifies that the pseudonym p_n does not appear in any of the previously enrolled platforms in the chain (recall that each Q_i is saved to the ledger). The verifier sends c_a to the attested enclave and constructs an enrolment message:

$$\mathsf{Enroll}_n = (Q_n, pk_n, r, h_b)$$

and broadcasts it to the network. Once the enrolment message is included in a new block, the new identity is established.

5.3.1.5 Re-enrolment

If an enrolled identity does not participate in the system (by sending confirmation messages) for sufficiently long, it will be excluded from selection as a leader candidate or endorser. In such cases, the platform can run the enrolment protocol again. In re-enrolment, a chosen verifier checks that the IAS service returns the same pseudonym p_i that was used for this identity (public key pk_i) during enrolment. If this is the case, the verifier can create and broadcast a new enrolment message with a flag that indicates re-enrolment. Once re-enrolment is recorded to a new block, the platform is included in leader candidate and endorser selection.

5.3.2 Mining identities

Our second approach is to "mine" identities, that is, reward successful block creation with new identities. A possible strawman solution would be to reward block creation with new coins and have each owned coin directly correspond to one or more identities in the system. As creation of new coins is recorded to the ledger, in each round the owner of the oldest identity can be chosen as the miner. However, this strawman has one major limitation: different coins of the same denomination would have different market values. If a coin is old and soon eligible for block creation, its market value is higher than that of a new coin due to the proximity of the future reward. It is a desirable property for any monetary system that units of the same denomination all have the same value. This is an issue if one were to use RRR for maintaining a cryptocurrency.

To avoid this problem, we *decouple* coins and identities. Every mining operation creates a value reward, such as new stake, and additionally an *identity reward*. A new identity to the system can then be priced at ($N_r \geq 1$) identity rewards. By adjusting N_r it is possible to control the rate of new identity creation. The identity rewards can be used in two ways: the block creator can enrol a new identity for herself or she can sell them to a new user that wants to participate. This new user can then use the identity to join the system.

5.3.2.1 Initialisation

The initial distribution of identities can be established using a preliminary PoW phase[5] or via an initial permissioned phase.

This initial distribution of identities is the series of public keys $pk_0, pk_1, ...$ from the sequence of blocks $\mathsf{InitBlock}_0 = (pk_0, pow_0), \mathsf{InitBlock}_1 = (pk_1, pow_1), ...$ Every initial block contains the public key pk_i of the miner that defines a new identity in the system, and a PoW solution pow. Initial blocks do not contain transactions; they are broadcast to the network, and their recipients store them. These blocks have a predetermined order in the chain, such that every initially mined identity has an associated age.

Once the initial n_0 identities have been created, the participants controlling these identities run a distributed randomness protocol, such as RandHound [201], to create the initial random $seed_0$ and the matching proof π_0 of the correctness of this protocol run that are both attached to the last initial block: $\mathsf{InitBlock}_{n_0} = (pk_{n_0}, pow_{n_0}, seed_0, \pi_0)$. The hash of this block is used as an identifier id for the new chain.

[5]It is worth noting that such a scheme would be vulnerable to "coin infanticide", as mentioned in § 4.1 (e.g., as we shall see with Ethereum's transition to PoS) . Using an atypical PoW function can help mitigate such attacks to some degree. It can also be argued that since this PoW phase only exists for a short initialisation phase, the window of opportunity—and the value of targets—available to attackers is small.

5.3.2.2 Enrolment

Once an identity pk_m has created N_r blocks, it creates a new key pair (pk_n, sk_n) that it uses for enrolment. The enrolment message

$$\mathsf{Enroll} = (h_1, h_2, ..., h_p, pk_n, sig_m)$$

contains a set of hashes $\{h_i\}$ that refer to the N_r previously created blocks by pk_m, the public key of the new identity pk_n, and a signature sig_m over these elements using the private part of pk_m. The participant broadcasts the enrolment message, and once it is included in a new block (see § 5.4), a new entity exists in the system. For an enrolment message to be valid, we require that (1) the set of hashes h_i refer to previous N_r valid blocks, (2) the previous blocks have not been used to create a new identity already, (3) all the referred previous blocks have been created by the same identity pk_m, and (4) the enrolment message signature sig_m is correct.

The same mechanism can also be used to allow new participants to join the system. The owner of the identity rewards can sell them to another participant by including a public key received from the buyer to the enrolment message. The payment from the buyer to the seller can be realised by using fiat money or smart contracts. The buyer should release the money only once he sees the correct enrolment message in the chain in a block that has been extended with d valid blocks (§ 5.5) to prevent double selling of identity rewards.

5.4 System operation

Once the initial n_0 identities are established, our system proceeds in rounds that have fixed length t_r. Algorithm 1 illustrates the sequence of operations per round. Next, we elaborate on the system operation in each round r.

5.4.1 Candidate and endorser selection

At the beginning of each round, every identity tests if it is a leader candidate or endorser. The number of leader candidates N_c and endorsers N_e are both fixed values (e.g. $N_c = 5, N_e = 100$). Below, we describe algorithms for candidate and endorser selection. We focus on presentation simplicity here; actual implementations will use straightforward optimisations like caching previous values.

SelectCandidates() parses the chain based on two adjustable parameters: activity threshold T_a and N_c. An example activity threshold is $T_a = 20,000$ rounds that matches one full day of operation. First, the algorithm selects the chain branch to use (see SelectBranch below). Then, it parses the selected branch starting from the newest block till the T_a oldest block. All the identities with recorded confirmation messages in this period are marked as active (SelectActive). Next, the algorithm finds the age of active

Algorithm 1 Pseudocode for single round of RRR

```
1: procedure RRRRound(chain, nodes{}, myID)
2:     candidates{} ← SelectCandidates(chain)
3:     endorsers{} ← SelectEndorsers(chain)
4:     txs{} ← receivePendingTxs()
5:     if myID ∈ candidates then
6:         goto: CandidateFlow
7:     if myID ∈ endorsers then
8:         goto: EndorserFlow
9:     CandidateFlow:
10:        intent ← createIntent()
11:        broadcast(intent)
12:        confirms{} ← receiveConfirms()
13:        for all confirm ∈ confirms do
14:            if validateConfirm(confirm, endorsers) ≠ true then
15:                confirms ← confirms \ confirm
16:            if myID ≠ confirm.candidate then
17:                confirms ← confirms \ confirm
18:        if |confirms| < q then
19:            goto: BlockFlow
20:        newBlock ← createBlock(txs, confirms)
21:        broadcast(newBlock)
22:        chain ← chain ∪ newBlock
23:        return true
24:    EndorserFlow:
25:        intents{} ← receiveIntents()
26:        oldest ← ∅
27:        for all intent ∈ intents do
28:            if validateIntent(intent, candidates) ≠ true then
29:                intents ← intents \ intent
30:            if oldest.candidate.age < intent.candidate.age then
31:                oldest ← intent
32:        if oldest ≠ ∅ then
33:            confirm ← createConfirm(oldest)
34:            sendToCandidates(confirm)
35:        goto: BlockFlow
36:    BlockFlow:
37:        blocks{} ← receiveBlocks()
38:        newBlock ← ∅
39:        for all block ∈ blocks do
40:            if validateBlock(newBlock, candidates, endorsers) ≠ true then
41:                blocks ← blocks \ block
42:            if newBlock.age < block.age then
43:                newBlock ← block
44:        if newBlock = ∅ then
45:            return false
46:        chain ← chain ∪ newBlock
47:        return true
```

identities and it marks an identity as inactive, when it has been the oldest for previous N_C rounds (to exclude it from selection if it is not responding). Finally, it sorts this list by age and returns the N_c oldest identities and their ages.

SelectEndorsers() computes a list of recently active identities as explained above (SelectActive). If an identity was created less than enrolment threshold T_e rounds ago

Algorithm 2 SelectActive

```
1: procedure SelectActive(branch)
2:     iterBlock ← Top(branch)
3:     i ← 0
4:     ActiveSet{} ← ∅
5:     while i < T_a do
6:         i ← i + 1
7:         BlockEndorsers ← GetEndorsers(iterBlock)
8:         for all Endorser ∈ BlockEndorsers do
9:             if Endorser ∉ ActiveSet then
10:                ActiveSet ← ActiveSet ∪ Endorser
11:        iterBlock ← Next(iterBlock)
12:    return ActiveSet
```

Algorithm 3 SelectCandidates

```
1: procedure SelectCandidates(branch)
2:     ActiveSet{} ← SelectActive(branch)
3:     SortByAge(ActiveSet)
4:     CandidateSet{} ← ActiveSet[0:N_{c−1}]
5:     while InactiveRounds(CandidateSet[0]) ≥ N_c do
6:         CandidateSet[0] ← Inactive
7:         CandidateSet[N_c] ← ActiveSet[N_c]
8:         ShiftLeft(CandidateSet,1)
9:     return CandidateSet
```

(e.g. $T_e = 100$) it will be excluded from selection to prevent grinding attacks. The algorithm selects N_e identities using standard simple random sampling (without replacement), where identities are sorted based on their public key binary. Random sampling uses $seed_{r-d}$ from the stable part of the chain.

Algorithm 4 SelectEndorsers

```
1: procedure SelectEndorsers(branch)
2:     ActiveNodes{} ← SelectActive(branch)
3:     for all node ∈ ActiveNodes do
4:         if enrolmentAge(node) < T_e then
5:             ActiveNodes = ActiveNodes \node
6:     EndorserSet ← RandomSampling(Seed_{r−d}, ActiveNodes)
7:     return EndorserSet
```

5.4.2 Endorsement protocol

Once the leader candidates and endorsers have been selected, each candidate runs an interactive protocol, shown in Figure 5.4, with the endorsers. The protocol consists of three fixed-length phases.

Intent phase. Each leader candidate c broadcasts :

$$\mathsf{Intent} = (id, pk_c, r, h_p, h_{tx}, sig_c)$$

This contains the chain identifier id, the candidate's identity pk_c, the current round

number r, the hash of the previous block h_p, the hash of the transactions h_{tx} the candidate proposes to include in the next block, and the candidate's signature sig_c over these elements. If multiple chain branches exist, the candidate uses SelectBranch, described below, to choose which branch to extend.

Confirmation phase. Each endorser e verifies all Intent messages received during the intent phase by checking that the sender is a valid leader candidate. Among the valid Intent messages, the endorser selects the oldest candidate and sends to it:

$$\mathsf{Confirm}_{e \to c} = (id, h_i, h(pk_v), sig_e)$$

This indicates that endorser e has confirmed candidate c. This message contains the chain identifier id, hash of the intent message h_i, the endorser identity $h(pk_e)$, and a signature sig_e over the previous elements. If multiple candidates have the same age (i.e. they were enrolled in the same block), we choose the oldest candidate in the order in which their enrolment messages appear in the block. If an endorser receives intent messages that refer to more than one chain branch, the endorser picks the branch to confirm using SelectBranch.

Block dissemination phase. If candidate c receives at least q Confirm messages, it is chosen as the leader to create a new block. The leader creates a new random seed $seed_r$ and the matching proof π_r using the previous seed: $\{seed_r, \pi_r\} \leftarrow VRF(sk_m, seed_{r-1})$.

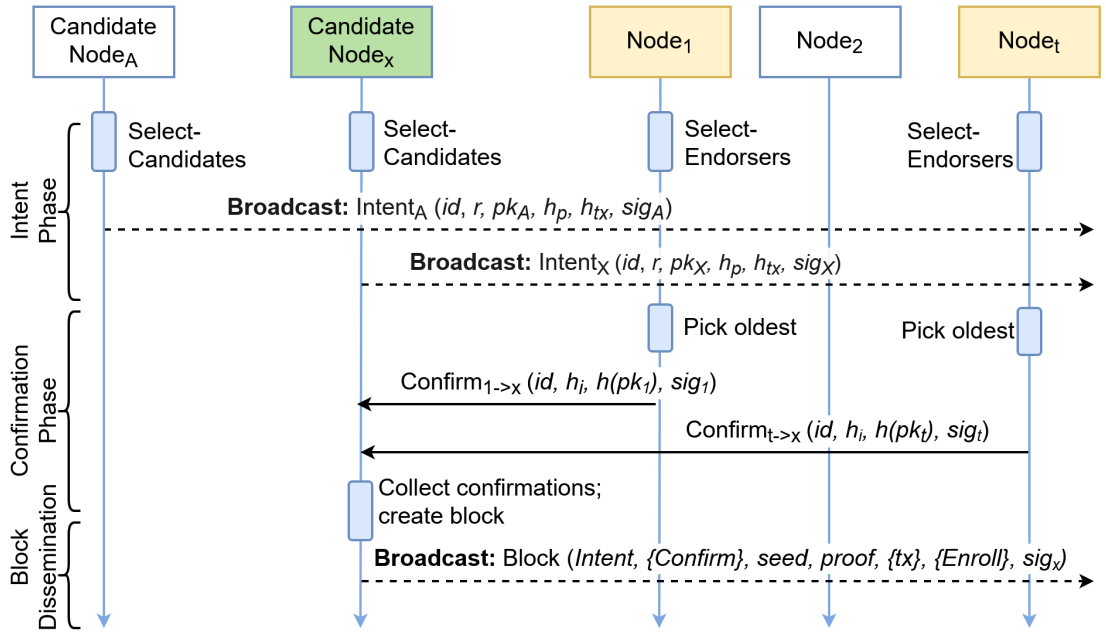


Figure 5.4: Endorsement protocol. Each leader candidate broadcasts Intent messages and endorsers reply with Confirm messages. A node that receives the required quorum of confirmations becomes an eligible leader for that round and can broadcast a new block.

The used VRF should be such that given a random input, the output should be random, even when the keys are generated by the adversary [105]. After that, the leader creates and broadcasts a new block:

$$\mathsf{Block}_r = (\mathsf{Intent}, \{\mathsf{Confirm}\}, \{tx\}, \{\mathsf{Enroll}\}, seed_r, \pi_r, sig_c)$$

This contains his intent, the received confirmations, new transactions $\{tx\}$, any enrolment messages of new identities, the new seed $seed_r$, the matching proof π_r, and a signature sig_c over these elements.

5.4.3 Chain validation

Chain validity is verified using the following algorithms.

SelectBranch() selects the valid branch among multiple choices. First, it verifies the correctness of each branch using VerifyBranch. Then, it computes a length for each of the branches which is defined by the number of rounds with missing blocks and selects the longest branch. If more than one branch has the same length, it chooses the branch with the older leader at the point of divergence.

Algorithm 5 SelectBranch

```
1: procedure SelectBranch(Branches{})
2:     for all branch ∈ Branches{} do
3:         if VerifyBranch(branch) ≠ true then
4:             Branches ← Branches \branch
5:     Branches ← SortByLength(Branches)
6:     Longest{} ← SelectLongest(Branches)
7:     if |Longest| = 1 then
8:         return Longest[0]
9:     else
10:        Selected ← Longest[0]
11:        counter ← 1
12:        while counter < |Longest| do
13:            current ← Longest[counter]
14:            Divergent ← GetFork(Selected,current)
15:            if LeaderAge(Selected, Divergent) < LeaderAge(current, Divergent) then
16:                Selected ← current
17:        return Selected
```

VerifyBranch() checks that a given chain branch is correctly constructed (pseudocode in Appendix A.3). It traverses the chain and checks that each block contains a correct hash of the previous block. For each block, it verifies the VRF proof of the random seed. All new identities must have correct Enroll messages (with valid attestation evidence or identity rewards). The algorithm verifies that the miner of each block was a candidate on that round (SelectCandidates), the block contains q confirmations, the confirmation messages contain the hash of the Intent message included to the block, the set of included transaction match h_{tx} from Intent, and the endorsers were eligible on that round (SelectEndorsers).

5.5 Security analysis

In this section, we analyse the security of our proposal. For our analysis we use the definition of *stability* from Bonneau et al. [47] with minor adaptations. We say that a consensus scheme is stable if it provides:

- *Eventual consensus.* At any time, all honest nodes agree on a *prefix* of what will eventually become the valid blockchain.
- *Exponential convergence.* The probability of a fork at depth d in the chain is proportional to (2^{-d}). That is, after a transaction is added to a block that is extended with a small number of valid blocks, the transaction is permanently part of the chain with a very high probability.
- *Liveness.* New blocks continue to be added and valid transactions included in the blockchain within a reasonable amount of time.
- *Correctness.* All the blocks in the prefix of the eventually valid chain will only include valid transactions.
- *Fairness.* In expectation, consensus participants with a fraction α of all stake will create a fraction α of all blocks, and collect a similar fraction of block creation rewards.

5.5.1 Consensus and convergence

We first consider the benign case where no participant intentionally manipulates the random seed. After that, we consider the more complicated case where the attacker does manipulate the seed.

To examine the different possible cases in leader endorsement, consider an example where the three oldest leader candidates are A, B and C. The endorser committee is sampled based on $seed_{r-d}$. When $seed_{r-d}$ is unbiased and the identities that take part in the sampling have been fixed before $seed_{r-d}$ is known (as is the case in our solution), on average α of the sampled endorsers are adversary-controlled. Fraction β of the endorsers may not receive Intent sent by the oldest candidate A. The remaining fraction $1-\alpha-\beta$ of endorsers who received all messages, confirm A as the leader. Those endorsers that did not receive all messages may confirm another candidate (B or C). The adversary-controlled endorsers may confirm more than one candidate (A and B), although such equivocation leaves evidence that can be easily used to penalise malicious identities (see § 5.7 for discussion).

In a rare case, at least q endorsers are sampled from the fraction $\alpha+\beta$ of active identities. In such a case, the second-oldest candidate B may also receive the required confirmations, causing two eligible leaders (A and B) and a fork in the chain. We denote the probability of such *benign fork sampling* as Pr(BFS). Assuming sufficiently

many active identities n_a, it can be computed as:

$$\mathsf{Pr(BFS)} = \sum_{i=0}^{N_e-q} \left(\binom{N_e}{q+i} (\alpha+\beta)^{q+i} (1-\alpha-\beta)^{N_e-q-i} \right).$$

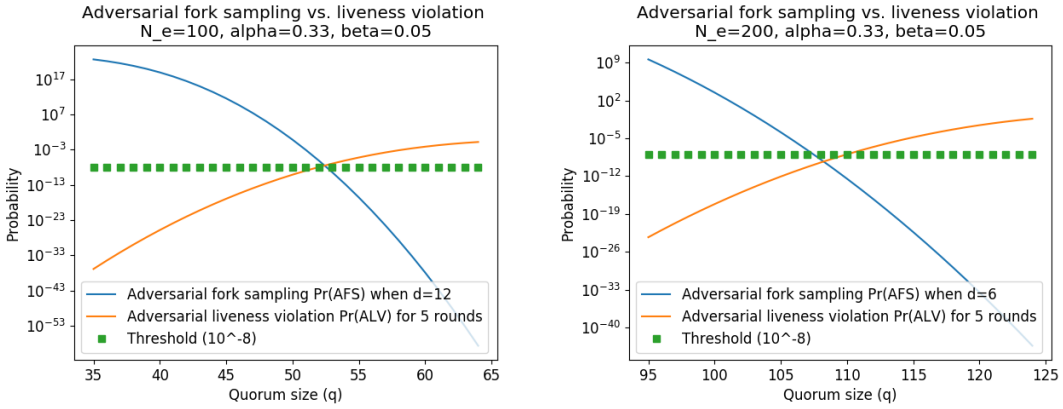
For example, when $\alpha = 0.33$, $\beta = 0.05$, $N_e = 100$ and $q = 54$, then $\mathsf{Pr(BFS)} = 0.0008$. That is, such sampling would take place, on average, every 1200 rounds. Extending both forked branches requires another similar sampling. As the probability of consecutive sampling decreases exponentially and the probability of three consecutive samplings is already very low (5.78×10^{-10}) for such parameter values, we consider the maximum depth of forks $d = 3$ in the absence of seed manipulation.

Next, we consider the adversarial case where the attacker intentionally manipulates $seed_r$ to bias endorser selection. Recall that we use VRFs to update the seed for each new block. If the adversary controls more than one oldest leader candidate, it may choose which one of these identities it uses to create the block and update the seed. This gives the adversary more than one seed to choose from. If the adversary similarly controls more than one oldest-leader candidate on the next round, he can again choose which candidate to use to update the seed. Such a process allows the adversary to build a "seed prediction tree" whose expansion factor is the number of controlled oldest candidates in each round and whose depth is the number of successive rounds where the adversary controls more than one oldest candidate. Since identity enrolment is permissionless and open, the adversary may control multiple oldest candidates on several successive rounds and build a large seed-prediction tree.

Assume an adversary that in round r builds a seed prediction tree of depth d_t and with 2^{80} leaves. We assume that building a tree larger than that is infeasible, as the tree needs to be constructed online without pre-computation. This tree allows the adversary to pick the seed-update schedule in round r that will give the best endorser sampling sequence starting from round $r + d_t$ out of the 2^{80} predicted options. Recall from our analysis above that, given our example parameter values, benign fork sampling probability $\mathsf{Pr(BFS)} = 0.0008$. The probability of finding such sampling on, for example, $d = 12$ successive rounds reduces exponentially and becomes very low (6.87×10^{-32}). With the above seed-prediction tree, the adversary has 2^{80} attempts to find such a sequence of samples. We call the probability that the adversary finds such *adversarial fork sampling* $\mathsf{Pr(AFS)} = 6.87 \times 10^{-32} \times 2^{80} = 8.3 \times 10^{-8}$. Thus setting the maximum depth of forks to $d = 12$ prevents such attacks even if we grant the adversary the unreasonably excessive power to compute and store such a large seed-prediction tree. This, incidentally, is the same number of confirmations that merchants are recommended to wait for before considering a transaction "final" by the Ethereum community. We chose the values for N_e and q after calculating the probability curves for several different parameter values which can be found in the additional analysis in Appendix B.

Because identity enrolment is open, seed-prediction attacks cannot be prevented

altogether. However, in § 5.7 we discuss how such attacks can be made difficult to realise in practice by using multiple identity queues and forcing the adversary to plan the attack years before its execution.



(a) For parameter values $N_e = 100$, $\alpha = 0.33$, $\beta = 0.05$ quorum $q = 54$ prevents forks at depth $d = 12$ without compromising liveness.

(b) For parameter values $N_e = 200$, $\alpha = 0.33$, $\beta = 0.05$ quorum $q = 108$ prevents forks at depth $d = 6$ without compromising liveness.

Figure 5.5: The quorum size q represents a trade-off between security and liveness. As q increases, the adversarial fork sampling probability Pr(AFS) reduces and the adversarial liveness violation probability Pr(ALV) increases.

Increasing the quorum size q reduces the probability of forks at depth d, but weakens liveness guarantees as explained later. Figure 5.5a shows that the quorum value $q = 54$ provides a good balance of security and liveness when $N_e = 100$.

Increasing the number of endorsers N_e also reduces d. As shown in Figure 5.5b, when $N_e = 200$ endorsers are used, forks can be reduced to $d = 6$ rounds without compromising liveness. The main drawback of larger N_e is that such a solution requires more communication in each round. We discuss system performance and communication complexity in more detail in § 5.6.

In Appendix B we extend this analysis to consider different parameter values, including stronger adversaries (e.g. $\alpha = 0.4$), better connectivity (e.g. $\beta = 0.01$), and larger endorser committees (e.g. $N_e = 400$).

5.5.2 Liveness

Block creation requires at least one of the leader candidates to receive q confirmations. We first consider the benign case where all endorsers confirm the oldest Intent they receive. We denote the probability that more than $N_e - q$ endorsers will be sampled from the fraction of β identities that did not receive the Intent message as *benign liveness violation* Pr(BLV) and compute it as:

$$\mathsf{Pr(BLV)} = \sum_{i=0}^{q} \left(\binom{N_e}{N_e - q + i} \beta^{N_e - q + i} (1 - \beta)^{q - i} \right).$$

Given the previous example parameters, this probability is negligible (6.96×10^{-33}). Next, we consider the case where the adversary reduces the probability of successful block creation by intentionally not sending Confirm messages to targeted leader candidates. Such *adversarial liveness violation* probability Pr(ALV) can be computed as:

$$\mathsf{Pr(ALV)} = \sum_{i=0}^{q} \left(\binom{N_e}{N_e - q + i} (\alpha + \beta)^{N_e - q + i} (1 - \alpha - \beta)^{q - i} \right).$$

Given the previous parameters, the adversary can prevent mining with probability 0.062, that is, on average they can block it every 16th round. The probability of being able to prevent mining on five successive rounds is 9.37×10^{-7} (see Figure 5.5a). If the adversary continues this strategy longer than the activity period T_a, its identities will be considered inactive and thus can no longer reduce the mining probability for other participants.

As in any leader-based blockchain consensus scheme, the chosen leader can exclude transactions from targeted users, so no such scheme can provide an absolute guarantee that a new transaction is included in the next block. In our approach, this problem is somewhat exacerbated. Since the adversary may control block creation on multiple successive rounds, it may prevent inclusion of specific transactions for an even longer period. So our proposal provides weaker resistance to such censorship than schemes based on random leader selection. While we cannot prevent denial-of-service attacks altogether, in § 5.7 we will discuss how such attacks can be made difficult to realise in practice by using multiple identity queues.

5.5.3 Correctness

Regarding transaction correctness, as in any other leader-based consensus scheme, the chosen leader can include invalid transactions in the published block. Users can detect and ignore incorrectly formatted transactions. Transactions that appear valid in the current branch but contradict transactions in another branch (e.g. double spending) can be detected by waiting d rounds. Thus all transactions in the chain prefix up to Block_{r-d} are either valid or ignored.

5.5.4 Fairness

The adversary can attempt to violate fairness in few ways. The first approach is that the adversary does not include Enrol messages from the targeted victim participant in its blocks. This approach can delay enrolment of a new identity by a few rounds, but cannot prevent it, and thus does not violate fairness in the long term. The second approach is that the adversary does not include Confirm messages from the victim in its blocks, so after T_a rounds the victim is excluded from miner candidate selection and has to

re-enrol. This *adversarial exclusion* probability can be computed as

$$\mathsf{Pr}(\mathsf{AE}) = (1 - N_e/n_a)^{T_a(1-\alpha)}.$$

Assuming $n_a = 10,000$ active participants and our example parameters, the adversarial exclusion probability is negligible (3.25×10^{-59}). If the size of the system increases to $n_a = 100,000$ the adversarial exclusion probability is still low (1.5×10^{-6}). If the grows larger than that, the value of T_a may have to be increased to ensure that active identities are not excluded from selection.

5.5.5 SGX considerations

Any attempt to enroll non-SGX platforms should fail, as the IAS will not return the signed QUOTE needed for enrolment. Enrolling the same SGX platform multiple times would also fail, because the IAS would return a pseudonym p_n that is already recorded for another identity on the chain. The third alternative is to enrol the same SGX platform to multiple chains and try to reuse enrolment from one chain to another. The QUOTE contains the chain identifier *id* to block this.

The adversary gains no advantage (in terms of more identities or selection bias) by breaking into their own SGX processors. Besides attestation, we only use enclaves to protect the IAS credential, whose leakage does not allow the adversary to create additional identities. A malicious chain creator could initialise an invalid chain, whose members are not constrained to be SGX processors, but any legitimate participant can detect this from the missing QUOTEs in the genesis block.

If the attestation service (IAS) is temporarily unavailable, new identities cannot be enrolled. However, the system can produce new blocks and process incoming transactions normally. Therefore, the IAS is not critical for operational liveness.

5.5.6 Privacy considerations

Since block creation is based on long-term identities, correlation of block creation events by the same participant becomes trivial. This is a limitation of our approach compared with systems where participants pick new identities for every round. However, the identities used for transactions can be completely separate from those used for consensus and block creation. For example, transactions can be based on changeable pseudonyms or cryptographic commitments that hide user identities and transaction values [150, 208]. Such patterns are already common in permissioned blockchains where the consensus-layer keys are disjoint from the smart-contract layer keys.

5.6 Performance evaluation

In this section we explain the experiments we performed in order to estimate suitable round duration t_r, transaction latency and throughput.

5.6.1 Experimental setup

We built a globally distributed peer-to-peer network using Amazon's AWS infrastructure. We instantiated nodes in Frankfurt, London, Singapore, Mumbai and Oregon. We used EC2 compute services with nodes ranging from t2.micro (single vCPU with 1 GB RAM) to m4.2xlarge (8 vCPUs and 16 GB RAM). The node software was written in Java and run on Ubuntu/Linux OS. To simulate the maximum round trip time, we ensured that the leader candidate was never located in the same data centre as any of the endorsers. To simulate global distribution of participants, we enforced that messages travel through at least x different nodes ($x = 0$, 2 or 6) before reaching their destination. We set the Intent and Confirm message sizes to 1 KB (although actual messages are smaller). Lastly, we tested for three block sizes: 500KB, 1MB and 2MB.

We note here that these experiments do not incorporate adversarial or offline nodes. The goal here is to provide a baseline for parameter values rather than to test the robustness of the implementation we used. Additional testing is required to evaluate the behaviour of our code under faulty or adversarial network conditions.

Network optimisations During testing we observed that most of the block dissemination delay came from the initial block transmission by the leader candidate, due to a high out-degree and multiple hops across geographically distant locations. To tackle these issues, we implemented a networking structure as shown in Figure 5.6, where we selected some nodes within a cluster of geographically distant nodes to serve as *top-level nodes*, i.e. nodes that are directly connected to by leaders when broadcasting the block. These top nodes have a large out-degree to mid-level nodes within the same geographical area. This optimisation led to a significant reduction in block dissemination latency. We did not see the need to use the same approach in the intent phase, as it did not lead to any noticeable improvement.

We emphasise that the top and middle nodes are not different from the others. Any node could be chosen as a top or middle node and messages may be broadcast to multiple top nodes within a cluster. In a large deployment, the top-level nodes may be chosen by reliability and performance metrics as in the Tor network.

5.6.2 Results

We measured message-delivery times for various network and block sizes. Figure 5.7 summarises the results of our experiments.

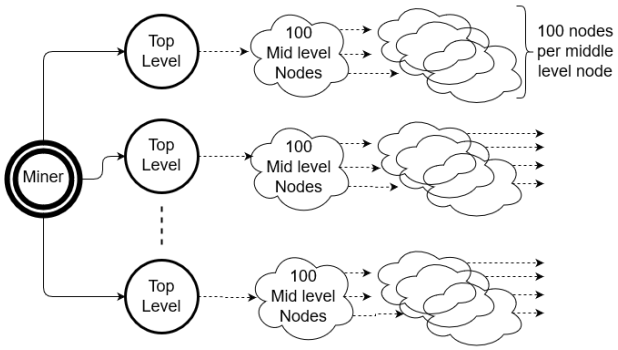
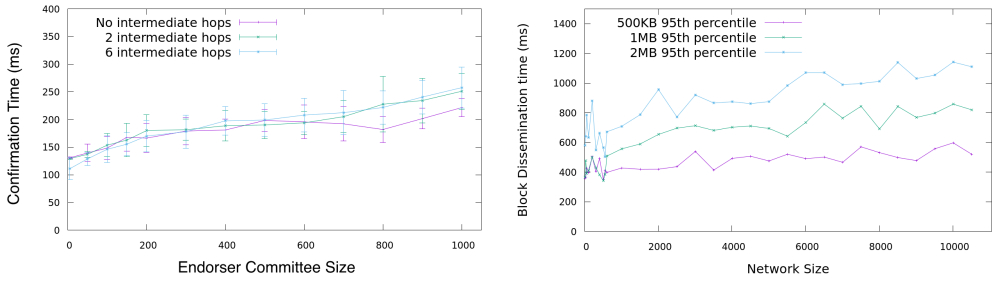


Figure 5.6: Network layout optimisation used in our experiments.

In Figure 5.7a we plot the time required for leader selection (combined Intent message delivery and Confirm message reception). This time grows from 130 ms for small endorser committee size $N_e = 5$ to 257 ms for large committee size $N_e = 1000$. We conclude that setting the combined duration of these two phases to one second is sufficient in a network environment like ours. We include the arguably excess buffer to account for clock drift and network jitter.

Figure 5.7b shows the time required for block dissemination (95th percentile) that grows from 357 ms for a system size of $n_a = 10$ active nodes to 1.1 seconds for a system size of $n_a = 10,000$ active nodes. We conclude that setting the duration of block dissemination phase to 4 seconds is sufficient for our network. The above two values give us a round duration of $t_r = 5$ seconds.



(a) Intent and confirmation time.

(b) Block dissemination time.

Figure 5.7: Experimental results for message delivery times in our test setup, a globally distributed peer-to-peer network using Amazon's AWS infrastructure with no faulty nodes.

Throughput and latency. Given $t_r = 5$ seconds, we can now estimate system throughput tp for our solution as follows:

$$tp = \frac{\frac{1}{t_r} \times (B - H - (N_e \times S_C) - (n_a \times S_E))}{T},$$

where H is the invariant block header (280 bytes), S_C is the size of Confirm message (416 bytes), B is the used block size, T is the transaction size, and S_E is the size of the Enroll message. Assuming $B = 2$ MB and $T = 250$ bytes, similar to Bitcoin [41],

$N_e = 100$ endorsers and few enrolments per round (owing to fast rounds), 99%+ of the block is left for the transactions and the system throughput is approximately 1500 transaction per second. Transaction latency is one minute (when $d = 12$) or 30 seconds ($d = 6$).

5.7 Discussion

In this section we briefly discuss security and performance improvements, alternative identities, the role of endorsers, penalisation of malicious participants, privacy implications, denial-of-service attacks and economic aspects.

Improved latency and liveness. An adversary who controls multiple oldest identities can predict seed evolution which enables deeper forks and thus higher latency. Although seed prediction cannot be prevented completely, it can be made difficult to realise in practice. One possible defensive approach is to use *multiple queues*. Recall that in our approach all identities are placed into one queue in the order of their age and the oldest identities are picked as leader candidates in round robin. Instead of using a single queue, identities could be placed into multiple queues. For example, in a system that has been operational for five years and each year equally many identities were enrolled, a separate queue could be established for each enrolment year. The leader selection could happen such that the oldest identity is picked from each queue in turn. To perform successful seed prediction, the adversary would now have to plan the attack years in advance, so that they control identities in all queues at the right places. The same applies for targeted liveness attacks.

Multiple queues could be used also as a performance enhancing mechanism. As with sharding, each queue could process a separate set of incoming transactions in parallel to increase the overall throughput. Multiple identity queues are thus a promising direction for future work.

Predictable leader selection can make denial-of-service attacks easier. For example, the adversary can prepare the attack in advance and launch it when the victim becomes a leader candidate. On the other hand, such predictability can help participants avoid DoS attacks. Participants can obtain multiple IP addresses and switch to a different one before their identity becomes the leader candidate. Such strategies are harder to realise with randomised leader selection.

Bootstrapping from other infrastructures. As in Intel SGX, reliable identities could be bootstrapped from some other infrastructure. For example, mobile-phone operators, credit-card companies or passport issuers could take the role of IAS and provide an interface that allows their customers to enrol new identities, such as one identity per person or per mobile-phone subscription. Another attestation infrastructure that could be leveraged is TrustZone [25]. New and emerging secure processor architectures that

are designed specifically for distributed ledger technologies [218] could also be used to create Sybil-resistant unique identities. Recent efforts to standardise EPID provisioning and attestation across manufacturers [124, 123] could provide a vendor-independent way of bootstrapping these identities.

Expanding the role of endorsers. In our proposal, the endorsers confirm the oldest leader candidate they observe, regardless of the content of the block the candidate proposes to create. The endorsers' role could be expanded to examine the proposed block, e.g. for transaction validity. Such optimisations could allow the endorsers to ignore leaders that try to extend the chain with invalid blocks.

Penalising malicious behaviour. In most permissionless consensus schemes, identities can be changed easily. For example, a Bitcoin miner can use a different public key every time they start mining a new block. In our approach identities cannot be changed after initial enrolment, as they are recorded in the blockchain. One advantage of long-lived identities is that penalising malicious behaviour becomes possible. For example, if an endorser confirms multiple intents in the same round, any entity that observes this can broadcast the conflicting (and signed) confirmation messages and the next miner can include them to a new block as evidence of cheating. Broadcast denunciation can help eliminate malicious nodes from the system, providing an incentive to avoid misbehaviour – similar incentives are missing in systems where identities are freely changeable.

Economic aspects. In case of SGX identities, participants are incentivised to buy the cheapest processors that enable enrolment. If processors differ significantly in processing and purchase costs, this raises questions about fairness [35]. We argue that no leading manufacturer is likely to sell unused but outdated products at scale. Moreover, cheap second-hand processors may not provide an advantage because those CPUs may have already been enrolled. Lastly, the enrolment of old CPUs can be prevented: in SGX, the attestation group signature does not identify the individual CPU but it does reveal the manufacturing batch.

An important economic side-effect of deterministic leader selection is that it negates the need for pooled mining. In random leader selection (whether PoW or PoS), there is an incentive for small miners to collaborate with others and create a mining pool. This gives them assurance of predictable earnings but it centralises the network. Bitcoin, for example, has just four mining pools controlling 55.4% of its mining power [52]. By making rewards predictable, we take away incentives for anyone to join or create a mining pool thus eliminating this vector for centralisation.

Implementation. This research was motivated by a desire to find a scalable consensus algorithm for permissioned blockchain frameworks. Robust Round Robin, while also providing an incentive structure that makes it amenable to permissionless settings, is

still adept at providing a consensus mechanism for permissioned networks. In this sense, it resembles PoET which can also be used in either setting. Our efforts recently have thus been to get a production version integrated with a permissioned blockchain framework so the broader community can benefit from it. With this view in mind, we have developed an implementation of RRR that is fully compatible with Quorum; the code for this can be found at the project repository [51].

5.8 Related work

In § 5.1 we outlined the limitations of several related proposals. In this section we review additional related work. For a general comparison and classification of blockchain consensus, we refer the reader to the Systematisation-of-Knowledge paper by Bano et al. [35]

Other Proof-of-Stake schemes. Ouroboros Praos [75] is another PoS scheme that leverages VRFs for new random value generation in each round, similar to Algorand [154]. Its main limitation is that the randomness can be biased, so it does not provide fairness.

RapidChain [221] samples all consensus participants to get a reference committee, which is then responsible for running a distributed randomness-generation protocol in the start of each epoch to create new randomness for that epoch. The protocol is based on verifiable secret sharing (VSS). The main limitations are that the reference committee becomes an obvious target for attacks and the distributed random generation protocol is expensive.

DFINITY [111] introduces a novel decentralised random beacon that uses BLS threshold signatures to generate periodic unbiased random values. This scheme requires a setup phase during which an expensive distributed key generation (DKG) protocol is run. New random values can then be derived by collecting signature shares from sufficiently many participants. The per-round or per-epoch randomness generation has low communication complexity, but at the cost of an expensive DKG protocol that needs to be repeated when participants join or leave.

Other TEE solutions Proof of Luck (PoL) [156] is an SGX-based protocol with the same basic idea and the same main limitations as PoET (recall § 5.1): participants have an incentive to compromise their own platforms.

PoTS [21] is another PoS solution that uses SGX and is designed to tolerate compromised TEEs that control up to 50% of the stake. One drawback is that by compromising a small number of high-stake TEEs an attacker might compromise the entire system (due to concentration of stake by a few rich individuals). Moreover, the approach does not provide fairness. Finally, PoTS requires TEEs for its operation, while our proposal works also without them.

Resource Efficient Mining (REM) [223] replaces the hash computation of PoW with attested enclave computation. This approach allows more useful usage of energy, but does not eliminate the need for massive collective computation nor does it take away incentives for participants to compromise their own TEEs. Our approach requires no computationally intensive puzzles, saving significant amounts of energy compared to protocols based on proof of work.

Coin aging. PPCoin [137] proposed that each coin have an associated age and leader selection be based on a hashing procedure where the target difficulty is coin-specific and lower for older coins. However, it is vulnerable to a simple attack where the adversary waits until they own enough old coins, then creates a deep fork for double spending. The authors suggest that such attacks could be blocked by a central time-stamping mechanism – a strange argument to make for an ostensibly decentralised and permissionless blockchain. Additionally, the leader selection is not fair, because selection can be manipulated with simple grinding.

5.9 Conclusion

We have proposed an alternative idea for blockchain consensus—selecting consensus leader candidates *deterministically* instead of the common *random* selection approach and complementing such selection with a simple interactive endorsement protocol. The main benefits of our solution are fairness and resilience to TEE compromise, as well as relatively high throughput and scalability. As our analysis shows, fairness is especially important in systems where block creation is rewarded with new stake which is a common practice in permissionless blockchains. Although deterministic selection also has its own limitations (weaker DoS resilience), our work provides a viable and previously unexplored alternative to random selection.

"Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won't come in."

—Isaac Asimov

6 Rethinking consensus

The previous chapter presented a scalable blockchain consensus algorithm in the form of Robust Round Robin. Looking at the performance of that system, the dissemination of new blocks ends up taking the vast majority of time as the network scales beyond a hundred or so nodes. This is a fundamental limitation of operating on a shared ledger: every node must have all the information on the ledger.

In this chapter, we present a new kind of decentralised network – one that builds not upon a linear ledger but on a tree-like structure. We call this system *Cambium*[1] and demonstrate how it seeks to achieve consensus with only logarithmic communication costs in the number of nodes. To understand how it works and how such a system could be useful, let us discuss its operational model with an illustrative example.

6.1 Lightweight yet verifiable commitments

When Alice is proving commitments to Bob, there are two kinds of commitments Bob may be interested in: positive commitments, which Alice has an incentive to share, and negative commitments which Alice may have an incentive to hide. Take for example car registration and maintenance records: if Alice is selling her car to Bob, she's happy to share the times she has taken it in for scheduled maintenance, but she would prefer to not show Bob the repairs from any accidents she had.

Let us consider this example of car maintenance in greater detail. Currently, there is a trend of car manufacturers seeking greater control over the maintenance of the

[1] Cambium is named after a layer of plant tissue that plays a crucial role in plant growth and the merging (inosculation) of trees.

cars they produce. Mercedes-Benz, for example, no longer provides a physical service book to owners as all maintenance records are now recorded in their database [14]. In this model, the manufacturer becomes the sole source of truth for car history as well as its gatekeeper. Tesla goes a step further, with the company actively campaigning against right-to-repair laws claiming vaguely specified "cyberattacks" [141], restricting access to repair information from garages and maintaining tight control of after-sale servicing [142].

As it becomes the norm for cars to be Internet-connected and keep a record of maintenance and repair actions, could we design a system that is less centralised than Mercedes' and Tesla's vision? The users of such a system would be diverse and numerous: service centres, car owners, buyers, and even insurance companies. The goal of this system would be to have verifiable records of cars (maintenance, insurance claims, recalls) in order to make the used car market less of a market for lemons without handing total control to manufacturers or insurance companies. To simplify matters, we will ignore repairs that don't touch the electronics, such as when a car owner pays cash to a panel-beater to straighten out a minor dent. We will assume that repairs of substance involve interaction between a car's systems and those of a garage, and can thus in principle be logged electronically by the car, the garage or both. If lawmakers wish to open up aftermarkets, are there any decentralised alternatives to OEM control?

One possible design would be to put everything on a blockchain. But that would hit several scalability limits: with millions of cars on the road, the throughput ceilings of permissionless blockchains would be breached. If we went the permissioned route, then given the state of the art in blockchain consensus algorithms, it would not be feasible to have every car and maintenance shop act as consensus nodes. Thus, we'd have to trust the government or a small set of companies to run things, raising many policy issues from cartels to privacy.

If we want to avoid big centralised systems and cannot use blockchains, what's left? Let us consider a simpler model where every stakeholder – car, maintenance shop or insurance company – maintains their own records locally, and build a system that enables each of them to prove exactly those facts that they need to prove. Let us assume, for now, a central timestamping service that is universally trusted. When Alice takes her car to the maintenance shop, her car and the maintenance shop create a packet that states the date, the car ID and the work performed. They send this to the service which returns a signed packet with the time of signing. The service itself need not store these records, as it simply attests the time at which commitments were made. When Alice wants to sell her car to Bob, and prove that the last scheduled maintenance was done, the car can simply show him the signed timestamp.

The next step would be to decentralise the timestamping service. Existing distributed timestamping schemes either build on top of blockchains [98, 113] or have multiple rounds of communication between all nodes in the network [145]. While these schemes

could provide us with eventual consistency and greater resilience to DoS attacks than a centralised service, we do not believe any current designs scale enough to accommodate the hundreds of millions of users of a car maintenance system deployed in say, dozens of European countries. A possible approach could be to use one of these distributed timestamping solutions among successive subsets of participants. We would then have some nodes acting as delegates, running the timestamping protocol among themselves and then broadcasting it to the network. However, such a delegation model is vulnerable to adversarial take-over: compromising a very small proportion of the network gives the adversary total control.

There is also a more fundamental issue with using a simple timestamping service. Such systems work well for positive information, like scheduled maintenance, which Alice has an incentive to show to Bob. *Negative information* is different: Alice needs a way to prove to Bob that the history he has been shown is the complete record, with nothing left out. We need an electronic equivalent of a paper logbook.

The obvious first attempt would be to make the car's engine control unit tamper-resistant. OEMs do this to some extent in order to demonstrate compliance with environmental regulations and to control aftermarkets. However it is hard to do well enough to exclude capable motivated opponents, given the cost pressures on OEMs and the complexity of the ecosystem [185]. So, just as card payment systems rely to some extent on the difficulty of cloning cards but also on logs and journals kept on bank and merchant servers, we might ask whether we can fortify any tamper-resistance using a trustworthy distributed system.

A simplistic approach might involve mandating that the timestamping service signs exactly one packet per day (or per other time epoch) from every entity. If the car doesn't have anything to commit on some day then it must send a `NULL` message to the service and store the signed acknowledgement.

Now, suppose Alice gets into a minor accident and goes to a repair shop to get it fixed. The car maintains a record of the daily packets including the one for the day of the repair. If she tampers with it and removes that record, then when she sells the car to Bob, he knows that there is information missing. In this manner, we can add state to a timestamping service so that a car can get attestation of a maintenance record, without relying on a centralised database.

The timestamping service is still a single point of failure, so our next step will be to allow all the cars, garages and other stakeholders to collaboratively act as such a distributed timestamping service. In this new model, nodes aren't expected to process and store data that is irrelevant to them: Alice's car only stores records for itself, not for Dave's, Charlie's and Eve's cars as it would in a blockchain system. Each stakeholder stores a set of message hashes at each round, and these hashes together enable it to prove both positive and negative assertions about commitments it has made in the past.

In the remainder of this chapter we will present a novel consensus algorithm, Cam-

bium, that can be used to construct such a lightweight decentralised system.

6.2 Achieving consensus with logarithmic costs

Research on consensus has tended to assume that everyone must necessarily hear what everyone else has said. Here, I challenge that implicit assumption and present a consensus algorithm[2] that achieves global consensus despite each node communicating only with a small subset of the network.

Recent research in Byzantine fault tolerance has been driven by the surge of interest in Bitcoin and other ledger-based systems, where transactions sent by clients are seen by every other node, and recorded in a globally-visible append-only ledger. This requires communication costs to increase (at least) linearly in the number of nodes. If the transactions contain code that must be executable, or at least verifiable, by all, then the ledger must hold them globally. This means that we also have a linear increase in per-node storage and computation costs. These issues inherently limit the scalability of ledger-based systems.

Cambium solves the scaling problem in two ways. First, it builds not on a linear ledger but rather on a data structure that we call the *banyan trie*. Banyan tries only require nodes to store their own data and the proofs that support them, not blocks of irrelevant data; the reader of a piece of data is responsible for validation, rather than the writer (§ 6.4.10). Second, every node only communicates with a logarithmic number of other nodes (§ 6.3.2). This provides a more scalable way of achieving decentralised consensus, as the computation and storage costs per node grow only logarithmically in the number of nodes, rather than more than linearly. It does not do everything that a bitcoin or ethereum blockchain can do, but in some applications it is quite capable.

Many applications do not need a public ledger as the data are private by default. The typical clients of the original Haber-Stornetta timestamping service [110] were patent attorneys wishing to establish priority for inventions that were not yet public. Similarly, many proposed blockchain applications assume that sensitive plaintext payloads are stored off-chain, and only their hashes appear on the chain – so-called "anchoring"[3].

Cambium has two further ideas. It is divided into time periods in each of which we run a cycle of the protocol, and at each cycle, we run something similar to the Merkle trees in a traditional timestamping service but in a distributed way. Hashes from each node propagate to a root, from which hashes propagate out again to provide a basis for the next epoch. We might call these 'banyan trees' after the trees that drop 'prop roots' to the ground, which sprout new trunks leading to a thicket of linked trees. The idea is that the history of each thicket will eventually become consistent, just as we expect in a blockchain.

[2]Designed in collaboration with Dann Toliver as detailed in § 1.2.
[3]This is also done to reduce storage costs and transaction fees.

The final idea is that rather than using the traditional Merkle trees, we use Merkle tries instead. A trie (which some people pronounce as 'try'), also known as a prefix tree, is an ordered tree used to store an associative data structure. All the descendants of a node have a common prefix of the string associated with that node [163]. Tries are used in applications such as dictionaries and search autocomplete. Our Banyan trie data structure enables a proving node to marshal all the relevant commitments it's made over a series of epochs and display them to a verifying node. The nature of the trie data structure means that any commitments of a certain type must appear in certain places, so these places can be examined to prove a negative, without the exhaustive enumeration required to establish a negative on a blockchain. It also means that the nodes can all work out the current cycle root in parallel; in the absence of a severe attack or partition, most will get the same value. If this description seems too telegraphic, we expand it in the next section, and then describe the components fully in the section after that.

The banyan trie is closely related to the linked tries of the TODA Proof of Provenance data structure [210], and is fully compatible with that structure. Our original contribution in Cambium is a novel algorithm for distributed consensus, as the original TODA-as-a-Service system uses banyan tries in a centralised system for proof of provenance [209]. We retain compatibility with its use cases and explore others in § 6.7.

The design of Cambium is built up from a small number of sub-protocols, or modules, described in § 6.4. Several of these modules have parameters that are tunable on a per-installation basis, to allow interoperable accommodation of a wide variety of use cases and network conditions. The source code for most modules is available at the project repository [10].

Cambium's security guarantees against forking attacks appear surprisingly robust given its communication constraints, and are discussed in § 6.5. Furthermore, Cambium provides for good censorship resilience due to its logarithmic communication patterns. On the flip side, Cambium provides relatively poor liveness guarantees as compared to blockchain consensus algorithms.

6.3 Design overview

In this section we introduce the core data structure underlying Cambium: the banyan trie. Then, we show the "happy path" process for Cambium to illustrate protocol flow under ideal conditions, and present some preliminary observations.

6.3.1 Banyan tries

As Cambium uses a trie-based data structure instead of a ledger, we first need to understand the trie as a data structure and see why it is a natural choice.

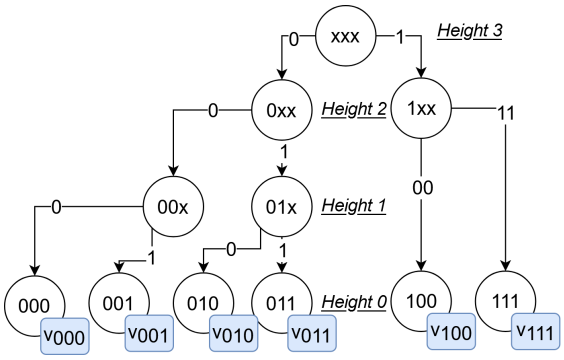


Figure 6.1: Illustration of a binary trie.

6.3.1.1 Merkleised data structures

A Merkle tree uses a cryptographic hash function, which is collision-resistant and second-preimage resistant, to hash the values at its leaves up to a value at the root which serves as a commitment to all the values at the leaves. Structurally, it is a tree in which every leaf node is labelled with a hash and every non-leaf node is labelled with a hash of the concatenation of the hashes of its child nodes. Operationally, we can think of a Merkle tree as a list of elements with a unique fingerprint and a short proof of membership for a given element. If Alice has the list and Bob has the fingerprint, Alice can prove that her list matches Bob's fingerprint by sending the list to Bob. She can prove that an element is in the list by sending the element and a short proof to Bob.

A Merkle trie is a similar data structure that in addition to a hash also assigns an index to each node. Figure 6.1 illustrates a binary trie; note how the index of a node is arrived at by traversing the trie. As the figure shows, the nodes at each level are lexicographically ordered. Because of this, tries support efficient find and insert operations using the indices [49].

Operationally, we can think of a Merkle trie as a key-value data structure (or dictionary) with a unique fingerprint, a short proof of membership for a given key/value pair, and a guarantee that each key has exactly one value (which may be null). If Alice has a dictionary, then she can prove to Bob that some key/value pair $x : y$ is in the dictionary by sending that pair to Bob along with a short proof. And we get the crucial proofs of non-existence: once Bob has validated that proof he knows that the *only* possible value for x in that dictionary is y.

This operational view of Merkle tries makes clear why it is an obvious data structure for committing to a single value. Recalling our discussion about the car maintenance system, this is exactly what we want from our underlying data structure. Transactions made with the car can be represented by Merkle proofs in a binary trie in Cambium.

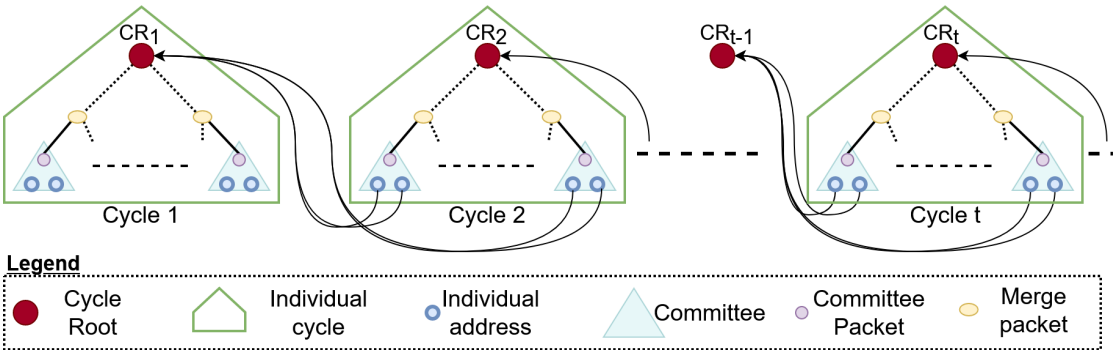


Figure 6.2: Illustration of a banyan trie; note the cycle root from the previous cycle trie being used as an input to the next cycle trie, similar to the linking of blocks seen in blockchains.

6.3.1.2 Linking Merkle tries

Cambium proceeds in a series of discrete time periods, called *cycles*. The goal of each cycle is to produce a *cycle trie*, containing all the activity in that cycle. Each leaf of this cycle trie contains the Merkle root of the previous cycle trie, called its *cycle root*. A *banyan trie* is a succession of cycle tries, just like a blockchain is a succession of blocks.

This banyan trie data structure is the core of Cambium. Each node has a different view of it, and typically no node will have complete knowledge of any cycle trie. This is unlike a blockchain, where every node sees the entire ledger. Here each node knows only its own Merkle proof in each cycle trie; the only globally shared knowledge is the sequence of cycle roots. Figure 6.2 illustrates this structure.

Each cyle trie within the banyan trie has a depth of $log(n/c)$ where n is the total number of nodes and c is a configurable parameter denoting the size of a *committee* (§ 6.3.2). Each cycle trie has n leaves, one for each of the nodes in the network; we shall see how nodes are linked to these leaves in § 6.4.1. Furthermore, we will discuss how each of these nodes use this mechanism to prove commitments in § 6.4.9. The goal of Cambium is to securely and efficiently build up these cycle tries—and consequently the banyan trie—for a large number of nodes. This is done by achieving local consensus first in committees, and then merging the committees' tries in a binary recursive structure as shown in Figure 6.3. At the end of this process, most nodes should agree on the value of the new cycle root.

6.3.2 Protocol flow

Here we present a more detailed summary of the protocol flow within Cambium. This overview is intended to provide the intuition behind the design and enable to reader to make sense of the details that follow in § 6.4. We follow Alice's node as she builds up a cycle in Cambium.

To generate the genesis cycle root, CR_0, all the nodes must share a common piece of information, in this case the list of all participating node addresses. We consider

identity management to be out of scope here. As already discussed in Section 5.2.2, we can bootstrap identity from whatever mechanisms are appropriate to the application (national ID, a corporate asset register, Trusted Execution Environments, proof of work, proof of stake, etc.) [43, 9, 63].

To obtain CR_0, Alice sorts the list of addresses of all the nodes in the system and then builds a binary Merkle tree with these as inputs; the Merkle root of this tree is CR_0. The rank of Alice's address in the ordered list is her *raw index*. The nodes store address-to-index and index-to-address mappings for efficient lookups.

Note that C_t refers to the cycle trie built in cycle t, while CR_t refers to the cycle root for that cycle trie.

We now join Alice at the beginning of cycle $t \geq 1$, where her first task is to find a fresh committee. Committees consist of the neighbouring c nodes in a permuted index list (c is a configurable parameter for a network, usually a small number that is a power of 2 e.g. 16). To calculate the permuted index, Alice uses a Pseudo-Random Permutation (PRP) algorithm such as FastPRP [198] with the previous cycle root as the key and her raw index as the input. Alice then calculates her committee for cycle t using PRP on the permuted indices for the neighbouring $c-1$ nodes. Alice now knows the raw indices of her committee members, and looks up their addresses in the genesis Merkle tree.

Next, Alice attempts to achieve committee-level consensus. She sends a hash of her input to her $c-1$ committee members, and receives their hashes. The committee then executes a Byzantine fault tolerant (BFT) consensus algorithm (e.g. PBFT [59]) and repeats it with unhashed inputs to arrive at the committee signed packet (CSP) – the concatenation of every node's input and signatures (§ 6.4.4).

Alice can now move beyond her committee and begin the merge process with neighbouring committees. The first step is to find *allies*: a group of nodes with knowledge of the neighbouring branch of the cycle trie under construction. For the first merge, if Alice is in the mth committee in the permuted index list then her allies are the c addresses in the $m-1$ or $m+1$ committee, depending on the parity of m (§ 6.4.5).

Alice finds the addresses of her allies in the same manner as she did for her committee (i.e., using PRP). Next, she sends her CSP to those allies, and receives theirs. Given a well-formed allied CSP, Alice can form a merge packet, called MP_1. This packet is the hash of both CSPs concatenated together with some data and signatures (§ 6.4.6).

Ally selection is more complex after the first merge level (since there are more than c nodes in the subtree at higher levels), but amounts to much the same thing: from all the nodes that have knowledge of the neighbouring branch in the cycle trie, Alice selects c using the deterministic algorithm in § 6.4.5. The merge packet is formed for this level, and is used as input for the next merge level.

Alice continues this process through $\log(n/c)$ merge levels, where n is the total number of nodes. At the top level, once $MP_{\log(n/c)}$ is formed, she needs to confirm that the total hole count and dissent count carried within it do not exceed the *hole threshold* and

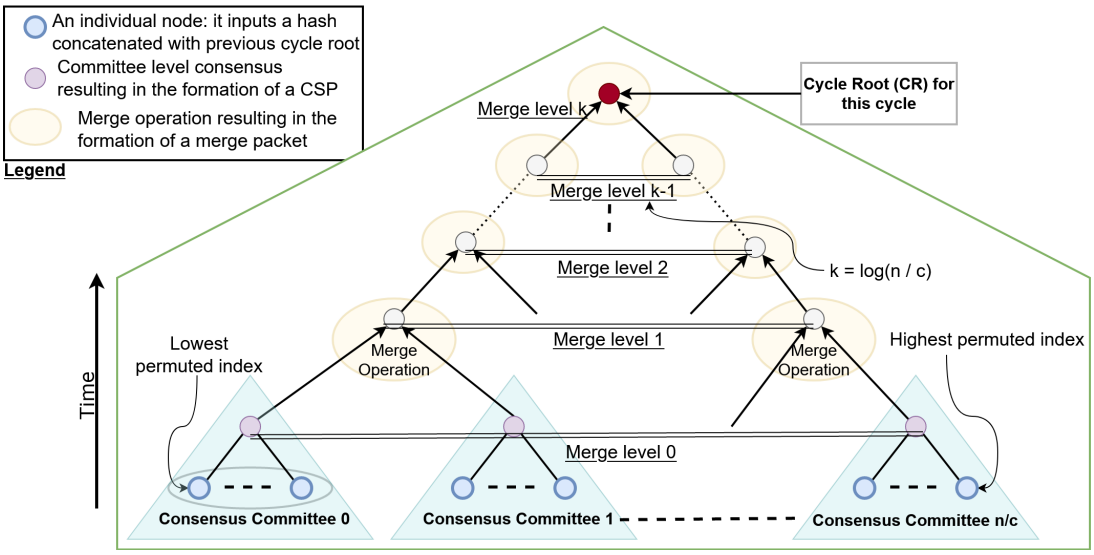


Figure 6.3: Illustration of the build process for a single cycle trie. Subsections in § 6.4 are ordered according to this build process: § 6.4.1 and 6.4.2 cover the set-up phase, § 6.4.3 and 6.4.4 cover committee consensus (blue triangle), § 6.4.5 and 6.4.6 detail the merge process (yellow ovals), and § 6.4.6.1 provides conditions for successfully forming the CR (red circle).

the *dissent threshold* respectively (§ 6.4.6.1).

Holes are a measure of how many committees were unsuccessful in participating in the trie building process. A hole could happen because too many nodes were offline for the committee to form a valid CSP, or because a partition kept the committee from communicating with other nodes during the merge process. If the number of holes exceeds the hole threshold then we failed to build this cycle trie, and we have to have another go.

Dissent is a measure of disagreement over the value of the previous cycle root. Nodes may disagree on the outcome because partitions cause holes for some but not others, or because of Byzantine behaviour such as equivocating merge packets. Committees can include dissenting nodes, but their CSPs denote the existence of dissent by setting a flag in the CSP data structure. These dissent flags are added as we build up the trie. Exceeding the dissent threshold in cycle t causes a rollback, with the result that we have to rebuild cycle trie $t-1$. This should be a rare occurrence under normal circumstances (§ 6.4.7).

If the hole and dissent thresholds are not exceeded then we say that the cycle trie has been successfully built, and $MP_{\log(n/c)}$ is the *cycle root* for cycle trie t. If the subsequent two cycles, C_{t+1} and C_{t+2}, are also successfully built, then we say that cycle trie C_t is *planted*. Once a cycle trie is planted no rollback is possible; it is "set in stone". Figure 6.3 illustrates the protocol flow for a single cycle as discussed.

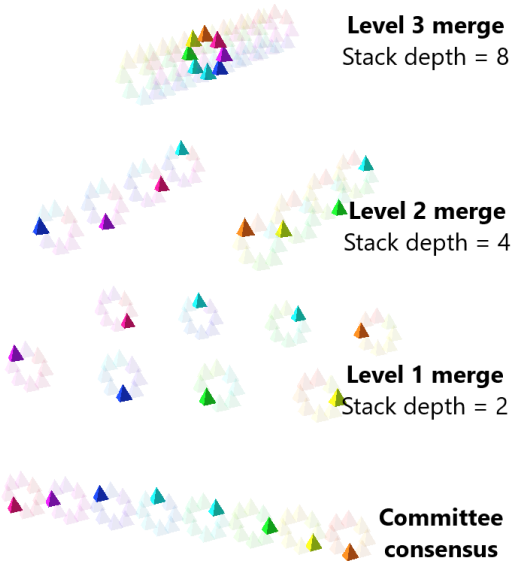


Figure 6.4: Illustration of a cycle trie build with $n = 64$ and committee size, $c = 8$. Each triangle represents a node. We have highlighted some nodes and uniquely coloured all so you can follow the path each node takes. At the bottom, you see 8 committees (n/c) and at each consecutive level you see the stack depth doubling. There are a total of 3 merge operations $(\log_2(n/c))$. This figure is a screenshot from our open-sourced cycle trie visualisation tool [10].

6.3.3 Intracycle communication

Having walked through the build process from a single node's perspective, it is worth looking at what this accomplishes from a global perspective as well.

A cycle trie, being a binary Merkle trie, is a key/value data structure. Keys in the cycle tries are each node's permuted index in that cycle. Values are hashes: each node is able to enter a single hash as the value of its key in that cycle trie. In the car maintenance example, keys are permuted indices of the users, whether cars or garages, and values are hashes of their commitments about car maintenance transactions. Cycle tries therefore have a fixed height for a given number of participants n, equal to $\log(n/c)$ where c is the committee size. Let us assume for the time being that there are c million nodes and therefore 20 levels in the cycle trie.

At the bottom of the cycle trie is the level of the committees, which can be thought of as the first *groups*. A group has the following characteristics: there are c nodes in a group; its nodes are building the same branch of the cycle trie; their permuted indices all fall in the same bucket; they all communicate their values with each other; and they all communicate their value with their *allied group*.

The allied group of a committee, as mentioned above, is its neighbouring committee in the cycle trie. By sharing their information, nodes from both the left and right branches can determine the value of their common ancestor branch. Concretely, the value of depth 19 from the top (i.e. merge level 1) is the hash of its left and right child branches, which in this case are the committees (depth 20). See § 6.4.6 for a more

concrete view of this.

Each node at level 1 now knows the value of that branch in the cycle trie. They would like to know the value of level 2. To do this, they must communicate with nodes building their neighbouring branch in the cycle trie. This communication must be limited (i.e., a constant overhead at each level), to avoid full broadcast and maintain logarithmic scaling.

We therefore partition the nodes at this level 1 branch into two groups, and say the *group stack* here has a depth of 2. Each group has a neighbouring allied group it merges with. In particular, the first group in the left hand branch at level 1, L_0, merges with the first group of the right hand branch R_0. Similar, the second group L_1 merges with R_1. Once this is complete all these nodes have a value for level 2, which now has a group stack depth of 4. Each of those groups then merges across again to derive a value for level 3.

At each subsequent level of the cycle trie, the number of branches halves, thus, the group stack depth at each level goes up by a factor of two. Each new group mixes nodes from across all the committees under that branch. This mixing algorithm, described in § 6.4.5, is chosen to minimise the pairwise intersection of the group's shared history; thus maximising information spread. Interestingly, given a fixed total number of nodes, this mixing is also fixed (see Figure 6.5). Figure 6.4 shows the shape of the group stack in a cycle trie for a small network.

6.3.4 Intercycle communication

It is important to shuffle the system to prevent persistent local errors or attacks, such as a node being stuck in a committee or branch of the trie with many offline or adversarial nodes. One of our design decisions was to use a fixed communication layout while building each cycle trie, but shuffle the index list between cycles using the previous cycle root as the seed (§ 6.4.1).

This shuffling also helps solve a second problem: at the top of the cycle trie, how does Alice know how widely shared her value for that cycle trie is? She may be successful, in the sense that she arrived at a cycle root that meets the hole and dissent requirements, but it might also happen that a partition created holes that affected Alice but not many other nodes.

In such cases, the next cycle itself provides a resolution process. Permuting the indices results in multiple new committee invitations: the ones Alice sends out to "her" committee members, based on the cycle root she arrived at; and the ones sent to Alice from other nodes based on the cycle roots they arrived at.

If there was a large partition event there may be many different roots for cycle t, and Alice may be invited to many different committees. This will result in high dissent counts for every cycle trie that is built in cycle $t + 1$, and cycle t will have to be redone. However, one of those committees may be based on a very popular cycle root,

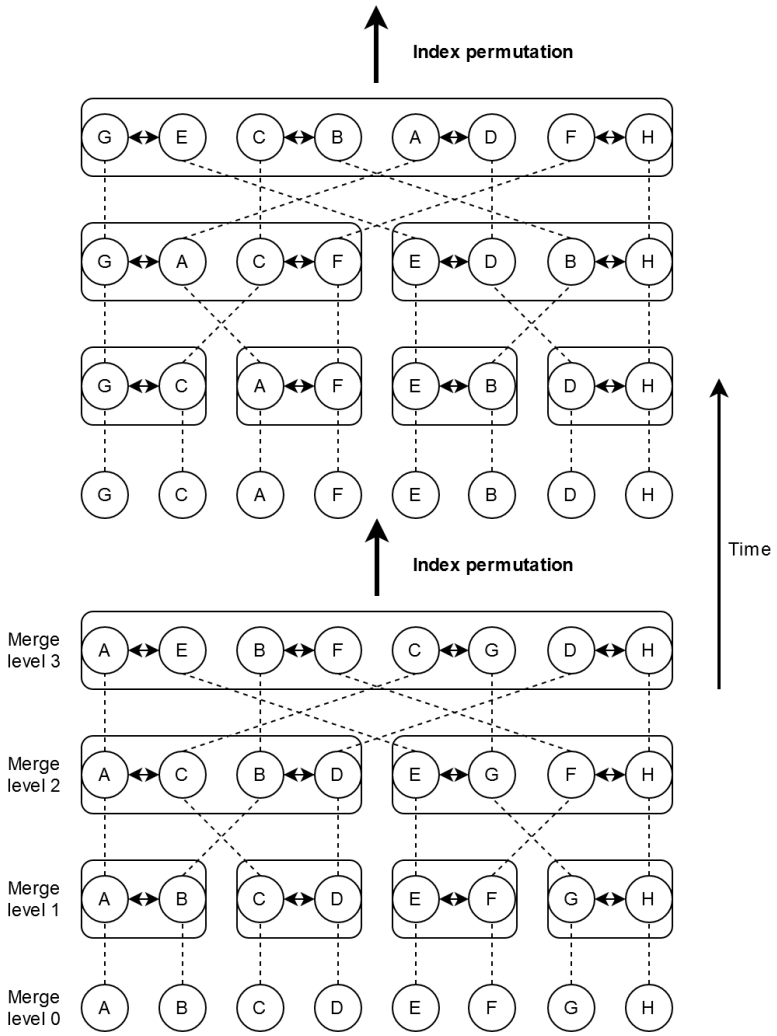


Figure 6.5: Illustration of the intercycle communication pathways. Notice how the position of nodes in the communication graph gets permuted but the pathways in the graph itself look identical going from cycle to cycle.

which builds a successful cycle in cycle $t + 1$. Alice has very limited (i.e. logarithmic) information about the global system, so there is no way to tell which committee she should be in until their cycle tries are built.

6.3.5 Preliminary observations

We will discuss the security and performance characteristics of this protocol in detail later, but a few things are worth observing right away. First, notice how we arrive at a globally shared piece of information, the cycle root, without global broadcast communication. In fact, each node only communicates with $O(\log(n))$ nodes in each cycle. This has obvious scalability benefits: *consensus with 10 million nodes only requires communication with approximately 300 nodes*, thus making it feasible over such large networks for the first time.

Second, the construction of the banyan tries gives us data privacy by default. Each node only shares a hash of its input, and that too only with its committee and first merge allies. Thus, even for networks with hundreds of millions of nodes, only tens of nodes hear each node's input.

These two salient features benefit applications such as tracking maintenance and provenance in companies, where permissioned blockchains are currently being used. In permissioned settings, there is often an expectation of data privacy as well as a need for widespread collaboration. This collaboration stands to become far more decentralised and efficient (as discussed in § 6.1) by moving to Cambium, while an adversary who compromises a small number of corporate assets cannot use the inventory and maintenance system to get an overall view of a company's operations.

More generally, applications that lend themselves to shared, diffuse power structures – a collective insurance pool, or a political movement, for instance – can grow and scale without linear per-action cost increase and without building up a large pool of sensitive information that needs to be protected, thereby increasing cost and risk. Cambium could serve as the substrate for a new class of applications, allowing previously unexplorable applications to be built.

6.3.6 Cambium and consensus

The traditional definition of a consensus algorithm usually mandates three properties: *termination*, *agreement* and *validity* [72]. Termination refers to the property whereby eventually all non-faulty nodes decide on some value. Agreement refers to the property whereby all non-faulty nodes agree on the same value. Validity states that if all non-faulty nodes proposed a value v then the accepted value must be v. Cambium aims to provide termination and agreement but it explicitly is not "valid". The value that is agreed upon (i.e., the merkle root) is not proposed by any of the nodes but is rather a culmination of all their inputs. A similar concern applies to some blockchain "consensus" mechanisms [99] where the produced block isn't typically proposed by any node but rather contains the set of transactions proposed by all nodes. Cambium's lack of validity is stronger than those seen in blockchain systems and we acknowledge this difference between classical consensus and Cambium, but refer to it as a consensus algorithm in the looser sense of the word as has been used by the blockchain community.

6.4 Protocol module definitions

We now define and examine the steps performed in a Cambium system more formally. They are ordered chronologically as we build up a cycle, as shown in Figure 6.3. Algorithm 6 shows the sequence in which the modules are executed within a round. In the module descriptions that follow, we assume a node with address A has just completed cycle t with cycle root CR_t.

Algorithm 6 Overall flow for Cambium round. c stands for committee size, n for number of nodes, HC for hole count, DC for dissent count, CSP for committee signed packet and CR for cycle root.

procedure DoCambiumRound(*prevCR, myRawIndex, IndexList, c, n*)
  $cycleIndex \leftarrow PRP(prevCR, myRawIndex)$
  $myCommittee\{\} \leftarrow selectPeers(cycleIndex, 0)$
  $myPath \leftarrow \emptyset$
  $CSP, myPath \leftarrow doCommitteeConsensus(myCommittee)$
  $i \leftarrow 1$
  $prevMP \leftarrow CSP$
  while $i \leq log(n/c)$ **do**
    $myPeers_i\{\} \leftarrow selectPeers(cycleIndex, i)$
    $myAllies_i\{\} \leftarrow selectPeers(getAlly(cycleIndex, i), i)$
    $toSend \leftarrow ResolvePeerMessages(myPeers_i, prevMP)$
    $sendToAllies(toSend)$
    $AP_i \leftarrow ResolveAllyMessages(myAllies_i)$
    if $AP_i = hole$ **then**
      $HC \leftarrow toSend.holecount + 2^i$
    else
      $HC \leftarrow toSend.holecount + AP_i.holecount$
    $DC \leftarrow toSend.dissentcount + AP_i.dissentcount$
    $temp \leftarrow (AP_ihash||prevMPhash||prevCR||HC||DC||i)$
    $MP_i \leftarrow temp||hash(temp)$
    $prevMP \leftarrow MP_i$
    $myPath \leftarrow myPath||temp$
  if $countHoles(myPath) > holeThreshold$ **then**
    return false
  if $countDissent(myPath) > dissentThreshold$ **then**
    return false
  $CR \leftarrow MPhash_{log(n/c)}$
  return $CR, myPath$

6.4.1 Index permutation

At the start of each cycle, every node calculates its cycle index. This is done by using the permute function of a small domain Pseudo-Random Permutation (PRP) on the raw index with the previous cycle root as the key. The requirements for the PRP function are that it should map inputs from a domain of $\{0, 1, ..., n-1\}$ to outputs in the same domain $\{0, 1, ..., n-1\}$, give a unique bijection for each key, support both permute and unpermute operations efficiently and, be pseudo-random. FastPRP [198], as an example, appears to fulfil these requirements.

6.4.2 Committee selection

Cycle root CR_t is called the *natural* root for A. Other nodes may have arrived at a different natural cycle root in the case of network partitions or adversarial behaviour (§ 6.5). Each node performs its index permutation using its natural root, as described above.

The cycle index space is split into (n/c) contiguous committee buckets giving a committee size of c. The committee size c is thus a tunable parameter, with higher values of c yielding more committee work (larger committees) but fewer merge operations

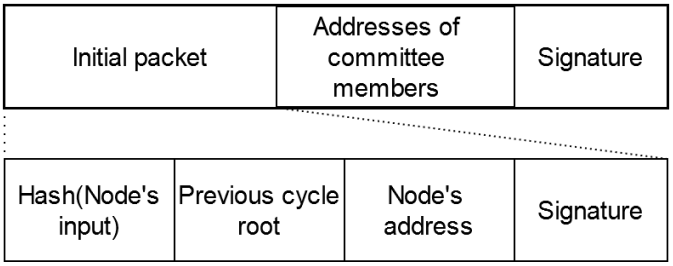


Figure 6.6: Initial packet and envelope structure

(fewer committees). In our descriptions here, we will often assume the value of c to be 16. This number is derived from the fact that it is desirable for c to be a power of 2 for a simple implementation of the SelectPeers module (§ 6.4.5) and that PBFT, which is used for arriving at committee-level consensus (§ 6.4.4), does not scale well beyond a dozen or so nodes.

Each node finds its natural committee for cycle $t + 1$, which are the $c - 1$ other nodes that fall in the same c-sized bucket. To put it more concretely, for a node with cycle index a', its committee are the nodes with cycle indices i such that $(a'/c) \leq i \leq (a'/(2c - 1))$, excluding a' itself. Once A knows the cycle indices of its committee members, it calculates their raw indices by performing the PRP unpermute operation. Then, it looks up their addresses in CR_0.

6.4.3 Initial committee packets

Node A forms its *initial packet*, puts it into an envelope containing the addresses of the committee members as shown in Figure 6.6 and sends it to its committee members. Node A should receive initial packets from those nodes as well. If A only receives packets that have the same natural cycle root as A, then we say that A has received no *dissenting messages*. In this case, A can proceed to committee consensus (§ 6.4.4).

If A did receive a dissenting message, it checks its validity as described in algorithm 7. A then sends its initial packet to all the committee members listed in the envelope of the dissenting message, and proceeds to join their committees as a dissenting presence, as described below.

6.4.4 Committee consensus

By this point A has received some natural initial packets, and maybe some dissenting messages. The committee's goal is to arrive at a committee signed packet (CSP), and in the full version of Cambium it involves two rounds of committee consensus using PBFT.

In the first round, nodes attempt to arrive at consensus over the set of initial packets. If $\geq 2c/3$ nodes arrive at the same (natural) value then we say that we have successfully completed the first round of consensus. This threshold of $2c/3$ is inherited from the consensus algorithm used, in this case, PBFT [59]. For $c = 16$, this would mean that

Algorithm 7 Check Validity of Dissenting Initial Committee Packet

```
1: procedure ValidityCheck(packet, myRawIndex, myAddr)
2:     CR_t ← packet.cycleroot
3:     if myAddr ∉ packet.addresses then
4:         return false
5:     senderRawIndex ← lookup(packet.sender, CR_0)
6:     senderIndex ← permute(senderRawIndex, CR_t)
7:     for all a ∈ packet.addresses do
8:         rawIndex ← lookup(a, CR_0)
9:         tempIndex ← permute(rawIndex, CR_t)
10:        if tempIndex ≥ (senderIndex/c + c) then
11:            return false
12:        if tempIndex < (senderIndex/c) then
13:            return false
14:    return true
```

≥ 11 nodes would need to be in agreement.

Next, all nodes commit a new packet, this time with their actual inputs instead of the hash of inputs. This packet is the unrolled packet. The committee now attempts to arrive at consensus over the committee's dissent score and the set of unrolled packets. Again, if $\geq 2c/3$ nodes arrive at the same value then the agreed packet is termed the CSP for this committee. The committee members can now move on to the first merge level. This two step consensus is done to prevent the (unlikely) attack where the adversary does a grinding attack simultaneously on all committees to bias the cycle root; in a lower-threat environment such as within a company, we can happily dispense with the first of the two rounds.

Given successful committee consensus, we get a CSP with the following fields: 1. Each committee member's input transaction packet as a hash TP_i as well as the transaction packet itself which contains the member's input, natural cycle root and address; 2. Dissenting messages, and any metadata values; and, 3. Committee members' signatures over the data structure.

Let us now consider failures in the two-step process. The first is failure to arrive at consensus in the first round. This indicates that more than a third of the nodes are either offline or that they believe in a different cycle root (since we do not use dissenting messages for consensus). At this point, this committee is considered a *hole* and they go to the next merge level with a hole (a set string signifying a hole) as the CSP. If the committee failed at second round consensus, then it indicates that a formerly online node is now unreachable. Here too, nodes mark their CSP as a hole and go to the next merge level.

It is possible, due to network partitions, for some nodes in a committee to believe that their committee is a hole while others believe it is not. However, it is not possible for nodes to believe in two different non-hole CSPs for the same committee.

6.4.5 Merge group selection

Now we need the CSP-bearing nodes to merge with their CSP-bearing neighbouring allies, following the pattern up the cycle trie. As we move up the trie, the number of nodes on either side of the merge operation increases exponentially. At the first merge level, there are c nodes on either side, at the second merge level there are $2c$ nodes on either side, third, there's $4c$ and so on. This is illustrated in Figure 6.4.

So, we need a way to segment the *peers* (nodes with whom we already share a packet) and the *allies* (nodes with whom we want to merge), in order to maintain our $\log(n)$ communication threshold. Otherwise, at the top, all nodes would need to talk to each other and we'd be back in the ledger-based communication model.

We have the following design principles for this operation: a small size for groups at each level of the cycle trie, and maximum intermingling. Ideally your peers would be different each time, and their previous peers would have been different, and so on. This increases the opportunity for sharing missing information (e.g. data for filling holes), as well as for detecting equivocation.

We achieve these properties by having nodes first discover their *peer group* at the merge level: the set of nodes they share *allies* with. Allies are the c number of nodes that a node is supposed to communicate with at a particular merge level. At the first merge level, the choice is obvious: your peer group is your committee and your ally group is the neighbouring committee.

Things get trickier as we go up. At every merge level, we visualise groups lying on group stacks as shown in Figure 6.4. In this way, we can assign each group a *group stack index* on both sides of the merge. Also, at every successive merge level the shared *prefix* of the respective cycle indices among the nodes merging goes down by one bit; this is clear to see when one looks at the top of the trie: at the final merge, there is no common prefix among nodes but at one level down, all nodes on the left have 0 as their common prefix and those on the right have 1. Thus, the common prefix keeps increasing as we go down the trie. We use this prefix and the group stack index to choose allies and peers.

For a node with group stack index i at level l of the cycle trie we can determine its group members (peers) by using Algorithm 8. The ally stack index is determined similarly. Working code for this is available in our repository [10].

6.4.6 Merge process

As shown in Algorithm 6, this and the previous module are executed repeatedly by nodes as they build the cycle trie and constitute the bulk of the time spent attaining consensus in Cambium.

The goal is for groups to come to agreement on the branch value they share. Specifically, two neighbouring ally groups at a merge branch need to create a new merge packet (MP). The MP is the concatenation of the left-hand and right-hand branches' merge packets, along with the cycle root CR, the hole count HC, the dissent count DC,

Algorithm 8 Merge Group Selection Algorithm. ci is the cycle index, l is the level of the cycle trie and c is the committee size.

```
1: procedure SelectPeers(ci, l)
2:     peers{} ← ∅
3:     logc ← log2(c)
4:     prefix ← ShiftLeft(ShiftRight(ci, l + logc), l + logc)
5:     d ← (ci ∧ 2^(l−1)) − ShiftLeft(ci − p − (ci mod 2^l), l)
6:     if d ≥ 0 then
7:         i ← d
8:     else
9:         i ← d + 2^l
10:    for all j ∈ [0...c] do
11:        batch ← (i + j) AND (2^l − 1)
12:        group ← batch OR (j × 2^l)
13:        peers ← {peers ∪ (prefix + group)}
14:    return peers
```

the merge level l and the hash of these components. So, if we are at merge level l, let $temp = MPhash_{l-1}^{left} \| MPhash_{l-1}^{right} \| CR \| HC \| DC \| l$, then $MP_l = temp \| Hash(temp)$ where $MPhash_y^x$ refers to the hash component of MP_y^x that is, the last 32 bytes of MP_y^x. The top level $MPhash_{log(n/c)}$ is the new cycle root.

At every merge level l, each node comes in with its previous merge packet (MP_{l-1}), knowledge of its peer group (X_l) and of its ally group (A_l). At the end of the process, we want all $2c$ nodes (c each in peer and ally group) to agree upon the merge packet of this level, MP_l. Each node signs their merge packet before sending it. For the sake of simplicity, we refer to the MP brought in by an ally a as AP_l^a for ally packet; also without loss of generality, we assume that the node we're following up the trie (node A) is on the left hand side of the merge at this level.

The merge process starts with A creating two queues ALLY-QUEUE and PEER-QUEUE of length c each for receiving incoming messages. It then sends its MP_{l-1}, to all peers in X_l. Simultaneously it should receive messages from its peers, which it stores in PEER-QUEUE. Once it receives all $c-1$ messages, or the timeout expires, it executes a RESOLVE-PEER-MESSAGES algorithm.

Next, A sends its MP_{l-1} to all the nodes in its A_l and receives their APs into ALLY-QUEUE. Once it receives all c messages, or the timeout expires, it executes a RESOLVE-ALLY-MESSAGES algorithm which result in the selection of a value for the accepted AP. Then, A concatenates this AP with its MP_{l-1} and hashes it to arrive at MP_l. The two resolution algorithms, illustrated in Figure 6.7, ensure homogeneity of operations across all $2c$ nodes. Pseudocode for the resolution algorithms is in Appendix A.4. We discuss calculation of the other data that go into the MP next.

6.4.6.1 Hole and dissent counts

At each merge level, the right-hand side and left-hand side nodes come with their hole and dissent counts. Calculating the dissent count for MP_l is then a simple matter of

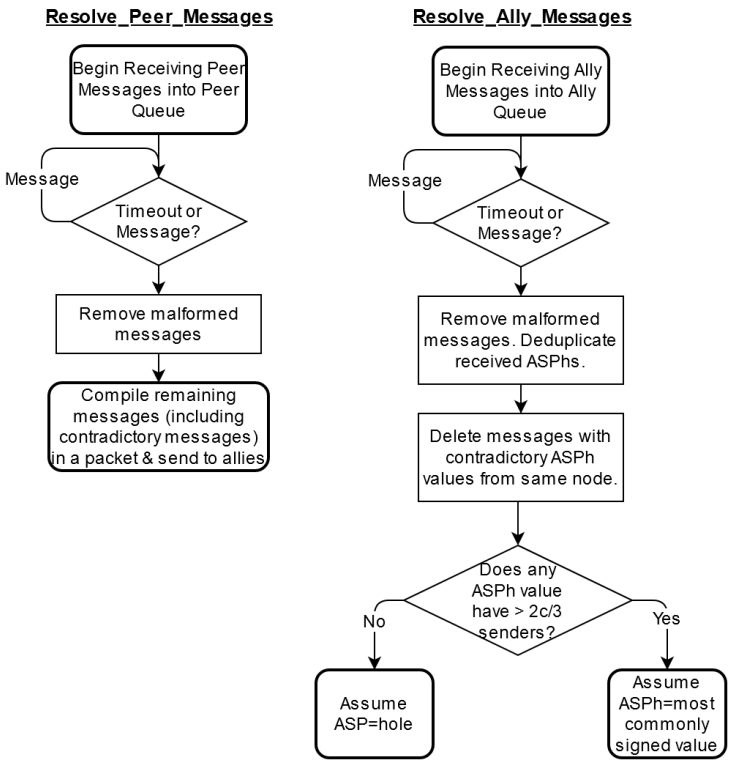


Figure 6.7: Flow for the message resolution algorithms.

adding the two values. Holes are slightly more complicated, however. If x found the AP to be a hole then the hole count for MP_l would be the sum of the previous hole count and 2^{l-1}. 2^{l-1} represents the number of committees that now must be assumed to be offline because we couldn't get appropriate information for their subtrie. Thus, lapses in communication higher up the trie mean larger hole counts. It is possible to do a *hole filling* algorithm to increase the odds of arriving at consensus, which we discuss in § 6.9. Regardless, large hole counts would result in having to rebuild the cycle, as configured by the hole threshold.

Note that if a node encounters ally or peer messages based on an unknown cycle root, it will not only store those messages but fully join the merge process. It not only carries the values it is passed, but records all messages as well (in case this new cycle trie turns out to be successful instead of its natural cycle).

6.4.6.2 Resolving disagreements

Earlier we mentioned the RESOLVE-ALLY-MESSAGES and RESOLVE-PEER-MESSAGES algorithms used to resolve disagreements. Figure 6.7 shows the flow of these algorithms while Appendix A.4 contains the detailed steps.

We note that this version of these algorithms works well for reliable networks. However, it is fragile: once a node chooses a value for a level, it never goes back to re-evaluate it. This is a problem for holes. If, due to a network failure, a node puts a hole at level

l where other nodes have a value, there's no way for it to correct that mistake later. To solve this issue, we expect that some deployments will use *windowing* in conjunction with the merge process. Windowing allows that node to use information from its peers in level $l + 1$ to fill holes in l. We discuss windowing in § 6.9.

6.4.7 Rollbacks

Rollbacks are caused by missing the required thresholds for dissent or holes, which are global parameters. If both are missed the dissent rollback takes precedence. The precise values for these thresholds will need to be tuned according to networking conditions (the hole threshold can be lower if networking is robust) and adversarial assumptions (a higher risk may entail a lower dissent threshold).

If there are too many holes in the cycle trie, this means that a significant number of nodes did not have the opportunity to participate in building it. They might not have direct knowledge of their path within it, requiring a cold-boot-like process to acquire it. They may have also formed their own trie, for example due to a network partition, in which case neither cycle trie should make progress. It is prudent to try building cycle t again, since the goal is cohesion – even if that comes at the expense of availability during network partitions. If the choice is whether your bank should provide a wrong balance, or to be temporarily unavailable, we opt for unavailability. We expand upon this and other limitations and trade-offs of our approach in § 6.9.

When the dissent scores are too high, that tells us there was widespread disagreement about the previous cycle. If we had just built cycle t, we must now return to our mutually shared cycle root $t - 2$ and rebuild cycle $t - 1$ from there. Dissent can arise from node equivocation, but it can also arise naturally from network partitions, which cause some nodes to believe a hole exists while others have a value. In either case, the result is that different roots were arrived at for cycle $t - 1$ by a significant number of nodes, and therefore cycle $t - 1$ must be rebuilt.

The other cause of rollback is discovering that two conflicting pairs of successful cycles have occurred. If Alice discovers that Bob has different successful cycle tries for t and $t + 1$, then equivocation has occurred and cycle trie t should be rebuilt.

While rollbacks of up to two cycles is possible, it is not possible to rollback further. Once a cycle is planted it can not be changed: having successfully built cycle t, now cycle $t - 2$ is set in stone. If it is discovered that there are contradictory planted $t - 2$ cycle roots present in the network, then the security assumptions of the network have been violated. Our security analysis (§ 6.5) shows that this ought to be extremely unlikely even in the presence of powerful adversaries. There are a variety of responses imaginable here, such as detecting and removing equivocating nodes, or rolling back further and rebuilding, but for many use cases halting the system while retaining the integrity of the earlier attestations is the natural solution, to be followed by a manual restart or other operator intervention. While some may want a static system in which maintenance

interventions become impossible, reality is dynamic, and the design of robust automatic restart mechanisms depends on a good understanding of the attacks and other failures from which recovery may be needed.

6.4.8 Cold boots

If you have been offline for a while and want to join the network again then you need to do a cold boot. To re-join the network, Bob randomly picks m nodes and sends them a request message. In response, these m nodes send back the entire sequence of cycle roots for the banyan trie up to the current cycle t.

If all the responses agree on their cycle roots up to $t-2$ then Bob can ask any of the m nodes to send its Merkle proof for cycle t. Bob can then use this value as the natural root and start participating in Cambium. The only way for the nodes not to agree on values up to $t-2$ in an uncompromised network is if some nodes send Bob a false value. In case of differing values, Bob must ask for build paths (Merkle proofs plus all the packets that went into making the proof) for cycle $t-2$ from all m nodes. Equivocating nodes will not be able to provide them, leaving Bob to communicate with honest nodes.

6.4.9 Proving commitments

Suppose Bob wants to prove to Alice that he committed a value x in cycle t. We assume that both Bob and Alice agree upon the value of the cycle root for cycle $t+2$, which is assured in an uncompromised network.

If they do, then all Bob has to do to convince Alice is send his build path for cycle t. This includes his input into the CSP and all the MPs going up. Alice then verifies the build process by first checking if Bob was in the right committee given C_{t-1} and then calculating the successive MPs. If this arrives at the agreed root value for cycle t, then Alice has proof that Bob had committed x in cycle t.

6.4.10 Double-spend prevention

A mechanism for double-spend prevention follows naturally from the above. Let us suppose that Alice wants to prove to Bob that an asset in her possession has not been sent to anyone else, say in the last 10 cycles (from cycle $t-10$ to t). Let us assume an agreed semantic between Alice and Bob for what it means to "send an asset". This could, for example, be a UTXO-like model which requires Alice to specify a recipient address and sign over the packet. The recipient can then use this signature, along with Alice's commit in that cycle, as proof of possession of the asset.

First, Alice and Bob need to agree upon the sequence of cycle roots up to cycle $t+2$. Then, Alice needs to send her build proofs for cycles $t-10$ to t. Bob does the verification process laid out in § 6.4.9, repeating it ten times. Suppose for example that a Cambium

system were being used to support a distributed system for peer-to-peer car rental, so that the asset being verified is a car; this verification would need to be done from the point where that car was purchased, or registered in the system, to the latest planted cycle.

Optimisations are possible by encoding additional semantic information into the input values at the committee level that are understood by all nodes. One such optimisation that has been explored in prior work [210] is encoding information about a node's liveness into a commitment. So, for example, a node can signal that no inputs in its address are allowed for the next 100 cycles while it is offline. Thus, for example, if Alice rents her car to Bob for 100 cycles in return for a fixed payment, she could forego the ability to rent it to anyone else and go offline.

Such optimisations are important because double-spend prevention in our prototype version of Cambium incurs a linearly increasing cost: one Merkle proof per cycle needs to be sent to the recipient who then needs to validate it. Another thing to consider here is that if, in the above example, Alice was offline for cycle $t-5$, she would need to get the CSP from her committee (or first level allies) in that cycle. If her committee successfully built a non-hole packet in cycle $t-5$ but subsequently all committee members and first level allies went permanently offline then it is impossible for Alice to prove that she didn't do a double spend in $t-5$. This can be mitigated somewhat by windowing but not completely (see § 6.9). Do note that if the CSP or MP at any merge level was a hole then Alice can simply get any of the allies involved in committing that hole to provide her with sufficient proof.

6.5 Security analysis

Here we present a sketch of Cambium's security under two different adversarial models: a Byzantine adversary with control over some minority of nodes, and the Dolev-Yao model. We note that this is not an extensive security analysis and that more research is needed to prove Cambium's security against a wider range of attacks.

In this section, we will focus on a particular kind of attack where the goal of the adversary is to *fork* the network: to have two (or more) sets of honest nodes continue to make progress with divergent sequences of cycle roots. We do not analyse other kinds of attacks such as denial of service or liveness violations here.

In Cambium, a cycle trie is planted when it and the subsequent two cycle tries are successful. Therefore if the adversary can cause three successful cycles in a row on two or more parallel forks then security properties have been violated.

After analysing the security of Cambium against these two models of forking attacks, we briefly discuss its characteristics with regard to censorship resilience and privacy.

6.5.1 Byzantine adversary

The Byzantine adversary model traditionally grants total control over some portion of the network's nodes to the adversary. The adversary here is limited in a variety of ways: they have no ability to impede the operation of correct nodes, either directly or indirectly through actions on the network; they have no ability to change which nodes they control; and they have bounded computational power.

In order to fork the network, the Byzantine adversary has two strategies: top-down or bottom-up. These strategies differ only in the first cycle of divergence, and are identical from the second cycle onward. In pursuing these strategies, the adversary has two kinds of costs: computational costs and sampling costs.

We assume a bounded delay in communication between any two honest nodes. Furthermore, we assume that the adversarial nodes do not change, i.e. the same set of byzantine nodes remain adversarial throughout the analysis.

We will first look at the bottom-up attack, where the adversary picks a victim node and compromises its entire merge path up the cycle trie. We need to calculate how many nodes the adversary needs to fork this one node, and then calculate the probability that those adversarial nodes are in the right positions.

Suppose that the adversary wishes to convince Alice of some forked cycle. The adversary necessarily needs to have $2c/3$ nodes in Alice's committee. If this wasn't the case, then the adversary would be unable to meet the necessary consensus quorums on two different CSPs.

Similarly, at merge level 1, the adversary needs to have control over at least $2c/3$ allies in order to meet the threshold for Alice to accept the forked ally message. This continues at every merge level that follows, with the adversary needing at least $2c/3$ compromised allies for every merge group that Alice encounters on the way up.

Since there are $\log(n/c)$ merge levels, and $2c/3$ required allies at each level, the adversary requires at least $2c/3 \times \log(n/c)$ adversarial nodes to compromise Alice. To quantify the success probability of such a bottom-up attack, we calculate the probability of the adversary being able to gain at least $2c/3$ nodes at each merge level.

To make our analysis more tractable we make a couple of adversary-favouring assumptions. First, we assume that Alice will ignore ally messages beyond the $2c/3$ messages sent by the adversary. Second, we assume that even if Alice receives valid messages belonging to a conflicting cycle root, she will continue building the path she is on and not keep track of contradictory paths. In a real deployment, we would expect nodes to keep track of contradictory build paths so that they have the necessary proof paths in case their path fails and so that they may abandon a network should a fork deeper than two cycles is discovered. Similarly, we assume Alice will ignore contradictory messages sent by her peers.

Compromising one group of size c at a given level requires having at least $2c/3$ adversarial nodes, which must fit into c spots. For an adversary with p fraction of nodes,

and assuming that $n \times p \gg c$ (so each sampling is independent), this probability is equal to:

$$\left(\sum_{i=0}^{c/3} p^{c-i} \times (1-p)^i \times \binom{c}{i} \right)^{log_2(n/c)} \tag{6.1}$$

If instead of a targeted attack, the adversary wanted to compromise *any* node then we need to multiply this probability by $(n - np)$.To put this into perspective, if we consider a network with $n = 16,000$, $c = 16$ and adversarial share $p = 1/3$, then the probability of compromise for a targeted attack, $P_{compromise} = 1.97 \times 10^{-21}$. This is the initial probability that the attack works. If an adversary waits for a cycle with their nodes in all the right places, with this probability, they can convince Alice that the cycle root is X' while the other honest nodes believe it is X.

However, this is not enough. The adversary must find an X' that repeats its success in the previous cycle. This is the "grinding cost" for the attack, and it is substantial: each hash calculation and subsequent simulation of the consequent cycle by the adversary is equivalent to one sample. Assuming that the adversary canont bias Alice's input into a cycle, for a 50% chance of success, the adversary would need to do $1/P_{compromise}$ number of calculations. If the adversary does not choose to do these calculations then the odds of an adversary creating a fork of depth l is:

$$P_{fork} = (P_{compromise})^l \tag{6.2}$$

Now, let us consider what changes if we roll back our adversary-friendly assumptions. By doing the calculations, the adversary can make Alice only talk to adversary-controlled nodes in the next cycle, however the adversary still cannot control the other honest nodes. With a high probability, at least one honest node building the X-based cycle trie will be in Alice's X committee, and will invite her to join that committee. In a real deployment, Alice will record this as dissent, communicate that she is building an X'-based trie and follow the progress of the X-based tree. If at the end of the cycle, both X-based and X'-based tries are successful then both nodes will assume a failure in consensus and rollback two cycles (see § 6.4.7). Thus, with the parameters stated above, the adversary will not be able to keep Alice fooled for longer than one cycle. The same considerations apply to the top-down attack as well with only a minor difference in the initial probability[4].

6.5.2 Dolev-Yao adversary

In the Dolev-Yao model, while it is possible for the adversary to completely stop the building of cycle tries (a liveness violation), it is not possible for the adversary to fork the network for two or more cycles.

[4]The initial probability for a top down attack is equal to a single committee compromise.

We prove by induction that a Dolev-Yao adversary cannot produce a planted cycle trie with any minority of nodes. First, we make the assertion that the hole count at level 1 for every committee is an upper bound on the actual hole count (how many committees are inactive). This is easy to see: at level 1, the only valid values are 0 or 1. 0 is only recorded if a well-formed CSP with quorum number of signatures are received from the allies. Thus, inactive allies will necessarily be marked as 1 (in the absence of equivocating nodes). Similarly, at merge level 2, we note that the hole value is between 0 and 3. The value chosen by an honest node here will necessarily be an upper bound on the number of genuine holes. We have shown this to be the case at level 1; at level 2 if no message is received from allies then both the allying committees are recorded as a hole. The only way for this not to be the case is to receive a well formed AP. This AP will have a hole count of 0 or 1, and since there are no dishonest nodes in the Dolev-Yao model, this will also be an upper bound on the number of genuine holes.

Inductively, we thus assert that at any given level i, the accepted hole count is an upper bound on the genuine hole count. If the adversary censors communication from allies at level i then the hole count recorded at that level is $2^i - 1$, as the entire subtrie on the allying side underneath level i is considered inactive.

Let us assume that the adversary splits out a minority of size g in cycle t that believes in cycle root X'_t, whereas the majority believes in X_t. Then, in cycle $t+1$, if the adversary censors connections between X'_t and X_t based nodes, the hole count for the minority will be at least equal to $(n-g)$ as shown above which will necessarily exceed the hole threshold set for at least one of the forks. On the other hand, if the adversary does not censor the connections, each X_t committee member will be recorded as a dissent in the X'_t trie. This in turn will exceed the dissent threshold for the minority.

Thus, while it is possible for a Dolev-Yao adversary to fork a small minority for a single cycle, it is impossible for them to sustain that fork for a second cycle.

6.5.3 Censorship resilience and privacy

Cambium's censorship resilience stems from two features. First is the index permutation done every cycle. This ensures that given a cycle root not controlled by the adversary (the difficulty of doing this is discussed above in § 6.5.1), no honest node ends up stuck with the same set of nodes over multiple cycles. So, if a Byzantine adversary were trying to censor a particular node by dropping messages at the committee level, the censorship might succeed for a round or two before the victim would get bucketed with a set of honest nodes.

The second defence we have against censorship is the network's resistance to partitioning attacks. While a Dolev-Yao adversary can stop Cambium from making progress by causing a widespread network partition, as soon as this partition is lifted, the protocol can resume immediately with no need for a "reconciliation phase". This is because Cambium ensures that no two partitions can continue making progress; both partitions

effectively end up doing a busy wait for each other.

The banyan trie is inherently more amenable to privacy preserving use than blockchains due to the limited sharing of input data versus the global broadcast in the case of ledgers. Moreover, we have shown that it is even possible to guarantee double spend prevention while not submitting anything more than an opaque hash to the network. Of course, the flip-side of this approach of submitting opaque hashes is that in order to talk to counter-parties, you now need to have an out-of-band communication channel to send the inputs behind the hash. One can think of this as analogous to the anchoring [132] concept used in some private channel mechanisms in blockchains.

6.6 Performance analysis

Cambium is designed to have logarithmic complexity wherever feasible. The number of rounds of communication, number of messages sent per node, size of the Merkle proof, and computation per proof all scale logarithmically with the number of nodes in the network. The only linearly growing components are the size of the address list, and the list of transactions per node; both have a storage cost.

The main time-consuming activity is the intracycle network communication. To get a better understanding the time taken during this communication we built a prototype and performed some experiments on Amazon Web Services. We chose four geographically distant datacentres to host our nodes: Ohio, Sydney, Seoul and London. The machines used for these tests ranged from EC2's t2.micro to t2.medium. One simplification we made in the interest of ease of deployment is that we did not perform index permutation in this test setup. We do not expect this to have a big impact on performance given the ease of calculating the permutations.

Do note that our networking setup did not incorporate adversarial or offline nodes and was operating on a very robust infrastructure (namely, AWS). The numbers reported here should thus be understood as a best case from a networking standpoint.

To arrive at a number for timeouts, we needed to measure the average latency of messages as a node moved up the cycle trie. We used one node as the logging node while the rest were left unmonitored. Then we did several runs of the cycle build process with varying network sizes. However, we kept a constant committee size of 16. The fixed size means that increasing network size did not noticeably increase merge level times. The average time taken for a single merge level in this configuration was 1.154 seconds with a standard deviation of 0.13 seconds. Of this time, an average of 0.86 seconds was spent on networking with the remaining being local computation (signature verification taking the bulk of that time).

During more than 1000 runs, we never saw the time taken for a merge level exceed 2.5 seconds, with 99 percentile completion in under 1.9 seconds. This gives us confidence in setting a timeout threshold of 2 seconds for merge levels under these robust networking

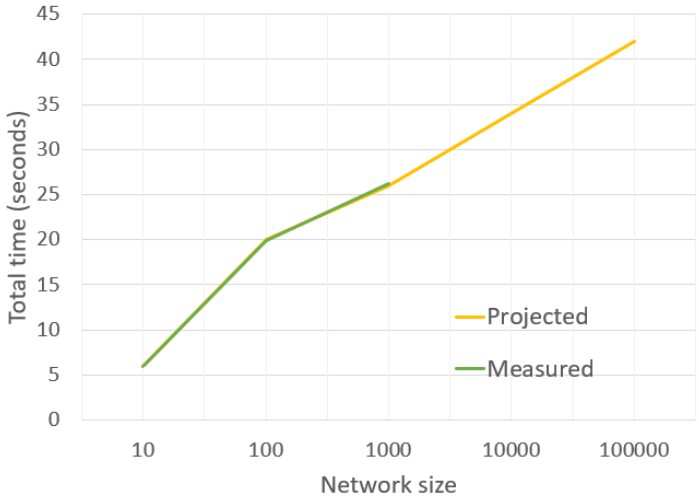


Figure 6.8: Predicted and measured latencies. Note that the measured latencies do not include address permutation. The reason why our projections and measurements correspond so tightly to each other is because nodes wait for timeouts even if they have received all expected messages. The predicted timings are calculated as $t = t_c + t_m log(n)$ where t_c is time taken for committee consensus and t_m is time taken for individual merge operations.

conditions. We set a timeout threshold of 6 seconds for the committee consensus due to the extra rounds of communication there. These numbers give us the values seen in Figure 6.8.

With our timeout of 2 seconds, we saw $>$ 99 percentile message delivery in our testing. Assuming this timeout, we see that the total time taken for single cycle trie build in a network with 1000 nodes is 26 seconds, for 100,000 nodes it would only increase to about 40 seconds. This demonstrates the logarithmic scaling of Cambium's latency with number of nodes (as shown by Figure 6.8's exponential x-axis).

For productions systems, the timeout selection would have to be re-done with more testing in the expected networking environment that a given production system is expected to operate in.

6.7 Use cases for Cambium

The differences between blockchains and Cambium are best illustrated by considering the hypothetical car maintenance record system we talked about in § 6.1. Suppose a permissioned blockchain like Hyperledger Sawtooth [120] is used to maintain data for all cars. In this case, every independent entity would typically wish to join as a validator. Every validator would need to maintain a complete copy of the ledger including all the data committed by all other entities. This not only incurs a large storage requirement but is also a liability – the committed data may contain sensitive information that should not leak to the outside world. Smaller organisations would struggle to meet these requirements and thus, would struggle to participate in the system. This would result

in a centralised system with only a few big players able to bear the costs and liabilities.

Moreover, such blockchain-based systems are inherently limited in the number of participants they can support if permissioned or impose high costs (energy, transaction fees) if not. For example, in the US, there are ∼230,000 auto repair shops [197] and ∼280 million cars [91]. It is infeasible to have all cars serve as validators[5] using any existing blockchain consensus algorithm. It might just about be possible to have all repair shops act as validators if we use Robust Round Robin and set block times to be very high (on the order of a day)[6]. Assuming it is possible, how would such a blockchain system compare to a similarly set up Cambium one?

First, if we assume that only hashes of transactions are stored on the blockchain (for storage and privacy reasons), we would be limited to about 30,000 transactions a day for a standard block size of 1 MB. This would mean that only about a tenth of repair shops would be able to commit data on a given day probably resulting in high transaction fees or late limiting of some form. In Cambium, each node is able to send a transaction per day. Second, the storage cost of the blockchain system would be about 1 MB per day while it is about 18 KB per day with Cambium. This difference may not be too relevant in case of repair shops using desktop computers but is significant in networks with constrained devices such as IoT sensors. Lastly, in the blockchain system it is possible for any participant to monitor the activity of any other participant; this is infeasible in Cambium since this information is only shared with $2c$ nodes per cycle.

In a more realistic deployment with existing production software, some form of escrow or middleware service must be employed to run this consensus algorithm. These services serve to centralise the system negating the desired trustlessness of blockchain systems. Cambium sidesteps the scalability issues. Nodes only store their own input data and Merkle proofs thus mitigating storage and data liability concerns. Logarithmic scalability opens up the possibility of more widespread collaboration on trustworthy data without centralised services.

Some novel applications are also opened up by Cambium. It provides a decentralised build process for TODA's Proof of Provenance (POP) data structure [210], which uses tries to provide efficient proof structures for maintaining the uniqueness of digital things. Cambium provides us with a way to have a massively distributed root of trust, something that we did not yet have.

As for real-world analogies, there may be some similarity between the Cambrium approach and traditional university governance, where staff (and some students) are allocated more or less at random to a thicket of committees that keep their own minutes but pass copies of minutes and of other paperwork to each other. There is no central

[5] I am omitting a discussion about proof of work based blockchains here due to their high energy cost and susceptibility to coin infanticide. It is hard to imagine car owners being okay with their cars burning more fuel to participate in some consensus protocol. Mining incentives are also quite tangential to the goal of such a car maintenance platform.

[6] Do note that this is conjecture; RRR has only been tested up to a network size of 10,000 in an AWS setup and I am not aware of any deployed permissioned blockchain with a larger network size.

controlling intelligence, and no central repository (at least until cloud-based word processing came along), but individual decision makers and the committees that support them can and do collect and file their own records of minutes to justify their decisions if challenged or audited.

6.8 Comparison to existing systems

Compared with Bitcoin, Cambium provides scalability, privacy, better participant management and energy efficiency. Bitcoin, however, does not require a sybil-resistant gatekeeping mechanism since Proof-of-Work acts as both a gatekeeper and a consensus mechanism. Compared with Robust Round Robin, Cambium yet again provides better scalability and privacy although they are closely matched in terms of participant management and energy efficiency.

Another interesting point of comparison is with sidechains. To some degree, sidechains make a similar trade-off of scalability vs. write validation. From an architectural point of view, the primary difference ends up being the existence of a "main" chain that can become a bottleneck. No such main chain exists in the case of Cambium; all nodes operate on the same data structure and follow the same consensus rules. This homogeneity allows for easier assurance of data integrity than with sidechains, where one needs to be party to the sidechain transaction to glean any trust information.

Perhaps the closest non-blockchain comparison to Cambium in terms of design is Hashgraph [33]. Hashgraph abandons a linear blockchain in favour of a graph that maintains a "gossip about gossip" record. Effectively, Hashgraph is contingent upon the idea that if every node knows what every other node knows then each node can calculate what another node *would have* voted for a given consensus question. This means that votes need not be cast using explicit messaging protocols. Moreover, adding new transactions is a matter of any node simply signing and gossiping it. Thus, the validation of the data occurs as part of the normal gossip protocol. One drawback of the Hashgraph approach is that nodes need to maintain the state of their peers, which limits scalability especially on resource constrained devices; in Cambium, we do not need peers to do this state maintenance. Moreover, the Hashgraph model is still based on a global data store since all the nodes need to validate data submitted to the network. Lastly, the throughput and latency of Hashgraph scales poorly, since all events need to be communicated to all nodes.

6.9 Future work and limitations

6.9.1 Windowing

In any system based on banyan tries (not just Cambium), there are two distinct things that can go wrong for node A at merge level k in cycle trie t. One is that A may fail to

receive sufficient information from nodes in its ally group, resulting in A putting a hole into the ally branch in cycle trie t. If A's peer B did receive sufficient information from allies at this same merge branch then B will put a value into that branch. This will ultimately result in A and B arriving at different cycle roots for cycle t, likely leading in cycle $t + 1$ to their joining committees with dissent. If this division with some nodes placing a value and others placing a hole happens often during cycle t it could lead to enough disagreement that we have to rebuild cycle t, slowing down the whole system. The other thing that can go wrong is that a node might lie about its value. This also leads to dissent in cycle $t + 1$, and is the primary tool of the Byzantine adversary, as explored in § 6.5.1.

There are many similarities between these two issues. In both cases nodes end up with different values for the same merge branch. And in both cases having more information – having access to previous merge branch values from your peers and allies, for example – makes it harder for things to go wrong. It makes holes easier to fill, and makes equivocation harder to hide.

Windowing can help mitigate both problems. This involves sending not only a signed version of your current value (your MP), but also a selection (i.e. a *window*) of the messages you received that lead you to that value. This provides more evidence to your peers and allies which can be used to fill holes when messages from previous allies did not meet thresholds. It also increases the difficulty of equivocation by requiring more nodes to take part in the collusion.

A window, then, is the section of the trie that you are required to send to your allies at each level. This window has a depth (number of preceding merge levels) and a breadth (number of messages per merge level). Determining the correct values for those parameters and coming up with a lightweight mechanism for dissemination of these windows is an active area of research.

6.9.2 Identity management

In the introduction, we mentioned that Cambium can be used with a large range of mechanisms such as PoW, PoS or centralised identity management. A gatekeeping layer for Cambium that supports pluggable identity management systems would make real-world deployments much simpler. The functions of any gatekeeping mechanism are the prevention of sybil attacks [84] and the dissemination of the participant list to all nodes. Going one step further, we believe it would be possible to design Cambium-like consensus algorithms even in the absence of a globally visible address list, as long as the total number of nodes is known. Exploring that possibility might lead to several new consensus algorithms.

6.9.3 Penalising misbehaviour

An avenue for penalising Byzantine nodes in Cambium would be by using Proofs of Equivocation (PoEs). If in the course of building up their trie, a node encounters equivocating messages from another node then those two messages together constitute a PoE. Ideally, we would want that PoE to be immediately disseminated to the entire network and thus exclude the equivocating node from participation. However, doing this as part of the build process introduces additional networking costs and doesn't guarantee complete dissemination. One alternative way to do it would be to have a parallel gossip network for the network. This is decoupled from the cycle trie so while the transmission isn't immediate, it also wouldn't have adverse effects on the core consensus protocol. Designing ways to broadcast PoEs quickly is a crucial defensive research question. There may be some interesting parallels to existing work in ad-hoc networks such as the so-called suicide attacks presented by Moore et al. [157].

6.9.4 Offline nodes

One problem with Cambium versus blockchains is that if a node is offline then it is difficult for it to prove to a counterparty that it did not perform any actions during that time period. This is crucial in any application where double-spend prevention is required, as we discussed in § 6.4.10. Currently, the best that this node (call it A) can do is ask its committee or merge level 1 allies in that cycle (call it cycle t) for their paths which will show that A did not commit an input in that cycle. There are two issues with this. First, the committee members and allies may be offline or may refuse to give A the paths. Second, the committee and allies in t might be adversarial and fill in a random hash value instead of null for A; this is called *junking*.

Preventing junking and helping with null retrieval is critical for applications that require double spend prevention. One avenue that we are currently exploring is introducing a "junk flag" that A can set when it comes back online. Setting this junk flag would invalidate all inputs made by A (maliciously or otherwise) in the last k cycles. This would mean that as long as A is not offline for longer than k cycles, junking it is impossible. However, it also means that any counterparty wishing to get a proof from A would now have to ask for k more cycle proofs. This is not ideal since issues might arise if A goes offline in those intervening cycles—it can't prove that it hasn't used the junk flag. We are exploring workarounds to fix this issue and aim to incorporate it into Cambium in the near future.

6.9.5 Liveness

As mentioned earlier, Cambium exhibits poorer liveness than traditional ledger based consensus. This weakness stems partly from the lower number of messages sent per node and partly because of the trie structure. Imagine a node attempting to perform

the merge operation at level $log(\frac{n}{c} - 1)$. At this level, its allies are responsible for informing of the information pertaining to half of the network. If the node fails to receive this information, it must attempt the cycle again. Thus, just c number of nodes being offline can cause a node to exceed its hole threshold. If this occurs for a large number of nodes (tunable by adjusting the dissent threshold), the next cycle will fail and cause the network to re-attempt this cycle. The windowing scheme mentioned above mitigates this issue to some degree but poorer liveness vis-a-vis ledgers-based systems seems to be a fundamentally difficult problem to solve for Cambium-like systems.

6.9.6 Defining semantics

In this chapter, I have spoken about the Cambium consensus mechanism in isolation. In order to make useful applications on top of Cambium, we need to also design a semantics layer that standardises and gives operational meaning to the data committed by the nodes. In our leading example of car maintenance, we would need some way to represent actions such as transfer of ownership, maintenance record update, signalling no update, etc. Designing a standardised format that is adaptable for a wide range of applications is an important step towards making Cambium-based systems a reality.

6.10 Conclusion

In this chapter, I presented Cambium and showed how it achieves its logarithmic scaling properties by using banyan tries. We've also sketched out its resilience to certain attack vectors by Byzantine and Dolev-Yao adversaries, giving us some security assurances of planted cycle tries at the cost of poorer liveness. These properties lend themselves to building consensus over larger-than-ever network sizes in relatively short periods of time. We fully expect new Cambium-like algorithms in the future, fine-tuned to particular applications and device constraints. We hope for Cambium to be to trie-based systems what Bitcoin is to blockchains.

"Neither question nor answer was meant as anything more than a polite preamble to conversation."

—Arundhati Roy, The God of Small Things

7 In closing

Centralised systems can be problematic because they skew power dynamics and create a single point of failure. This is doubly true for information networks because of their tendency towards centralisation due to well-known economic and technical factors. Making systems that are decentralised by design is a goal as old as the Web itself, yet the Web today resembles an oligopoly more than ever before.

The advent of Bitcoin and the blockchain networks that followed it has given many researchers hope that these systems offer a way out. This hope stems not just from the decentralised design of these networks but also from the possibility of cryptocurrencies enabling novel economic models to displace the surveillance-driven business methods of today.

I started off in chapter 3, by looking at the flipside of this hopeful new technology – the crimes enabled by cryptocurrencies and the lack of effective recourse for victims. I examined the scope of the problem and assessed the factors that make cryptocurrencies tempting for criminals. Having looked at legal precedents, we developed a system for tracking proceeds of crime as they are passed around on the blockchain. I also designed a visualisation tool to help investigators and researchers spot transaction patterns that could help single out suspicious actors.

Following the release of these tools, we got in touch with victims of cryptocurrency theft as well as regulators. This led us to change our viewpoint on the cryptocurrency ecosystem as we came to understand the exchanges' outsized influence and fragmentary oversight. This led us to draft eight recommendations for regulators including urging central banks to explore implementing central-bank cryptocurrencies.

We turned our attention to permissioned blockchains in chapter 4. I started off

by explaining why permissionless systems don't work for certain applications and why permissioned blockchains could improve transparency and efficiency. To go beyond just theory and see the issues involved in shipping a real production system to users, I worked with a startup to design their permissioned blockchain backend.

This started off by comparing existing blockchain frameworks and noting common issues. We then discussed two of those issues: the computational and storage costs associated with an ever-growing blockchain, and the difficulty with editing historical data. I also discussed the problems caused by the scalability deficiencies of existing consensus algorithms.

I tackled this last issue in chapter 5 where we looked at consensus algorithms. I presented Robust Round Robin (RRR), a consensus algorithm for blockchains. RRR works for both permissioned and permissionless settings. Compared with existing permissionless networks, it aims to provide fairer distribution of rewards and higher throughput. Compared with existing permissioned networks, it seeks to provide better scalability.

In chapter 6, I presented Cambium, a novel consensus algorithm that operates on a trie-based data structure instead of a linear ledger. This enables us build networks where transaction data is private by default and where the communication cost per node is logarithmic to network size. I hope that Cambium becomes to trie-based systems what Bitcoin is to blockchains.

Designing decentralised systems is hard, and designing them in such a way that they remain decentralised is harder still. Yet I believe that it is a goal worth pursuing. I hope that the work in this thesis helps nudge the designers of the next generation of decentralised systems to take a broader, more considered view of their systems and the effects their systems have on society. I also hope that the technologies presented here play a role—however small—in the engineering of these systems.

Bibliography

[1] A. Aghaseyedjavadi, B. Bloomer, and S. Giudici. *Coin viz*. Online: Accessed on 22-02-2019. URL: http://people.ischool.berkeley.edu/shaun/infoviz/bitcoin/index.html.

[2] Mansoor Ahmed, Ilia Shumailov, and Ross Anderson. "Tendrils of Crime: Visualizing the Diffusion of Stolen Bitcoins". In: *Proceedings of The Fifth International Workshop on Graphical Models for Security*. 2018. URL: https://link.springer.com/chapter/10.1007/978-3-030-15465-3_1.

[3] Mansoor Ahmed, Anirudh Sriram, and Sanjay Singh. "Short Term Firm-Specific Stock Forecasting with BDI Framework". In: *Computational Economics* 55.3 (Mar. 2020), pp. 745–778. ISSN: 1572-9974. DOI: 10.1007/s10614-019-09911-0. URL: https://doi.org/10.1007/s10614-019-09911-0.

[4] Emma Ahmed-Rengers and Mansoor Ahmed-Rengers. "Democracy on the Margins of the Market: A Critical Look Into the Privatisation of Cyber Norm Formation". In: The Hague Program for Cyber Norms. The Hague, The Netherlands, 2020. URL: https://www.thehaguecybernorms.nl/conference-2020-speakers/emma-ahmed-rengers.

[5] Mansoor Ahmed-Rengers. "FrameProv: Towards End-to-End Video Provenance". In: *Proceedings of the New Security Paradigms Workshop*. NSPW. San Carlos, Costa Rica: Association for Computing Machinery, 2019, 68–77. DOI: 10.1145/3368860.3368866. URL: https://doi.org/10.1145/3368860.3368866.

[6] Mansoor Ahmed-Rengers. *Mansoor-AR - GitHub*. Online: Accessed on 28-10-2020. https://mansoor-ar.github.io/Cambium/.

[7] Mansoor Ahmed-Rengers. "Trusting video in a fake news world". In: *OpenDemocracy* (Mar. 2020). Online: Accessed on 28-10-2020. URL: https://www.opendemocracy.net/en/digitaliberties/trusting-video-fake-news-world/.

[8] Mansoor Ahmed-Rengers, Ross Anderson, Darija Halatova, and Ilia Shumailov. "Snitches Get Stitches: On the Difficulty of Whistleblowing". In: *Security Protocols XXVII*. Ed. by Jonathan Anderson, Frank Stajano, Bruce Christianson, and Vashek Matyáš. Cham: Springer International Publishing, 2020, pp. 289–303. ISBN: 978-3-030-57043-9. URL: https://link.springer.com/chapter/10.1007/978-3-030-57043-9_27.

[9] Mansoor Ahmed-Rengers and Kari Kostiainen. “Don’t Mine, Wait in Line: Fair and Efficient Blockchain Consensus with Robust Round Robin”. In: *CoRR* (2019). arXiv: 1804.07391. URL: http://arxiv.org/abs/1804.07391.

[10] Mansoor Ahmed-Rengers and Dann Toliver. *Cambium - GitHub*. Online: Accessed on 28-10-2020. https://mansoor-ar.github.io/Cambium/.

[11] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. “Chainspace: A Sharded Smart Contracts Platform”. In: *CoRR* (Aug. 2017). arXiv: 1708.03778. URL: http://arxiv.org/abs/1708.03778.

[12] Knut Alicke, Alan Davies, Markus Leopoldseder, and Alex Niemeyer. “Blockchain technology for supply chains—A must or a maybe?” In: *McKinsey and Company* (Sept. 2017). Online: Accessed on 28-10-2020. URL: https://www.mckinsey.com/business-functions/operations/our-insights/blockchain-technology-for-supply-chainsa-must-or-a-maybe.

[13] Harald Tveit Alvestrand. *A Mission Statement for the IETF*. RFC 3935. IETF Network Working Group, Oct. 2004, pp. 1–6. URL: https://tools.ietf.org/html/rfc3935.

[14] AM Online. *Mercedes adopts digital service records*. Online: Accessed on 24-12-2020. https://www.am-online.com/news/2007/11/6/mercedes-adopts-digital-service-records/15985/. 2007.

[15] Ross Anderson. *GCHQ helps banks dump fraud losses on customers*. Online: Accessed on 28-10-2020. https://www.lightbluetouchpaper.org/2016/05/27/gchq-helps-banks-dump-fraud-losses-on-customers/. 2016.

[16] Ross Anderson. *Stolen Bitcoin Tracing*. Online: Accessed on 28-10-2020. https://www.youtube.com/watch?v=UlLN0QERWBs. 2018.

[17] Ross Anderson, Ilia Shumailov, and Mansoor Ahmed. “Making Bitcoin Legal”. In: *Proceedings of the Twenty-sixth International Workshop on Security Protocols*. 2018. URL: https://link.springer.com/chapter/10.1007/978-3-030-03251-7_29.

[18] Ross Anderson, Ilia Shumailov, Mansoor Ahmed, Johann Bezuidenhoudt, and Nicholas Bohm. *Failures of Trust and Regulation in Cryptocurrency*. Online: Accessed on 06-12-2020. http://data.parliament.uk/WrittenEvidence/CommitteeEvidence.svc/EvidenceDocument/Treasury/Digital%20currencies/written/82188.html. Apr 30 2018.

[19] Ross Anderson, Ilia Shumailov, Mansoor Ahmed, and Alessandro Rietmann. “Bitcoin Redux”. In: *Proceedings of the 17th Annual Workshop on the Economics of Information Security*. 2018. URL: https://www.repository.cam.ac.uk/handle/1810/287807.

[20] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2nd ed. Wiley Publishing, 2008. ISBN: 9780470068526.

[21] Sébastien Andreina, Jens-Matthias Bohli, Ghassan O. Karame, Wenting Li, and Giorgia Azzurra Marson. *PoTS - A Secure Proof of TEE-Stake for Permissionless Blockchains*. Cryptology ePrint Archive, Report 2018/1135. https://eprint.iacr.org/2018/1135. 2018.

[22] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. Porto, Portugal: ACM, 2018, 30:1–30:15. ISBN: 978-1-4503-5584-1. DOI: 10.1145/3190508.3190538. URL: http://doi.acm.org/10.1145/3190508.3190538.

[23] App Annie. *State of Mobile*. Tech. rep. Online: Accessed on 28-10-2020. 2020, pp. 1–49. URL: https://www.appannie.com/en/go/state-of-mobile-2020/.

[24] Ricardo Araujo. "Assessing the efficiency of the anti-money laundering regulation: an incentive-based approach". In: *Journal of Money Laundering Control* 11 (Jan. 2008), pp. 67–75. DOI: 10.1108/13685200810844505.

[25] ARM. *TrustZone*. Online: Accessed on 28-10-2020. https://developer.arm.com/ip-products/security-ip/trustzone.

[26] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. "Redactable Blockchain – or – Rewriting History in Bitcoin and Friends". In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2017, pp. 111–126. DOI: 10.1109/EuroSP.2017.37.

[27] Raphael Auer, Giulio Cornelli, and Jon Frons. "Rise of the central bank digital currencies: drivers, approaches and technologies". In: *Bank for International Settlements* (2020). Online: Accessed on 29-11-2020. URL: https://www.bis.org/publ/work880.pdf.

[28] BaFin Federal Financial Supervisory Authority. *Crypto.exchange GmbH: BaFin orders cessation of unauthorized principal broking services*. Online: Accessed on 28-10-2020. https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Verbrauchermitteilung/unerlaubte/2018/meldung_180129_Crypto_exchange_en.html. 5 Feb 2018.

[29] BaFin Federal Financial Supervisory Authority. *necoin Ltd, Dubai: Prohibition of involvement in unauthorised money remittance business*. Online: Accessed on 28-10-2020. https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Verbrauchermitteilung/unerlaubte/2017/vm_170418_Onecoin_Ltd_en.html. 18 Apr 2017.

[30] BaFin Federal Financial Supervisory Authority. *Onecoin Ltd (Dubai), OneLife Network Ltd (Belize) und One Network Services Ltd (Sofia/Bulgaria): BaFin issues cease and desist orders holding the companies to stop own funds trading in "OneCoins" in Germany*. Online: Accessed on 28-10-2020. https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Verbrauchermitteilung/unerlaubte/2017/vm_170427_Onecoin_Ltd_en.html. 20 Apr 2017.

[31] BaFin Federal Financial Supervisory Authority. *Virtual Currencies (VC)*. Online: Accessed on 28-10-2020. https://www.bafin.de/EN/Aufsicht/FinTech/VirtualCurrency/virtual_currency_node_en.html. 2018.

[32] BAE Systems and SWIFT. *Follow The Money: Understanding the money laundering techniques that support large-scale cyber-heists*. Online: Accessed on 28-10-2020. https://www.swift.com/sites/default/files/files/swift_bae_report_Follow-The%20Money.pdf.

[33] Leemon Baird. *The Swirlds Hashgraph consensus algorithm: Fair, Fast, Byzantine Fault Tolerance*. http://www.leemon.com/papers/2016b.pdf. May 2016.

[34] Billy Bambrough. "Coronavirus Is Shaping Up To Be Very Bad For Banks—But Not For Bitcoin". In: *Forbes* (Sept. 2020). Online. Accessed on 06-12-2020/ https://www.forbes.com/sites/billybambrough/2020/09/19/coronavirus-is-shaping-up-to-be-very-bad-for-banks-but-great-for-bitcoin/?sh=3d437c6f3184.

[35] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. "Consensus in the Age of Blockchains". In: *arXiv:1711.03936* (2017).

[36] Paul Baran. "The Future Computer Utility". In: *The Public Interest* (1967).

[37] G. D. Battista, V. D. Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia. "BitConeView: visualization of flows in the bitcoin transaction graph". In: *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*. 2015, pp. 1–8.

[38] BBC News. *Bitcoin: El Salvador makes cryptocurrency legal tender*. Online: Accessed on 20-06-2021. https://www.bbc.com/news/world-latin-america-57398274. 2021.

[39] Tim Berners-Lee. *The World Wide Web and the Web of Life*. Online: Accessed on 20-12-2020. https://www.w3.org/People/Berners-Lee/UU.html.

[40] Tim Berners-Lee. *Three challenges for the web, according to its inventor*. Online: Accessed on 28-10-2020. https://webfoundation.org/2017/03/web-turns-28-letter/. 2017.

[41] StackExchange Bitcoin. *What is the average size of a bitcoin transaction?* Online: Accessed on 28-10-2020. https://bitcoin.stackexchange.com/questions/31974/what-is-the-average-size-of-a-bitcoin-transaction.

[42] BitcoinStats. *Data Propagation: Daily Snapshots*. Online: Accessed on 12-01-2018. http://bitcoinstats.com/network/propagation/.

[43] Bitcoinwiki. *Proof of Stake*. Online: Accessed on 28-10-2020. https://en.bitcoin.it/wiki/Proof_of_Stake.

[44] Bitfury. "Use case – Crystal tracking ransomware payments". In: (2018). Online: Accessed on 14-04-2018. https://crystalblockchain.com/files/Crystal-Use-Cases-Ransomware.pdf.

[45] Apolline Blandin, Ann Sofie Cloots, Hatim Hussain, Michel Rauchs, Rasheed Saleuddin, Jason Grant Allen, Bryan Zhang, and Katherine Cloud. "Global Cryptoasset Regulatory Landscape Study". In: *Cambridge Centre for Alternative Finance* (2019). URL: https://www.jbs.cam.ac.uk/faculty-research/centres/alternative-finance/publications/cryptoasset-regulation/.

[46] Rainer Boehme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. "Bitcoin: Economics, Technology, and Governance". In: *Journal of Economic Perspectives v 29 issue 2* (July 2015).

[47] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. "Research Perspectives and Challenges for Bitcoin and Cryptocurrencies". In: *IACR Cryptol. ePrint Arch.* (2015), p. 261. URL: http://eprint.iacr.org/2015/261.

[48] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. "Software Grand Exposure: SGX Cache Attacks Are Practical". In: *CoRR* abs/1702.07521 (2017). URL: http://arxiv.org/abs/1702.07521.

[49] Dustin Brickwood. *Understanding Trie Databases in Ethereum*. Online: Accessed on 24-12-2020. https://medium.com/shyft-network-media/understanding-trie-databases-in-ethereum-9f03d2c3325d. 2018.

[50] Katrina Brooker. *Tim Berners-Lee, the Man Who Created the World Wide Web, Has Some Regrets*. Online: Accessed on 28-10-2020. https://www.vanityfair.com/news/2018/07/the-man-who-created-the-world-wide-web-has-some-regrets. 2018.

[51] Robin Bryce and Mansoor Ahmed-Rengers. *RRR Spec*. Online: Accessed on 04-01-2021. https://github.com/RobustRoundRobin/RRR. 2020.

[52] BTC.COM. *Pool Distribution*. Online: Accessed on 03-01-2021. https://btc.com/stats/pool?pool_mode=year.

[53] Ethan Buchman, Jae Kwon, and Zarko Milosevic. "The latest gossip on BFT consensus". In: *CoRR* abs/1807.04938 (2018). arXiv: 1807.04938. URL: http://arxiv.org/abs/1807.04938.

[54] Kent Bye. *Internet Co-Inventor Vint Cerf on Decentralized Internet Challenges*. Online: Accessed on 28-10-2020. https://voicesofvr.com/669-internet-co-inventor-vint-cerf-on-decentralized-internet-challenges/. 2018.

[55] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. "Secure and Efficient Asynchronous Broadcast Protocols". In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. 2001, pp. 524–541. DOI: 10.1007/3-540-44647-8_31. URL: https://doi.org/10.1007/3-540-44647-8_31.

[56] Ryan Calo and Alex Rosenblat. "The Taking Economy: Uber, Information, and Power". In: *Columbia Law Review* 117 (2017). URL: https://digitalcommons.law.uw.edu/faculty-articles/47.

[57] Cambridge Centre for Alternative Finance. *Cambridge Bitcoin Electricity Consumption Index*. Online: Accessed on 28-10-2020. https://cbeci.org/cbeci/comparisons.

[58] Augustin Carstens. "Money in the digital age: what role for central banks?" In: *Bank for International Settlements* (6 Feb 2018). URL: https://www.bis.org/speeches/sp180206.htm.

[59] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. OSDI '99. New Orleans, Louisiana, USA: USENIX Association, 1999, 173–186. ISBN: 1880446391.

[60] Chainalysis. *The 2020 State Of Crypto Crime*. Online: Accessed on 28-10-2020. https://go.chainalysis.com/2020-Crypto-Crime-Report.html. 2020.

[61] David L. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Commun. ACM* 24.2 (Feb. 1981), 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: https://doi.org/10.1145/358549.358563.

[62] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. "SGXPECTRE Attacks: Leaking Enclave Secrets via Speculative Execution". In: *arXiv:1802.09085* (2018).

[63] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. "On Security Analysis of Proof-of-Elapsed-Time (PoET)". In: *Stabilization, Safety, and Security of Distributed Systems*. Oct. 2017, pp. 282–297. ISBN: 978-3-319-69083-4. DOI: 10.1007/978-3-319-69084-1_19.

[64] Angshuman Choudhury. "How Facebook Is Complicit in Myanmar's Attacks on Minorities". In: *The Diplomat* (2020). URL: https://thediplomat.com/2020/08/how-facebook-is-complicit-in-myanmars-attacks-on-minorities/.

[65] Judd v Citibank. *107 Misc.2d 526*. 1980.

[66] Coinbase. *Coinbase User Agreement*. Online: Accessed on 06-12-2020. https://www.coinbase.com/legal/user_agreement. 2020.

[67] CoinTelegraph. *The History and Evolution of Proof of Stake*. Online: Accessed on 28-10-2020. https://cointelegraph.com/news/the-history-and-evolution-of-proof-of-stake.

[68] Companies House. *CB Payments Ltd, Report and Financial Statements*. Online: Accessed on 06-12-2020. shorturl.at/vHMU4. 2019.

[69] Companies House. *Coinbase UK Ltd, Report and Financial Statements*. Online: Accessed on 06-12-2020. shorturl.at/ouEY7. 2019.

[70] Kelly Cooper. *Burrow - The Boring Blockchain*. Online: Accessed on 28-10-2020. https://wiki.hyperledger.org/display/burrow/Burrow+-+The+Boring+Blockchain. Feb. 2020.

[71] *Corda documentation*. Online: Accessed on 28-10-2020. https://docs.corda.net/docs/corda-os/4.6.html.

[72] G.F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design*. International computer science series. Addison-Wesley, 2001. ISBN: 9780201619188. URL: https://books.google.co.uk/books?id=6KQZAQAAIAAJ.

[73] Cryptovoices. *24h trading volume Per On-Chain Payment*. 2018.

[74] George Danezis. *Distributed Ledgers: what is so interesting about them?* Online: Accessed on 28-10-2020. https://conspicuouschatter.wordpress.com/2018/09/27/distributed-ledgers-what-is-so-interesting-about-them/. Sept. 2018.

[75] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. "Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 66–98.

[76] Joshua Davis. "The Crypto-Currency". In: *The New Yorker* (2011). Online: Accessed on 28-10-2020. https://www.newyorker.com/magazine/2011/10/10/the-crypto-currency.

[77] Carlos De Meijer. "Blockchain versus GDPR and who should adjust most". In: *FinExtra* (Oct. 2018). Online: Accessed on 28-10-2020. https://www.finextra.com/blogposting/16102/blockchain-versus-gdpr-and-who-should-adjust-most.

[78] Christian Decker and Roger Wattenhofer. "Information Propagation in the Bitcoin Network". In: *International Conference on Peer-to-Peer Computing (P2P)*. 2013.

[79] Tuur Demeester. *Bitcoin: digital gold or digital cash? Both.* Online: Accessed on 28-10-2020. https://medium.com/@tuurdemeester/bitcoin-digital-gold-or-digital-cash-both-382a346e6c79.

[80] *Devaynes v Noble (Clayton's Case)*. http://www.commonlii.org/uk/cases/EngR/1815/77.pdf. 1816.

[81] *Directive 1999/13/EC of the European Parliament and the Council*. 13 December 1999.

[82] *Directive 2000/31/EC of the European Parliament and the Council*. 8 June 2000.

[83] *Directive 2009/110/EC of the European Parliament and of the Council of 16 September 2009 on the taking up, pursuit and prudential supervision of the business of electronic money institutions amending Directives 2005/60/EC and 2006/48/EC and repealing Directive 2000/46/EC*. 16 September 2009.

[84] John R. Douceur. "The Sybil Attack". In: *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 251–260. ISBN: 3-540-44179-4. URL: http://dl.acm.org/citation.cfm?id=646334.687813.

[85] Hannah Ellis-Peterson and Shaikh Azizur Rahman. "Facebook faces grilling by MPs in India over anti-Muslim hate speech". In: *The Guardian* (2020). Online: Accessed on 28-10-2020. URL: https://www.theguardian.com/world/2020/sep/01/facebook-faces-grilling-by-mps-in-india-over-anti-muslim-hate-speech.

[86] Emerging Technology from the arXiv team. "We're getting closer to being able to track stolen bitcoins". In: *MIT Technology Review* (Jan. 2019). Online: Accessed on 28-10-2020. URL: https://www.technologyreview.com/2019/01/18/137797/were-getting-closer-to-being-able-to-track-stolen-bitcoins/.

[87] Ethereum Foundation. *Consortium Chain Development*. Online: Accessed on 18-12-2020. https://github.com/ethereum/wiki/wiki/Consortium-Chain-Development.

[88] Yaya J. Fanusie and Tom Robinson. "Bitcoin Laundering: An Analysis of Illicit Flows Into Digital Currency Services". In: *Elliptic* (2018).

[89] FCA. *Financial Conduct Authority's Written Submission on Digital Currencies*. Online: Accessed on 06-12-2020. http://data.parliament.uk/WrittenEvidence/CommitteeEvidence.svc/EvidenceDocument/Treasury/Digital%20currencies/written/81677.html. April 2018.

[90] Federal Trade Commission. *Equifax to Pay $575 Million as Part of Settlement with FTC, CFPB, and States Related to 2017 Data Breach*. Online: Accessed on 28-10-2020. https://www.ftc.gov/news-events/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related. 2019.

[91] Finances Online. *The Number of Cars in the US in 2020/2021*. Online: Accessed on 02-01-2021. https://financesonline.com/number-of-cars-in-the-us/.

[92] Forrester. *Seize The Day: Public Blockchain Is On The Horizon*. Online: Accessed on 28-10-2020. https://assets.ey.com/content/dam/ey-sites/ey-com/en_gl/topics/blockchain/ey-public-blockchain-opportunity-snapshot.pdf. 2019.

[93] David Fox. "Cryptocurrencies in the Common Law of Property". In: *SSRN* (2018). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3232501.

[94] David Fox. *Property Rights in Money*. Oxford University Press, 2008.

[95] Matthew Frankel. *The 7 Biggest Challenges Facing Bitcoin*. Online: Accessed on 28-10-2020. https://www.fool.com/investing/2017/11/19/the-7-biggest-challenges-facing-bitcoin.aspx. 2017.

[96] Freshfields. *Virtual currencies: how regulators treat it*. Jan. 2018.

[97] Brian Fung. "Move deliberately, fix things: How Coinbase is building a cryptocurrency empire". In: *Washingoton Post* (May 17 2018). Online: Accessed on 28-10-2020. https://www.washingtonpost.com/business/economy/move-deliberately-fix-things-how-coinbase-is-building-a-cryptocurrency-empire/2018/05/17/623d950c-587c-11e8-858f-12becb4d6067_story.html.

[98] Yuefei Gao and Hajime Nobuhara. "A Decentralized Trusted Timestamping Based on Blockchains". In: *IEEJ Journal of Industry Applications* 6 (July 2017), pp. 252–257. DOI: 10.1541/ieejjia.6.252.

[99] Juan Garay and Aggelos Kiayias. "SoK: A Consensus Taxonomy in the Blockchain Era". In: *Topics in Cryptology – CT-RSA 2020*. Ed. by Stanislaw Jarecki. Springer International Publishing, 2020, pp. 284–318. ISBN: 978-3-030-40186-3.

[100] Gartner. *Hype Cycle for Blockchain Technologies*. Online: Accessed on 28-10-2020. https://www.gartner.com/document/3947355. 2019.

[101] Vishal Gaur and Abhinav Gaiha. "Building a Transparent Supply Chain". In: *Harvard Business Review* (May 2020). URL: https://hbr.org/2020/05/building-a-transparent-supply-chain.

[102] Paul George. *FRC submission to the Treasury Select Committee Digital Currencies Inquiry*. Online: Accessed on 06-12-2020. http://data.parliament.uk/WrittenEvidence/CommitteeEvidence.svc/EvidenceDocument/Treasury/Digital%20currencies/written/81400.html. April 13 2018.

[103] Arthur Gervais, Ghassan Karame, Srdjan Capkun, and Vedran Capkun. "*Is Bitcoin a Decentralized Currency?*" In: *IEEE Security and Privacy Magazine v 12 no 3* (2014).

[104] Phil Glazer. *State of Global Cryptocurrency Regulation*. Online: Last accessed on 28-10-2020. https://www.bitcoininsider.org/article/16609/state-global-cryptocurrency-regulation-february-2018. 21 Jan 2018.

[105] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, and Leonid Reyzin. "NSEC5 from Elliptic Curves: Provably Preventing DNSSEC Zone Enumeration with Shorter Responses". In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 83. URL: http://eprint.iacr.org/2016/083.

[106] Russell Golman, David Hagmann, and George Loewenstein. "Information Avoidance". In: *Journal of Economic Literature, 55 (1): 96-135* (2017).

[107] Mt. Gox. *Terms of Use,* https://web.archive.org/web/20130906174719/https://www.mtgox.com/terms_of_service?Locale=en_US. January 20, 2012.

[108] Andy Greenberg. "A 200-Year-Old Idea Offers a New Way to Trace Stolen Bitcoins". In: *WIRED* (May 2018). Online: Accessed on 28-10-2020. URL: https://www.wired.com/story/bitcoin-blockchain-fifo-dirty-coins/.

[109] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. "SoK: Layer-Two Blockchain Protocols". In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Springer International Publishing, 2020, pp. 201–226. ISBN: 978-3-030-51280-4.

[110] Stuart Haber and Scott Stornetta. "How to time-stamp a digital document". In: *Conference on the Theory and Application of Cryptography*. Springer. 1990, pp. 437–455.

[111] Timo Hanke, Mahnush Movahedi, and Dominic Williams. "DFINITY Technology Overview Series, Consensus System". In: *arXiv:1805.04548* (2018).

[112] Markus Held. "Internet payments: Minimum Requirements for Security". In: *BaFin* (2015). URL: https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Fachartikel/2015/fa_bj_1505_zahlungen_im_internet_en.html.

[113] Thomas Hepp, Alexander Schoenhals, Christopher Gondek, and Bela Gipp. "OriginStamp: A blockchain-backed system for decentralized trusted timestamping". In: *IT - Information Technology* 60 (Nov. 2018). DOI: 10.1515/itit-2018-0020.

[114] "How a few companies are bitcoining it". In: *The Economist* (May 19 2018).

[115] Hughes, Eric. *A Cypherpunk's manifesto*. https://www.activism.net/cypherpunk/manifesto.html. 1993.

[116] *Hyperledger Besu*. Online: Accessed on 28-10-2020. https://besu.hyperledger.org/en/1.5.5/.

[117] *Hyperledger Burrow Documentation*. Online: Accessed on 28-10-2020. https://hyperledger.github.io/burrow/.

[118] *Hyperledger Indy Documentation*. Online: Accessed on 28-10-2020. https://indy.readthedocs.io/en/latest/.

[119] *Hyperledger Iroha Documentation*. Online: Accessed on 28-10-2020. https://iroha.readthedocs.io/en/1.1.3/overview.html.

[120] *Hyperledger Sawtooth Documentation*. Online: Accessed on 28-10-2020. https://sawtooth.hyperledger.org/docs/core/releases/1.2.5/.

[121] IETF Network Working Group. *Architectural Principles of the Internet*. Online: Accessed on 28-10-2020. https://www.ietf.org/rfc/rfc1958.txt. 1996.

[122] Fred Imbert. "JPMorgan CEO Jamie Dimon says bitcoin is a 'fraud' that will eventually blow up". In: *CNBC* (2017). Online: Accessed on 28-10-2020. URL: https://www.cnbc.com/2017/09/12/jpmorgan-ceo-jamie-dimon-raises-flag-on-trading-revenue-sees-20-percent-fall-for-the-third-quarter.html.

[123] Intel. *A Cost-Effective Foundation for End-to-End IoT Security*. Online: Accessed on 28-10-2020. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/intel-epid-white-paper.pdf. 2017.

[124] Intel. *Atmel and Microchip Adopt Intel Identity Technology for IoT*. Online: Accessed on 28-10-2020. http://download.intel.com/newsroom/kits/idf/2015_fall/pdfs/Intel_EPID_Fact_Sheet.pdf.

[125] Intel. *Attestation Service for Intel® Software Guard Extensions (Intel SGX): API Documentation*. Online: Accessed on 28-10-2020. https://software.intel.com/sites/default/files/managed/7e/3b/ias-api-spec.pdf.

[126] Intel. *Intel SGX Homepage*. Online: Accessed on 28-10-2020. https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html.

[127] Intel. *Q3 2018 Speculative Execution Side Channel Update*. Online: Accessed on 28-10-2020. https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00161.html. Aug. 2018.

[128] Internet.org. *Internet.org by Facebook*. Online: Accessed on 28-10-2020. https://info.internet.org/en/.

[129] *Istanbul BFT's design cannot successfully tolerate fail-stop failures*. Online: Accessed on 28-10-2020. https://github.com/ConsenSys/quorum/issues/305.

[130] *Istanbul Byzantine Fault Tolerance*. Online: Accessed on 28-10-2020. https://github.com/ethereum/EIPs/issues/650.

[131] J.P. Morgan. *J.P. Morgan Creates Digital Coin for Payments*. Online: Accessed on 28-10-2020. https://www.jpmorgan.com/solutions/cib/news/digital-coin-payments. Feb. 2019.

[132] JPMorgan. *Tessera - Enterprise Implementation of Quorum's transaction manager*. Online: Accessed on 28-10-2020. https://github.com/jpmorganchase/tessera.

[133] JPMorgan and Consensys. *GitHub - Quorum*. Online: Accessed on 28-10-2020. https://github.com/ConsenSys/quorum.

[134] Robin Kar. "Contract as Empowerment". In: *Chicago Law Review* 83 (2 2016), pp. 759–834. URL: https://chicagounbound.uchicago.edu/uclrev/vol83/iss2/4/.

[135] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10401 LNCS (2017), pp. 357–388. ISSN: 16113349. DOI: 10.1007/978-3-319-63688-7_12.

[136] Nancy Kim. *Wrap Contracts: Foundations and Ramifications*. Oxford University Press, 2013. ISBN: 9780199336975.

[137] Sunny King and Scott Nadal. *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*. Aug. 2012.

[138] Christoph Kinkeldey, Jean-Daniel Fekete, and Petra Isenberg. *BitConduite: Visualizing and Analyzing Activity on the Bitcoin Network*. EuroVis 2017 - Eurographics Conference on Visualization, Posters Track. 2017.

[139] Hugo Krawczyk and Tal Rabin. "Chameleon Signatures". In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*. The Internet Society, 2000. URL: https://www.ndss-symposium.org/ndss2000/chameleon-signatures/.

[140] Sophia Lam. "It's Time to Break up Big Tech". In: *The Gate* (2019). URL: http://uchicagogate.com/articles/2019/10/20/its-time-break-big-tech/.

[141] Fred Lambert. *Tesla fights new 'Right to Repair' initiative over cybersecurity concerns*. Online: Accessed on 24-12-2020. https://electrek.co/2020/10/14/tesla-fights-right-to-repair-initiative-over-cybersecurity-concerns/.

[142] Andrew Lawrence. *Rich Benoit Built a Career by Rebuilding a Tesla. What's Next for Rich Rebuilds?* Online: Accessed on 24-12-2020. https://www.caranddriver.com/features/a33660589/rich-benoit-tesla-rich-rebuilds/. 2020.

[143] Jason Leopold, Anthony Cormier, John Templon, Jeremy Singer-Vine, Scott Pham, Richard Holmes, Azeen Ghorayshi, Michael Sallah, Tanya Kozyreva, and Emma Loop. "8 Things You Need To Know About The Dark Side Of The World's Biggest Banks, As Revealed In The FinCEN Files". In: *BuzzFeed News* (2020). Online: Accessed on 28-10-2020. URL: https://www.buzzfeednews.com/article/jasonleopold/fincen-files-8-big-takeaways.

[144] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. "Meltdown". In: *ArXiv e-prints* (Jan. 2018). arXiv: 1801.01207. URL: https://arxiv.org/abs/1801.01207.

[145] David Lomet, Philip Bernstein, James Johnson, and Kenneth Wilner. *System and method for consistent timestamping in distributed computer databases*. Patent. Available online: https://patents.google.com/patent/US5212788A/en. 1990.

[146] W Lu. *bitcoin-tx-graph-visualizer*. URL: http://www.npmjs.com/package/bitcoin-tx-graphvisualizer.

[147] Ewa Luger, Stuart Moran, and Tom Rodden. "Consent for All: Revealing the Hidden Complexity of Terms and Conditions". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. New York, NY, USA: Association for Computing Machinery, 2013, 2687–2696. DOI: 10.1145/2470654.2481371. URL: https://doi.org/10.1145/2470654.2481371.

[148] Malta Today. *Why world leader crypto exchange Binance moved to Malta*. Online: Accessed on 28-10-2020. https://www.maltatoday.com.mt/business/business_news/93170/why_world_leader_crypto_exchange_binance_moved_to_malta. 2019.

[149] Apostolaki Maria, Zohar Aviv, and Vanbever Laurent. "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies". In: *Security and Privacy (S&P), 2017 IEEE Symposium on*. IEEE. 2017.

[150] Gregory Maxwell. *Confidential Transactions*. Online: Accessed on 28-10-2020. https://www.weusecoins.com/confidential-transactions/. 2015.

[151] John McCrank. "Equifax says 15.2 million UK records exposed in cyber breach". In: *Reuters* (2017). Online: Accessed on 28-10-2020. URL: https://www.reuters.com/article/us-equifax-cyber/equifax-says-15-2-million-uk-records-accessed-in-cyber-breach-idUSKBN1CF2JU.

[152] Sarah Meiklejohn. "Top Ten Obstacles along Distributed Ledgers Path to Adoption". en. In: *IEEE Security & Privacy* 16.4 (July 2018), pp. 13–19. ISSN: 1540-7993, 1558-4046. DOI: 10.1109/MSP.2018.3111235. URL: https://ieeexplore.ieee.org/document/8425611/ (visited on 10/12/2020).

[153] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey Voelcker, and Stefan Savage. "A Fistful of Bitcoins: Characterising Payments Among Men with No Names". In: *IMC 2013* (2013).

[154] Silvio Micali. "ALGORAND: The Efficient and Democratic Ledger". In: *CoRR* abs/1607.01341 (2016). URL: http://arxiv.org/abs/1607.01341.

[155] Silvio Micali, Michael Rabin, and Salil Vadhan. "Verifiable random functions". In: *40th Annual Symposium on Foundations of Computer Science*. IEEE. 1999, pp. 120–130.

[156] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. "Proof of Luck: An Efficient Blockchain Consensus Protocol". In: *Proceedings of the 1st Workshop on System Software for Trusted Execution*. SysTEX '16. Trento, Italy: ACM, 2016, 2:1–2:6. ISBN: 978-1-4503-4670-2. DOI: 10.1145/3007788.3007790. URL: http://doi.acm.org/10.1145/3007788.3007790.

[157] Tyler Moore, Jolyon Clulow, Shishir Nagaraja, and Ross Anderson. "New Strategies for Revocation in Ad-Hoc Networks". In: *Security and Privacy in Ad-hoc and Sensor Networks*. Ed. by Frank Stajano, Catherine Meadows, Srdjan Capkun, and Tyler Moore. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 232–246. ISBN: 978-3-540-73275-4.

[158] Malte Möser, Rainer Böhme, and Dominic Breuker. "An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem". In: *IEEE eCrime* (2013).

[159] Malte Möser, Rainer Böhme, and Dominic Breuker. "Towards Risk Scoring of Bitcoin Transactions". In: *Financial Cryptography* (2014).

[160] Fedor Muratov, Andrei Lebedev, Nikolai Iushkevich, Bulat Nasrulin, and Makoto Takemiya. "YAC: BFT Consensus Algorithm for Blockchain". In: *CoRR* abs/1809.00554 (2018). URL: http://arxiv.org/abs/1809.00554.

[161] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Online: Accessed on 30-10-2020. http://www.bitcoin.org/bitcoin.pdf. 2009.

[162] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies*. Princeton University Press, 2016. ISBN: 9780691171692.

[163] National Institute of Standards and Technology. *Trie data structure*. Online: Accessed on 24-11-2020. https://xlinux.nist.gov/dads/HTML/trie.html.

[164] Paul Nemitz. "Democracy and Technology in the Age of Artificial Intelligence". English. In: *Philosophical Transactions Royal Society* (Oct. 2018). DOI: 10.1098/rsta.2018.0089. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2018.0089.

[165] Lily Hay Newman. "A New Google+ Blunder Exposed Data From 52.5 Million Users". In: *Wired* (2018). Online: Accessed on 28-10-2020. URL: https://www.wired.com/story/google-plus-bug-52-million-users-data-exposed/.

[166] Alfred Ng. "How the Equifax hack happened, and what still needs to be done". In: *CNet* (2018). URL: https://www.cnet.com/news/equifaxs-hack-one-year-later-a-look-back-at-how-it-happened-and-whats-changed/.

[167] Devaynes v Noble. *35 ER 767, 781*. 1816.

[168] Niels ten Oever and Kathleen Moriarty. *The Tao of IETF*. Online: Accessed on 28-10-2020. https://ietf.org/about/participate/tao/. 2019.

[169] Diego Ongaro and John Ousterhout. "In Search of an Understandable Consensus Algorithm". In: *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, July 2014, pp. 305–319. ISBN: 978-1-931971-10-2. URL: https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro.

[170] *Orion Private Transaction Manager*. Online: Accessed on 28-10-2020. https://docs.orion.consensys.net/en/1.6.0/.

[171] Kin Pan. "Protecting Innovation: Antitrust Laws and the Tech Giants". In: *Law In Society* (2019). URL: https://www.lawinsociety.org/protecting-innovation-antitrust-laws-and-the-tech-giants.

[172] Helen Partz. "Hacked Crypto Exchange Coincheck Confirms Removal of Four Anonymity-Focused Altcoins". In: *Cointelegraph* (May 20 2018). Online: Accessed on 28-10-2020. https://bitlyfool.com/?p=15976.

[173] Steven Pearlstein. "Beating up on Big Tech is fun and easy. Restraining it will require rewriting the law." In: *Washington Post* (2020). Online: Accessed on 28-10-2020. URL: https://www.washingtonpost.com/business/2020/07/30/antitrust-amazon-apple-facebook-google/.

[174] *PoET 1.0 Specification*. Online: Accessed on 28-10-2020. https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html.

[175] The Solid Project. *What Is Solid*. Online: Accessed on 28-10-2020. https://solid.mit.edu/.

[176] Newley Purnell and Jeff Horwitz. "Facebook's Hate-Speech Rules Collide With Indian Politics". In: *The Wall Street Journal* (2020). Online: Accessed on 28-10-2020. URL: https://www.wsj.com/articles/facebook-hate-speech-india-politics-muslim-hindu-modi-zuckerberg-11597423346.

[177] Ilham A. Qasse, Manar Abu Talib, and Qassim Nasir. "Inter Blockchain Communication: A Survey". In: *Proceedings of the ArabWIC 6th Annual International Conference Research Track*. ArabWIC 2019. Rabat, Morocco: Association for Computing Machinery, 2019. ISBN: 9781450360890. DOI: 10.1145/3333165.3333167. URL: https://doi.org/10.1145/3333165.3333167.

[178] Wannan v Her Majesty the Queen. *Ottawa, 2003 FCA 423*. 2003.

[179] Max Raskin. "The Law and Legality of Smart Contracts". In: *Georgetown Law Technology Review* (2016). https://ssrn.com/abstract=2959166.

[180] Michel Rauchs, Apolline Blandin, Keith Bear, and Stephen McKeon. "Second Global Enterprise Blockchain Benchmarking Study". In: *Cambridge Centre for Alternative Finance* (2019). URL: https://www.jbs.cam.ac.uk/faculty-research/centres/alternative-finance/publications/2nd-global-enterprise-blockchain-benchmarking-study/.

[181] F. Reid and M. Harrigan. "An Analysis of Anonymity in the Bitcoin System". In: *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 2011, pp. 1318–1326.

[182] *Replay attack due to unsynchronized lastMessageReceived*. Online: Accessed on 28-10-2020. https://github.com/bft-smart/library/issues/63. 2018.

[183] Sylvain Ribes. *Chasing fake volume: a crypto-plague*. Online: Accessed on 28-10-2020. https://medium.com/@sylvainartplayribes/chasing-fake-volume-a-crypto-plague-ea1a3c1e0b5e. 15 Jan 2018.

[184] Dorit Ron and Adi Shamir. "How did Dread Pirate Roberts acquire and protect his bitcoin wealth?" In: *IACR preprint 2013/782* (Financial Cryptography 2013).

[185] Florian Sagstetter, Martin Lukasiewycz, Sebastian Steinhorst, Marko Wolf, Alexandre Bouard, William Harris, Somesh Jha, Thomas Peyrin, Axel Poschmann, and Samarjit Chakraborty. "Security Challenges in Automotive Hardware/Software Architecture Design". In: Jan. 2013, pp. 458–463. ISBN: 978-1-4673-5071-6. DOI: 10.7873/DATE.2013.102.

[186] *Sale of Goods Act [RSBC 1996], Section 27*. Online. Accessed on 29-10-2020. https://www.bclaws.ca/civix/document/id/complete/statreg/96410_01#section27.

[187] *Sale of Goods (Amendment) Act*. Online. Accessed on 29-10-2020. https://www.legislation.gov.uk/ukpga/1994/32/pdfs/ukpga_19940032_en.pdf. 1994.

[188] Sandvine. *The Global Internet Phenomena Report. COVID-19 Spotlight*. Tech. rep. Online: Accessed on 28-10-2020. May 2020, pp. 1–19. URL: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2020/Phenomena/COVID%20Internet%20Phenomena%20Report%2020200507.pdf.

[189] Sandvine. *The Mobile Internet Phenomena Report*. Tech. rep. Online: Accessed on 28-10-2020. Feb. 2020, pp. 1–14. URL: https://www.sandvine.com/download-report-mobile-internet-phenomena-report-2020-sandvine.

[190] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. "Malware Guard Extension: Using SGX to Conceal Cache Attacks". In: *CoRR* abs/1702.08719 (2017). URL: http://arxiv.org/abs/1702.08719.

[191] Meryl Sebastian. "Facebook's Ankhi Das Apologised For Sharing Anti-Muslim Post: Report". In: *Huffington Post* (2020). Online: Accessed on 28-10-2020. URL: https://www.huffingtonpost.in/entry/facebook-ankhi-das-anti-muslim-post_in_5f45de23c5b6cf66b2b02390.

[192] Meryl Sebastian. "Facebook's Ankhi Das Facebook's Ankhi Das Supported Modi, BJP, Openly Talked Of Efforts To Help Them Win 2014 Election: Report". In: *Huffington Post* (2020). Online: Accessed on 28-10-2020. URL: https://www.huffingtonpost.in/entry/ankhi-das-facebook-india-modi-bjp_in_5f4c7f4fc5b697186e37f2d1.

[193] Kai Sedgwick. *No, Visa Doesn't Handle 24,000 TPS and Neither Does Your Pet Blockchain*. Online: Accessed on 28-10-2020. https://news.bitcoin.com/no-visa-doesnt-handle-24000-tps-and-neither-does-your-pet-blockchain/. Apr. 2018.

[194] Jeremy Singer-Vine, Jason Leopold, Anthony Cormier, John Templon, Scott Pham, Richard Holmes, Azeen Ghorayshi, Michael Sallah, Tanya Kozyreva, and Emma Loop. "We Got Our Hands On Thousands Of Secret Documents. Let's Break Them Down." In: *BuzzFeed News* (2020). Online: Accessed on 28-10-2020. URL: https://www.buzzfeednews.com/article/jsvine/fincen-files-explainer-data-money-transactions.

[195] João Sousa and Alysson Bessani. "From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation". In: *Proceedings of the 2012 Ninth European Dependable Computing Conference*. EDCC '12. USA: IEEE Computer Society, 2012, 37–48. ISBN: 9780769546711. DOI: 10.1109/EDCC.2012.32. URL: https://doi.org/10.1109/EDCC.2012.32.

[196] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. "BitIodine: Extracting Intelligence from the Bitcoin Network". In: *Financial Cryptography and Data Security*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Springer Berlin Heidelberg, 2014, pp. 457–468.

[197] Statistica. *Number of auto repair and maintenance establishments in the U.S.* Online: Accessed on 02-01-2021. https://www.statista.com/statistics/436416/number-of-auto-repair-and-maintenance-shops-in-us/.

[198] Emil Stefanov and Elaine Shi. "FastPRP: Fast Pseudo-Random Permutations for Small Domains". In: *IACR ePrint Archive* (2012), p. 254. URL: https://eprint.iacr.org/2012/254.pdf.

[199] Kai Stinchcombe. *Ten years in, nobody has come up with a use for blockchain.* Online: Accessed on 28-10-2020. https://hackernoon.com/ten-years-in-nobody-has-come-up-with-a-use-case-for-blockchain-ee98c180100. 22 December 2017.

[200] Fran Strajnar. "Banking The Unbanked – Catalyst For Bitcoin's Mass Adoption?" In: *Brave NewCoin* (2014). Online: Accessed on 28-10-2020. URL: https://bravenewcoin.com/insights/banking-the-unbanked-catalyst-for-bitcoins-mass-adoption.

[201] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. "Scalable bias-resistant distributed randomness". In: *IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 444–460.

[202] Nick Szabo. "Smart Contracts: Building Blocks for Digital Markets". In: *Extropy* 16 (1996).

[203] *TaintChain - GitHub*. Online: Accessed on 28-10-2020. https://github.com/TaintChain.

[204] The Economist. *American tech giants are making life tough for startups.* Online: Accessed on 28-10-2020. https://www.economist.com/business/2018/06/02/american-tech-giants-are-making-life-tough-for-startups. 2018.

[205] The Guardian. *Twitter hack: US and UK teens arrested over breach of celebrity accounts.* Online: Accessed on 28-10-2020. https://www.theguardian.com/technology/2020/jul/31/twitter-hack-arrests-florida-uk-teenagers. July 2020.

[206] The Linux Foundation. *Linux Foundation Unites Industry Leaders to Advance Blockchain Technology.* https://www.linuxfoundation.org/news-media/announcements/2015/12/linux-foundation-unites-industry-leaders-advance-blockchain. 2015.

[207] Peter Thiel and Blake Masters. *Zero to One: Notes on Startups, or How to Build the Future.* Crown, 2014. ISBN: 978-0-8041-3930-4. URL: https://books.google.co.uk/books?id=ZH4oAwAAQBAJ.

[208] Tom Elvis Jedusor. *Mimblewimble.* Online: Accessed on 28-10-2020. https://docs.beam.mw/Mimblewimble.pdf. July 19, 2016.

[209] TRIE. *TODA Introduction.* Online: Accessed on 28-10-2020. https://engineering.todaq.net/toda_brief_intro.pdf.

[210] TRIE. *TODA POP Structure.* Online: Accessed on 28-10-2020. https://engineering.todaq.net/todapop.pdf.

[211] European Union. *PE CONS 72/17: Directive of the European Parliament and the Council amending Directive 2015/849 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing, and amending Directives 2009/138/EC and 2013/36/EU*. May 12 2018.

[212] Ann Marie Utratel. *Decentralising the web: The key takeaways*. Online: Accessed on 20-12-2020. https://blog.p2pfoundation.net/decentralising-the-web-the-key-takeaways/2018/09/14.

[213] Jo Van Bulck, Frank Piessens, and Raoul Strackx. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution". In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018.

[214] David Vorick. *The State of Cryptocurrency Mining*. Online: Accessed on 28-10-2020. https://blog.sia.tech/the-state-of-cryptocurrency-mining-538004a37f9b. May 13 2018.

[215] Alex de Vries. "Bitcoin's Growing Energy Problem". In: *Joule v 2 no 5 p801–805* (16 May 2018).

[216] Orla Ward and Sabrina Rochemont. "Understanding Central Bank Digital Currencies (CBDC)". In: *Institute and Faculty of Actuaries* (2019). Online: Accessed on 06-12-2020. URL: https://www.actuaries.org.uk/system/files/field/document/Understanding\%20CBDCs\%20Final\%20-\%20disc.pdf.

[217] Chris Welch. "Google begins shutting down its failed Google+ social network". In: *The Verge* (2019). Online: Accessed on 28-10-2020. URL: https://www.theverge.com/2019/4/2/18290637/google-plus-shutdown-consumer-personal-account-delete.

[218] Christian Wentz. *Introducing Gradient*. Online: Accessed on 28-10-2020. https://medium.com/gradient-tech/introducing-gradient-715e11be685b. Aug. 2018.

[219] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum Project* 151.2014 (2014), pp. 1–32.

[220] Yahoo News. *World Bank rejects El Salvador request for help in adopting bitcoin*. Online: Accessed on 20-06-2021. https://news.yahoo.com/world-bank-rejects-el-salvador-174415765.html. 2021.

[221] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. "RapidChain: Scaling Blockchain via Full Sharding". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2018, pp. 931–948.

[222] *ZCash*. Online: Accessed on 28-10-2020. https://z.cash/.

[223] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert van Renesse. "REM: Resource-Efficient Mining for Blockchains". In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 179. URL: http://eprint.iacr.org/2017/179.

[224] Nils Zimmermann. "Backing up the history of the internet in Canada to save it from Trump". In: *TechCrunch* (2016). Online: Accessed on 23-10-2020. URL: https://techcrunch.com/2016/12/08/backing-up-the-history-of-the-internet-in-canada-to-save-it-from-trump/.

[225] Nils Zimmermann. "Network effects helped Facebook win". In: *Deutsche Welle* (2017). Online: Accessed on 23-10-2020. URL: https://www.dw.com/en/network-effects-helped-facebook-win/a-40418818.

[226] Shoshana Zuboff. *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*. 1st. Profile Books, 2018. ISBN: 9781610395694.

A

Supplementary pseudocode

In this section, I present the pseudocode for several algorithms used in this thesis in the interests of reproducability.

A.1 Blockchain archival

Algorithm 9 Pseudocode for creating an archival sub-chain

```
1: procedure CreateSubchain(signatories{}, currentChain, quorum)
2:     regenReq ← ∅
3:     regenReq ← {currentChain.regenNum + 1, currentTS, currentChain.subBlockNum}
4:     regenrReq ← {currentChain.chainBlockNum, signatories{}}
5:     sendToNetwork(regenReq)
6:     sigs{} ← ∅
7:     sigs{} ← receiveFromNetwork()
8:     regenReq ← sigs{}
9:     if |sigs| < quorum then
10:        return false
11:    transBlock ← {regenReq, tx}
12:    transBlock ← createBlock(transBlock)
13:    sendToNetwork(transBlock)
14:    runConsensus()
15:    if currentChain.block ≠ transBlock then
16:        return false
17:    regenBlock ← ∅
18:    regenBlock ← {hash(transBlock), signatories{}}
19:    regenBlock ← createBlock(regenBlock)
20:    sendToNetwork(regenBlock)
```

```
21:     sigs{} ← ∅
22:     sigs{} ← receiveFromNetwork()
23:     if |sigs| < quorum then
24:         return false
25:     if currentChain.block ≠ transBlock then
26:         return false
27:     return true
```

A.2 Modifiable blockchain storage

As described in § 4.4, the protocol for creating a modification block is similar to the one for creating a transition block shown in Algorithm 9. Here, I present the pseudocode for validating a chain containing modifications.

Algorithm 10 Pseudocode for validating a modified chain

```
1: procedure ValidateChain(chain)
2:     currBlock ← chain.genesis
3:     pending ← ∅
4:     while currBlock.num < chain.current.num do
5:         nextBlock ← getBlock(chain, currBlock.num + 1)
6:         if ∃MT|MT ∈ currBlock.txs then //MT stands for modification transaction
7:             for all MT ∈ currBlock.txs do
8:                 bool ← validate(MT)
9:                 if bool ≠ true then
10:                    return false
11:                if MT.blockHash ∈ pending then
12:                    pending \ MT.blockhash
13:                else
14:                    return false
15:        if hash(currBlock) ≠ nextBlock.prevHash then
16:            pending ← currBlock
17:        currBlock ← nextBlock
18:    if pending ≠ ∅ then
19:        return false
20:    return true
```

A.3 Robust Round Robin

Here I present the VerifyBranch algorithm used as part of the chain validation process in RRR.

Algorithm 11 VerifyBranch

```
1: procedure VerifyBranch(currentBranch)
2:     prevBlock ← Genesis(currentBranch)
3:     iterBlock ← Next(prevBlock, currentBranch)
4:     currentBlock ← Top(currentBranch)
```

```
    while iterBlock ≠ CurrentBlock do
        if Hash(prevBlock) ≠ prevHash(iterBlock) then
            return false
        iterLeader ← Leader(iterBlock)
        if iterLeader ∉ SelectCandidates(currentBranch[0, iterBlock]) then
            return false
        intent ← GetIntent(iterBlock)
        if GetTxHash(intent) ≠ Hash(GetTx(iterBlock)) then
            return false
        counter ← 0
        for all endorsement ∈ Endorsements(iterBlock) do
            if VerifyEndorsement(endorsement,iterBlock) ≠ true then
                return false
            counter ← counter+1
        if counter < q then
            return false
        if VerifyVRF(iterBlock) ≠ true then
            return false
        for all enrolment ∈ Enrolments(iterBlock) do
            if verify(enrolment) ≠ true then
                return false
        prevBlock ← iterBlock
        iterBlock ← Next(iterBlock, currentBranch)
    return true
```

A.4 Cambium modules

Here I present the pseudocode for the RESOLVE-PEER-MESSAGES and RESOLVE-ALLY-MESSAGES algorithms shown in Figure 6.7.

Algorithm 12 Resolve Peer Messages

```
procedure ResolvePeerMessages(Peers X_l, myMP_{l-1})
    peerQueue{} ← ∅
    toSend{} ← ∅
    while timeout() ≠ 1 do
        tempMP ← ReceiveMP()
        if tempMP.sender ∉ X_l then
            continue
        if isValid(tempMP) then
            peerQueue ← tempMP
    toSend ← peerQueue ∪ myMP_{l-1}
    return toSend
```

Algorithm 13 Resolve Ally Messages

```
1: procedure ResolveAllyMessages(Allies A_l)
2:     allyQueue{} ← ∅
3:     AP ← hole
4:     while timeout() ≠ 1 do
5:         tempPacket ← ReceiveAllyPacket()
6:         if tempPacket.sender ∉ A_l then
7:             continue
8:         if isValid(tempPacket) then
9:             allyQueue ← allyQueue ∪ tempPacket
10:    allyAPMap{}{} ← ∅, {hole}
11:    APCountMap{}{} ← {hole}, {0}
12:    advAllies{} ← ∅
13:    for all p ∈ allyQueue do
14:        for all a ∈ p do
15:            if a.sender ∈ advAllies then
16:                continue
17:            if allyAPMap[a.sender] = ∅ then
18:                allyAPMap[a.sender] ← a
19:                APCountMap[a] + +
20:            else
21:                tempPacket ← allyAPMap[a.sender]
22:                if tempPacket ≠ a then
23:                    APCountMap[tempPacket] − −
24:                    advAllies ← a.sender
25:    maxCount ← GetMaxCount(APCountMap)
26:    if maxCount ≥ (2/3|A_l|) then
27:        AP ← MaxCountPacket(APCountMap)
28:    return AP
```

Algorithm 14 Get ally algorithm for a node with cycle index i at merge level l.

```
1: procedure GetAlly(i, l)
2:     l2 ← 2^l
3:     offset ← log_2(k) + l
4:     x ← ShiftLeft(i, offset)
5:     p ← ShiftLeft((i − ShiftLeft(x, offset) − (i mod l2)), l)
6:     zp ← (i & (l2 − 1)) − p
7:     z ← ((zp mod l2) + l2) mod l2
8:     if (x mod 2) = 0 then
9:         x ← x + 1
10:    else
11:        x ← x − 1
12:    a ← ((p + z) & 2^(l−1)) | ShiftRight(p, l) | ShiftRight(x, (l + log2(k)))
13:    return a
```

B

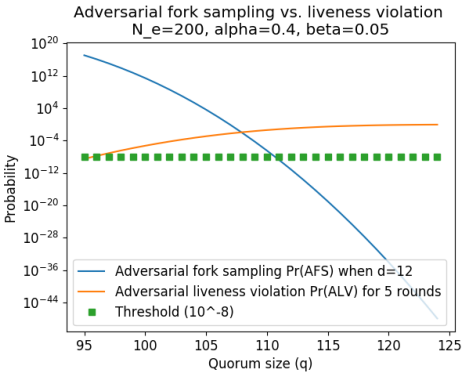
Additional parameter values for RRR

In the interest of thoroughness, I will now extend the analysis from § 5.5.1 to consider further values for Robust Round Robin's system parameters.

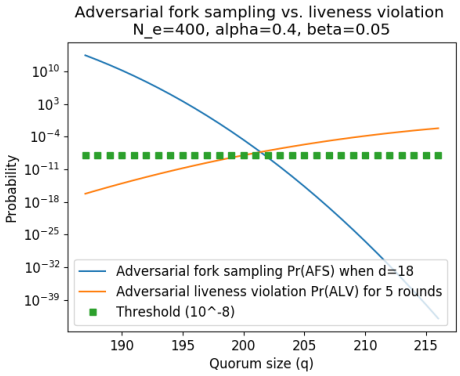
We start by examining the effect of larger α, i.e. cases where the adversary controls a larger fraction of all identities in the system. As can be see from Figure B.1a, when $\alpha = 0.4$ and the fraction of non-responsive identities remains as before ($\beta = 0.05$), our previous example value of $N_e = 200$ endorsers leads to no quorum value q that would prevent forks at the same depth $d = 12$ without reducing liveness. To handle such cases, we must either increase the endorser committee size or the maximum depth of forks. Figure B.1b shows that increasing the size of the endorser committee moderately to $N_e = 400$ and simultaneously increasing the depth of the forks to $d = 18$ allows us to find a quorum value $q = 202$ that provides good security and liveness at the same time.

Tolerating such stronger adversaries ($\alpha = 0.4$) becomes significantly easier in our solution when the connectivity between consensus nodes is better. As shown in Figure B.1c, if we assume the fraction of non-responsive identities in each round to be smaller ($\beta = 0.01$), it is possible to find a suitable quorum size ($q = 104$) that provides prevents forks at depth $d = 12$ using $N_e = 200$ endorsers. Figure B.1d shows that in principle stronger adversaries ($\alpha = 0.4$) can be handled without increasing the committee size ($N_e = 100$) by only increasing the maximum depth of forks ($d = 22$) which would mean a latency of almost two minutes.

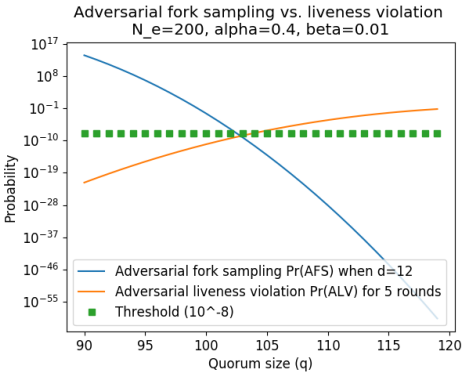
Allowing longer adversarial liveness violation enables shallower forks. Next, we consider the case where we allow the adversary to prevent block creation for 10 rounds. Figure B.1e shows that when $\alpha = 0.4$, $\beta = 0.01$ and we demand $N_e = 200$ endorsers, there is a quorum value $q = 111$ that prevents forks at depth $d = 7$ (in contrast to the



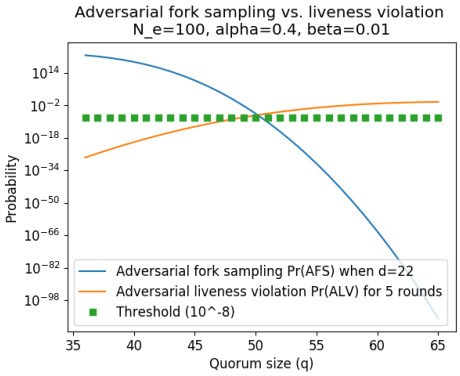
(a) When $\alpha = 0.4$ and $\beta = 0.05$ using $N_e = 200$ endorsers there is no quorum value q that prevents forks at depth $d = 12$ and provides good liveness.



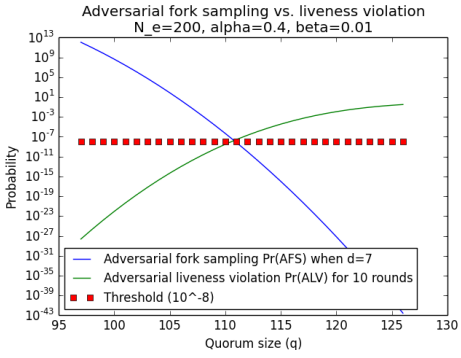
(b) When $\alpha = 0.4$ and $\beta = 0.05$ using $N_e = 400$ endorsers there is a quorum value $q = 202$ that prevents forks at depth $d = 18$ and ensures good liveness.



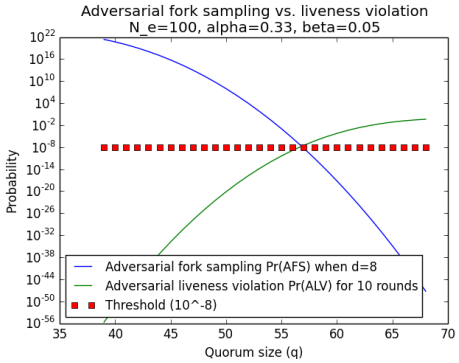
(c) When $\alpha = 0.4$ and $\beta = 0.01$ using $N_e = 200$ endorsers there is a quorum value $q = 104$ that prevents forks at depth $d = 12$ and provides good liveness.



(d) When $\alpha = 0.4$ and $\beta = 0.01$ using $N_e = 100$ endorsers there is a quorum value $q = 51$ that prevents forks at depth $d = 22$ and provides good liveness.



(e) When $\alpha = 0.4$ and $\beta = 0.01$ using $N_e = 200$ endorsers there is a quorum value $q = 111$ that prevents forks at depth $d = 7$, when adversarial liveness violation is increased to 10 rounds.



(f) When $\alpha = 0.33$ and $\beta = 0.05$ using $N_e = 100$ endorsers there is a quorum value $q = 57$ that prevents forks at depth $d = 8$, when adversarial liveness violation is increased to 10 rounds.

Figure B.1: Security versus liveness with additional example parameter values.

previous value $d = 12$). Similarly, Figure B.1f shows that when $\alpha = 0.33$ and $\beta = 0.05$, using $N_e = 100$ endorsers there is a quorum value $q = 57$ that prevents forks at depth $d = 8$.

We conclude that our proposal can be adapted to handle various assumptions regarding the strength of the adversary and connectivity between the system participants. While it is best suited to scenarios where the adversary controls up to one-third of all identities, stronger adversaries can be tolerated by using larger endorser committees or by reducing liveness guarantees. The exact tuning of parameter values should therefore be done based on the application and on the expected levels of adversarial activity.