

EXHIBIT 1

Expert Report

Jeremy Clark, Ph.D., P.Eng.

Contents

1	Introduction	4
1.1	Assignment	4
1.2	Qualifications	4
1.3	Facts, data, and documents relied upon	5
1.4	Principles and methods	6
1.5	Disclaimers	6
2	Summary of Opinions	7
3	Overview of technology	12
3.1	Distributed systems	12
3.2	Bitcoin	14
3.3	Ethereum	19
3.4	The XRP Ledger	24
4	Analysis and opinions	31
4.1	The XRP Ledger consensus protocol requires validators to agree on a list of trusted validators and use of the list published by <i>Ripple Labs</i> is a de facto requirement	31
4.2	The XRP Ledger is a distributed system but <i>Ripple Labs</i> remains the root of trust	33

4.3	<i>Ripple Labs</i> is a gatekeeper to full participation	35
4.4	XRP distribution favors <i>Ripple Labs</i>	35
4.5	Validators in the XRP Ledger require external incentives	37
5	Declaration	38
A	The Bitcoin protocol and implementation	41
A.1	Cryptographic properties and primitives	42
A.2	The blockchain data structure	46
A.3	Transactions	47
A.4	Nakamoto consensus	50
A.5	Issuance and fees	58
B	The Ethereum protocol and implementation	60
B.1	Smart Contracts	60
B.2	Primitives and data structures	61
B.3	Ethereum consensus	61
B.4	Issuance and fees	64
C	The XRP Ledger protocol and implementation	67
C.1	Overview of the XRP Ledger and XRP	67
C.2	Primitives and data structures	68
C.3	XRP Ledger consensus	71
C.4	Transactions	81
C.5	Issuance and fees	82
D	Complete List of Materials Considered	83
D.1	Materials	83
D.2	Articles	83
D.3	Class Certification Documents	83
D.4	Public Court Filings	83

D.5	Discovery Responses and Objections	83
D.6	Depositions and Exhibits	84
D.7	Document Production	85
D.7.1	a. As follows	85
D.7.2	b. SEC v. Ripple – Depositions and Exhibits	89
D.7.3	c. SEC Investigative Testimonies and Exhibits	91
D.8	Other	91
E	Curriculum Vitae	92

1 Introduction

2 1.1 Assignment

3 I have been engaged by Lead Plaintiff Bradley Sostack (“Plaintiff”), through his counsel,
 4 to provide expert testimony in the case captioned *In re Ripple Labs Litigation*, Case No.
 5 4:18-cv-06573, pending in the United States District Court for the Northern District of
 6 California. Lead Plaintiff has retained me to independently analyze and opine on the XRP
 7 Ledger protocol and the differences between it and the Bitcoin and Ethereum protocols.

8 1.2 Qualifications

9 I am an associate professor at the Concordia Institute for Information Systems Engineering
 10 (CIISE) at Concordia University in Montreal, QC, Canada. I hold the NSERC/Raymond
 11 Chabot Grant Thornton/Catallaxy Industrial Research Chair in Blockchain Technologies. I
 12 hold a Ph.D. in Computer Science from the University of Waterloo, awarded in 2011 with
 13 the university’s Alumni Gold Medal, and in the discipline of applied cryptography. I am a
 14 professional engineer (P.Eng.) with the Professional Engineers of Ontario (PEO).

15 I have over 10 years of research expertise in digital assets and blockchain, and even more
 16 experience with related areas of cryptography. My expertise includes material knowledge of
 17 Bitcoin and Ethereum.

18 Bitcoin was described in late 2008 and released as software in early 2009. I began pursu-
 19 ing academic research on Bitcoin in 2011 and have published over 20 peer-reviewed papers
 20 on Bitcoin, Ethereum, digital assets, blockchain technology, and similar topics. Research
 21 highlights include CommitCoin [11], one the earliest academic works on Bitcoin published in
 22 *Financial Cryptography and Data Security* (Conference Rank:¹ A) in 2012; our 2015 system-
 23 ization of knowledge on Bitcoin and blockchain research [4] published in *IEEE Symposium*
 24 *on Security and Privacy* (Conference Rank:² A+; citations 1000+³); and our 2017 article

¹CORE Conference portal, Feb. 2023.

²CORE Conference portal, Feb. 2023

³Google Scholar, Feb. 2023.

on Bitcoin’s academic pedigree [20] published in the *Communications of the ACM* (Journal Impact Factor:⁴ 14.065; downloads: 300K+⁵).

I have testified on digital assets to the Standing Senate Committee on Banking, Commerce and Economy of the Senate of Canada (April 3, 2014), and to the Standing Committee on Finance of the House of Commons of Canada (March 27, 2018). I have given over 50 presentations on digital assets to companies, government agencies, law enforcement, pension plans, and academic groups.

My attached CV contains further evidence of my expertise and research impact in these subjects.

1.3 Facts, data, and documents relied upon

To prepare this report, I read technical reports on the XRP Ledger, reviewed the technical information distributed online by Ripple Labs, Inc. (*Ripple Labs*) and the *XRP Ledger Foundation*, examined the source-code made available on GitHub, and reviewed the academic literature on the XRP Ledger. I also reviewed technical documents produced by *Ripple Labs* in this litigation and the transcripts of depositions in this litigation and the SEC Action, including the transcripts of the depositions of the Chief Technical Officer (“CTO”) of *Ripple Labs*, David Schwartz. I also reviewed the technical details of Bitcoin and Ethereum, as necessary.

When I draw directly on the documents I considered, I will provide a citation inline. Additional documents which I considered are itemized in Appendix D. These include deposition transcripts and documents that were provided to me. The documents in Appendix D were not used as a primary source for any facts or opinions in this report.

Bitcoin, Ethereum, and the XRP Ledger are evolving protocols with substantial changes and modifications made over time. If not otherwise stated, I make assertions that are generally true across the lifespan of the respective system. In cases where I make an assertion

⁴Clarivate / Web of Science, Feb. 2023.

⁵293 530 Queue + 41 257 CACM, ACM Digital Library, Feb. 2023

1 with a citation, it is asserted to be true as of the publication date of the cited document.

2 **1.4 Principles and methods**

3 For purposes of this report, I reviewed portions of the XRP Ledger code, known as `rippled`.
4 I relied on the assertions made in the documents I cite and for critical facts, I examined
5 the software using a time-box method to confirm the critical fact before including it in this
6 report.

7 Based on all the above, I formed opinions on what points of commonality and what points
8 of difference exist between the XRP Ledger, Bitcoin, and Ethereum.

9 **1.5 Disclaimers**

10 For serving as an expert witness, I am remunerated by *Susman Godfrey L.L.P.* at \$175 CAD
11 per hour (before University fees). My compensation is not dependent upon me reaching any
12 specific conclusion or opinion.

13 All opinions are mine and do not necessarily reflect those of Raymond Chabot Grant
14 Thornton, Catallaxy, the Natural Sciences and Engineering Research Council of Canada
15 (NSERC), or Concordia University.

2 Summary of Opinions

I have been asked by counsel for Lead Plaintiff to analyze *Ripple Labs*' role in the creation the creation and functioning of the XRP Ledger protocol and to compare the XRP Ledger protocol to Bitcoin and Ethereum, the two most prominent blockchain technologies. For the purposes of this report, I specifically focus on the creation and distribution of the native digital assets for each protocol (XRP, BTC, and ETH) and each protocol's mechanism for validating transactions, otherwise referred to as its "consensus mechanism," which is performed by a set of entities called validators, rather than a single entity. The idea of relying on multiple validators is the defining feature of blockchain technology, and distinguishes it from traditional online systems that are controlled or operated by a single entity.

A blockchain's consensus mechanism is a critical component that ensures the reliability, integrity, and consistency of data transactions. The consensus mechanism is responsible for validating transactions, resolving potential conflicts, and ensuring that all validators within the system agree on the order and legitimacy of transactions. It is this aspect that provides the core trust and security feature of blockchain technology and makes the network resistant to fraudulent activity. A blockchain's consensus mechanism is the primary means of determining who validates transactions and how this validation process works. A blockchain's consensus mechanism thus provides critical insight into the degree of control any particular entity has over a protocol.

The creation and distribution of a blockchain protocol's native asset also plays a vital role in establishing the protocol's internal incentives. Firstly, the native asset is often a primary source of revenue for validators, as the validators are rewarded with newly created assets. This reward incentivizes validators to undertake the task of running a server and paying networking costs, thereby securing the system. Transaction fees can also be paid from users to validators to further compensate them. The initial allocation and method for any further distribution of the native asset can provide insight into the degree of control a particular entity has over a protocol.

Based upon my analysis of Bitcoin, Ethereum, and the XRP Ledger, I reach five con-

clussions, which are laid out in greater detail in the conclusion section (Section 4). In this subsection, I provide a high-level summary for each finding. The main takeaway is that the XRP Ledger differs in material ways from both Bitcoin and Ethereum. The differences result in the XRP Ledger having been, and continuing to be, subject to a greater influence and control by one entity, *Ripple Labs*, than other participants. There is no equivalent entity which exerts a similar level of influence or control over Bitcoin or Ethereum.

Opinion 1: The XRP Ledger consensus protocol requires validators to agree on a list of trusted validators and use of the list published by *Ripple Labs* is a de facto requirement

The XRP Ledger Consensus Mechanism requires that every validator have a list of other validators that they trust to provide truthful and timely decisions. As explained in Section 4.1, the XRP Ledger operates best when all validators use exactly the same trusted validator list as each other. There is an academic consensus that the properties of the XRP Ledger degrade when validators use lists that differ by 10% or more, with some models suggesting that some properties degrade with as little as a single different validator. In any case, the consistent operation of the XRP Ledger requires that all or almost all validators agree to adopt effectively the same list of trusted validators.

To coordinate validators, *Ripple Labs* provides a “recommended” list which it refers to as the “default Unique Node List (dUNL).” The *Ripple Labs* validator list is in fact a preset default in the original software implementation of the XRP Ledger (the `rippled` software). The `rippled` software warns validators not to change the preset list from *Ripple Labs* and online documentation from *XRP Ledger Foundation* reinforces this warning.

As the original default distributor of such a list in the original XRP Ledger software client, utilizing the *Ripple Labs* validator list is a *de facto* requirement for anyone using the XRP Ledger. The *Ripple Labs* validator list is not just used by validators—it is an inherent assumption in the data displayed by exchanges, websites, user software, and visualization tools. While the entity maintaining the list does not have to be *Ripple Labs* specifically, *Ripple*

Labs has played this role historically, and some central entity must maintain coordination between validators.

If one or a few validators decided not to utilize the preset list provided by *Ripple Labs*, these nodes could desynchronize from the rest of the network. The only tenable way to smoothly transition control from *Ripple Labs* to another entity is for *Ripple Labs* to choose to do this itself.

Opinion 2: the XRP Ledger is a distributed system but *Ripple Labs* remains the root of trust

Ripple Labs maintains, in practical terms, the ability to modify or distribute a new list of its choosing (*i.e.*, with only itself on the list), resulting in a hierarchy of trust where it is the trust anchor for the whole system. An inside attack or data breach within *Ripple Labs* (compromising its cryptographic signing key) would suffice to cause a catastrophic protocol failure in the XRP Ledger that would require coordinated human intervention from nearly all the XRP Ledger participants to repair.

Opinion 3: *Ripple Labs* is a gatekeeper to full participation

If a new validator (or miner in Bitcoin terminology) joins Bitcoin or Ethereum for the first time, they are able to immediately contribute to the consensus mechanism without any other entity on the network knowing who they are. This is because their contribution will be judged by “the work itself,” not by “who they are” or “who they know.” Bitcoin operates without a list of a validators (and this is in fact, its novel contribution) while Ethereum currently operates with an open list that anyone can join or leave without authorization from any company or entity.

In contrast, contributing meaningfully to the XRP Ledger consensus mechanism is based on “who you know,” namely whether *Ripple Labs* has placed you on the preset list of validators (in which case, your contributions will be considered) or not (in which case, your contributions will be ignored). From January 2013 through June 2018, this list contained

only nodes controlled by *Ripple Labs*. Since July 2018, *Ripple Labs* has diversified the list to include a mix of validators. However, *Ripple Labs* still controls who is placed on, or removed from, this list. Some of the validators on the list are entities that have received funding from *Ripple Labs*. This positions *Ripple Labs* as a more influential entity than any equivalent entity in Bitcoin or Ethereum.

Opinion 4: XRP distribution favors *Ripple Labs*

Bitcoin, Ethereum, and the XRP Ledger each have an intrinsic asset that can be transferred between users, respectively called BTC, ETH, and XRP. How this asset comes into circulation and how it is used to incentivize validators differs between Bitcoin, Ethereum, and the XRP Ledger.

In Bitcoin, all BTC in circulation was created slowly over time and paid to a miner for the job of validating transactions as a part of the consensus process. As anyone can become a miner at any time, the issuance of BTC does not favor any one company, entity, or individual. By contrast to BTC, the Ethereum foundation conducted an initial offering of ETH (ETH was offered for sale in exchange for BTC), which was conducted before Ethereum was completed and deployed. During this offering, newly created ETH was also given to the foundation and the founders/developers affiliated with it. After this initial sale and distribution, all subsequent ETH that has come into circulation is paid as a reward to a validator for validating transactions, as in Bitcoin.

In contrast, all XRP was issued at the initialization of the XRP Ledger to *Ripple Labs* or its founders. No subsequent XRP will come into circulation. Validators are not compensated in XRP (see opinion 5). This allocation of all XRP to *Ripple Labs* and individuals closely associated with it is a material difference between the XRP Ledger and Bitcoin, where all BTC is issued directly to miners. It also differs, to a lesser degree, from Ethereum, where a portion of ETH is paid to validators.

As the largest shareholder of XRP, *Ripple Labs* is invested in advancing the functionality of the XRP Ledger. Nearly all of the significant contributors to the *rippled* software are

1 either current or former *Ripple Labs* employees or have some other financial relationship with
 2 *Ripple Labs*.

3 **Opinion 5: Validators in the XRP Ledger require external incentives**

4 Validators and miners in Bitcoin and Ethereum are “for profit” participants—they profit
 5 from the new issuance of BTC and ETH over time, as well as transaction fees. These rewards
 6 provide an incentive for acting as validators on the Bitcoin and Ethereum protocols. The
 7 growth in the numbers of validators and miners on Bitcoin and Ethereum is thus the result of
 8 a free market and internal protocol incentives. The XRP Ledger is materially different because
 9 no XRP is provided to validators as a reward or payment for participating in the consensus
 10 process and transaction fees are burned instead of being given to validators. XRP Ledger
 11 validators “work for free.” A consequence is a relatively small number of validators—35
 12 recommended validators in XRP⁶ compared to $\approx 600\,000$ validators in Ethereum.⁷ Validators
 13 in the XRP Ledger are recruited with an external incentive to participate (*e.g.*, alignment with
 14 the XRP industry, capital investments from *Ripple Labs* or *XRP Ledger Foundation*, grants
 15 to academic institutions, altruism, *etc.*), whereas participation in Bitcoin and Ethereum is
 16 driven by their internal incentives.

⁶“XRP Validator registry,” XRPSCAN, Retrieved Feb–May 2023.

⁷“BeaconScan statistics,” Etherscan, Retrieved Feb–May 2023.

3 Overview of technology

In this section, I provide an overview of the relevant technology and a summary of Bitcoin, Ethereum, and the XRP Ledger. In the appendix (Sections A–C), I provide in-depth technical descriptions of the protocols for Bitcoin (Section A), Ethereum (Section B), and the XRP Ledger (Section C).

3.1 Distributed systems

Many industries have been transformed by the process of digitizing records that were traditionally stored in paper format. In a digital system, a purpose-built computer (called a server) collects and stores the records, performs operations on the records, and is accessible over the internet (called a network connection) by users who may be permissioned to read from the database and/or write to the database. Among its benefits, digitalization often allows faster access and more efficient transactions.

Ownership of assets has been digitized to a large extent. Users access their bank accounts through their bank’s server; banks send inter-bank payments through the Federal Reserve’s server; users and their brokers trade stocks on NYSE or NASDAQ servers; and stock ownership is tracked on the Depository Trust Company server.

Digitization requires trust that the digital records are accurate. Often times this trust is placed in one central entity or authority, which operates the server(s) (*e.g.*, bank servers). This operator performs critical and necessary technical tasks like maintaining the infrastructure, ensuring continual up-time, and processing requests to access the database. But the operator’s control of the server also means that the operator could potentially overwrite, remove, or even lie about digital records. The operator could also ban competitors, or could undertake other conduct that is not in the interests of those relying on the server. In sum, while delegating control over digital records to a single entity is simple, it is accompanied by significant risks that the entity will abuse that trust⁸ or will itself be exploited by malicious

⁸*cf.* fraud at the FTX exchange with an estimated \$8B USD in losses.

1 third parties.⁹

2 Resolving the issue of trusting a single server has occupied academics since the 1970s and
3 1980s, who proposed distributed systems where a group of independent servers, operated by
4 mutually distrustful entities, work together on storing digital records and operating on the
5 records as if they were a single server. A key ingredient to a distributed system is its
6 “consensus mechanism,” which ensures that no action is taken until a majority (or greater)
7 of the independent servers agree it is the correct action. Modern distributed systems are
8 designed to be resilient to a limited set of servers leaving the system (“aborts”), generating
9 errors (“faults”), or acting maliciously (“Byzantine faults”). A consensus mechanism is
10 described as being Byzantine Fault Tolerant (BFT) if the protocol will still work (“liveness”)
11 correctly (“safety”) if the total number of servers that abort, fault, and/or act maliciously
12 remains under a determined threshold (such as 25% or 33%).

13 An issue with many BFT protocols is that they do not necessarily eliminate the role of a
14 single trusted server operator because a traditional BFT system must maintain a list of the
15 servers who operate the system. Someone must control this list and that entity is still in a
16 powerful position—the list controller can remove all independent servers and replace them
17 with new servers belonging to the list controller, hijacking the system at any time.

18 A long-standing academic question was whether a distributed system could be built
19 without a list of operators. Put another way, the question was whether a distributed system
20 could be built so that anyone on the internet could decide to become a server at any time and
21 join the network and participate in the consensus mechanism without needing permission
22 from a single trusted service operator.

23 Bitcoin was the first widely used deployment of a consensus mechanism (called Nakamoto
24 Consensus) that completely replaces a single, trustworthy operator with a set of independent
25 servers. Bitcoin is designed to run on the internet and since the internet contains hostile
26 entities, the system is designed to run correctly even when a fraction of the servers are
27 malicious and try to attack the system.

⁹*cf.* exploits at the Ronin network, Poly network, and BNB Bridge, each with over \$500M USD in losses.

1 A note on terminology: Nakamoto Consensus is a type of BFT protocol in that it is able
2 to continue functioning correctly if the total number of servers that abort, fault, and/or act
3 maliciously remains under a determined threshold (*e.g.*, 51% percent in Bitcoin). However
4 Nakamoto Consensus did not originate from the same academic stream as BFT protocols
5 and the Bitcoin whitepaper does not cite or use the language of the BFT literature, thus it
6 has become conventional to use the term “BFT protocol” to refer to pre-Bitcoin distributed
7 systems that use a list manager (or post-Bitcoin systems based on them). Distributed systems
8 (which do protect against byzantine faults) based on Nakamoto Consensus are generally
9 called blockchain systems. Systems that merge ideas from both are sometimes called “BFT
10 blockchains.” This language can vary widely across the literature and community. For our
11 purposes, when we say “traditional BFT protocol” or simply “BFT protocol,” we are referring
12 to distributed systems that operate with a list of trustworthy servers; a list established and
13 maintained by some entity in the system.

14 We also say that Nakamoto Consensus provides an “open” and “permissionless” system.
15 Open means that anyone on the internet with the appropriate technical capabilities is invited
16 to join the system. Permissionless means that joining the system (and leaving the system)
17 does not require the authorization of an entity in the system. The protocol itself may impose
18 rules about how and when servers can join but ultimately, a permissionless system will let
19 anyone join eventually if they meet the in-protocol prerequisites. For example, in Ethereum,
20 described below, validators can join if they run validation software, lock up ETH as a fidelity
21 bond that they will follow the protocol, and wait in a queue to be added to the set of active
22 validators.

23 3.2 Bitcoin

24 Bitcoin was designed to address a single task: maintaining a system for owning and trans-
25 ferring a digital asset called BTC. Unlike a traditional financial system that is operated by a
26 single trustworthy entity, Bitcoin is operated by an open set of participating servers, called

1 miners,¹⁰ that can be anyone on the internet running software compatible with the Bitcoin
2 protocol.

3 In order for miners to check the validity of new transactions, they need to reference a
4 log of all past transactions. A consequence of the open and permissionless nature of Bitcoin
5 is that all transactions are made public. Every miner begins by obtaining a list of all past
6 transactions (from others on the network called “archive nodes”), processes them, and works
7 over time to ensure their copy of new transactions always remains identical to that of the
8 other miners (“consensus”). For technical reasons, there are occasions where a miner might
9 temporarily maintain a different set of new transactions than its peers, however Bitcoin has
10 mechanisms to resolve (“re-organize”) these divergences (temporary “forks”) over the span
11 of less than 1 hour (6 “blocks”).

12 **Bitcoin blockchain.** For performance reasons, miners do not try to finalize new trans-
13 actions one-by-one. Instead they group together (often thousands of) transactions into a
14 “block” of transactions. Blocks of transactions are not free-floating blocks that can be
15 assembled in any particular order. Instead miners proposing a new block must, in their
16 proposal, identify which existing block (“previous block”) they are extending with their new
17 block. Likewise, the specified previous block itself has specified its own previous block.
18 These backward references enforce an ordered “chain” of blocks (the “blockchain”) that go
19 back to the very first block, called the “genesis block,” which is the only part of the chain
20 programmed into the software itself to ensure universal agreement on it. The consequence
21 of storing transactions in a blockchain is that any modification to any transaction anywhere
22 in the chain will necessarily cause every block that follows it to change as well, making any
23 attempts at tampering evident to validators, who will dismiss the modification. Bitcoin has
24 mechanisms to rate-limit the creation of new blocks to once every 10 minutes (on average).

¹⁰Miners act as validators on the Bitcoin network and the two terms are used interchangeably in this section.

Bitcoin accounts. For users, the process of owning BTC begins with setting up the ability to create a “digital signature.” A digital signature is a well-studied primitive from cryptography and digital signatures are in wide use today (*e.g.*, the lock that appears in web browsers beside websites accessed over `https://`). Using software, a user generates two “keys” (or a “key pair”): one key is a secret signing key that is used to generate signatures on any piece of data, and the other is a verification key that can be shared publicly and serves as a unique identifier for the user. The keys are linked together mathematically. Unlike other applications of digital signatures (*e.g.*, `https://`), Bitcoin makes no attempt to link a public key to a user’s real-world identity—it is simply a random-looking number that serves as a pseudonym. Users can create as many key pairs as they want.

To receive BTC for the first time, the receiver can communicate their public key to another user (the “sender”) that already has BTC (where BTC comes from in the first place is described below). Or, more commonly, the receiver will first create a shorter version of their public key, called a “BTC address,” and communicate their address to the sender. The sender will create a new transaction that specifies the receiver’s address (or full public key) as the recipient (“output”) of some portion of BTC. This transaction is broadcast on the network to the miners operating the blockchain. At this time, the miners have never seen this particular BTC address before. Because BTC is open and permissionless, this is acceptable—new users can enter the system at any time without enrollment or permission from any operator. When the transaction is finalized in a block, the user now owns BTC and can transfer it by creating a new transaction and signing the transaction using the private signing key associated with their BTC address.

Nakamoto Consensus. The key component to Bitcoin is the “consensus” mechanism that promotes agreement amongst all miners about the validity of transactions. In determining whether a transaction is valid, miners verify that the sender has enough BTC to complete the transaction, that all BTC in the transaction (“inputs”) have not been spent before, and that all inputs have the appropriate digital signature authorizing their use in the transaction, amongst other details.

1 Bitcoin's solution is not built on a BFT protocol as that would require someone to
2 maintain a list of trusted validators. Nor is it as simple as running a BFT protocol with an
3 open list of servers that anyone can add themselves to. The problem with such an approach
4 is that malicious entities could add themselves to the list a million times and thereby take
5 control of the system. Consensus works through a process similar to voting, and so one
6 malicious entity with millions of fake identities ("sybil identities") can overwhelm the voting
7 process and hijack a BFT system with an open list of servers. Bitcoin solves this problem with
8 "Nakamoto consensus" (named for the creator of Bitcoin) or "proof of work consensus" by
9 asking servers to solve a computational puzzle that is designed to be hard (but not impossible)
10 for computers to solve. The faster a server can solve this puzzle, the more influence they have
11 in the consensus process. If an entity wants to increase their influence within the consensus
12 mechanism by, say, 100 times, they must solve the puzzle 100 times faster and that can only
13 be accomplished by purchasing and running additional computer hardware. The capital
14 costs for a miner to increase their influence over the consensus mechanism discourages and
15 disincentivizes malicious attacks. Generally, increases in influence in Nakamoto Consensus
16 are only achievable through fair means (with some technical caveats discussed in §A.4).

17 Nakamoto Consensus is full of important nuances and details described in §A.4, however
18 I offer an overview here. Assume the latest block of transactions has been seen by every
19 miner on the network and is considered valid by each of them. Each miner will construct a
20 new block of transactions that have been broadcast to the network but are not yet included
21 in the chain. This new block will extend the latest block. Even though the miners all see
22 the same set of transactions, each miner's new block will be unique to that miner—first,
23 because they might include different transactions in a different order, and secondly, because
24 the Bitcoin protocol allows the miner to mint new BTC and collect transaction fees for every
25 transaction in the block (described below). The miner will create a transaction to claim the
26 newly minted BTC and the fees and place it at the start of the block. Since each miner will
27 specify their own unique BTC address as the recipient for this BTC, this transaction will
28 differ in who the recipient is. If at least one transaction in the block differs, the entire block

1 is different, thus each miner's new block is unique to them.

2 It is not enough for a miner to calculate a new block, the miner must also solve a
 3 computational puzzle. The exact puzzle they solve is unique to them because it is based
 4 on their block, which as laid out above is unique to them. For example, if Alice and Bob
 5 are both miners, their puzzles will be the same type of puzzle with the same difficulty, but
 6 the exact puzzle will be different between them, such that a solution to Alice's puzzle is
 7 not a solution to Bob's puzzle (and vice-versa). This prevents another miner from stealing
 8 a solution and using it for themselves. The first miner to find a solution will broadcast
 9 both their block and their solution to the rest of the miners. The rest of the miners will
 10 check the block and solution, and if it is valid, they will mark this block as the latest block,
 11 abandon their current block (as it no longer extends the latest block), build a new block
 12 of transactions that extends the latest block, and begin solving the puzzle instantiated by
 13 their new block. The key idea is that miners always work on extending the 'longest chain'
 14 of blocks and the miner that solves a puzzle is rewarded with BTC (explained below).

15 As Bitcoin operates over the internet amongst anonymous miners who can join and leave
 16 at any time, miners cannot be assumed to altruistic and well-intentioned entities that will
 17 follow the rules of the protocol. Bitcoin is designed to operate correctly even when a signif-
 18 icant fraction of miners are deviant: *e.g.*, buggy software, malicious behaviour, censorship,
 19 profit opportunities, and accepting bribes. For most security properties, a malicious miner
 20 (or set of colluding miners) would need a majority of the computational power (to solve
 21 puzzles) and then use their 51%, or greater, of computational power to attack the Bitcoin
 22 system.

23 **Issuance of BTC.** The Bitcoin protocol governs how BTC comes into circulation. All
 24 new BTC is given to the servers that are validating transactions—for this reason, they are
 25 called miners (the term miner is also a homage to the difficult work these servers do by
 26 continually solving computational puzzles). The inflation schedule is hardcoded into the
 27 Bitcoin protocol: for the first 4 years, 50 BTC is given out every 10 minutes (on average),
 28 then it is halved to 25 BTC for the next 4 years, and then halved again and again every 4

1 years until the amount reaches 0. In the end, 21 million BTC will be produced. These block
 2 rewards provide an incentive to validate transactions on the network.

3 The puzzle being solved by miners is not configured like a race where the fastest miner
 4 with the most computational power always wins—finding the solution to the puzzle first.
 5 Instead it is configured like a lottery where the miner with the most computational power
 6 has the most lottery tickets—an advantage but not a guarantee they will win. In fact, a
 7 miner with, say, 3% of the computational power is similar to a customer who purchases
 8 3% of the tickets in a lottery—they can expect to win 3% of the time. This provides an
 9 incentive for miners to participate in the consensus process even if they do not have the most
 10 computational power.

11 **Bitcoin fees.** Miners collect new transactions (“mempool”) as they are broadcast and
 12 relayed across the Bitcoin network (as well as relaying transactions themselves). Miners are
 13 free to assemble blocks of new transactions however they want: including or not including
 14 any transaction, and ordering included transactions. The Bitcoin protocol specifies an upper-
 15 limit on how large a block can be, so miners need to prioritize which transactions to include
 16 if there are more transactions than space in the block. To incentivize miners to include
 17 their transactions, users will offer a fee that can be claimed by the miner that includes
 18 the transaction in the miner’s block. Users compete for inclusion of their transactions by
 19 offering higher fees in times of congestion. Naturally, miners then prioritize transactions by
 20 the highest fee. The result is an auction of space in the block given to the highest fees.
 21 Bitcoin miners are enterprising “for-profit” entities that choose to operate Bitcoin miners
 22 because it is profitable.

23 **3.3 Ethereum**

24 After the initial success of Bitcoin, a group of enthusiasts believed that an open blockchain
 25 could be useful beyond use-cases like the transfer of assets. Bitcoin itself is limited in terms
 26 of what it can do beyond this. After failing to convince the Bitcoin community to expand

the scope of Bitcoin, they created a competitor called Ethereum. The Ethereum blockchain began producing blocks in July 2015. Ethereum’s protocol can broadly be split into two eras: pre-Paris (July 2015–September 2022) and post-Paris (or Ethereum 2.0 or eth2) (September 2022–present), where Paris is the name assigned to the protocol change (or “hard fork”) that was agreed to and actuated by (a super-majority of) miners at the time.

Ethereum originally used Nakamoto Consensus with only minor differences from Bitcoin (pre-Paris) but has since transitioned to a new consensus mechanism described below. Post-Paris, miners are called validators as they no longer solve computational puzzles. Like Bitcoin, Ethereum is an open and permissionless blockchain where anyone can join or leave at any time. As in Bitcoin, users have addresses controlled by digital signatures. Ethereum has a native asset called ETH that can be transferred between addresses. As of the time writing, ETH has the second highest market capitalization (to BTC) of all digital assets. Validators receive newly minted ETH and fees from users with some differences from Bitcoin elaborated on below.

Ethereum contracts. The key difference between Bitcoin and Ethereum (pre-Paris) was that users can design and deploy custom software applications (called “smart contracts”) and have the miners/validators run these applications for them. Smart contracts might allow users to make custom tokens, trade Ethereum’s digital asset ETH for these tokens, borrow tokens, invest in tokens, purchase financial derivatives based on tokens, and many other use-cases that are now called “decentralized finance (DeFi).” The most popular smart contracts in addition to DeFi, according to the website DappRadar,¹¹ allow gambling, gaming, social platforms, and transacting digital art. “Smart contracts” are essentially computer programs or applications. They are sometimes called “decentralized applications” or Dapps instead.

Ethereum begins with the same capabilities as Bitcoin: users can create addresses to receive and send ETH. It then adds a new kind of transaction where a user can submit the code of a computer application (or a “contract”) to Ethereum. The contract will be assigned an address and its code will be stored on the blockchain at this address. The user pays a fee

¹¹ “Top Blockchain Dapps,” Dapp Radar, Retrieved Feb–May 2023.

1 to deploy a contract (proportional to the size of the contract). At this point, the user who
 2 created the contract could disappear, and the application will still live on the blockchain
 3 and be accessible to current and future **Ethereum** users. Contracts are “autonomous” which
 4 means they cannot perform computations by themselves (“in the background”) the way
 5 a computer or smartphone application might. Contracts only run code when users ask
 6 **Ethereum** to run the contract (and pay for it). Once the user-requested computation is
 7 completed, the contract code hibernates until the next user requests that it runs. What
 8 users are allowed to run computations and what computations a contract can perform are
 9 contained in the code of the contract itself (and can be anything the programmer of the
 10 contract decides when programming it).

11 While a sophisticated user might interact with a smart contract directly on **Ethereum**,
 12 most contracts are accompanied by a website with graphics, text input, buttons, and other
 13 user interface elements that will interact with **Ethereum** and the smart contract. A user will
 14 navigate to the website and if they wish to use the contract, they will “connect” the website
 15 to the **Ethereum** (or **Ethereum-compatible**) software they are using to manage their signing
 16 keys (called a “wallet”). The website will pass the cost and other details of what the user
 17 wants to do (called a “transaction”) to the user’s wallet software. The wallet software will
 18 display the information to the user and ask the user for consent to execute the transaction
 19 (typically requiring a password) or provide an option to cancel the transaction.

20 **Ethereum consensus.** Pre-Paris, **Ethereum** used **Nakamoto Consensus** that was essentially
 21 the same as **Bitcoin**, having miners solve computational puzzles. Leading up to the Paris
 22 fork, **Ethereum** began a slow switch from this “proof of work” mechanism to an alternative
 23 called “proof of stake.” Post-Paris, **Ethereum** has dropped the computational puzzles as the
 24 mechanism that makes “sybil identities” expensive to create. It now requires anyone wanting
 25 to serve as a validator to obtain and lock up (or “stake”) 32 **ETH** (approximately \$60K USD
 26 at time of writing) as a fidelity bond for participating in a timely manner and taking correct
 27 actions (according to the majority of validators).

28 **Ethereum’s** proof of stake consensus has similarities to both **BFT** protocols and **Nakamoto**

1 **Consensus.** Like Nakamoto Consensus, it is permissionless: anyone able to stake 32 ETH is
 2 able to join the set of validators. Proof of stake is sybil-resistant since validators wanting to
 3 inflate their influence need to stake more ETH, which entitles them to have greater influence.
 4 Since staking is an on-chain action, the list of validators is visible to everyone at all times.
 5 The result is a sybil-resistant list of validators that anyone can join or leave at any time.
 6 With a list of validators, Ethereum can then use a traditional BFT protocol to complete the
 7 consensus mechanism.

8 The Ethereum protocol creates a sequence of 32 slots at a time for the creation of the
 9 next 32 blocks. Ethereum uses an in-protocol system to assign a validator at random to
 10 each slot (with some nuance in what random means here). The validator assigned to a slot
 11 waits until the slot is reached (12 seconds between each slot) and then proposes a block of
 12 fresh transactions that have been broadcast to the validators but have not been included yet
 13 in any previous blocks (from any previous slot). Once a set of 32 blocks is created, other
 14 validators vote on the validity of the set. The set is considered final when it receives votes
 15 from validators representing 2/3 of all the ETH staked by validators.

16 Validators that do not participate in timely manner, sign conflicting messages (equivocate),
 17 or perform other faulty/malicious actions that can be adjudicated by the Ethereum
 18 protocol itself will be penalized. For minor infractions (*e.g.*, going offline), validators will
 19 simply forgo the rewards (described below) they would otherwise have earned. Major infrac-
 20 tions (*e.g.*, voting both for and against a fork) will see a fine levied against their deposited
 21 ETH ('slashing'), and they will lose their validation status if their deposit ends up below 32
 22 ETH. In contrast, new ETH is provided to validators that participate actively and perform
 23 actions that align with the majority of other validators. Validators are thus economically
 24 incentivized to align their actions with the majority of other validators and punished when
 25 they fail to do so. All rewards, fees, and slashing are fully automated within the Ethereum
 26 protocol itself, and do not require any external adjudicator or authority.

27 **Issuance of ETH.** The method for allocating ETH in Ethereum is different from Bitcoin.
 28 Ethereum began with an 'initial coin offering' of ETH, where 60,000,000 ETH was auctioned.

1 12,000,000 was given to **Ethereum** developers directly (or indirectly through a fixed price
 2 purchase program) and an endowment for investing in **Ethereum** technology overseen by the
 3 *Ethereum Foundation*. After this initial allocation, more new **ETH** continues to be issued over
 4 time. This reward is claimed by the validator who is randomly selected to propose a block,
 5 similar to **Bitcoin**, with some **ETH** also being given to validators who vote in the consensus
 6 process. These rewards provide an incentive to validate **Ethereum** transactions. Today the
 7 total **ETH** that has been issued is $\approx 120,000,000$. The fees collected by validators along
 8 with the rewards from participating in consensus creates revenue for those participating.
 9 Like **Bitcoin** miners, **Ethereum** validators are enterprising “for-profit” entities that choose to
 10 operate **Ethereum** validators because it is profitable.

11 **Ethereum fees.** To ensure validators are fairly compensated and to combat malicious ac-
 12 tors from stalling the network (“denial of service” attacks) by asking for a long-running
 13 computation to be performed, all computations are broken into small steps (“instructions”
 14 or “opcodes”) where each step is assigned a value in a unit called “gas.” The value represents
 15 how complex the computation step is to execute or store (*e.g.*, a multiplication has a higher
 16 gas value than an addition). Users then pay two types of fees. The first component is the
 17 priority fee: the user specifies a rate of **ETH** per unit of gas that they are willing to pay
 18 as a fee to the validator who includes their transaction in a block. This works like fees in
 19 **BTC**—the user is bidding to have their transaction included ahead of other user transactions.
 20 In practice, the user’s software examines the current conditions of **Ethereum** and suggests a
 21 rate to the user.

22 The second component is the base fee: the blockchain specifies a rate of **ETH** per unit of
 23 gas that is a mandatory fee and is burned from circulation once paid. The base fee dynam-
 24 ically increases (and decreases) in value if the **Ethereum** networks becomes more congested
 25 (less congested) with transactions. This creates an incentive for users to wait during times
 26 of congestion. The main takeaways are: (1) all computations cost the user **ETH** in fees, (2)
 27 more complex computations cost more than simpler ones, and (3) validators earn revenue
 28 by performing computations on **Ethereum**.

1 To prevent users from asking for a computation to be run without realizing it will consume
2 more **ETH** than they are willing to pay, users can cap the maximum amount they will pay for
3 a computation. If a cap is used and a computation “runs out of gas” before it is completed,
4 the user will lose their entire fee but no more than it. The validator will abandon the
5 computation at the point that it runs out of gas, record an error on the blockchain, and
6 “revert” any changes that the computation made (leaving it as if the computation was never
7 run in the first place).

8 **3.4 The XRP Ledger**

9 Like **Ethereum**, the **XRP Ledger** is a blockchain which followed in the footsteps of **Bitcoin**. It
10 predates **Ethereum** by many years—the **XRP Ledger** began producing blocks (called “ledgers”)
11 in December 2012. The **XRP Ledger** is designed to be faster than **Bitcoin**, producing ledgers
12 every few seconds instead of every 10 minutes. It also abandons **Nakamoto Consensus** and
13 does not require validators to solve computational puzzles, in favor of something closer to a
14 **BFT** protocol. The compromise the **XRP Ledger** makes to achieve these improvements is to
15 reduce the degree to which it is open and permissionless, at least in practice.

16 Aside from its deviation from **Nakamoto Consensus**, the **XRP Ledger** is a blockchain-based
17 protocol with many similarities to **Bitcoin**. Users obtain an address to receive its native
18 asset which is called **XRP** and use digital signatures to authorize transmissions. The **XRP**
19 **Ledger** offers standard transactions, along with more complex transactions, custom tokens,
20 and financial services (such as an on-ledger exchange service for trading assets). Unlike in
21 **Ethereum** where users can create new kinds of financial services at any time by deploying
22 custom contracts, each type of transaction is hard-coded into the **XRP Ledger** protocol. New
23 transaction types cannot be added to the **XRP Ledger** without changing the protocol itself.

24 The **rippled** code is the open-source code underlying the **XRP Ledger**. While anyone may
25 technically suggest modifications to **rippled**, nearly all significant contributors to **rippled**
26 are current or former *Ripple Labs* employees or have some other relationship with *Ripple*
27 *Labs*. Of the 20 most significant contributors to **rippled**, one is Arthur Britto a co-founder

of *Ripple Labs*, 16 are current or former *Ripple Labs* employees, and *Ripple Labs* has made payments to 2 of the remaining 3 contributors.¹² *Ripple Labs* thus has or has had a financial relationship with 19 out of the 20 most significant contributors to *rippled*.

The XRP Ledger Consensus. However, the XRP Ledger’s consensus protocol (called XRP-LCP) differs significantly from that of Bitcoin and Ethereum. XRP Ledger uses a consensus mechanism that is much closer to a traditional BFT protocol than Nakamoto Consensus. Recall that a traditional BFT protocol is predicated on having a list of trusted validators. The challenge is determining who manages the list. If a blockchain is open and permissionless, a new validator must be able to join the consensus mechanism at any time without the permission of another entity. However if anyone can add themselves to the list, a malicious validator can hijack the network by adding themselves thousands of times using “sybil identities.” Nakamoto Consensus combats sybils by requiring miners to have computational power while Ethereum (post-Paris) requires validators to stake a material value of ETH to participate. I first describe how the XRP Ledger consensus works assuming there is an available list of validators, then I will describe how the XRP Ledger arrives at such a list.

Transactions in the XRP Ledger are created and signed by users and broadcast to the network of validators. Validators batch new transactions into a ledger (*cf.* block) every 3–5 seconds. The consensus process aims to quickly (‘liveness’) finalize valid XRP Ledger transactions, in the same order (‘total order’) across all validators (‘safety’). Unlike Bitcoin and Ethereum where one validator proposes a block of transactions and the remaining validators vote in support or opposition of the block, validators in the XRP Ledger construct the ledger together, transaction-by-transaction. Every validator constructs a list of transactions it has seen and considers valid, and circulates it to the other validators. Each validator keeps the transactions that are being supported by the other validators and discards the transactions that are not. Eventually a set of transactions will gain the support of 80% (or more) of all validators on the list and this ledger will be passed around to be digitally signed by each validator. Once it obtains signatures from 80% of validators, it is considered final. The

¹²Defendants’ Supplemental Response to Lead Plaintiff’s Interrogatory No. 7.

validators then repeat the process to produce the next ledger.

Validator lists in the XRP Ledger. In a traditional BFT protocol, a centralized entity would manage the list of validators. However, this is not open and permissionless. The solution of the XRP Ledger is to allow (at least in principle) every validator to choose their own list of trustworthy validators. While this proposal appears elegant and simple, it creates new difficulties. If two validators have lists that are completely different, they are likely to develop different ledgers over time as distributed networks do not always see the same transactions at the same time. Developing different ledgers is called an “unintentional fork” and once it happens, there will be two versions of the XRP Ledger and no (in-protocol) way to say which one is right and which one is wrong.

As a result, the proper functioning of the XRP Ledger requires that validators utilize lists (called “unique node lists” or “UNLs”) which must overlap substantially with other validators’ UNLs. Experts have reached differing conclusions regarding what degree of overlap between validator lists is required to prevent forks and ensure ledgers get enough validator support to finalize. *XRP Ledger Foundation* documentation suggests overlap should be at least 90% and some experts argue there are conditions where it should be even higher. The requirement depends on what assumptions are made about validator behaviour (*e.g.*, greater overlap is required when some validators might be malicious), network conditions, and what exactly it means for the XRP Ledger to function correctly.

The *XRP Ledger Foundation* warns on its website: “if your UNL does not have enough overlap with the UNLs used by others, there is a risk that your server forks away from the rest of the network. As long as your UNL has > 90% overlap with the one used by people you’re transacting with, you are completely safe from forking. If you have less overlap, you may still be able to follow the same chain, but the chances of forking increase with lower overlap, worse network connectivity, and the presence of unreliable or malicious validators on your UNL.”¹³ Validators must therefore coordinate to ensure that their UNL lists overlap with those of other validators.

¹³*XRP Ledger Foundation* FAQ

1 The next problem is how validators coordinate those lists to ensure at least 90% overlap.
 2 The solution in practice has been that software implementing the **XRP Ledger** protocol comes
 3 with a preset list of validators. The original software, **rippled**, comes with a preset, or
 4 default, list. Until July 2021, **rippled** contained only one distribution point for obtaining a
 5 **recommendedValidatorList**. This distribution point was maintained by Ripple Labs itself
 6 at <https://vl.ripple.com>. A second distribution point was added in July 2021. The
 7 recently added second distribution point is provided by the *XRP Ledger Foundation* at
 8 <https://vl.xrplf.org>. At the time of writing, both preset distribution points have 100%
 9 overlap and contain an identical set of 35 validators. Given the importance of overlapping
 10 validator lists, it is unsurprising that the *XRP Ledger Foundation* has exactly matched the
 11 validator list provided by *Ripple Labs*. Using two identical lists adds redundancy to **rippled**
 12 but does not enhance trust.

13 This list is widely referred to as the *default unique node list* (or **dUNL**) although I use
 14 the more neutral term **recommendedValidatorList**. “Validators” is more descriptive than
 15 “nodes,” and **dUNL** validators are not necessarily “unique” (*e.g.*, from January 2013–June
 16 2018, 100% of the validators on the **dUNL** were operated by *Ripple Labs*¹⁴).

17 There is no incentive for validators to deviate from the **recommendedValidatorList** and
 18 no evidence to suggest that validators risk deviating from or altering the preset list. My
 19 opinion (§4.1) is that validators in practice need to treat this list as canonical and a *de facto*
 20 requirement, rather than a recommendation. The risks of modifying the list are reinforced
 21 through comments in the code such as, “Changing [the distribution points] can cause your
 22 **rippled** instance to see a validated ledger that contradicts other **rippled** instances’ validated
 23 ledgers (aka a ledger fork) if your validator list(s) do not sufficiently overlap with the list(s)
 24 used by others.”¹⁵ The canonical nature of the recommended list is also reinforced by
 25 the *XRP Ledger Foundation* network visualizer, which flags these validators with a special
 26 “UNL” visual cue in the list of validators.¹⁶

¹⁴Disclosure, Document RPLI02460831.

¹⁵GitHub

¹⁶XRPL Live Data, 2023.

Given the presets in `rippled` and the strong warnings against deviating from them, *Ripple Labs*, as the author of the `recommendedValidatorList`, is a *de facto* single point of failure for the XRP Ledger. *Ripple Labs* can change or remove validators from the `validatorList` without any action or approval from the validators or other network participants. This gives *Ripple Labs* significant control over the consensus process (see Opinion 2 in §4.2). For example, *Ripple Labs* can modify the `recommendedValidatorList` to include only validators under its control and overtake the network. Under such a scenario, validators would require out-of-protocol coordination to recover.

The main takeaway is that the XRP Ledger is open and permissionless “on paper” but the realities of operating a validator that reaches consensus with other validators dictate that validators must coordinate on a single list of validators (Opinion 1: §4.1). *Ripple Labs* plays this role currently by setting the default list in `rippled` (Opinion 2: §4.2), and *Ripple Labs* will continue to play this role until *Ripple Labs* itself decides to delegate it to someone else (*e.g.*, *XRP Ledger Foundation*). In this case, the entity will change but the idea of coordinating on a list controlled by a single entity will continue. Since validators cannot meaningfully contribute to consensus until they are added to the list, the XRP Ledger is not open and permissionless in practice (Opinion 3: §4.3).

Issuance of XRP. Unlike in Bitcoin, where BTC is released over time to miners, all XRP units (100B units) were created and allocated at or near the start of the ledger (December 2012). This allocation is hardcoded into the XRP Ledger, the original software client created by *Ripple Labs*.¹⁷ Development on `rippled` began in 2011 by programmers that included Ripple co-founder Jed McCaleb¹⁸ and David Schwartz (presently Chief Technology Officer of *Ripple Labs*).¹⁹ I rely on their depositions to understand the history behind the development of `rippled`.

The XRP Ledger was first deployed in December 2012. The first ledger updates were

¹⁷McCaleb Depo. at 75-76

¹⁸McCaleb Depo. at 13-17

¹⁹Schwartz Depo. at 54-62.

created and validated by three servers running `rippled`. These three validators were run by McCaleb and Schwartz.²⁰ The exact date of deployment is unknown because the initial ledgers, 1-32,569, including the genesis block at ledger 1, were lost due to a technical issue with the validators. The oldest existing block 32,570 was created on January 1, 2013.

The XRP Ledger was apparently reset a number of times before the current deployment—“resetting the network” in McCaleb’s words— which I understand means that any transaction data was forgotten and the ledger began again with the initial allocation of 100B XRP units.²¹

XRP is a fungible digital asset that can be divided into one million subunits called drops. The XRP in existence today was created when McCaleb’s and Schwartz’s validators began running, in 2012, the `rippled` code for the current version of the XRP Ledger. The total supply of 100B XRP is hard coded in the `rippled` code and the current version of the `rippled` code does not allow for the creation of additional XRP. As such, no XRP has been created since December 2012. The overall supply of XRP actually decreases over time, as each transaction on the XRP Ledger requires the sender to destroy a small, variable amount of XRP as a financial deterrent against spamming the network with transactions in order to disrupt it or delay other transactions.

Because ledgers 1–32,569 were lost, the initial allocation of XRP is based upon the status of the XRP Ledger at 32,570 and *Ripple Labs*’ records and statements. It is widely understood that the initial allocation was as follows: 80B units were allocated to the company now known as *Ripple Labs* (*née* OpenCoin) and the remaining 20B units were split between three of the founders of the company. After this initial allocation, the protocol does not allow for the creation of additional XRP. Importantly, validators are not rewarded with newly created XRP, as in Bitcoin/Ethereum, which means they operate without internal incentives.

The XRP Ledger fees. Fees are charged of users for every transaction but these fees are removed from circulation (“burned”) instead of being paid to validators, reducing the total

²⁰Disclosure, Schwartz Depo. at 101–02.

²¹McCaleb Depo. at 72–75; Schwartz Depo. at 69–76, 84, 99–108.

1 amount of XRP in circulation. This is also a difference from both Bitcoin/Ethereum and
2 removes the other incentive revenue stream for validators. As a consequence, validators in
3 XRP Ledger do not earn revenue and have no incentive to operate that is internal to the
4 protocol. Operating a validator uses computational resources and network capacity and
5 is costly. Thus, validators must have some external incentive to operate (Opinion 5: §4.5
6 below).

4 Analysis and opinions

In the previous section, I provided a high-level overview of some technical details of Bitcoin, Ethereum and the XRP Ledger. A deeper description is provided in the Appendix. Based upon each protocol's structure, and particularly their consensus mechanisms and the distribution of the native assets, I reach the following conclusions.

4.1 The XRP Ledger consensus protocol requires validators to agree on a list of trusted validators and use of the list published by *Ripple Labs* is a de facto requirement

The XRP Ledger consensus protocol (XRP-LCP) requires that every validator have a list of other validators that they trust to provide truthful and timely decisions. As explained in Appendix C.3, the XRP Ledger exhibits no issues when all validators use exactly the same trusted validator list as each other. There is an academic consensus that the properties of the XRP Ledger degrade when validators use lists that differ by 10% or more with some models suggesting that some properties degrade with as little as a single different validator. In any case, the consistent operation of the XRP Ledger requires that all or almost all validators agree to adopt effectively the same list of trusted validators.

The `recommendedValidatorList` provided by *Ripple Labs* is a preset, default in `rippled`. While the protocol theoretically allows that the recommendation of *Ripple Labs* can be overridden by validators, the safety and liveness requirement of at least a 90% overlap with the `validatorList` of each validator is a guardrail against using a list other than the "recommended" list published by *Ripple Labs*. I suspect all validators use the *Ripple Labs* list verbatim and this is consistent with my observations of the network.

The following cases illustrate that trust cannot be smoothly transitioned away from *Ripple Labs* by individual validators, at least not without some collective action taken by their human operators in the real world.

1. If the `rippled` software preset changes from vesting *Ripple Labs* with control over

1 the `recommendedValidatorList` to another entity with materially the same list there
 2 would be no impact on operation of the XRP Ledger.

3 2. If the `rippled` software preset changes from vesting *Ripple Labs* with control over the
 4 `recommendedValidatorList` to another entity with a materially different list there
 5 would be no impact on operation of the XRP Ledger if most validators are up-to-date
 6 because all validators with up-to-date software would simultaneously be directed to
 7 the same new list.

8 3. If *Ripple Labs* stops distributing a new list after current list expires (*e.g.*, 3 weeks) then
 9 `rippled` validators default to *XRP Ledger Foundation* list which, currently, is Case 1
 10 as there is total overlap between the *Ripple Labs* list and the *XRP Ledger Foundation*
 11 list.

12 4. If one or a few validators override the preset from *Ripple Labs* to another entity that
 13 replicates the *Ripple Labs* list there would be no impact on the operation of the XRP
 14 Ledger.

15 5. If one or a few validators override the preset from *Ripple Labs* to another entity with
 16 a materially different list (or customize the list for themselves) then those validators
 17 could experience safety issues (*i.e.*, fork) and, if joined by others, could contribute to
 18 liveness issues for the network.

19 6. If a super-majority (*e.g.*, over 80%) of validators override the preset from *Ripple Labs*
 20 to another entity with a materially different list then the network will fork but most
 21 validators will join the new fork, so it becomes the *de facto* ledger, which is not a safety
 22 issue.

23 These options illustrate an asymmetric power dynamic where trust can be smoothly
 24 transitioned by *Ripple Labs* (as in Cases 1–3) but it cannot be by individual validators (Cases
 25 4–5) unless the new list merely replicates the *Ripple Labs*’s list (Case 4)). For validator-
 26 initiated actions, the only safe option is Case 6 which requires the collective action of most

validators. Case 5 is allowed but avoided by validators (because of the resulting safety and liveness risks) which results in a lock-in to the *Ripple Labs* list.

Finally, it is important to note that the choice of which `validatorList` to use is not just made by validators, it needs to be made by everyone interacting with the XRP Ledger. If a user wants to check her XRP balance with a software wallet or with a website, the balance is fetched from the XRP Ledger. However the “official” (or “preferred”) XRP Ledger is the version that has been validated by a quorum of validators, and establishing this requires a list of validators. So the `recommendedValidatorList` is implicitly used by all software that fetches data from the XRP Ledger.

My conclusion is that there is a material difference between the XRP Ledger and Bitcoin/Ethereum in that the latter does not use a root of trust at all. Bitcoin/Ethereum are open, permissionless systems without a `validatorList`.

4.2 The XRP Ledger is a distributed system but *Ripple Labs* remains the root of trust

In the XRP Ledger, the operation of the ledger is delegated to a set of validators. In security, this is called “distributed trust” because more than one validator needs to be compromised in order to compromise the system. However the realization of distributed trust can be illusory. For example, from January 2013 through June 2018, the list contained only validators controlled by *Ripple Labs*. During this time period all validators on the `recommendedValidatorList` could have been vulnerable to the compromise of a single entity, *Ripple Labs*. In security, such an entity is called a “root of trust,” “trust anchor,” or “single point of failure.”

Even after diversifying the validators on the `recommendedValidatorList`,²² *Ripple Labs* remains a root of trust in the XRP Ledger because it controls the list (and using this list is a de facto requirement as set forth in Section 4.1 above). *Ripple Labs* maintains, in practical

²²Since July 2018, the `recommendedValidatorList` in the XRP Ledger has diversified somewhat to a mix of entities, with only a single *Ripple Labs* validator amongst the 35 validators.

1 terms, the ability to modify or distribute a new `recommendedValidatorList` of its choosing
2 (*i.e.*, it could decide in the future to issue a new list where all validators are controlled by
3 *Ripple Labs*). So while trust is delegated, creating a trust hierarchy, *Ripple Labs* is at the
4 top of the hierarchy.

5 This implies *Ripple Labs* is a single point of failure for the system. An attack (insider or
6 external) directly on *Ripple Labs* or its cryptographic signing key (hardcoded as a verification
7 key into `rippled`) could lead to a catastrophic protocol failure that would require human
8 intervention to repair.

9 It is also worth noting that distributed trust can fail in other ways relevant to the XRP
10 Ledger. Even if validators are different entities, they might collude if they share a common
11 interest—the recommended validators include companies and university groups funded by
12 *Ripple Labs*.²³ While it is harder to compromise a set of validators than a single validator, sets
13 of validators have been compromised simultaneously before. Key compromise is a common
14 attack vector in blockchain systems, such as those running on top of `Ethereum`, and can
15 occur even when operations are split or require multiple independent keys (*cf.* attacks on
16 Ronin Network which required 5 of 9 validators²⁴ and Harmony Bridge which required 2 of
17 5 validators²⁵).

18 My conclusion is that the XRP Ledger has failed to realize a meaningful advance, in
19 practice, over simply using a centralized root of trust. The critical role played by *Ripple*
20 *Labs* in the XRP Ledger protocol represents a material difference between the XRP Ledger and
21 Bitcoin/Ethereum. While Bitcoin/Ethereum have foundations that provide software support
22 and public awareness, neither has any direct (or *de facto*) influence over the consensus
23 protocol itself.

²³Disclosure, Document RPLI_02460831.

²⁴“Ronin Network,” rekt.news, Retrieved Feb–May 2023.

²⁵“Harmony Bridge,” rekt.news, Retrieved Feb–May 2023.

1 4.3 *Ripple Labs* is a gatekeeper to full participation

2 In XRP-LCP, a ledger is accepted by other validators (*e.g.*, `rippled`, wallet software, and
3 ledger explorer tools) when it receives a sufficient number of validations from validators
4 on their `validatorList`. In the Bitcoin/Ethereum protocols, a block is accepted by other
5 validators if the block is valid. This represents a categorical difference where the XRP Ledger
6 consensus is based on *who* is making an assertion, while Bitcoin/Ethereum is based on *what*
7 is being asserted. Given the lack of practical trust agility over the `recommendedValidator-`
8 `List`, and given *Ripple Labs* position at the top of the coordinated trust hierarchy, decisions
9 on ledgers in the XRP Ledger are effectively rooted in one entity: *Ripple Labs*.

10 Consider a new validator that joins the Bitcoin network for the first time. If it is able to
11 solve a valid block before other miners, other miners will only consider if the block is valid
12 or not before choosing to adopt it, even when they have never heard of or had any past
13 interactions with this new validator. Bitcoin examines the work being done and is oblivious
14 to who is doing the work. By contrast, a new validator without a pre-existing relationship
15 with *Ripple Labs* that joins the XRP Ledger network for the first time will presumably not
16 yet be on the `recommendedValidatorList`. If it creates valid ledgers and votes on ledgers,
17 its contributions will be effectively ignored by the network because of who it is (or who it is
18 not: it is not a recommended validator).

19 My conclusion is that the lack of meritocracy toward validators in the XRP Ledger repre-
20 sents a material difference between the XRP Ledger and Bitcoin/Ethereum. Bitcoin/Ethereum
21 are open, permissionless systems that accept new validators without any enrollment or au-
22 thorization from trust anchors. The XRP Ledger is not a permissionless system, with respect
23 to validators, in practice.

24 4.4 XRP distribution favors *Ripple Labs*

25 The method for allocating XRP in the XRP Ledger is materially different than BTC in Bitcoin,
26 and somewhat different from ETH in Ethereum. Bitcoin begins with a supply of 0 BTC and
27 slowly releases BTC over time to Bitcoin miners. No BTC was allocated to any entity and

1 all BTC originates from mining (including the entity named the “Bitcoin foundation” which
2 has no more or less influence of Bitcoin than any other entity).

3 In Bitcoin, all BTC in circulation ultimately originates from a payment to a miner, whose
4 primary job is validating transactions. The issuance of BTC does not favor any company,
5 entity, or individual. In contrast to BTC, the Ethereum foundation profited from the public
6 offering of ETH (ETH was offered for sale in exchange for BTC), which was conducted before
7 Ethereum was completed and deployed, as well as payments in ETH to the foundation and
8 the founders/developers affiliated with it. After this initial sale, all subsequent ETH that
9 has come into circulation originates from a payment to a validator.

10 In the XRP Ledger, all 100B XRP (*née* XNS) units were created and allocated at or
11 near the start of the ledger. It is reported that 80B units were allocated to *Ripple Labs*
12 (*née OpenCoin*) and the remaining 20B units were given to the founders of the project.
13 Regardless of the details of this allocation, there is no dispute that all 100B XRP were initially
14 allocated to Ripple or its founders. When combined with the lack of internal incentives
15 described below, this centralized initial distribution, reduces the economic rationale for new
16 participants to join the XRP Ledger.

17 As all XRP was distributed at the beginning of the the XRP Ledger, validators have no
18 incentive internal to the protocol (*cf.* newly issued assets and fees in the Bitcoin protocol)
19 for their participation, which uses computational resources and network capacity. Fees are
20 charged to transaction users but these are removed from circulation (“burned”) instead of
21 being paid to validators.

22 This allocation of all XRP to *Ripple Labs* and its founders also differs, albeit to a lesser
23 degree, from Ethereum where a portion of ETH is paid to validators, leaving some internal
24 incentives for validators.

4.5 Validators in the XRP Ledger require external incentives

The XRP Ledger currently runs with 35 validators contributing to consensus and ≈ 100 total validators (the role of these extra validators is explained in Appendix C).²⁶ Since updating its consensus mechanism, Ethereum validators are registered on the blockchain itself and there are $\approx 600\,000$ validators.²⁷

Bitcoin does not track miners easily as miners are unknown until they solve a block, and their number is difficult to measure when many miners operate in mining pools (hiding thousands of miners behind a single address). Another measurement is the number of “full nodes” which are participants that forward and validate transactions. Bitcoin has $\approx 17\,000$,²⁸ in comparison to the XRP Ledger which has ≈ 275 .²⁹ Another signal of the scale of Bitcoin is that its mining is reported to consume the same annual electricity as a small country like Malaysia or Sweden.³⁰ The conclusion is that validating transactions on Bitcoin and Ethereum are large-scale operations compared to the XRP Ledger—why is that?

Validators and miners in Bitcoin and Ethereum are “for profit” participants (they profit from the new issuance of BTC and ETH over time, as well as transaction fees). The growth in their numbers is the result of a free market, as validators are free to join and leave according to their changing incentives. The XRP Ledger is materially different because there is no issuance of XRP and transaction fees are burned instead of given to validators. The XRP Ledger validators “work for free” and require some external incentive to participate (*e.g.*, alignment with the XRP industry, capital investments from *Ripple Labs* or *XRP Ledger Foundation*, grants to academic institutes, altruism, *etc.*), which one would expect to come from entities that already have a financial interest in the operation of the ledger, such as *Ripple Labs*. As mentioned, validators also cannot meaningfully contribute without being added to the `recommendedValidatorList`.

²⁶ “XRP Validator registry,” XRPSCAN, Retrieved Feb–May 2023.

²⁷ “BeaconScan statistics,” Etherscan, Retrieved Feb–May 2023.

²⁸ “Reachable Bitcoin nodes,” Bitnodes, Retrieved Feb–May 2023.

²⁹ “Nodes,” XRPL Livenet, Retrieved Feb–May 2023.

³⁰ “Nic Carter: How Much Energy Does Bitcoin Actually Consume?,” Harvard Business Review, Retrieved Feb–May 2023.

1 The requirement that economically rational participants are incentivized to act as valida-
2 tors on the XRP Ledger only through outside financial incentives is a plausible explanation for
3 why *Ripple Labs* was the only entity acting as a validator on the `recommendedValidatorList`
4 until July 2018. Since then, the number of validators has expanded. However, *Ripple Labs*
5 does maintain financial ties to many of the other participants currently acting as validators
6 today. For example, 10 of the 35 validators in the `recommendedValidatorList`³¹ belong to
7 universities that are part of the University Blockchain Research Institute (“UBRI”)³² which
8 may have received funding from Ripple through this program.³³ Additional validators belong
9 to companies funded by *Ripple Labs* or related entities.

10 My conclusion is that the lack of internal rewards is a material difference between the
11 XRP Ledger and Bitcoin/Ethereum, which is exhibited by less interest in entities running
12 validators on the XRP Ledger.

13 5 Declaration

14 The opinions expressed in this report are based on my review and analysis of the documents
15 I cite. I reserve the right to supplement my report and analysis based on any new evidence
16 brought to my attention.



17
18 June 07, 2023

19 Montreal, QC, Canada

³¹“Validators,” XRPL Explorer, Retrieved Feb–May 2023.

³²“University Blockchain Research Initiative,” Ripple.com, Retrieved Feb–May 2023.

³³Disclosure, Document RPLI_02460831.

References

- [1] I. Amores-Sesar, C. Cachin, and J. Micic. Security analysis of ripple consensus. In *OPODIS*, 2020.
- [2] F. Armknecht, G. O. Karame, A. Mandal, A. Youssef, and E. Zenner. Ripple: Overview and outlook. In *TRUST*, 2015.
- [3] A. Back. Hashcash: a denial of service counter-measure, 2002.
- [4] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. Kroll, and E. Felten. Bitcoin and second-generation cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.
- [5] A. Bracciali, D. Grossi, and R. de Haan. Decentralization in open quorum systems: limitative results for ripple and stellar. In *Tokenomics*, 2020.
- [6] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *OSDI*, 1999.
- [7] P. Champagne. *The Book of Satoshi: The Collected Writings of Bitcoin Creator Satoshi Nakamoto*. E53 Publishing, 2014.
- [8] B. Chase and E. MacBrough. Analysis of the xrp ledger consensus protocol. Technical Report 1802.07242, arXiv, 2018.
- [9] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.
- [10] J. Clark, D. Demirag, and S. Moosavi. Demystifying stablecoins. *Communications of the ACM*, 63(7), July 2020.
- [11] J. Clark and A. Essex. Commitcoin: Carbon dating commitments with bitcoin. In *Financial Cryptography*, 2012.
- [12] A. Di Luzio, A. Mei, and J. Stefa. Consensus robustness and transaction de-anonymization in the ripple currency exchange system. In *ICDCS*, 2017.

- [13] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *IEEE Symposium on Security and Privacy*, 2014.
- [14] S. Gilbert and N. Lynch. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.
- [15] S. Haber and W. S. Stornetta. How to time-stamp a digital document. In *CRYPTO*, 1990.
- [16] S. A. Haber and W. S. Stornetta. Secure names for bit-strings. In *CCS*, 1997.
- [17] A. Lewis-Pye and T. Roughgarden. Resource pools and the cap theorem. Technical Report 2006.10698, arXiv, 2020.
- [18] L. Mauri, S. Cimato, and E. Damiani. A formal approach for the analysis of the XRP ledger consensus protocol. In *ICISSP*, 2020.
- [19] A. Narayanan, J. Bonneau, E. W. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies*. Princeton, 2016.
- [20] A. Narayanan and J. Clark. Bitcoin’s academic pedigree. *Communications of the ACM*, 60(12), 2017.
- [21] S. Sankagiri, X. Wang, S. Kannan, and P. Viswanath. Blockchain cap theorem allows user-dependent adaptivity and finality. In *Financial Cryptography and Data Security*, Berlin, Heidelberg, 2021.
- [22] D. Schwartz, N. Youngs, and A. Britto. The ripple protocol consensus algorithm. Whitepaper, 2014.

1 **A The Bitcoin protocol and implementation**

2 In this and the following sections, I provide an in-depth technical description of the protocols
3 for Bitcoin (Section A), Ethereum (Section B), and the XRP Ledger (Section C).

4 The concept of electronic cash is widely attributed to David Chaum in 1982 [9]. The
5 Chaum design uses cryptography to protect user privacy and stop users from spending money
6 they do not have. While it was commercialized by the company DigiCash, which operated
7 through most of the 1990s, it relied on commercial banks to play the role of a centralized
8 issuer and processor of transactions [19]. The DigiCash currency, CyberBucks, was a dema-
9 terialization of government money on deposit at the commercial bank (today, such a design
10 is called a stablecoin [10]).

11 Bitcoin was described in a whitepaper in late 2008 by Satoshi Nakamoto (thought to be
12 a pseudonym [19]) and deployed in January 2009. Nakamoto notes that he “started with
13 the usual framework of coins made from digital signatures, which provides strong control of
14 ownership, but is incomplete without a way to prevent double spending.” Digital signatures
15 will be described below but the key idea is that the commercial bank will endorse it digitally.
16 The double spending problem refers to the challenge of preventing a digital asset from being
17 spent more than once, a critical issue for digital assets. Prior to Bitcoin, existing digital
18 payment systems struggled to prevent this issue without relying on a central authority to
19 verify transactions. Bitcoin was designed to solve the double spend problem through a decen-
20 tralized, trustless, and transparent mechanism called the blockchain. Nakamoto describes
21 Bitcoin’s solution to the double spend problem as “a peer-to-peer network using proof-of-work
22 to record a public history of transactions that quickly becomes computationally impractical
23 for an attacker to change if honest nodes control a majority of CPU power.”

24 Nakamoto contrasts Bitcoin to the Chaumian model of digital cash (“the old Chaumian
25 central mint stuff”) when he writes, “A lot of people automatically dismiss e-currency as a
26 lost cause because of all the companies that failed since the 1990’s. I hope it’s obvious it
27 was only the centrally controlled nature of those systems that doomed them. I think this is
28 the first time we’re trying a decentralized, non-trust-based system [7].” Bitcoin provides its

own stand-alone digital asset, Bitcoin (BTC), and operates through a peer-to-peer network that anyone can join or leave at any time with no one in charge.

This section is structured as follows. First, I discuss the basic building blocks underlying the Bitcoin protocol, known as cryptographic primitives, in §A.1. Second, I discuss Bitcoin’s data structure in §A.2. Third, I discuss transactions on the Bitcoin protocol in §A.3. Fourth, I describe Bitcoin’s consensus mechanism in §A.4. Finally, I discuss the creation and distribution of BTC and its role in compensating miners in §A.5.

A.1 Cryptographic properties and primitives

The Bitcoin protocol uses a number of primitives from cryptography. A primitive is a building block that can be combined with other primitives to make a protocol. Cryptography is the study of keeping data confidential and integral. Cryptographic primitives are often designed with security properties that prevent an adversary from accomplishing some malicious action, such as forging a digital signature on a document.

In applied cryptography, researchers often say it is *infeasible* for a computer to perform some action. Infeasible is a weaker assertion than impossible—there is at least a naive algorithm to break the property, but the primitive is parameterized such that this algorithm would require more computation than what is currently possible, often by a large safety margin (*e.g.*, more computational effort than a supercomputer running for a billion years). If it is infeasible to find an example value of something, I say it is *negligible* rather than saying it is impossible. Similarly, if it is infeasible to find a counter-example of a property, I say the property is *overwhelmingly* true rather than saying it is exactly true.

It is always possible that some weakness in the primitive can be found that means an efficient algorithm can break the system. Cryptography relies on assumptions, based on evidence but not conclusive proof, that certain computational tasks are difficult. A different computational model, such a quantum computer, might also suffice to break a primitive. Specifically, a large quantum computer (beyond what is feasible today) would break the digital signature primitive used in Bitcoin but is not known to break the hash function

1 primitive—both primitives described below.

2 The following basic components are designed to be secure, efficient, and mathematically
 3 robust, allowing developers to use them to create complex cryptographic systems. I briefly
 4 introduce them here.

5 **Hash functions.** An important component of **Bitcoin** is a hash function, $\mathcal{H}(\cdot)$, which is a
 6 deterministic function that makes an input x of any length and produces an output y that
 7 is exactly d bits long: $y = \mathcal{H}(x)$. Conceptually, a hash function is applied to data x and
 8 produces a unique “fingerprint” or “digest” of the data. If the data input changes even by
 9 a small amount (such a single bit), the output will change with overwhelming probability.
 10 Even for the best known attacks on the hash function, it should be infeasible for a computer
 11 to find two inputs, $x_1 \neq x_2$ with the same output ($\mathcal{H}(x_1) = \mathcal{H}(x_2)$)—a property called
 12 (strong-)collision-resistance. It should be even harder when x_2 needs to match a specific x_1
 13 and $\mathcal{H}(x_1)$ —a property called weak collision resistance (“weak” because it is easier for the
 14 protocol designer to achieve). Finally, it should be infeasible to determine an input x to a
 15 hash function given only its output $y = \mathcal{H}(x)$ —a property called pre-image resistance (where
 16 “pre-image” is a more formal mathematical term for “input”).

17 In most places in the protocol, **Bitcoin** uses the SHA-256 hash function from SHA-2 family
 18 of hash functions. As of February 2023, the SHA2 family of hash functions are considered
 19 collision resistant and pre-image resistant by NIST.³⁴ In one specific place, **Bitcoin** uses
 20 the RIPEMD-160 hash function which is pre-image resistant and weak collision resistant
 21 by NIST’s general standards (the hash is not specifically identified by NIST but hashes
 22 with outputs of 160 bits and no other known weaknesses can be quantified) but it is not
 23 (strong) collision resistant. In its specific application within **Bitcoin**, weak collision resistance
 24 is thought to be the only necessary property for how RIPEMD-160 is used and therefore its
 25 lack of strong collision resistance is not a concern.

26 At a high level, **Bitcoin** uses hash functions in many places: to store short representations
 27 of transactions, blocks of transactions, chain together blocks of transactions, instantiate a

³⁴“Hash Functions,” NIST Computer Security Resource Centre, Retrieved Feb–May 2023.

1 computational puzzle, generate a short digest of a message before signing it, and to create
 2 digests of cryptographic keys to serve as addresses in the system.

3 **Commitment Functions.** A commitment function is a direct application of a hash func-
 4 tion. It allows someone to take data, called the message m , and create a succinct (*e.g.*,
 5 256-bit) fingerprint of it that *locks-in* the data: $c = \text{Comm}(m)$. Later the commitment can
 6 be opened to check if it matches m exactly: $\{\text{Accept}, \text{Reject}\} = \text{Open}(c, m)$. A commitment
 7 is required to be *binding* which means it is infeasible to find a different message $m' \neq m$ such
 8 that $\text{Accept} = \text{Open}(c, m')$ when $c = \text{Comm}(m)$ for any m and m' .

9 A commitment function may also be required to be *hiding* which means it is infeasible
 10 to compute any information about m given $c = \text{Comm}(m)$. Even if the adversary guesses m
 11 correctly, it is infeasible to determine if c is actually a commitment to m . Bitcoin primarily
 12 uses commitments that are only binding, except in one location (described below as *proof of*
 13 *work*) where a hiding commitment is used indirectly.

14 The simplest binding-only commitment is to directly use a hash function $c = \text{Comm}(m) =$
 15 $\mathcal{H}(m)$, where the binding property follows from the (strong) collision resistance of the hash
 16 function. The remaining problem is how to commit to multiple (*e.g.*, thousands of) mes-
 17 sages: $\{m_1, m_2, m_3, \dots\}$. If 1000 messages are committed to individually, the result is 1000
 18 commitments; however any individual message can be opened independently of any other
 19 message. Alternatively, 1000 messages could be concatenated together and the concatenation
 20 could be committed to, resulting in only 1 commitment. However opening one message
 21 requires knowledge of the other 1000 messages.

22 Bitcoin uses commitment functions to lock-in transactions and blocks of transactions. It
 23 opts for a balanced data structure called a hash tree or Merkle tree. A Merkle tree arranges
 24 a batch of messages in the leaves of a binary tree and nodes of the tree compute a hash of
 25 its two children nodes. Using a Merkle tree to commit to 1000 messages produces only 1
 26 commitment value. Opening 1 message requires knowledge of some of the other values in the
 27 tree but not all of them. Specifically it requires $\log_2 n$ for a Merkle tree with n messages, or
 28 10 values for a commitment to 1000 messages. Each of the 1000 messages to be committed

are each stored in their own *Merkle leaf*, the final single commitment value is called the *Merkle root*, and the 10 values revealed to prove a particular message is committed to in a given Merkle root are called the *Merkle path*, as these values trace the root of the tree to the leaf containing the message.

Digital signatures. The second important cryptographic primitive in Bitcoin is a digital signature. A digital signature allows Alice to sign a message m in a way that cannot be forged by Eve. Instead of signing her name, Alice is identified by a numeric value called a “public key.” Technically, the digital signature proves that the person who controls a given public key signed a given message and does not prove anything about Alice, the person, specifically. It is up to system to additionally provide a binding between identities and their public keys—called public key infrastructure (PKI)—which might utilize a directory, certificates signed by an authority, and/or credentials which allow private disclosure of information in a certificate. Finally, a designer of a system might also deliberately forgo PKI and let the public keys operate as pseudonyms for the users. This is the approach that Bitcoin takes.

To generate a public key, the user first choses a secret value at random called the private, secret, or signing key sk . Next, a deterministic algorithm $\text{KeyGen}(\cdot)$ generates the public key from the signing key: $pk = \text{KeyGen}(sk)$. It is infeasible to invert the function $\text{KeyGen}()$ and compute sk from pk . If sk is chosen randomly, the number of possible sk values is infeasible to exhaustively search. Two users will chose the same sk (which would result in the same pk since $\text{KeyGen}()$ is deterministic) with negligible probability.

Given a signing key and some data to be signed, called the message m , the function $\sigma = \text{Sign}(m, sk, r)$ is a randomized function (randomized by the parameter r selected at random, while $\text{Sign}()$ itself is deterministic) that produces signature value σ . To verify σ is a correct signature, the deterministic function $\text{Verify}(\sigma, m, pk)$ takes the signature, the message, and the public key of the signer and returns accept if the signature is valid, and reject otherwise. For a given pk , it is infeasible to produce a σ on any chosen message m without knowing sk —a property known as forgery resistance.

Proof of work. The final primitive used by Bitcoin is a computational puzzle, which is a function run by a computer that is designed to take a moderate amount of computation effort (and thus time) to solve. One application of puzzles is to combat email spam: an email client might require every email has the solution to a puzzle, which is easy for the sending computer to generate once but would become expensive to generate thousands of times (every puzzle would be uniquely instantiated with the recipient's address and the contents of the email). Bitcoin uses proof of work puzzles to deter a single entity from hijacking an unfair portion of influence on consensus.

This idea was deployed in the system Hashcash [3] where the puzzle is defined as follows. Given data or message m (*e.g.*, an email subject, body, and recipient), the computer is challenged to find a value **nonce** such that $y = \mathcal{H}(m, \text{nonce})$ is a small value with, say, its first 10 bits are equal to 0 (or “10 leading zeros”). As a consequence of the pre-image resistance of $\mathcal{H}()$, the best known way to solve this puzzle is to try unique values of **nonce** over and over until y happens to match the solution. Each try has probability 2^{-d} of succeeding for d leading zeros, and therefore a computer would expect to hash 2^{d-1} unique **nonce** values to find a solution.

A.2 The blockchain data structure

An important component of Bitcoin is the record or log of all transactions, which are stored in a data structure called the blockchain. Instead of adding transactions one-by-one to the log, they are added in batches (*e.g.*, up to 1MB) called a *block*. The block itself uses a Merkle tree so that all the transactions in the block can be locked-in with a single commitment: the block's Merkle root. Every block commits (by concatenation) to three core pieces of data and this commitment is called the *block header*.

$$\text{blockHeader} = \mathcal{H}(\langle \text{merkleRoot}, \text{prevBlock}, \text{nonce} \rangle)$$

The first core piece of data in a block header is the block's Merkle root, which is itself a commitment to all the transactions in the block. The second is the block header of the

most recent block. This forms a commit chain (or hash chain) where the block header commits to the previous block header before it, which commits to the block header before it, all the way back to the first block (the first block is called the genesis block, explained below). The term blockchain arises from this chain of blocks, as was previously called linked timestamping [15, 16]. The key idea of the blockchain is that the most recent blockheader is in fact a commitment to the entire history of every past transaction.

The third core piece of data in the block header is called the nonce and it serves to integrate a proof of work puzzle into extending the blockchain, along with two complementary values called the timestamp and the target which are also used in the proof of work. I describe their purpose in §A.4. In addition to the core pieces of data, block headers contain a version number.

A.3 Transactions

The core ability of Bitcoin is to perform a standard transaction, moving BTC from one user (Alice) to a second user (Bob).

Double-spending. The notorious difficulty with any digital system is what stops a user from “copy and pasting” their digital coins multiple times. This is called the double spending problem. The solution used by Bitcoin to solve this problem is to have a single, canonical ledger of transactions, even if it is not maintained by a single centralized entity. Transactions that try to input spent outputs will be rejected by Nakamoto consensus under its threat model of an honest majority of hashrate amongst its miners.

For Alice to have BTC, she must generate a public and private (signing) key for the digital signature scheme used by Bitcoin (called ECDSA-over-secp256k1). Alice’s balance is recorded on the blockchain and assigns units of BTC to her public key. For usability, it is standard to use a hash of a public key, which results in a smaller value that can fit into a standard QR code (*i.e.*, 2D barcode). This smaller value is called a “Bitcoin address” and is obtained with a hash function (technically two in sequence, called SHA256 and RIPEMD-160).

1 Alice can generate as many **Bitcoin** addresses as she wants and since they are not tied
2 to her real world identity by the protocol itself, the protocol itself provides pseudonymity—
3 transactions to/from the same address can be linked together but not to a real world identity.

4 **Bitcoin** deploys a more complex model that is akin to having actual coins: to pay \$1.17,
5 one must select from their wallet or purse which actual coins to hand over, often paying
6 greater than \$1.17 and receiving the difference in coins back (*i.e.*, receiving change). It is
7 argued by **Bitcoin** enthusiasts that this model is more efficient for miners to determine if a
8 transaction is valid but the differences are not material for the purposes of this report. Users
9 generally do not experience these details personally as the software they use hides them.

10 A transaction consists of a set of inputs, which are units of **BTC** that have not yet
11 been spent, and a set of outputs that consist of where the **BTC** should be transferred to,
12 typically identified by addresses. The outputs can split the total amount of **BTC** into chunks
13 of any size and allocate these chunks amongst the addresses in the output. The smallest
14 transactional unit is 0.00000001 **BTC** (worth approximately \$0.00030 USD at the time of
15 writing). A transaction cannot output a greater amount of **BTC** than what is input (with
16 one exception called the coinbase transaction below), however a transaction can output less
17 than the amount that is input—the difference is claimed as a fee by the miner that solves
18 the block that includes the transaction.

19 **Bitcoin** miners maintain a set of all **BTC** balances that have not yet been spent, called the
20 unspent transaction output (UTXO) pool, as a convenient data structure for determining if
21 transactions are valid. Once a transaction is included, the inputs are marked as spent and
22 cannot be respent. So if Alice is trying to pay Bob and Carol at the same time, she must
23 choose non-overlapping sets of **BTC** from the units of **BTC** she owns. Or she can merge the
24 two payments into a single transaction—outputs can go to different addresses (and inputs
25 can also come from different addresses and different people, although this is uncommon with
26 the exception of users trying to better anonymize their **BTC**). Finally the transaction must
27 be signed by each address that contributes an input to the transaction, which is the basic
28 property that stops Bob from spending Alice's **BTC**.

1 A slightly different type of double spending attack is possible in Bitcoin, given that blocks
2 are reorganized and result in small forks. Alice might purchase, say, a car with a specific set
3 of UTXOs, while creating a second transaction that sends the same UTXOs to herself. Either
4 transaction is valid independent of the other, but both cannot be included as they contradict
5 each other. If she releases the car transaction first and the car dealership sees it on the
6 network (but not in a block, which is called 0-confirmed) and gives possession of the car to
7 Alice, she can try (perhaps colluding with a miner) to have her other transaction included in
8 a block instead. It is more difficult to achieve, but even if the car transaction is 1-confirmed
9 or 2-confirmed, it is possible a reorganization will ultimately result in the other transaction
10 being confirmed instead. Waiting 6 blocks prevents this attack but also implies a one-hour
11 wait for a Bitcoin transaction to be fully confirmed. In practice, small value transactions
12 (*e.g.*, a coffee shop) might accept 0-confirmed or 1-confirmed transactions as final, given the
13 technical sophistication of the attack. But an exchange service that accepts deposits of BTC
14 before letting a user trade and withdraw will generally wait for full confirmation.

15 **Scripts.** Bitcoin transactions are implemented in a general way. Instead of sending BTC
16 to the Bitcoin address of another user, a transaction includes computer code (called a script)
17 that describes the conditions under which the recipient of an output can spend the funds.
18 The receipt demonstrates they are authorized to spend the output by composing their own
19 script for each input. This allows Bitcoin users to create new types of transactions, assuming
20 the scripting language available to them is verbose enough to capture the idea. Some scripts
21 are very elaborate, such as the prepaid offline payments. However generally Bitcoin is very
22 conservative about their scripting language and offers a limited set of instructions. By
23 contrast, as will be discussed in more detail below, Ethereum takes scripting further and
24 allows very verbose scripts called smart contracts, while XRP Ledger goes in the opposite
25 direction and restricts transactions to a set of pre-determined types. While scripting is
26 technically available, the majority of transactions follow a simple script called a “standard
27 transaction” which operates as described above (outputs are paid to BTC addresses, and a
28 digital signature is required to spend them).

1 **A.4 Nakamoto consensus**

2 The core problem Bitcoin attempts to solve is how to process accurate transaction records
 3 without control or oversight of a single trusted authority. Bitcoin does this through a mech-
 4 anism known as “Nakamoto consensus.” I describe this mechanism in detail below.

5 **Peer-to-peer network.** Bitcoin runs over a peer-to-peer network. The set of peers, which
 6 I will call nodes, is open and permissionless—anyone can become a node at any time. Bitcoin
 7 software comes with a set of known nodes. A new node can connect to these nodes and ask
 8 for other nodes to add to its list. Nodes are not identified with real world identities, but
 9 rather correspond to network addresses (or IP addresses). Nodes do not try to maintain a
 10 full list of every other node, instead they rely on other nodes to relay their messages to the
 11 nodes they know about, which will propagate messages to all nodes eventually—this is called
 12 a gossip protocol. Gossip protocols are simple and effective although generally inefficient due
 13 to redundancies in received messages and agnosticism over the network topology.

14 **Fault tolerance.** The main innovation of the Bitcoin protocol is to process transactions,
 15 not by a single entity, but rather across an open peer-to-peer network where (i) no one is
 16 in charge and (ii) anyone can join or leave at any time. Distributed systems have been
 17 studied that allow (i), but resolving (ii) in addition to (i) was largely an open research
 18 problem, and no solution had been found and widely deployed prior to Bitcoin. In such a
 19 network, participants are generically called nodes, peers, or participants. When they process
 20 transactions, they are more specifically called miners (Bitcoin and Ethereum 1.0) or validators
 21 (Ethereum 2.0 and XRP Ledger).

22 Ignoring (ii) for now, the typical solution to the problem is to use a consensus protocol
 23 which allows nodes on the network to vote on whether transactions are valid or invalid, with
 24 the majority view taken. The literature on consensus mechanisms is vast and addresses
 25 challenges such as:

- 26 1. **Reset tolerance:** nodes can go offline for periods of time,

- 1 2. **Fault tolerance:** nodes that have been offline do not realize it and send the wrong
2 messages at the wrong time,
- 3 3. **Byzantine fault tolerance** (or BFT): nodes actively attack the consensus mechanism,
- 4 4. **No broadcast:** nodes can equivocate and send different messages to different nodes,
- 5 5. **Partially connected network:** nodes are not fully connected to every node on the
6 network and may lie about the connections they have, attempt to change messages
7 relayed from other nodes, and other attacks,
- 8 6. **Unreliable network:** nodes that send a message do not have a guarantee that the
9 message was transmitted and successfully received (or a time-bound on how long mes-
10 sage transmission takes),

11 Solutions to each of the above challenges had been found prior to Bitcoin and work if
12 the number of honest nodes is greater than some threshold, such as 75% or 80%. Generally,
13 the threshold becomes higher as the protocol becomes more robust. The drawback of these
14 solutions is that the nodes participating in the protocol (validating nodes) must be agreed
15 upon prior to running the protocol, which requires at least one participant—the one creating
16 and maintaining the list—to be trusted above the others. A new node cannot join the
17 protocol without first being authorized by the list authority (sometimes called a gateway).
18 If the list authority decides to add only themselves to the list, the protocols generally do not
19 prevent this. Bitcoin avoids this problem by not using a list of nodes as part of its consensus
20 mechanism.

21 **Sybil-resistance.** Running a voting protocol with a trusted list of eligible voters is a
22 familiar setting, even outside of technology. The primary challenge of designing a consensus
23 mechanism over the internet *without* a list of nodes/validators, as Bitcoin does, is that a
24 single node might pretend to be, say, a thousand or million unique nodes (fake identities are
25 called “Sybils”), as there are no reliable identifiers online. IP addresses that identify internet

connected devices can be obtained in bulk, and MAC addresses that identify hardware network cards can be spoofed.

To provide Sybil-resistance, Nakamoto consensus employs proof of work puzzles. Any node can join the network at any time, and the rate at which it can solve proof of work puzzles (relative to the rest of the network) will dictate how much influence (or weight) it has in the consensus mechanism. Roughly speaking, nodes obtain one “vote” per computational unit. This provides Sybil-resistance because nodes cannot increase their influence without actually increasing their computational power, assuming the proof of work puzzle cannot be cheated. Nakamoto consensus, moreover, also innovates by not using an explicit voting protocol, as described next.

Proof of work consensus. Recall the blockchain data structure that is used to record transactions. The main components of a `blockHeader` is:

$$\text{blockHeader} = \mathcal{H}(\langle \text{merkleRoot}, \text{prevBlock}, \text{nonce} \rangle)$$

The nonce is how the proof-of-work (POW) puzzle is incorporated into the Nakamoto consensus mechanism. The POW puzzle is to find a `nonce` such that `output` is a small number:

$$\text{output} = \mathcal{H}(\langle \text{message}, \text{nonce} \rangle)$$

These have the same template, where $\langle \text{merkleRoot}, \text{prevBlock} \rangle$ can serve as the `message` in the PoW, while the `blockHeader` can serve as the `output`. In Bitcoin, each block header is a solution to a proof of work puzzle, which is to say the value of `blockHeader` must be smaller than some integer called the `target`:

$$\text{target} \geq \text{blockHeader} = \mathcal{H}(\langle \text{merkleRoot}, \text{prevBlock}, \text{nonce} \rangle)$$

The key concept is that finding a `blockHeader` is a moderately difficult task for computer because it must try `nonce` values one-by-one until it finds one that is satisfactory. First I discuss who is performing this task, and then I discuss how the difficulty is set.

Mining. Any node on the network can participate in the Nakamoto consensus mechanism by determining a set of valid transactions that are not yet in the blockchain, determining the values (other than the **nonce**) in the **blockHeader** to represent these transactions, and then searching for a **nonce** value that satisfies the proof of work condition of an output below the target.

Roughly because of the pre-image resistance of the hash function used to compute the **blockHeader**, there is no known method for starting with a specific output or a structure of an output (*i.e.*, a small value), and finding an input that satisfies it. The best approach is to choose values of the **nonce** arbitrarily (such as 0, 1, 2, ...), then compute the hash of the result to create a **blockHeader**, and then check if the **blockHeader** satisfies the constraint. This task can be parallelized with multiple computers searching different regions of possible **nonce** values. The probability that at least one **nonce** value will satisfy the constraint is overwhelming.

Once a **nonce** is found, the blockchain is extended and nodes begin the process again with the next **blockHeader** (a few more details on this below). The main point is that nodes engaging in this process are being fully utilized continuously and this explains the high energy consumption of **Bitcoin**. I call these nodes “miners,” both to communicate the difficult work they are doing and to communicate that they make money when they find a **nonce** (described below). While simple computers were used in the early days of **Bitcoin**, mining is now a commercial industry with large warehouses full of custom computer chips (ASICs) that can only do one computation: compute a **Bitcoin blockHeader**. All **Bitcoin** miners around the world are together capable, at the time of writing, of trying over 300 000 000 000 000 000 000 **nonce** values each second, which is called a hashrate.³⁵

Heaviest chain rule. Miners have two basic tasks before searching for a **nonce**: (i) determine the best set of transactions to put into the Merkle tree, and (ii) determine which existing and valid **blockHeader** to use as their previous **blockHeader**, or to put it another way, which block to extend. I will deal with (ii) first and then with (i) below. I will also

³⁵“Total Hash Rate (TH/s),” Blockchain.com, Retrieved Feb–May 2023.

1 explain the mechanism by which miner are paid below, but note that miners are paid if
2 their **blockHeader** becomes part of the blockchain and this is their primary motivation in
3 participating. Any behaviour that jeopardizes the inclusion of a miner's **blockHeader** will
4 be avoided by the miner. Nakamoto consensus uses incentives to guide miners into correct
5 behaviour, and generally works as long as a majority (51%) of the hashrate is contributed
6 by miners that follow the protocol.

7 Assume everyone on the network sees exactly 81 blocks forming a chain. Consider a
8 simplification of the rule miners will adopt: a miner should extend the "longest" valid chain.
9 This rule says that miners should set the 81th block as the previous block and work on finding
10 a nonce and blockheader for the 82nd block. However because it is an open, permissionless
11 system, miners are free to do whatever they want and one miner might instead try to extend
12 the 80th block instead of the 81th. Assume this miner has a minority of the computational
13 power (or hashrate) relative to the rest of the miners on the network. With high probability,
14 the rest of the network will find a 82nd block before this miner finds a 81st, and even if
15 this miner gets lucky, it will eventually be unable to grow its chain as fast as the other
16 chain. Therefore miners are incentivized to extend the longest existing chain, to increase
17 their probability of success.

18 A similar logic applies to the validity of the chain. A minority miner who extend a chain
19 that includes at least one invalid transaction will see their blocks rejected by the honest
20 majority of miners and is therefore incentivized to check the validity of every transaction
21 in every block in the chain it is extending. This is a lot of work at first, as it needs to
22 synchronize from the start of **Bitcoin** to the most current block, but after this overhead cost,
23 miners will check each block as it is released by other miners. For reasons to be discussed,
24 block arrival time is every 10 minutes on average which provides ample time for these checks.

25 Finally this logic also extends, with some important caveats, to incentivize minors to
26 release a valid block when it finds one. Generally, a miner's best chance at having its own
27 block added to the blockchain happens when other miners hear about the block as quick
28 as possible, so they can validate it and start extending it. For technical reasons, this logic

breaks down if a miner obtains a high portion of the networks' hashrate, such as 25% or 33% — in this case, it becomes more profitable to withhold a block, and be the only miner working on the next block while the rest of the network is one block behind [13].

This subsection is titled the “heaviest” chain rather than the “longest” for a small technicality: the proof of work difficulty changes dynamically (see below). So it is theoretically possible that chain A is longer than chain B, but the proof of work difficulty in chain B is so much higher, that it is more difficult to produce even though it is shorter.

Forks and reorganizations. If two independent miners are both working on extending block 81, their candidates for forming the 82nd block in the chain will be different from each other. They might include different transactions or different orderings of the same transactions, and they will include one transaction where they pay themselves (called the coinbase transaction) which will have a different recipient.

It is possible that two independent miners find block 82 at approximately the same time (call these Block 82a and Block 82b). Because the network is partially connected, both blocks will eventually reach all other miners but might propagate at different speeds or different orders. When miners receive one of the two block 83s, they will validate it and immediately start working on extending it into block 84. The result is that part of the network will work on extending block 82-a if they hear it first, and the rest will work on block 82-b if they hear it first. This is called a “fork” and is a natural occurrence in Nakamoto consensus.

Forks resolve under a condition such as the following: assume the miners working on extending block 82-a find a block 83-a before the miners working on block 82-b. The miner will broadcast it to the network and all miners working on 82-a will switch to 83-a after hearing it and validating it. Importantly, all miners working on 82-b will also switch because the chain ending with $\langle 81, 82 - a \rangle$ is now heavier (and longer) than the chain ending with $\langle 81 \rangle$. So miners re-synchronize their efforts.

It is possible that ties between blocks happen several times in a row but eventually, with overwhelming probability, one side of the fork will become heavier than the other with enough for miners to switch. This is called a re-organization. Under normal circumstances

(excluding an error in the software), temporary forks and re-organizations have historically been shorter than 6 blocks.

Finality. Consider the case where Alice cares about a specific transaction and wants to know if it will be finalized or lost in a re-organization. She will wait until it is included in, say, block 81 and this is called one “confirmation.” She will continue to wait until a block 82 extends the block of interest, 81, and this is called two confirmations. She will wait for six confirmations and consider the transaction final.

Target. To be an acceptable solution to the proof of work, a blockheader value needs to be smaller than a certain value called the target value. The target value is adjusted by the protocol itself and the goal of the protocol is to have blocks appear, on average, once every 10 minutes. Since blocks include a timestamp, the target value is recomputed every 2016 blocks (which corresponds to approximately 2 weeks) using the timestamps included in each block. Miners compute the difference between timestamps of each successive block (block interval) and compute a weighted average over the 2016 blocks. If the average block interval is faster than 10 minutes, the difficulty will be increased by making the target value smaller. Conversely, if the average block interval is slower than 10 minutes, the difficulty will be decreased by making the target value larger. An algorithm in the protocol makes these adjustments, and miners refuse to accept blocks where the timestamp is greater than around 1-2 hours from their own local clock (the exact determination is more complex but is immaterial for this report).

Liveness and safety. The ability to agree (eventually) on a single blockchain is called safety in the distributed systems literature. The basic assumption made by Nakamoto consensus is that a majority of the hashrate at every time is being used by honest miners that follow the protocol. For the specific property of fairly distributing of new BTC (mechanism described below) in proportion to a miner’s hashrate (*i.e.*, a miner with 5% of the hashrate should expect to solve 5% of the blocks and be awarded as such), this property requires a

1 greater number (*e.g.*, 3/4) of honest nodes [13].

2 The basic attack that Bitcoin admits is called a 51% attack, where a miner is able to
 3 obtain 51% or more of the hashrate. With this level of hashrate, the miner can monopolize
 4 the blockchain going forward, ensuring it is the only miner of new blocks. Other miners might
 5 produce blocks, but the 51% miner will ignore them and eventually extend its own chain
 6 to be longer than any chain they contribute to. Further, a 51% attacker could modify past
 7 blocks and eventually “catch up” to the current honest chain and become the heaviest chain.
 8 Finally, a 51% attacker could also modify the rules of the protocol to anything arbitrary.

9 This vulnerability is accepted by the Bitcoin community and not considered worrisome
 10 for several reasons. The first is that the current hashrate of Bitcoin is so vast, that the cost is
 11 prohibitive even for nation states. The second is that even if someone acquired this hashrate,
 12 using it to attack the system will kill the “golden goose” of being able to mine new BTC.³⁶
 13 Last, miners often form coalitions with each other, called mining pools, and these pools have
 14 occasionally approached or exceeded 51% of the hashrate. This differs from a single entity
 15 acquiring 51% of the hashrate because pool members can leave at any time, as they would
 16 likely do if a mining pool used a 51% attack. Leaving is as simple as stopping to work on
 17 the pool’s suggested blockheader (*i.e.*, it does not require a specific action, just inaction is
 18 enough).³⁷

19 Another consideration is liveness, which is a miner’s ability to make forward progress at
 20 all times, and a user’s goal to have their valid transactions eventually included. The liveness
 21 of Bitcoin is akin to its safety. A 51% attacker can completely censor any of any users’
 22 transactions if it wants to. A miner’s ability to censor a transaction results in a delay that
 23 is proportional to its share of the hashrate. A selfish miner can inflate this delay in greater
 24 proportion but cannot ultimately prevent a transaction’s inclusion if the majority of nodes
 25 are honest.

³⁶ “Ed Felten: Bitcoin Mining Now Dominated by One Pool,” Freedom to Tinker, Retrieved Feb–May 2023.

³⁷ “Ed Felten: Bitcoin Mining Now Dominated by One Pool,” Freedom to Tinker, Retrieved Feb–May 2023.

1 A.5 Issuance and fees

2 The Bitcoin protocol was originally created to process transactions of the newly-created
3 BTC digital asset. But BTC plays an additional, critical role in the Bitcoin protocol—it
4 compensates (and therefore incentivizes) miners to participate in Nakamoto consensus enabling
5 transactions to be processed and blocks to be added to the blockchain.

6 **BTC as a block reward.** A digital asset like BTC needs to come into circulation through
7 some process. In Bitcoin, BTC is released slowly over time to the miners who are creating
8 blocks. The schedule for releasing new BTC is hardcoded into the Bitcoin protocol. Miners
9 claim new BTC as a reward. They record this in the first transaction of a block, called the
10 coinbase transaction. It is unique in that it has no inputs, only an output (or outputs). It
11 is created by the miner and generally pays out to the miner. The payment consists of two
12 parts: the block reward and the transaction fees. Nakamoto consensus ensures miners follow
13 the rules of claiming new BTC (or their blocks will be rejected as invalid).

14 Bitcoin's inflation schedule works as follows: initially, miners can claim 50 BTC per
15 block (which corresponds to 50 BTC every 10 minutes on average). Once 210,000 blocks
16 pass (which corresponds to approximately 4 years), the mining reward halves to 25 BTC.
17 It continues halving every 210,000 blocks until it eventually reaches 0 BTC (projected to
18 happen around the year 2140). The most recent halving happened in May 2020, when the
19 block reward was halved from 12.5 BTC to 6.25 BTC. The total number of BTC is capped
20 at 21,000,000 BTC, with most coming into circulation early in Bitcoin history and tapering
21 off as time progresses. At the time of writing, 92% of all BTC has been created.

22 **Transaction fees.** Users entice miners to include their transactions in a block by offering
23 a fee. Since users do not know which miner will ultimately create the block that includes
24 their transaction, they cannot simply output a fee to the miner's address. Instead, fees are
25 specified in the transaction itself by having the total output of the transaction be slightly
26 less than the total input. Miners take these nominal amounts and add them to the mining
27 reward, and create the coinbase transaction to output this total. Other miners use Nakamoto

1 consensus to validate that a block computes the coinbase amount correctly.

2 Transaction fees are set freely through market forces, namely the congestion of the **Bitcoin**
3 network at any given time. Transactions are broadcast to miners who place them in a data-
4 structure called their `mempool`, and generally sort transactions according to fees, prioritizing
5 transactions with higher fees. Fees are not a proportional to the value of the transaction, un-
6 like other payment systems where fees are typically a percentage of the transactional amount.
7 Instead fees in **Bitcoin** are charged in proportion to how large (in data) the transaction is.
8 The value of the transaction has a negligible impact on the size of the transaction, while in-
9 cluding more inputs and/or more outputs has a large impact. While miners are incentivized
10 to include as many transactions as possible, **Bitcoin** limits the size of any block. This limit
11 was initially 1 MB, and now fluctuates around 1 MB to 2 MB due to a more complicated
12 data structure for storing transactions and their accompanying signatures (called segmented
13 witness). At the time of writing, blocks have around 4000 transactions. The block limit
14 helps ensure miners can transmit blocks quickly and validate them quickly, which reduces
15 the amount of forking and reorganization that occurs.

16 **Mining incentives.** As an open and permissionless system, **Bitcoin** allows anyone to join
17 or leave mining at any time. Miners participate in **Bitcoin** if and when it is profitable for
18 them. It is profitable when the cost of mining is less than the amount of **BTC** earned
19 through mining rewards and transaction fees. The cost of mining includes capital costs, like
20 the computer equipment, as well as marginal costs like electricity and cooling. The **Bitcoin**
21 protocol itself does not influence how many miners participate or what the hashrate of the
22 network should be. The increase in mining over the history of **Bitcoin** is largely due to the
23 increase in the **BTC/USD** exchange rate.

1 **B The Ethereum protocol and implementation**

2 **Ethereum** is a popular blockchain. Its native digital asset, **ETH** (or **Ether**), has the second
 3 highest market capitalization (to **Bitcoin**) of all digital assets. **Ethereum** is a blockchain with a
 4 consensus mechanism that runs over a gossip network by an open, permissionless set of miners
 5 (or validators) that anyone can join or leave at any time. **Ethereum**’s protocol can broadly
 6 be split into two eras: **Ethereum 1.0** (July 2015–September 2022) and **2.0** (September 2022–
 7 present). **Ethereum**’s founders have stated that the primary goal of **Ethereum** is to expand
 8 on the idea of transaction scripts in **Bitcoin** and allow more verbose smart contracts. Since it
 9 shares many common details with **Bitcoin**, I only survey the significant points of difference.
 10 First, I will describe smart contracts, the original differentiator of the **Ethereum** protocol,
 11 in §B.1. Second, I will describe unique aspects of the cryptographic primitives used in the
 12 **Ethereum** protocol in §B.2. Third, in §B.3, I will describe the consensus protocols (Proof of
 13 Work and Proof of Stake) that **Ethereum** has used. Finally, I will describe the creation and
 14 distribution of **ETH** in §B.4.

15 **B.1 Smart Contracts**

16 While **Bitcoin** allows transactions to be customized through scripting, the scripting lan-
 17 guage is very restricted to prevent denial-of-service attacks or other unanticipated behaviour.
 18 **Ethereum** was proposed as a **Bitcoin**-variant with a full-fledged scripting language that any
 19 user can use. Instead of attaching the scripts to a transaction, users can push a code as a
 20 stand-alone “contract” to **Ethereum** where it will be assigned an address, and its code and
 21 data will be stored on the blockchain at that address. After deploying a contract, users
 22 can use one type transaction to interact with deployed contracts—in addition to “standard”
 23 transactions of **ETH** payments between a sender and set of receivers. With contract trans-
 24 actions, users specify a function that is implemented by the code of the contract, and ask
 25 for the function to be run on user-supplied data (“parameters”). Miners will run the code,
 26 update the data stored in the contract (“state variables”), and propose the result (“state
 27 transition”) in a block.

The term “smart contract” is a misnomer to some extent. Contracts are essentially computer programs or applications. They are sometimes called “decentralized applications” or Dapps instead. Contracts are stored on **Ethereum** and can be interacted with by users sending commands to them. These commands are called transactions but they are general interactions, although users have to pay for the interaction, so a financial transactions is always a component—more below on how users pay for transactions. The most popular smart contracts, according to the website DappRadar,³⁸ allow gambling platforms, games, social platforms, token exchange services, digital art, and financial applications.

B.2 Primitives and data structures

There are many small differences in data structures and algorithms between **Ethereum** and **Bitcoin**, but they are immaterial for the purposes of this report (*e.g.*, Merkle trees are augmented with a Patricia trie to maintain key-value pairs³⁹). I highlight two ways **Ethereum** (like the XRP Ledger, as will be discussed below) differentiates itself from **Bitcoin**. The first is that **Ethereum** maintains a set of balances for each user address in **ETH** and transactions increment/decrement these balances. This is in opposition to the UTXO model of **Bitcoin**. The second is that **Ethereum** commits the current state (state root) of all contracts on **Ethereum** in every block, whereas **Bitcoin**’s current state (*e.g.*, UTXO pool) requires a full node to run all **Bitcoin** transactions from the genesis block to the present.

B.3 Ethereum consensus

The consensus method of **Ethereum** was originally Nakamoto Consensus and it operated essentially the same as **Bitcoin**, using “proof of work” sybil-resistance (*i.e.*, based on computational puzzles), although blocks were produced every ≈ 12 seconds (rather than every 10 minutes). **Ethereum** then slowly switched to an alternative called “proof of stake.” Important protocol changes affirming this change were adopted, through in-protocol voting, beginning in 2020

³⁸ “Top Blockchain Dapps,” Dapp Radar, Retrieved Feb–May 2023.

³⁹ “Merkle Patricia Trie,” ETH Documentation, Retrieved Feb–May 2023.

1 and other changes rolled out up to the finalization of the switch (called “the merge”) in
2 September 2022 through a hard fork called the “Paris” upgrade.

3 Post-Paris, **Ethereum** has dropped the computational puzzles as the mechanism that
4 makes “sybil identities” expensive to create. Since computational puzzles, by design, require
5 large computational resources to be competitive in **Bitcoin** and pre-Paris **Ethereum**, this
6 change reduces the energy consumption of **Ethereum** by orders of magnitude, a change that
7 coheres with environmental concerns over **Nakamoto Consensus**.

8 Post-Paris **Ethereum** aims to also make sybil identities expensive to create but uses a
9 more direct mechanism: anyone can serve as a validator but they must first obtain and lock
10 up (or “stake”) 32 **ETH** (approximately \$60K USD at time of writing) as a fidelity bond
11 for participating in a timely manner and taking correct actions (according to the majority
12 of validators). By participating, they will earn **ETH** through rewards and fees (described
13 below) but they also could be penalized and lose a fraction of their staked **ETH** (“slashed”)
14 for performing predefined actions that are considered malicious by the protocol. Rewards,
15 fees, and slashing are all automated within the **Ethereum** protocol itself, and do not require
16 any external adjudicator or authority.

17 **Ethereum’s** proof of stake consensus has similarities to both **BFT** protocols and **Nakamoto**
18 **Consensus**. Like **Nakamoto Consensus**, it is permissionless: anyone able to stake 32 **ETH** is
19 able to join the set of validators. Proof of stake is sybil-resistant since validators wanting to
20 inflate their influence need to stake more **ETH**, which entitles them to have greater influence.
21 **Ethereum** benefited from having run **Nakamoto Consensus** for many years before switching,
22 as this circulated **ETH** widely and diversified the set of people holding it. Blockchains that
23 implement proof of stake from the very beginning are challenged to ensure the digital asset
24 to be staked does not start off concentrated in the hands of a few.

25 Since staking is an on-chain action, the list of validators is visible to everyone at all times.
26 The result is a sybil-resistant list of validators that anyone can join or leave at any time.
27 With a list of validators, **Ethereum** can then use a traditional **BFT** protocol to complete the
28 consensus mechanism.

1 The **Ethereum** protocol creates an “epoch” which is a sequence of 32 “slots” for the
 2 creation of the next 32 blocks. Slots are produced every 12 seconds and an epoch is completed
 3 every 6.4 minutes. **Ethereum** uses an in-protocol system to assign a validator at random to
 4 each slot. Technically, **Ethereum** cannot produce truly random numbers because the numbers
 5 would be part of a block, and a validator has a lot of latitude over what they chose to include
 6 in a block, what they do not, and whether they even broadcast the block they create.
 7 These are opportunities to bias any randomness inside a block. **Ethereum** uses an elaborate
 8 procedure (called **RANDAO**) that we will not detail here. In short, **RANDAO** harvests randomness
 9 from validators and produces psuedo-random numbers that are difficult to predict or bias in
 10 any way, assuming a quorum of honest validators.

11 Once an epoch is assigned, a chosen validator for a slot waits until the slot is reached (12
 12 seconds between each slot) and then proposes a block of fresh transactions that have been
 13 broadcast to the validators (*i.e.*, are in the **mempool**) but have not been included yet in any
 14 previous block (in any previous slot). Once a set of 32 blocks is created, other validators
 15 vote on the validity of the epoch. The epoch is considered final when it receives votes from
 16 validators representing 2/3 of all the **ETH** staked by validators.

17 Validators that do not participate in timely manner, sign conflicting messages (equivocate), or perform other faulty/malicious actions that can be adjudicated by the **Ethereum**
 18 protocol itself will be penalized.⁴⁰ For minor infractions (*e.g.*, going offline), validators will
 19 simply forgo the rewards (described below) they would otherwise have earned. Major infrac-
 20 tions (*e.g.*, voting both for and against a fork) will see a fine levied against their deposited
 21 **ETH** (‘slashing’), and they will lose their validation status if their deposit ends up below 32
 22 **ETH**. In contrast, new **ETH** is provided to validators that participate actively and perform
 23 actions that align with the majority of other validators. Validators are thus economically
 24 incentivized to align their actions with the majority of other validators and punished when
 25 they fail to do so. All rewards, fees, and slashing are fully automated within the **Ethereum**
 26 protocol itself, and do not require any external adjudicator or authority.

⁴⁰ “Proof-of-stake rewards and penalties,” **ETH** Documentation, Retrieved Feb–May 2023.

1 B.4 Issuance and fees

2 **Initial issuance of ETH.** The method for allocating ETH in Ethereum is different from
 3 Bitcoin. Ethereum began with an ‘initial coin offering’ (or ICO) of ETH, where 60,000,000
 4 ETH was auctioned (for BTC). An additional 12,000,000 was given to Ethereum developers
 5 directly (or indirectly through a fixed price purchase program) and to an endowment fund for
 6 investing in Ethereum technology overseen by the *Ethereum Foundation*. This allocation took
 7 place before the Ethereum software existed or the Ethereum blockchain had been deployed,
 8 so it was a pledge to allocate ETH according to the auction once Ethereum was created.
 9 Approximately one year later, Ethereum was deployed with this allocation encoded into its
 10 first (genesis) block (and the genesis block is hardcoded into the original Ethereum software
 11 `geth`).

12 **Ethereum rewards.** After the ICO of ETH, new ETH continues to be issued over time.
 13 All new ETH is given to validators who participate in the consensus mechanism. Pre-Paris,
 14 it worked essentially the same as Bitcoin with some minor differences. For example, a miner
 15 who created a valid block and solved the computational puzzle for it, only to realize they
 16 were beaten by a slightly faster miner, could still receive a fraction of the block reward for
 17 it—it would be stored on the blockchain as an “uncle block” that is not part of the heaviest
 18 chain.

19 Post-Paris, every slot is assigned, unpredictably, to validator from the set of validators,
 20 called the “block leader” or “block proposer.” If this validator is online and live, it will
 21 propose a block and receive a reward in new ETH if the block is validated (as part of the
 22 epoch of slots). Other validators who vote on the validity of an epoch, and their votes
 23 match the quorum, are also rewarded with new ETH for participating in this portion of
 24 the consensus. This is different than Bitcoin and pre-Paris Ethereum, where miners do not
 25 explicitly vote on the validity of blocks, they implicitly support a block by proposing new
 26 blocks that extend it.

27 An additional difference between BTC and ETH is the “inflation schedule.” Bitcoin is

1 capped at 21M BTC and decreases the block reward over time according to its programmed
 2 schedule. **Ethereum** has no cap and **ETH** rewards in **Ethereum** do not decrease over time
 3 as in **Bitcoin**. The amount of **ETH** in circulation grew (“inflationary”) until recent changes
 4 (described next) added a type of user fee that burns **ETH** (called the base fee). At the time of
 5 writing, the **ETH** burned through fees has outpaced the **ETH** created through rewards since
 6 these changes, but both rates are dynamic and depend on the conditions of the network
 7 (*e.g.*, the amount of congestion and the amount of **ETH** staked by validators).

8 **Ethereum fees.** To ensure validators are fairly compensated and to combat malicious ac-
 9 tors from stalling the network (“denial of service” attacks) by asking for a long-running
 10 computation to be performed, all computations are broken into small steps (“instructions”
 11 or “opcodes”) where each step is assigned a value in a unit called “gas.” The value represents
 12 how complex the computation step is to execute or store (*e.g.*, a multiplication has a higher
 13 gas value than an addition).

14 As with consensus, **Ethereum** has changed how gas works through a hard fork. In this
 15 case, **Ethereum**’s gas model was largely unchanged until a hard fork called “London,” which
 16 was deployed about a year before “Paris,” and was designed to make fees more equitable,
 17 particularly in times of network congestion. The differences are not important to the opinions
 18 in this report, so I will explain only the post-London fee structure. Post-London in **Ethereum**,
 19 users pay two types of fees. The first component is the priority fee: the user specifies a rate
 20 of **ETH** per unit of gas that they are willing to pay as a fee to the validator who includes
 21 their transaction in a block. This works like fees in **BTC**—the user is bidding to have
 22 their transaction included ahead of other user transactions. In practice, the user’s software
 23 examines the current conditions of **Ethereum** and suggests a rate to the user.

24 The second component is the base fee: the blockchain specifies a rate of **ETH** per unit of
 25 gas that is a mandatory fee and is burned from circulation once paid. The base fee dynam-
 26 ically increases (and decreases) in value if the **Ethereum** networks becomes more congested
 27 (less congested) with transactions. This creates an incentive for users to wait during times
 28 of congestion. The main takeaways are: (1) all computations cost the user **ETH** in fees, (2)

1 more complex computations cost more than simpler ones, and (3) validators earn revenue
2 by performing computations on **Ethereum**.

3 To prevent users from asking for a computation to be run without realizing it will consume
4 more **ETH** than they are willing to pay, users can cap the maximum amount they will pay for
5 a computation. If a cap is used and a computation “runs out of gas” before it is completed,
6 the user will lose their entire fee but no more than it. The validator will abandon the
7 computation at the point that it runs out of gas, record an error on the blockchain, and
8 “revert” any changes that the computation made (leaving it as if the computation was never
9 run in the first place).

10 **Internal incentives.** The main takeaway of issuance and fees in **Ethereum** is that valida-
11 tors earn **ETH** by participating in consensus. Specifically, they are awarded new **ETH** and
12 they also earn fees from users. These rewards provide an incentive to validate **Ethereum**
13 transactions. Today the total **ETH** that has been issued is $\approx 120,000,000$. The fees col-
14 lected by validators along with the rewards from participating in consensus creates revenue
15 for those participating. Like Bitcoin miners, **Ethereum** validators are enterprising “for-profit”
16 entities that choose to operate **Ethereum** nodes because it is profitable.

C The XRP Ledger protocol and implementation

The XRP Ledger, like Bitcoin and Ethereum, is a blockchain. The native digital asset of XRP Ledger is XRP.

At the outset, I would like to provide a note on vocabulary. I understand that the XRP Ledger has also been referred to by other names including Ripple, the Ripple Consensus Protocol, and the Ripple Protocol. For purposes of this report, I will use XRP Ledger exclusively. Similarly, I understand that XRP has also been referred to by other names, including ripples. For purposes of this report, I will use XRP exclusively. I use *Ripple Labs* to refer to the company also known as Ripple and previously known as NewCoin and OpenCoin.

First, I provide an overview of XRP Ledger in §C.1. Second, I describe XRP Ledger's unique cryptographic primitives and data structures in §C.2. Third, I describe the XRP Ledger consensus protocol (XRP-LCP) in §C.3. Fourth, I describe the transactions that can be accomplished using XRP Ledger in §C.4. Finally, I discuss XRP and the lack of validator fees in §C.5.

C.1 Overview of the XRP Ledger and XRP

The XRP Ledger is a distributed system that “stores and processes transactions to move XRP and other digital assets.”⁴¹ The software code that runs the XRP Ledger is known as *rippled* and was created by *Ripple Labs*.⁴² Development began on the *rippled* code in 2011 by individuals including Ripple co-founder Jed McCaleb and current Ripple Chief Technology Officer David Schwartz.⁴³ The current version of the XRP Ledger was deployed sometime in December 2012. The ledger was created when three validators, also known as servers, ran the *rippled* code and began agreeing on ledgers. These three validators were run by McCaleb and Schwartz.⁴⁴ The exact date of deployment is unknown because the initial

⁴¹Disclosure, Document RPLI00339374.

⁴²McCaleb Depo. at 75-76

⁴³McCaleb Depo. at 13-17; Schwartz Depo. at 54-62.

⁴⁴Disclosure, Schwartz Depo. at 101-02.

ledgers, 1–32,569, including the genesis block at ledger 1, were lost due to an issue with validator storage space. The oldest existing block 32,570 was created on 01 Jan 2013. The XRP Ledger was apparently reset a number of times before the current deployment—McCaleb testified that resetting the XRP Ledger meant “resetting the network,” which meant that they started the whole network again and any pre-reset transaction data would be forgotten.⁴⁵

XRP is a fungible digital asset that can be divided into one million subunits called drops. The XRP in existence today were created when McCaleb’s and Schwartz’s validators began running the `rippled` code for the current version of the XRP Ledger in 2012. No XRP has been created since December 2012, and the creation of additional XRP in the future is unlikely. The overall supply of XRP is instead decreasing, as each transaction on the XRP Ledger requires the destruction of a variable amount of XRP to prevent spam attacks.

C.2 Primitives and data structures

Hash Functions. Like the Bitcoin protocol, the XRP Ledger protocol uses hash functions to generate unique commitments to data and both use the SHA-2 family of hash functions. As of February 2023, the SHA2 family of hash functions are considered collision-resistant and preimage-resistant by NIST.⁴⁶

Hash Trees. As in the Bitcoin protocol, the XRP Ledger protocol will commit to multiple data values by accumulating them in a hash tree structure, where the root value of the tree represents all the data values in the tree. Verifying a data value was included in a leaf of the tree is accomplished by providing the hash values along the path from the root to leaf.⁴⁷

Proof-of-Work. Unlike the Bitcoin protocol, the XRP Ledger protocol does not utilize proof-of-work puzzles anywhere in its protocol, nor does it use the proof-of-stake protocol of Ethereum. The consequences of this will be described below in §C.3.

⁴⁵McCaleb Depo. at 72–75; Schwartz Depo. at 69–76, 84, 99–108.

⁴⁶“Hash Functions,” NIST Computer Security Resource Centre, Retrieved Feb–May 2023.

⁴⁷“Ledgers: Tree Format,” XRPL Documentation, Retrieved Feb–May 2023.

Digital Signatures. As in Bitcoin, participants in XRP Ledger are identified pseudonymously by values called addresses, which are hashes of the public key for a digital signature scheme. The XRP Ledger protocol supports as default the signature scheme called ECDSA (elliptic curve digital signature algorithm) over the elliptic curve secp256k1 with DER-encoded parameters (as in Bitcoin). XRP Ledger later added support for curve Ed25519 which offers greater transparency and is hardened against side-channel attacks.⁴⁸

Ledger Data Structure. The Bitcoin, Ethereum, and XRP Ledger protocols all propose updates to the record of all transactions with a batch of new and validated transactions. In XRP Ledger, the update is called a ledger (*cf.* a block in Bitcoin and Ethereum) and is appended to the XRP Ledger (*cf.* the blockchain in Bitcoin and Ethereum). A finalized ledger will contain the following (non-exhaustive list of) core data elements:⁴⁹

⟨seq, hash, time, prevLedger, status, transactionSet, stateData⟩

The **seq** parameter is a sequence number (or index index) assigned to the ledger by the validator. The ledger index starts at 1 for the genesis ledger, and increments by 1 for each subsequent ledger. Ledger updates happen every 3-5 seconds.⁵⁰ Ledgers 1 to 32,569 (inclusive) are lost and modern clients begin from ledger 32,570⁵¹ (protocol differences from Bitcoin described shortly ensured that no balance of XRP was lost as a result). Near the time of writing, an example sequence number is 78,000,000 finalized on February 23, 2023 at 09:38:32 PM UTC containing 47 transactions.⁵²

The **hash** parameter is a hash-based commitment to all of the data elements of the ledger. Like **seq**, **hash** serves (with overwhelming probability) as a unique identifier for the ledger, and it additionally ensures the contents of the ledger are binding. However **seq** is used as an extra index, which offers a user-friendly way to traverse the XRP Ledger. The **time** parameter is a

⁴⁸“Cryptographic Keys: ed25519 Key Derivation,” XRPL Documentation, Retrieved Feb–May 2023.

⁴⁹“XRP Ledger Protocol Reference,” XRPL Documentation, Retrieved Feb–May 2023.

⁵⁰“Decentralized Exchange,” XRPL Documentation, Retrieved Feb–May 2023.

⁵¹“Configure Full History,” XRPL Documentation, Retrieved Feb–May 2023.

⁵²“Block 78000000,” XRPL Explorer, Retrieved Feb–May 2023.

roughly correct assertion of finalization time for the ledger. In a distributed network, there is no ‘wall clock’ to reference which accounts for why this parameter is imprecise and validated within a certain tolerance. The **prevLedger** parameter references the **hash** from the previous ledger to make the **XRP Ledger** conform to a hashchain data-structure. As in the **Bitcoin** and **Ethereum** protocols, this means that the most current ledger is a binding commitment to every previous transaction going back to ledger index 1. The **status** parameter indicates if the ledger is finalized yet (*i.e.*, open or closed in **XRP Ledger** terminology; discussed further in §C.3). The ledger contains a hash-tree commitment to all transactions in the **transactionSet** parameter where transactions are sequenced in a specified order. The types of possible transactions will be discussed in §C.4.

The final illustrated parameter is **stateData**, which reflects a material deviance from the design of the **Bitcoin** protocol. Consider how a **Bitcoin** node might obtain a user’s current balance. This cannot be determined from the most recent block. The node must have reconstructed the balance from the entire blockchain, starting with the genesis block. Thus each node in **Bitcoin** begins by building the **utxo-pool** to determine the current state of accounts, balances, and anything else future valid transactions might make reference to. In contrast, **XRP Ledger** commits the current state of all accounts in every ledger with hashtree commitment.⁵³ This allows a new node to obtain a copy **XRP Ledger** archive from any untrusted source and validate its correctness using only the most recent ledger. Recall the 32,569 lost ledgers—while past transactions are indeed lost, the result of those transactions on each user’s balance is reflected in the **stateData** of ledger 32,570 and can be validated provided that one trusts that the signatories on ledger 32,570 (all of which were operated by Ripple Labs) would not have signed if they did not validate ledgers 1 to 32,569. In addition to account balances, **stateData** also commits to data used in more exotic payment types (escrow, payment channels, checks, *etc.*) and non-payment transactions (*e.g.*, non-fungible tokens). It tracks proposed amendments to the **XRP Ledger** protocol that may be open for voting and offers a mechanism to flag offline validators (see §C.3). Several other less

⁵³ “Ledger Data Formats: State Data,” XRPL Documentation, Retrieved Feb–May 2023.

significant data fields are also committed to by `stateData`.

C.3 XRP Ledger consensus

In this subsection, I discuss the unique consensus mechanism used by XRP Ledger to validate transactions for the ledger.

I begin with another note on terminology. I use the term *full node* for a network participant that propagates data through the peer-to-peer network. I use the term *validating node* or *validator* for a node that additionally participates in the consensus mechanism I will describe below. Finally, I use the term *recommended validator* for a validating node which is (or has been) part of a list of validators distributed by *Ripple Labs*. This list is a preset or default in `rippled`, which is the original software implementation of XRP Ledger.

Peer-to-peer network. Like Bitcoin, the XRP Ledger protocol uses a peer-to-peer network.⁵⁴ Each full node (including validators) maintains a set of known nodes and can indirectly reach the rest of the full nodes on the network by having nodes propagate all messages—which, as previously discussed, is called a gossip protocol. `rippled` provides a hardcoded set of nodes on first use. Messages include pending transactions, proposed ledger updates, consensus messages, and requests for archived transactions and ledgers. Gossip protocols are effective although generally inefficient due to redundancies in received messages and agnosticism over the network topology.

Consensus landscape. The XRP Ledger’s process of reaching consensus on a new ledger is materially different than Bitcoin and pre-merge Ethereum (Nakamoto Consensus based on proof of work), and post-merge Ethereum (proof of stake). Chase and MacBrough (with *Ripple Labs* listed as their affiliation) describe the XRP Ledger Consensus Protocol (XRP-LCP) as belonging to the family of Byzantine fault tolerant (BFT) protocols [8]. As discussed above, research on BFT protocols for distributed systems was popularized in the 1980s and predates Bitcoin and XRP Ledger [20]. A ‘classic’ BFT protocol (*e.g.*, PBFT [6]) applied to

⁵⁴“Peer Protocol,” XRPL Documentation, Retrieved Feb–May 2023.

a blockchain data structure with blockchain terminology, would assume that all validating nodes are specified in a list generated by a trusted entity, and all validating nodes have the same list of validating nodes. With this assumption in place, validators would proceed to decide on blocks/ledgers. Nakamoto Consensus is a material departure from ‘classic’ BFT protocols (*e.g.*, Castro and Liskov’s Practical BFT protocol—PBFT [6]) because Nakamoto Consensus allows anyone to join and leave the network as a validator without permission or authorization (this is sometimes called a permissionless blockchain). The XRP Ledger, by contrast, is more grounded in a ‘classic’ BFT approach, as described below.

Validator lists. In the XRP Ledger protocol, every validator maintains a list of validators considered by the node to be available (online) and trustworthy, where trustworthy means they do not deviate from the XRP-LCP. Examples of deviations include buggy software, malicious behaviour, censorship, profit opportunities, and accepted bribes. In a classic BFT, a centralized entity would govern over a `validatorList` that every node is *required* to use. XRP Ledger is the same except the `validatorList` that is the default preset in `rippled` is described as *recommended* instead of *required*. One of my opinions is that there are significant potential risks in deviating from the `recommendedValidatorList` and using a customized `validatorList`. Validators should thus treat the `recommendedValidatorList` as a *de facto required*.

Another note on terminology: XRP Ledger documentation, whitepapers, and academic papers often refer to the `validatorList` as a *unique node list* (or UNL), and a `recommendedValidatorList` as the *default unique node list* (or dUNL). Our terminology is more consistent with comments and naming conventions in the `rippled` software. I note that, while not all nodes on the XRP Ledger network are validating nodes, all nodes on these lists are validating nodes. Second, I note that there is no restriction against the same organization having multiple nodes on a UNL or dUNL so validators are not necessarily ‘unique.’ Finally the term dUNL is used inconsistently to mean: (i) any list that a validator sets as its own UNL, (ii) the list of validators distributed from *Ripple Labs*, or (iii) the list of validators which is the default in `rippled`. As the default list of validators in `rippled` presently directs to a list distributed

by *Ripple Labs*, (ii) and (iii) are presently the same. For us, `recommendedValidatorList` means (iii), the list of validators which is the default in `rippled`.

From January 2013–June 2018, 100% of nodes on the dUNL were operated by *Ripple Labs*.⁵⁵

Quorum. Once a validator has selected a `validatorList`, it uses the list in determining when proposed ledger updates should be considered closed; this happens when the ledger has been validated by at least a given fraction of the validators on its `validatorList`. This fraction is called the `validation_quorum`.⁵⁶ The XRP-LCP currently specifies a quorum of 80%, which is consequentially hardcoded into `rippled`. All analysis of XRP-LCP known to us (including [22, 2, 12, 8, 18, 5, 1]) is premised on the quorum being 80%, thus the extent to which it can be considered a free parameter (that can be adjusted up or down) has not been adequately analyzed in the literature. While a validator strives to only add trustworthy validators to its `validatorList`, a quorum is used so that it can tolerate some number of validators acting untrustworthy.

Note that the number of untrusted validators that can be tolerated is complex, as this threshold is also dependant on other properties: *e.g.*, how much overlap the validator has with other validators in terms of its `validatorList`; how many validators are offline (and thus unreachable by all validators); how many validators are separated (but connected to the separated validators on a different partition of the network); and what security goal is being achieved (liveness or safety). These issues are discussed in turn below.

Consensus phases. The consensus process aims to quickly (‘liveness’) finalize valid XRP Ledger transactions, in the same order (‘total order’) across all validators (‘safety’). A simple transaction is a payment of some amount of XRP from one account (identified by a public key) to another account. However, the XRP Ledger protocol supports more advanced types kinds of transactions (see §C.4). Consensus is the process by which validators come to agree

⁵⁵Disclosure, Document RPLI.02460831.

⁵⁶“`server_info (rippled)`,” XRPL Documentation, Retrieved Feb–May 2023.

1 on a common, ordered sequence of transactions, updated in batches called ledgers.

2 Finalizing a ledger is a process that happens across three main phases and completes
3 within a set of time called the update interval. Ledgers are produced every 3-5 seconds,
4 which is materially faster than Bitcoin which targets a block every 10 minutes. Phases are
5 timed and validators must be able to reference their own clock which are assumed to be
6 synchronized (within a predefined tolerance) with the clocks of other validators.

7 The first phase of the consensus mechanism is dependent on the ledger that has been
8 built so far. To have validators coalesce around a single ledger, the XRP Ledger protocol
9 defines a heaviest chain (called the preferred ledger) for deciding between forks (more than
10 one different ledger at the same index). Unlike Bitcoin, the heaviest chain is not determined
11 by proof of work, but is determined by which branch has obtained the most signatures from
12 other validators on the validator's `validatorList`. A validator constantly monitors the
13 network for a its preferred ledger. If it ever discovers its own last ledger is not the preferred
14 ledger (i.e. does not have the most signatures from other validators on the `validatorList`),
15 it will abandon any progress on consensus, set the preferred ledger as its most recent ledger
16 and begin the first phase again.

17 In the first phase, all validators will add to their collection of unprocessed transactions,
18 pruning any transactions that are in an invalid format (*e.g.*, incorrect digital signature).
19 Validators learn of new transactions through the gossip protocol, which means the set of all
20 transactions known to one specific validator at a particular moment in time may differ from
21 another validator. Once half of the update interval has expired, the validators move to the
22 second phase.

23 In the second phase of the protocol, each validator converts its collection of transactions
24 into an ordered sequence of transactions. The ordering rule is global, so validators with
25 the same transactions will establish the same order. Transactions that have valid format
26 but cannot be finalized because of some other reason (*e.g.*, insufficient funds, action after
27 a time expiration, reference to something that does exist, *etc.*) are still added in sequence,
28 but are then reverted once they fail (returning the state of all accounts to its original state

before the transaction was attempted). Validators record an appropriate error code for failed transactions and still apply the transaction fee. Once a validator completes the process, it has a candidate ledger.

Still in the second phase, a validator circulates its candidate ledger through the gossip protocol. When a validator receives a candidate ledger from a different validator on its `validatorList`, it compares the sequence of transactions to its own candidate ledger and marks any transaction that does not appear in both as a disputed transaction. As the validator receives more candidate ledgers, it keeps a tally of how many validators are including or excluding each disputed transaction. Roughly speaking, if the validator finds itself in enough of a minority position with respect to a disputed transaction, it will modify its candidate ledger to match the majority position and rebroadcast. The thresholds at which a validator makes a modification to its candidate ledger are complex and change as time passes [1]. If and when a validator has 80% agreement from its `validatorList` on every disputed transaction, it considers itself as reaching quorum. The decision of the quorum on each disputed transaction (*i.e.*, to include or exclude it) is applied.

In the final phase, the validator takes the final set of transactions and computes the remaining values of the ledger, in particular `stateData`. Then the validator signs the ledger and sends it through the gossip protocol to the other validators. The validator considers this ledger as closed, and returns to first phase of the protocol in order to generate the next ledger. However as it is collecting transactions for the next ledger, it is also tracking how many signatures the closed ledger has received, as well as any other closed ledgers that are gathering signatures. A ledger is considered finalized by a validator once it obtains L signatures from L different validating nodes on a `validatorList` of $L + L/5$ nodes (*i.e.*, an 80% quorum).

Overlap requirements. Consider two validating nodes, Alice and Bob, with `validatorList` A and B respectively. They accept a quorum of 80%. A and B have c common nodes ($c = A \cap B$), while Alice has a nodes unique to her ($a = A \cap (\neg B)$) and Bob has b nodes unique to him ($b = B \cap (\neg A)$). Assume for convenience that Alice and Bob have the lists of

the same size, otherwise the entity with the larger list is called Alice. This implies: $|A| \geq |B|$; or $(a + c) \geq (b + c)$; or more simply $a \geq b$.

The following claims are made in the literature:

1. The original whitepaper claims: $|c| \geq \frac{1}{5}|A|$ to ensure safety and liveness [22]. Stated otherwise, two nodes must share at least 20% common nodes to ensure safety and liveness.
2. Armknecht *et al.* claims: $|c| \geq \frac{2}{5}|A|$ to ensure safety and liveness [2]. Stated otherwise, two nodes must share at least 40% common nodes to ensure safety and liveness.
3. Chase and MacBrough claim: $|c| \geq \frac{9}{10}|A|$ to ensure safety [8]. Stated otherwise, two nodes must share at least 90% common nodes to ensure safety and liveness.

Analysis of more than two validators is complicated by the fact that overlaps are defined pairwise between each pair of validators, and they are non-commutative. To illustrate this, consider Alice, Bob and Carol. Alice and Bob may overlap by 90% and Bob and Carol may overlap by 90% but this does *not* imply that Alice and Carol overlap by 90%: in fact, Alice and Carol could overlap by anywhere between 80% and 100%. Modelling 50 validators would require an assumption about 1225 overlaps between each pair of validators. Without empirical data about overlaps between validators in XRP Ledger network, modelling is speculative. Instead, researchers tend to show minimum conditions for a breach of liveness or safety.

Safety. Amores-Sesar *et al.* consider a model where $2n$ honest validators use one of two possible lists: consider that half will use Alice's list for example, and half will use Bob's. The model adds f malicious nodes who are assumed to be in both lists from Alice and Bob ($f = |\text{faulty}_A| = |\text{faulty}_B|$). To simplify, assume the lists are equal in size, $|a + c| = |b + c|$, and thus the overlap, $\omega = \frac{|c|}{|a|+|c|}$, is equal for both sets of validators. They show a safety violation when f exceeds the following value:

$$|\text{faulty}_A| \geq 2n \cdot \frac{5\omega - 2}{12 - 10\omega}$$

In practical terms, assume 9 of the 35 validators currently on the `recommendedValidatorList` were malicious, they could violate the safety of XRP-LCP if half of the validators used `recommendedValidatorList` and half used a different `validatorList` that still had a 60% overlap with `recommendedValidatorList`.

Liveness. In perfect conditions, every validator uses the exact same `validatorList` and 0 nodes on the `validatorList` are malicious. In this case, XRP-LCP maintains liveness. However if these perfect conditions are disrupted at all, liveness fails. Chase and MacBrough show a liveness violation when the overlap between validators decreases from 100% to 99% (even with 0 malicious nodes). Amores-Sesar *et al.* show a liveness violation when the number of malicious nodes increases from 0 to 1 (even with 100% overlap).

Network partitions. Consider a network partition of validators in a distributed system. Researchers consider a common assertion, called the CAP theorem [14], when reasoning about partitions and I describe it informally as follows. As an example of a partition (the ‘P’ of CAP), assume half of the validators are connected to each other on network A but inaccessible, directly or indirectly, from the other half of validators which are connect to amongst themselves on network B. The theorem asserts that a system must sacrifice consistency (the ‘C’) or availability (the ‘A’) (or both). A system prioritizing consistency over availability might halt in all but one network, preserving one consistent record. A system prioritizing availability over consistency would continue operations on both networks, resulting in inconsistency.

The CAP theorem was not articulated with blockchain technology in mind and requires some adaptation to be useful at a technical level [17, 21]. However at a high level, it can illustrate some properties of the XRP Ledger protocol. If quorum is set at 80% of validators and all validators have a common `validatorList`, XRP Ledger prioritizes consistency over availability. Three cases are possible: (1) network A has quorum and network B stalls; (2) network B has quorum and network A stalls; or (3) neither network has quorum and both stall. None of the cases will result in inconsistent ledgers being finalized.

1 If the validators do not share a common (or highly overlapping) `validatorList`, then
 2 XRP Ledger fails to guarantee consistency. For example, if all validators on network A have a
 3 `validatorList` with only validators from network A and likewise with network B, there will
 4 be a logical partition even if the networks are fully connected. Both networks will finalize
 5 their own independent ledgers, providing availability but not consistency. In XRP Ledger
 6 documentations, this is referred to as parallel networks: “When different consensus groups
 7 of rippled instances only trust other members of the same group, each group continues as
 8 a parallel network.”⁵⁷ Preventing forks like these is a reasonable explanation for why XRP
 9 Ledger documentation and `rippled` configuration files strongly discourage modifying the
 10 `recommendedValidatorList`.

11 Proof of work protocols are less intuitive to analyze. If half the computational power is on
 12 network A and half on network B, both will continue to operate producing different chains of
 13 transactions. At first glance, this appears to prioritize availability (both networks continue)
 14 over consistency (the chains are different). However it is not technically inconsistent because
 15 neither chain is considered final. If the partition is removed, one chain will have the heavier
 16 chain and the other chain will be discarded. The actuality of the situation is that the network
 17 with lighter chain is unknowingly suffering from unavailability, but no one on the network
 18 realizes it at the time, as it appears operational.

19 **Validator lists.** The most used validator software on XRP Ledger is `rippled` from *Rip-*
 20 *ple Labs*. The `rippled` client contains a configuration file for directly or indirectly listing
 21 trusted validators.⁵⁸ The current recommendation, communicated through a comment in
 22 the configuration file, is that a `validatorList` is hosted online at a specified URL and is
 23 signed by a signing key that corresponds to a specified public key. The URL and public
 24 key for the `validatorList` distribution point are placed in the configuration file. This is in
 25 contrast to listing validator keys directly in the configuration file.

26 The motivation for this indirection to a distribution point is that modifications to the

⁵⁷ “Parallel Networks and Consensus,” XRPL Documentation, Retrieved Feb–May 2023.

⁵⁸ “`validators-example.txt`,” GitHub, Retrieved Feb–May 2023.

1 `validatorList` can be made by the entity that controls the `validatorList`, and be accepted
 2 by `rippled` clients without requiring a software update or other user action. Thus indirection
 3 can be used for agility in the face of an attack by malicious validators, by providing the entity
 4 controlling the distribution point, in this case *Ripple Labs*, to remove validators that are
 5 malicious, offline, unresponsive, or unable to cope with the level of network communication
 6 required. However, indirection also creates the risk that the distribution point itself becomes
 7 unresponsive or is taken over. It also allows *Ripple Labs* to change or remove validators from
 8 the `validatorList` without any action or approval from the validators. This gives *Ripple*
 9 *Labs* significant control over the consensus process.

10 A recent modification to `rippled` adds the ability for nodes to gossip about validators
 11 that are non-responsive. If this “negative” `validatorList` (negative UNL or `nUNL`) achieves
 12 80% support from other validators on a given validator’s `validatorList`, the validator is
 13 not considered during the consensus phase. The motivation is to enhance liveness in the
 14 scenario that validators go offline slowly over time, however does not help if many validators
 15 go offline simultaneously.

16 Finally, lists expire quickly (*i.e.*, within weeks) and it is necessary that at least one node
 17 on the network is able to obtain a fresh list, otherwise nodes will not be able to operate a
 18 `validatorList` (impacting the liveness of XRP-LCP). To mitigate this, `rippled` allows for
 19 more than one distribution point to be listed.

20 **Coordination through presets.** As previously noted, maintaining a `validatorList` (or
 21 UNL) that overlaps materially with other validators’ lists is essential for the correct operation
 22 (liveness and safety) of XRP-LCP. The *XRP Ledger Foundation* warns on their website: “if
 23 your UNL does not have enough overlap with the UNLs used by others, there is a risk that
 24 your server forks away from the rest of the network. As long as your UNL has > 90%
 25 overlap with the one used by people you’re transacting with, you are completely safe from
 26 forking. If you have less overlap, you may still be able to follow the same chain, but the
 27 chances of forking increase with lower overlap, worse network connectivity, and the presence

1 of unreliable or malicious validators on your UNL.”⁵⁹

2 To affirm overlapping lists, **rippled** comes with a set of presets or defaults. Until
 3 July 2021, **rippled** contained only one distribution point for obtaining a **recommended-**
 4 **ValidatorList**. This distribution point was maintained by Ripple Labs itself at **https:**
 5 **//vl.ripple.com**. A second distribution point was added in July 2021 and is only refer-
 6 enced by the client under failure to reach the first list. The second distribution point is
 7 provided by *XRP Ledger Foundation* at **https://vl.xrplf.org**. At the time of writing,
 8 both preset distribution points have 100 percent overlap and contain an identical set of 35
 9 validators.

10 While the software does not provide additional preset distribution points, some ledger
 11 explorers (e.g., XRPSCAN) also reference the distribution point from the company *Coil* at
 12 **https://vl.coil.com**. At the time of writing, *Coil*’s list is no longer available.

13 To the extent that accepting the *Ripple Labs* list of 35 validators is considered canonical by
 14 **rippled** clients, the overlap issue is solved. The concept of these being canonical is reinforced
 15 through comments in the code such as, “Changing [the distribution points] can cause your
 16 **rippled** instance to see a validated ledger that contradicts other **rippled** instances’ validated
 17 ledgers (aka a ledger fork) if your validator list(s) do not sufficiently overlap with the list(s)
 18 used by others.”⁶⁰ The canonical nature of the recommended lists are also reinforced by
 19 the *XRP Ledger Foundation* network visualizer, which flags these validators with a special
 20 “UNL” visual cue in the list of validators.⁶¹

21 **Risks of presets.** Given the presets in **rippled** and the strong warnings against deviat-
 22 ing from them, *Ripple Labs* is a *de facto* single point of failure for XRP-LCP. *Ripple Labs*
 23 can modify the **recommendedValidatorList** to include only validators under its control and
 24 overtake the network. Under such a scenario, nodes would require out-of-protocol coordina-
 25 tion to recover.

⁵⁹*XRP Ledger Foundation* FAQ

⁶⁰GitHub

⁶¹XRPL Live Data, 2023.

For most of its history, not only did *Ripple Labs* maintain the `recommendedValidatorList`, but it also operated all or a super-majority of the validators on the list. At these times, a broad compromise of *Ripple Labs* infrastructure would have been sufficient to attack the liveness and safety of the XRP-LCP. While distributing trust to multiple, independent servers reduces risk, blockchain systems with a small number of independent validators have been attacked in a coordinated fashion, where the adversary gains access to multiple authorities simultaneously (*cf.* attacks on Ronin Network which required 5 of 9 validators⁶² and Harmony Bridge which required 2 of 5 validators⁶³).

C.4 Transactions

Recall that transactions in Bitcoin are technically specified by the users of the system through a scripting language. While common scripts (with names like P2PKH or P2SH) account for most transactions, technically Bitcoin allows any type of transaction that can specified within its scripting language. Ethereum allows the same thing except with a verbose scripting language that allows essentially any computation (“Turing completeness”), as long as it can run to completion within Ethereum’s gas limit for a block.

By contrast, users in XRP Ledger can only use transaction types that have been predefined, or are added to the protocol, currently through an amendment system that requires an 80% quorum of the `recommendedValidatorList`.⁶⁴ One common transaction type is `payment` which transfers a cryptoasset like XRP from account to another (possibly new) account. Payments can also be for non-XRP tokens that also exist in the XRP Ledger. More advanced payment types, including checks and escrows, are supported. Ripple also provides an on-ledger orderbook for trading assets, with transactions for creating and canceling offers (*i.e.*, buy/sell limit orders).⁶⁵

⁶² “Ronin Network,” rekt.news, Retrieved Feb–May 2023.

⁶³ “Harmony Bridge,” rekt.news, Retrieved Feb–May 2023.

⁶⁴ “Amendments,” XRPL Documentation, Retrieved Feb–May 2023.

⁶⁵ “Transaction Types,” XRPL Documentation, Retrieved Feb–May 2023.

1 C.5 Issuance and fees

2 Unlike in Bitcoin, where the circulation of BTC begins at zero and is released over time
 3 to miners, all 100B XRP (*née* XNS) units were created at the start of the ledger. The
 4 ledger starts when validators start producing and closing ledgers. The genesis ledger date
 5 is unknown but can be extrapolated from the oldest known ledger (32,570 in January 2013)
 6 back to mid-December 2012 since ledgers are updated every few seconds. From testimony,
 7 *Ripple Labs* initially provided the three original validators⁶⁶ that operated the XRP Ledger
 8 through to 2013 when *Ripple Labs* started expanding with additional validators operated by
 9 *Ripple Labs*, and eventually (after June 2018) validators operated by other entities.⁶⁷

10 The total supply of 100B XRP is hard coded in the `rippled` code and the current version
 11 of the `rippled` code does not allow for the creation of additional XRP. No XRP has been
 12 created since the beginning of the XRP Ledger. Of the 100B XRP, 80B units were allocated
 13 to *Ripple Labs* and the remaining 20B units were given to the founders of the project.

14 Importantly, validators are not rewarded with newly created XRP, as in Bitcoin/Ethereum,
 15 which means they operate without this revenue stream and internal incentive. In fact, they
 16 are not rewarded by fees either. Fees are charged to the sender of a transaction but the XRP
 17 is removed from circulation (“burned”) instead of being paid to validators. This reduces
 18 the total amount of XRP in circulation over time, unlike Bitcoin/Ethereum which expand (at
 19 least, until hitting the upper cap of BTC in Bitcoin). Validators are still using computational
 20 resources and network capacity to operate, which are not free, and so external incentives
 21 must be at play (see Opinion 54.5).

⁶⁶Disclosure, Schwartz Depo. at 101–02.

⁶⁷Disclosure, Document RPLI.02460831.

D Complete List of Materials Considered

D.1 Materials

D.2 Articles

D.3 Class Certification Documents

- Lead Plaintiff's Motion for Class Certification, Supporting Documents and Exhibits
- Defendants' Opposition to Lead Plaintiff's Motion for Class Certification, Supporting Documents and Exhibits
- Lead Plaintiff's Reply in Support of Motion for Class Certification, Supporting Documents and Exhibits

D.4 Public Court Filings

- SEC v. Ripple, Case 1:20-cv-10832-AT-SN (SDNY) dkt 814
- In re Ripple Labs Inc. Litigation, Case 4:18-cv-06753-PJH (N.D. Cal) dkt 85
- In re Ripple Labs Inc. Litigation, Case 4:18-cv-06753-PJH (N.D. Cal) dkt 87
- In re Ripple Labs Inc. Litigation, Case 4:18-cv-06753-PJH (N.D. Cal) dkt 115

D.5 Discovery Responses and Objections

- Defendants' Responses to Lead Plaintiff's Interrogatories, Set Two
- Defendants' Responses to Lead Plaintiff's Requests for Admission, Set Two
- Defendants' Responses to Lead Plaintiff's Requests for Production of Documents, Set Two
- Defendants' Responses to Lead Plaintiff's Interrogatories, Set Three

- 1 • Lead Plaintiff’s Response and Objections to Defendant Ripple Labs Inc.’s First Set of
- 2 Requests for Admission
- 3 • Lead Plaintiff’s Response and Objections to Defendant Ripple Labs Inc.’s Second Set
- 4 of Requests for Admission
- 5 • Lead Plaintiff’s Response and Objections to Defendants’ Second Set of Requests for
- 6 Production of Documents
- 7 • Lead Plaintiff’s Response and Objections to Defendant Ripple Labs Inc.’s Third Set
- 8 of Requests for Admission
- 9 • Lead Plaintiff’s Response and Objections to Defendant XRP II’s First Set of Inter-
- 10 rogatories
- 11 • Lead Plaintiff’s Supplemental Responses and Objections to Defendant Ripple Labs
- 12 Inc.’s First Set of Interrogatories

13 **D.6 Depositions and Exhibits**

- 14 • 1/13/2023 Deposition of Bradley Sostack and accompanying exhibits
- 15 • 1/20/2023 Deposition of Cameron Azari and accompanying exhibits
- 16 • 1/20/2023 Deposition of Steven Feinstein and accompanying exhibits
- 17 • 1/31/2023 Deposition of Bradley Sostack and accompanying exhibits
- 18 • 2/21/2023 Deposition of Dinuka Samarsinghe and accompanying exhibits
- 19 • 2/28/2023 Deposition of Miguel Vias and accompanying exhibits
- 20 • 3/8/2023 Deposition of Mukarram Attari and accompanying exhibits
- 21 • 3/10/2023 Deposition of Monica Long and accompanying exhibits
- 22 • 3/16/2023 Deposition of David Schwartz and accompanying exhibits

- 1 • 3/22/2023 Deposition of Chris Larsen and accompanying exhibits
- 2 • 3/28/2023 Deposition of Bradley Garlinghouse and accompanying exhibits
- 3 • 3/29/2023 Deposition of Jed McCaleb and accompanying exhibits
- 4 • 3/30/2023 Deposition of Carolyn Dicharry as 30b6 representative of Ripple Labs and
- 5 accompanying exhibits
- 6 • 4/20/2023 Deposition of Carolyn Dicharry as 30b6 representative of Ripple Labs

7 D.7 Document Production

8 D.7.1 a. As follows

- | | | | | | |
|----|--------------------|----|-------------|----|-------------|
| 9 | • CIRCLE_00001472 | 21 | • MC0000020 | 33 | • MC0000054 |
| 10 | • EVERSPLIT0000001 | 22 | • MC0000022 | 34 | • MC0000056 |
| 11 | • MC0000001 | 23 | • MC0000023 | 35 | • MC0000065 |
| 12 | • MC0000002 | 24 | • MC0000026 | 36 | • MC0000067 |
| 13 | • MC0000004 | 25 | • MC0000028 | 37 | • MC0000068 |
| 14 | • MC0000007 | 26 | • MC0000030 | 38 | • MC0000069 |
| 15 | • MC0000008 | 27 | • MC0000031 | 39 | • MC0000070 |
| 16 | • MC0000010 | 28 | • MC0000032 | 40 | • MC0000072 |
| 17 | • MC0000012 | 29 | • MC0000033 | 41 | • MC0000075 |
| 18 | • MC0000013 | 30 | • MC0000035 | 42 | • MC0000077 |
| 19 | • MC0000016 | 31 | • MC0000038 | 43 | • MC0000078 |
| 20 | • MC0000018 | 32 | • MC0000053 | 44 | • MC0000080 |

1	● MC0000081	21	● MC0000117	41	● MC0000148
2	● MC0000084	22	● MC0000118	42	● MC0000149
3	● MC0000086	23	● MC0000119	43	● MC0000150
4	● MC0000087	24	● MC0000121	44	● MC0000151
5	● MC0000088	25	● MC0000124	45	● MC0000152
6	● MC0000089	26	● MC0000125	46	● MC0000154
7	● MC0000091	27	● MC0000127	47	● MC0000164
8	● MC0000092	28	● MC0000128	48	● MC0000167
9	● MC0000093	29	● MC0000129	49	● MC0000170
10	● MC0000094	30	● MC0000130	50	● MC0000172
11	● MC0000097	31	● MC0000131	51	● MC0000190
12	● MC0000100	32	● MC0000132	52	● MC0000191
13	● MC0000103	33	● MC0000133	53	● MC0000194
14	● MC0000104	34	● MC0000137	54	● MC0000195
15	● MC0000105	35	● MC0000139	55	● MC0000196
16	● MC0000107	36	● MC0000141	56	● MC0000197
17	● MC0000109	37	● MC0000143	57	● MC0000198
18	● MC0000110	38	● MC0000144	58	● MC0000199
19	● MC0000114	39	● MC0000145	59	● MC0000200
20	● MC0000115	40	● MC0000147	60	● MC0000202

1	● MC0000203	21	● MC0000244	41	● MC0000272
2	● MC0000205	22	● MC0000245	42	● MC0000273
3	● MC0000207	23	● MC0000247	43	● MC0000274
4	● MC0000208	24	● MC0000248	44	● MC0000276
5	● MC0000209	25	● MC0000249	45	● MC0000278
6	● MC0000210	26	● MC0000256	46	● MC0000280
7	● MC0000211	27	● MC0000257	47	● MC0000283
8	● MC0000212	28	● MC0000258	48	● MC0000284
9	● MC0000213	29	● MC0000259	49	● MC0000287
10	● MC0000214	30	● MC0000260	50	● MC0000289
11	● MC0000215	31	● MC0000261	51	● MC0000290
12	● MC0000216	32	● MC0000262	52	● MC0000292
13	● MC0000220	33	● MC0000263	53	● MC0000294
14	● MC0000222	34	● MC0000264	54	● MC0000296
15	● MC0000231	35	● MC0000265	55	● MC0000297
16	● MC0000232	36	● MC0000267	56	● MC0000298
17	● MC0000235	37	● MC0000268	57	● MC0000299
18	● MC0000237	38	● MC0000269	58	● MC0000302
19	● MC0000239	39	● MC0000270	59	● MC0000304
20	● MC0000241	40	● MC0000271	60	● MC0000305

1	● MC0000306	21	● RPLI_01141351	41	● RPLI_01676493
2	● RPLI_00074655	22	● RPLI_01141779	42	● RPLI_01676563
3	● RPLI_00132765	23	● RPLI_01165682	43	● RPLI_01676623
4	● RPLI_00296021	24	● RPLI_01166967	44	● RPLI_01676654
5	● RPLI_00303187	25	● RPLI_01250767	45	● RPLI_01676727
6	● RPLI_00303391	26	● RPLI_01250834	46	● RPLI_01676876
7	● RPLI_00307265	27	● RPLI_01250875	47	● RPLI_01677009
8	● RPLI_00339374	28	● RPLI_01251012	48	● RPLI_02426249
9	● RPLI_00538606	29	● RPLI_01675183	49	● RPLI_02460809
10	● RPLI_00544192	30	● RPLI_01675321	50	● RPLI_02460831
11	● RPLI_00551268	31	● RPLI_01675357	51	● RPLI_02566841
12	● RPLI_00781756	32	● RPLI_01675391	52	● RPLI_02745377
13	● RPLI_00816628	33	● RPLI_01675412	53	● RPLI_02910824
14	● RPLI_00816632	34	● RPLI_01675477	54	● RPLI_02911551
15	● RPLI_00816642	35	● RPLI_01675514	55	● RPLI_03148318
16	● RPLI_00816645	36	● RPLI_01675534	56	● RPLI_03182433
17	● RPLI_00885029	37	● RPLI_01675691	57	● RPLI_03452191
18	● RPLI_00885035	38	● RPLI_01675829	58	● RPLI_03455362
19	● RPLI_00885388	39	● RPLI_01676409	59	● RPLI_03529991
20	● RPLI_00897567	40	● RPLI_01676455	60	● RPLI_03546328

1	• RPLI_03552149	9	• RPLI_03565918	17	• SOSTACK0000001
2	• RPLI_03552227	10	• RPLI_03571088	18	• SOSTACK0000406
3	• RPLI_03556004	11	• RPLI_03628522		
4	• RPLI_03556733	12	• RPLI_03629580	19	• SOSTACK0000575
5	• RPLI_03559837	13	• RPLI_03635513	20	• SOSTACK0000658
6	• RPLI_03560993	14	• RPLI_03636583		
7	• RPLI_03561032	15	• RPLI_03640883	21	• SOSTACK0000735
8	• RPLI_03565053	16	• RPLI_03644313	22	• SOSTACK0000958

23 **D.7.2 b. SEC v. Ripple – Depositions and Exhibits**

24 (Produced as RPLI_03671786 – RPLI_03680600)

- 25 • 2/28/2021 Deposition of Daniel Fischel and accompanying exhibits
- 26 • 5/18/2021 Deposition of Breanne Madigan and accompanying exhibits
- 27 • 5/26/2021 Deposition of David Schwartz and accompanying exhibits
- 28 • 6/9/2021 Deposition of Dinuka Samarsinghe and accompanying exhibits
- 29 • 6/17/2021 Deposition of Monica Long and accompanying exhibits
- 30 • 6/23/2021 Deposition of Asheesh Birla and accompanying exhibits
- 31 • 6/28/2021 Deposition of Miguel Vias and accompanying exhibits
- 32 • 6/29/2021 Deposition of Patrick Griffin and accompanying exhibits
- 33 • 7/20/2021 Deposition of Ryan Zagone and accompanying exhibits
- 34 • 7/22/2021 Deposition of Phillip Rapoport and accompanying exhibits

- 1 • 7/27/2021 Deposition of William Harold Hinman and accompanying exhibits
- 2 • 7/30/2021 Deposition of Ron Wil and accompanying exhibits
- 3 • 8/4/2021 Deposition of Antoinette O’Gorman and accompanying exhibits
- 4 • 8/11/2021 Deposition of Cristian Gil and accompanying exhibits
- 5 • 8/24/2021 Deposition of Ethan Beard and accompanying exhibits
- 6 • 9/14/2021 Deposition of Chris Larsen and accompanying exhibits
- 7 • 9/20/2021 Deposition of Bradley Garlinghouse and accompanying exhibits
- 8 • 11/18/2021 Deposition of James Cangiano and accompanying exhibits
- 9 • 12/3/2021 Deposition of Bradley Borden and accompanying exhibits
- 10 • 12/8/2021 Deposition of Peter Easton and accompanying exhibits
- 11 • 12/17/2021 Deposition of Marko Vukolic and accompanying exhibits
- 12 • 12/20/2021 Deposition of Kristina Shampanier and accompanying exhibits
- 13 • 12/21/2021 Deposition of Carol Osler and accompanying exhibits
- 14 • 12/21/2021 Deposition of M. Laurentius Marais and accompanying exhibits
- 15 • 1/13/2022 Deposition of Anthony Bracco and accompanying exhibits
- 16 • 2/8/2022 Deposition of Peter Adriaens and accompanying exhibits
- 17 • 2/11/2022 Deposition of Alan Schwartz and accompanying exhibits
- 18 • 2/11/2022 Deposition of Yesha Yadav and accompanying exhibits
- 19 • 2/15/2022 Deposition of John Griffin and accompanying exhibits
- 20 • 2/16/2022 Deposition of Patrick Doody and accompanying exhibits

1 • 2/18/2022 Deposition of Albert Metz and accompanying exhibits

2 • 2/23/2022 Deposition of Allen Ferrell and accompanying exhibits

3 • 5/10/2022 Deposition of Albert Metz and accompanying exhibits

4 **D.7.3 c. SEC Investigative Testimonies and Exhibits**

5 (produced in this action)

6 • 12/5/2019 Investigative Testimony of Asheesh Birla and accompanying exhibits

7 • 12/17/2019 Investigative Testimony of Miguel Vias and accompanying exhibits

8 • 1/30/2020 Investigative Testimony of David Schwartz and accompanying exhibits

9 • 2/12/2020 Investigative Testimony of Patrick Griffin and accompanying exhibits

10 • 9/10/2020 Investigative Testimony of Bradley Garlinghouse, Jr. and accompanying
11 exhibits

12 **D.8 Other**

13 Any and all other materials referenced in my report.

¹ **E Curriculum Vitae**

June 7, 2023

A more recent version may be available here:

<https://www.pulpspy.com/cv/cv.pdf>

Jeremy Clark

**NSERC / Raymond Chabot Grant Thornton / Catallaxy
Industrial Research Chair in Blockchain Technologies**

Associate Professor
Concordia Institute for Information Systems Engineering (CIISE)
Concordia University

j.clark@concordia.ca
+1 (514) 848-2424 x5381
<https://pulpspy.com>

Table of Contents

Employment	3
Academic Background	4
Publications	5
Funding	11
Evidence of Impact	13
Highly Qualified Personnel	20
Teaching	23
Service to University	25
Service to Academia	27

Employment

Academic positions

- Associate Professor, Concordia Institute for Information Systems Engineering (CIISE), Concordia University. 1 Jun 2018 – present.
- Assistant Professor, Concordia Institute for Information Systems Engineering (CIISE), Concordia University. 1 Aug 2013 – 31 May 2018.

Professional designations

- Professional Engineer (non-practicing). Professional Engineers of Ontario (PEO). Dec 2018 — present.

Consulting work

- Subject matter expert on undisclosed digital asset subject, *Susman Godfrey LLP*. November 2022—present.
- Subject matter expert on undisclosed cryptocurrency subject, *Williams & Connolly LLP*. January 2018—March 2018.
- Subject matter expert on internet voting security, *City of Toronto*, RFP 3405-13-3197. November 2014 – September 2015.

Advisory boards

- Canadian Blockchain Supply Chain Association (CBSCA), Advisory Board, 2019—present.
- 3iQ Digital Asset Management, Advisory Board, 2017—2021.

Academic Background

Degrees

- Ph.D., Computer Science, University of Waterloo. Graduated: Jun 2011.
- M.A.Sc., Electrical Engineering, University of Ottawa. Graduated: Oct 2007.
- B.E.Sc., Computer Engineering, University of Western Ontario. Graduated: Apr 2004.

Post-Doctorate

- Post Doctoral Fellow, School of Computer Science, Carleton University. 1 Jul 2011 – 1 Aug 2013.

Awards & honours

- Excellence in Teaching Award, Junior Faculty Member. Concordia University, 2017.
- Postdoctoral Fellowships Program (PDF). Natural Sciences and Engineering Research Council of Canada (NSERC). 2011–2013
- Alumni Gold Medal (Top Graduating PhD Student). University of Waterloo. 2011
- Alexander Graham Bell Canada Graduate Scholarship (CGS). Natural Sciences and Engineering Research Council of Canada (NSERC). 2008–2011
- David R. Cheriton Graduate Scholarship. University of Waterloo. 2008–2011
- President's Graduate Scholarship. University of Waterloo. 2008–2011
- Ontario Graduate Scholarship (OGS). Declined. 2008
- Entrance Scholarship. University of Waterloo. 2007
- Grand Prize: Best Election System. "The Punchscan Voting System." University Voting Systems Competition (VoComp). 2007
- Best Project in Department. "Real-Time Encryption using Cellular Automata." University of Western Ontario Design Day Competition. 2004
- Honorable Mention. "Cellular Automata." Ontario Engineering Competition. 2004

Publications

Summary

Unlike other fields, the most active venues for security research are **refereed conferences**, as opposed to refereed journals. Given the competitive nature of the top tier conferences, mid-tier venues are often called **workshops**. Unlike in other fields, these are also rigorously peer reviewed venues for completed technical papers and are typically competitive. In our field, the term workshop denotes a venue that is specific to a narrow domain, as opposed to conferences and symposiums, which tend to accept a broad range of papers.

As one illustrative example, our well-publicized work on the Scantegrity voting system (see media below) appeared initially at a **workshop** (USENIX EVT/WOTE which is co-located with USENIX Security; a top-4). The following year, we published a fuller version of the paper in a **journal** (IEEE Transactions on Information Forensics and Security). The workshop version has been cited 206 times, while the journal version has been cited only 114 times.

Type	Lifetime	While employed
Journals	10	8
Refereed Conferences	48	28
Book Chapters	5	2

Statistics

Citations, h-index and i10 index is based on Google Scholar. Google Scholar is automated and not necessarily fully accurate; however it gives representative results. Our field does not have organizations providing rigorous citation counting or metrics (e.g., impact factor).

Updated Fall 2022	Lifetime
Citations	7439
h-index	27

Refereed conference publications

Abbreviations

**Supervised student* *AR = Acceptance rate* *Rank = Core2021*
LNCS XXXX = Volume XXXX of Springer's Lecture Notes in Computer Science

C48	A. Arun, J. Bonneau, J. Clark. Short-lived zero-knowledge proofs and signatures. <i>28th Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)</i> , 2022. [Rank: A]
C47	D. Demirag*, M. Namazi, E. Ayday, J. Clark. Privacy-Preserving Link Prediction. <i>17th DPM International Workshop on Data Privacy Management</i> , 2022.
C46	D. Chaum, R.T. Carback, J. Clark, C. Liu, M. Nejadgholi*, B. Preneel, A.T. Sherman, M. Yaksetig, F. Zagorski, B. Zhang. VoteXX: A Solution to Improper Influence in Voter-Verifiable Elections. <i>Seventh International Joint Conference on Electronic Voting (E-VOTE-ID)</i> , 2022.
C45	M. Salehi*, J. Clark, M. Mannan. Not so immutable: Upgradeability of Smart Contracts on Ethereum. <i>WTSC, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2022.
C44	M. Moosavi*, J. Clark. Lissy: Experimenting with on-chain order books. <i>WTSC, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2022.
C43	D. Demirag*, J. Clark. Opening sentences in academic writing: How security researchers defeat the blinking cursor. <i>ACM Technical Symposium on Computer Science Education (SIGCSE TS)</i> , 2022. [Rank: A]
C42	S. Eskandari*, M. Salehi*, W. C. Gu, J. Clark. SoK: Oracles from the Ground Truth to Market Manipulation. <i>ACM Advances in Financial Technology</i> , 2021
C41	M. Salehi*, J. Clark, M. Mannan. Red-Black Coins. <i>DeFi, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2021.
C40	D. Demirag*, J. Clark. Absentia: secure function evaluation on Ethereum. <i>WTSC, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2021.
C39	M. Nejadgholi*, N. Yang*, J. Clark. Ballot secrecy for liquid democracy. <i>VOTING, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2021.
C38	J. Clark, P.C. van Oorschot, S. Ruoti, K. Seamons, D. Zappala. Securing Email. <i>Proceedings of Financial Cryptography and Data Security (FC)</i> , 2021. [Rank: A]
C37	M Rahimian*, S Eskandari*, J. Clark. Resolving the Multiple Withdrawal Attack in ERC20 Tokens. <i>2019 IEEE Workshop on Security & Blockchains (IEEE S&B)</i> .
C36	E. Mangipudi, K. Rao, J. Clark, A. Kate. Automated Penalization of Data Leakage using Crypto-augmented Smart Contracts. <i>2019 IEEE Workshop on Security & Blockchains (IEEE S&B)</i> .
C35	S. Eskandari*, M. Moosavi*, J. Clark. Transparent Dishonesty: front-running attacks on Blockchain. <i>Trusted Smart Contracts, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2019. LNCS 11599.
C34	M. Elsheikh, J. Clark, A. Youssef. Deploying PayWord on Ethereum. <i>Trusted Smart Contracts, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2019. LNCS 11599.

C21	M. Backes, J. Clark, P. Druschel, A. Kate, M. Simeonovski. Back-Ref: Accountability in Anonymous Communication Networks. <i>Proceedings of the 12th International Conference on Applied Cryptography and Network Security (ACNS)</i> , 2014. LNCS 8479. AR: 22%.
C22	J. Bonneau, J. Clark, E. W. Felten, J. A. Kroll, A. Miller, A. Narayanan. On Decentralizing Prediction Markets and Order Books. <i>Proceedings of the 13th Annual Workshop on the Economic of Information Security (WEIS)</i> , 2014.
C23	D. Barrera, D. McCarney, J. Clark, P. C. van Oorschot, Baton: Certificate Agility for Android's Decentralized Signing Infrastructure. <i>Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)</i> , 2014.
C24	S. Eskandar*, D. Barrera, E. Stobert, J. Clark. A First Look at the Usability of Bitcoin Key Management. <i>Proceedings of the NDSS Workshop on Usable Security (USEC)</i> , 2015.
C25	J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. Kroll, E. W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. <i>Proceedings of the 34th IEEE Symposium on Security and Privacy (IEEE S&P)</i> , 2015. [Rank: A+] AR: 14%.
C26	G. Dagher*, B. Bünz, J. Bonneau, J. Clark, D. Boneh. Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges. <i>Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS)</i> , 2015. [Rank: A+] AR: 19%
C27	S. Eskandar*, J. Clark, A. Hamou-Lhadj. "Buy your Coffee with Bitcoin: Real-World Deployment of a Bitcoin Point of Sale Terminal." <i>Proceedings of the 13th IEEE International Conference on Advanced and Trusted Computing (Bitcoin Track)</i> , 2016.
C28	N. Yang* and J. Clark. Practical Governmental Voting with Unconditional Integrity and Privacy. <i>VOTING, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2017. LNCS 10323.
C29	S. Eskandar*, J. Clark, M. Adham, V. Sundaresan. On the feasibility of decentralized derivatives markets. <i>Trusted Smart Contracts, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2017. LNCS 10323.
C30	M. Moosavi*, J. Clark. Ghazal: toward truly authoritative web certificates using Ethereum. <i>Trusted Smart Contracts, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2018. LNCS 10958.
C31	C. Okoye*, J. Clark. Toward Cryptocurrency Lending. <i>Trusted Smart Contracts, Proceedings of Financial Cryptography and Data Security: FC Workshops</i> , 2018. LNCS 10958.
C32	S. Eskandar*, A. Leoutsarakos, T. Mursch, J. Clark. A first look a browser-based cryptojacking. <i>2018 IEEE Workshop on Security & Blockchains (IEEE S&B)</i> .
C33	V. Zhao, J. Choi, D. Demirag*, M. Mannan, K. Butler, E. Ayday, J. Clark. One-time programs made practical. <i>Proceedings of Financial Cryptography and Data Security (FC)</i> , 2019. LNCS 11598. [Rank: A]

C20	J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, E. W. Felten. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. <i>Proceedings of the 18th Conference on Financial Cryptography and Data Security (FC)</i> , 2014. LNCS 8437. [Rank: A] AR: 22%
C19	F. Zagorski, R. Carback, D. Chaum, J. Clark, A. Essex, P. Vora. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System. <i>Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS)</i> , 2013. AR: 23%.
C18	J. Clark and P. C. van Oorschot. SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. <i>Proceedings of the 34th IEEE Symposium on Security and Privacy (IEEE SSP)</i> , 2013. [Rank: A+] AR: 12%.
C17	D. McCarney, D. Barrera, J. Clark, S. Chiasson, and P. C. van Oorschot. Tapas: Design, implementation, and usability evaluation of a password manager. <i>Proceedings of the 2012 Annual Computer Security Applications Conference (ACSAC)</i> , 2012. AR: 19%.
C16	D. Barrera, J. Clark, D. McCarney, P. C. van Oorschot. Understanding and improving app installation security mechanisms through empirical analysis of Android. <i>Proceedings of the 2nd Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)</i> , 2012. AR: 37%.
C15	A. Essex, J. Clark, and U. Hengartner. Cobra: Toward concurrent ballot authorization for internet voting. <i>Proceedings of the 2012 USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)</i> , 2012. AR: 35%.
C14	J. Clark and A. Essex. CommitCoin: Carbon dating commitments with Bit-coin. <i>Proceedings of the 16th Conference on Financial Cryptography and Data Security (FC)</i> , 2012. LNCS 7397. [Rank: A]
C13	J. Clark and U. Hengartner. Selections: an internet voting system with over-the-shoulder coercion-resistance. <i>Proceedings of the 15th Conference on Financial Cryptography and Data Security (FC)</i> , 2011. LNCS 7035. [Rank: A]
C12	R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, P. L. Vora. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. <i>Proceedings of the 19th USENIX Security Symposium</i> , 2010. [Rank: A+] AR: 15%.
C11	A. Essex, J. Clark, U. Hengartner, C. Adams. Eperio: Mitigating Technical Complexity in Cryptographic Election Verification. <i>Proceedings of the 2010 USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)</i> , 2010.
C10	J. Clark, U. Hengartner. On the Use of Financial Data as a Random Beacon. <i>Proceedings of the 2010 USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)</i> , 2010.
C09	A. T. Sherman, R. Carback, D. Chaum, J. Clark, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, B. Sinha, P. L. Vora. Scantegrity Mock Election at Takoma Park. <i>Proceedings of the 4th International Conference on Electronic Voting (EVOTE)</i> , 2010.

C08	J. Clark, U. Hengartner, K. Larson. Not-So Hidden Information: Optimal Contracts for Undue Influence in E2E Voting Systems. <i>Proceedings of the Second IAVoSS International Conference on E-voting and Identity (Vote-ID)</i> , 2009, LNCS 5767.
C07	A. Essex, J. Clark, U. Hengartner, C. Adams. How to Print a Secret. <i>Proceedings of the 4th USENIX Workshop on Hot Topics in Security (HotSec)</i> , 2009. AR: 28%.
C06	D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen A. T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. <i>Proceedings of the 2008 USENIX Electronic Voting Technology Workshop (EVT)</i> , 2008.
C05	J. Clark, U. Hengartner. Panic passwords: Authenticating under duress. <i>Proceedings of the 3rd USENIX Workshop on Hot Topics in Security (HotSec)</i> , 2008. AR: 32%.
C04	A. Essex, J. Clark, C. Adams. Aperio: High integrity elections for developing countries. <i>Proceedings of the IAVoSS Workshop on Trustworthy Elections (WOTE)</i> , 2008.
C03	J. Clark, P.C. van Oorschot, C. Adams. Usability of anonymous web browsing: An examination of Tor interfaces and deployability. <i>Proceedings of the Third Symposium On Usable Privacy and Security (SOUPS)</i> . ACM International Conference Proceedings Series, vol 229, 2007, pp. 41–51. AR: 31%.
C02	J. Clark, A. Essex, C. Adams. On the security of ballot receipts in E2E voting systems. <i>Proceedings of the IAVoSS Workshop on Trustworthy Elections (WOTE)</i> , 2007.
C01	A. Essex, J. Clark, R. T. Carback III, S. Popoveniuc. Punchscan in practice: An E2E election case study. <i>Proceedings of the IAVoSS Workshop on Trustworthy Elections (WOTE)</i> , 2007.

Articles in journals & periodicals

*Supervised student

JIF = 2021 Journal Impact Factor, Journal Citation Reports, Web of Science / Clarivate

J10	Raphael Auer, Rainer Böhme, Jeremy Clark, Didem Demirag*. Mapping the Privacy Landscape for Central Bank Digital Currencies. <i>ACM Queue</i> , June/July 2022.
J09	E. Pimentel, E. Boulianne, S. Eskandari,* J. Clark. Systemizing the Challenges of Auditing Blockchain-Based Assets. <i>Journal of Information Systems</i> , Summer 2021.
J08	J. Clark, D. Demirag*, S. Moosavi*. Demystifying Stablecoins. <i>Communications of the ACM</i> . 63(7):40-46. Jul 2020. [JIF: 14.065]
J07	S. Ruoti, B. Kaiser, A. Yerukhimovich, J. Clark, R. Cunningham. Blockchain Technology: What is it good for? <i>Communications of the ACM</i> . 63(1):46-53. Jan 2020. [JIF: 14.065]
J06	G. Dagher*, B. Fung, N. Mohammad, J. Clark. SecDM: Privacy-preserving Data Outsourcing Framework with Differential Privacy. <i>Knowledge and Information Systems</i> . 62:1923–1960, 2020.
J05	A. Narayanan, J. Clark. Bitcoin's Academic Pedigree. <i>Communications of the ACM</i> . 60(12):36-45. 2017. [JIF: 14.065]

J04	E. Moher, J. Clark, A. Essex. Diffusion of voter responsibility: potential failings in E2E receipt checking. <i>USENIX Journal of Election Technology and Systems</i> . 3(1):1-17. 2014.
J03	J. Clark. Enhancing Anonymity: Cryptographic and statistical approaches for shredding our digital dossiers. <i>ACM Computing Reviews</i> . 2014. Invited.
J02	D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, P. L. Vora. Scantegrity II: End-to-End Verifiability by Voters of Optical Scan Elections Through Confirmation Codes. <i>IEEE Transactions on Information Forensics and Security</i> , 4(4):611-627, 2009. [JIF: 7.231]
J01	D. Chaum, A. Essex, R. T. Carback III, J. Clark, S. Popoveniuc and A. T. Sherman, P. Vora. Scantegrity: end-to-end voter verifiable optical-scan voting. <i>IEEE Security & Privacy</i> , vol. 6, no. 3, pp. 40–46, May/June 2008. [JIF: 3.105]

Book chapters

B05	J. Clark. The Long Road to Bitcoin. Foreword to: “Bitcoin and Cryptocurrency Technologies.” <i>Princeton University Press</i> , 2016.
B04	R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, P. L. Vora. The Scantegrity Voting System and its Use in the Takoma Park Elections. Chapter 10 in: “Real-World Electronic Voting: Design, Analysis and Deployment.” <i>CRC Press</i> , 2016.
B03	S. Popoveniuc, J. Clark, R. Carback, A. Essex, D. Chaum. Securing Optical-Scan Voting. Chapter in: “Toward Trustworthy Elections: New Directions in Electronic Voting.” State of the Art Survey Series, <i>Springer</i> , 357–369. 2010.
B02	A. Essex, J. Clark, C. Adams. Aperio: High Integrity Elections for Developing Countries. Chapter in: “Toward Trustworthy Elections: New Directions in Electronic Voting.” State of the Art Survey Series, <i>Springer</i> , 388–401. 2010.
B01	J. Clark, P. Gauvin, C. Adams. Exit Node Repudiation for Anonymity Networks. Chapter 22 in: “Lessons from the Identity Trail: Anonymity, Privacy and Identity in a Networked Society.” <i>Oxford University Press</i> . 399-415, 2009.

Editorial activities

E03	Bracciali, A., Clark, J., Pintore, F., Roenne, P., Sala, M. (Editors). “Financial Cryptography and Data Security: FC Workshops 2019.” Lecture Notes in Computer Science (LNCS) 11599. <i>Springer</i> , 2020.
E02	A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, M. Sala (Editors). “Financial Cryptography and Data Security: FC Workshops 2018.” Lecture Notes in Computer Science (LNCS) 10958. <i>Springer</i> , 2019.
E01	J. Clark, S. Meiklejohn, P.Y.A.Ryan, D. Wallach, M. Brenner, K. Rohloff (Editors). “Financial Cryptography and Data Security: FC Workshops 2016.” Lecture Notes in Computer Science (LNCS) 9604. <i>Springer</i> , 2016.

Funding

External Funding

Year	Title, Program, Agency	Amount	PI	Co-Applicants
2021	"Privacy Design Landscape for Central Bank Digital Currencies," Contributions Program, Office of the Privacy Commissioner of Canada (OPC)	\$26,450 once Share: 100%	Y	
2021	"Understanding Blockchains through Experimentation," Extension to previous project, Autorité des marchés financiers (AMF)	\$100,000 once Share: 50%	Y	Emilio Boulianne (JMSB)
2021	"Enhancing transparency, inclusion, and privacy for financial and democratic technologies," Discovery Grant (DG), Natural Sciences and Engineering Research Council of Canada (NSERC)	\$35,000/year for 5 years Share: 100%	Y	
2020	"The Human-Centric Cybersecurity Partnership (HC2P)," Partnership Grant, Social Sciences and Humanities Research Council (SSHRC)	\$2,434,323 over 5 years Share: TBD	N	Benoit Dupont + 32 others
2020	"Toward Scalable Systems for Securities on Blockchains," Fintech Chaire, Autorité des marchés financiers (AMF) and Finance Montreal	\$50,000 once Share: 50%	N	Kaiwen Zhang (ETS)
2019	"NSERC / Raymond Chabot Grant Thornton / Catalaxy Industrial Research Chair on Blockchain Technologies," Natural Sciences and Engineering Research Council of Canada (NSERC)	\$1,380,000 over 5 years Share: 100%	Y	
2017	"Understanding Blockchains through Experimentation," Education and Good Governance Fund (EGGF), Autorité des marchés financiers (AMF)	\$100,000/year for 2 years Share: 50%	Y	Emilio Boulianne (JMSB)
2016	"One Person, One Vote? Blockchain Technologies and Experiments in Voting and Party Governance," Seed Grant, Centre for the Study of Democratic Citizenship (CSDC)	\$6831 once Share: 50%	N	Fenwick Mckelvey (Comm)
2015	"Certificate Authority Report Card: Examining the Root of Data Protection on the Web," Contributions Program, Office of the Privacy Commissioner of Canada (OPC)	\$50,000/year for 1 year Share: 50%	Y	Mohammad Mannan (CIISE)
2015	"Vote par Internet : des technologies favorisant la démocratie," Programme Établissement de nouveaux chercheurs universitaires, Fonds de recherche du Québec - Nature et technologies (FRQNT)	\$19,000/year for 2 years Share: 100%	Y	

Year	Title, Program, Agency	Amount	PI	Co-Applicants
2014	"Secure online services for private user data," Discovery Grant (DG), Natural Sciences and Engineering Research Council of Canada (NSERC)	\$24,000/year for 5 years Share: 100%	Y	

Internal Funding

Year	Program	Amount	PI	Co-Applicants
2023	Aid to Research Related Events, Exhibition, Publication and Dissemination Activities (ARRE) Program	\$5K once	Y	
2020	Aid to Research Related Events, Exhibition, Publication and Dissemination Activities (ARRE) Program	\$5K once	Y	
2015	Aid to Research Related Events, Exhibition, Publication and Dissemination Activities (ARRE) Program	\$5K once	Y	
2015	Individual Seed Program	\$7K once	Y	
2013	Start-Up Grant	\$50K once	Y	

Research Centres/Networks

- Human-Centric Cybersecurity Partnership (HC2P). Co-Investigator, 2020—present.
- Centre for the Study of Democratic Citizenship (CSDC). Member, 2016—present. Advisory Board, 2022—present.
- Smart Cybersecurity Network (SERENE-RISC). Knowledge Mobilization Network, Networks of Centres of Excellence of Canada (NCE). Co-Investigator, 2016—2021.

Evidence of Impact

Invited Talks and Seminars

- Digital Economy Taxation Network / Revenu Québec, DET 2023, “Going Digital: Tax Systems and Emerging Technology,” June 18, 2023.
- C-Dem/CSDC Forum, “Roundtable: Electoral Integrity,” Panel, June 4, 2023.
- CIADI/GCS Aerospace Meets Cybersecurity Forum, “Cybersecurity challenges in aerospace,” Moderator, April 17, 2023.
- Financial Management Institute of Canada, PD Week. “Blockchain and DeFi: Landscape,” Nov 24, 2022.
- FIC, International Cybersecurity Forum, Nov 1-2, 2022.
- MTL Connect, “MTL Inspire.” Panel, October 19, 2022.
- ACT International Midterm Conference, “Policing Blockchain.” Panel, October 6, 2022.
- Fintech Cadence | Fintech Drinks, “Fintech & DeFi: How is fintech DeFi-ing the traditional banking system?” Panel, July 12, 2022.
- Blockchain Technology Symposium. “Blockchain Culture, Leisure and Luxury.” Panel, June 10, 2022.
- Quartier de l’innovation de Montréal. “Entre Terre et techno, ça clique ?” Panel, May 26, 2022.
- Fintech Cadence Certificate Program. “Understanding blockchain and its uses in the financial sector.” February 22, 2022.
- Autorité des marchés financiers. “Finance décentralisée et crypto : état de la situation, nouveaux risques et points de vigilance.” Panel, October 26, 2021.
- Smith School of Business, Queen’s University. “New Frontiers in Auditing: Risk and Opportunities in the Blockchain Sector.” Panel, October 7, 2021.
- Vancouver International Privacy & Security Summit (VIPSS). “Banking on the Future: How the Digital Surge Will Reshape How We Do Business.” Panel, May 6, 2021.
- CyberEco Cyber Conference. “Technology & blockchain.” May 5, 2021.
- Quartier de l’innovation de Montréal. “Blockchain - multiples usages.” Panel, April 28, 2021.
- Holt Accelerator, “[I AM PROTECTED].” Panel, April 21, 2021.
- UMBC Cyber Defense Lab Seminar. “Transparent Dishonesty: front-running attacks on Blockchain.” March 26, 2021.
- 1st Annual Lecture on Computer Science and Society. “The Blockchain and Cryptocurrency Landscape.” Carleton University. March 10, 2021
- Workshop on The State of Canadian Cybersecurity Conference: Human-Centric Cybersecurity. “Decentralized Finance: Landscape and Future Directions.” SERENE-RISC, February 18, 2021.
- Fintech Cadence Certificate Program. “Understanding blockchain and its uses in the financial sector.” January 30, 2021.

- Montreal Lakeshore University Women's Club. "Bitcoins: What, why and how..." February 10, 2020.

Note: Parental & sabbatical leave Fall 2019—Summer 2021.

- Elections Quebec. "Internet Voting." Nov 2, 2019.
- Blockchain at McGill. "Introduction to Blockchain for Non-Profits," Social Innovation: Int'l Development and Blockchain. 29 Mar 2019.
- Canada Mortgage and Housing Corporation (CMHC). "Blockchain Technologies: Landscape and Future Directions." 26 Feb 2019.
- CFA Montreal FinTech Rendez-vous. "Blockchain Technologies: Landscape and Future Directions." 7 Feb 2019.
- Loto-Quebec. "Lunch and learn." 22 Jan 2019.
- RISQ Colloquium. "Blockchain Technologies: Landscape and Future Directions." 29 Nov 2018.
- TriPAC Pension Advisory Committees. "Blockchain Technologies: Landscape and Future Directions." Treasury Board Secretariat. 21 Nov 2018.
- Defending Democracy: Confronting Cyber-Threats At Home And Abroad. "Liquid Democracy and Blockchains." October 26, 2018.
- Blockchain and National Security. "Blockchain Technology: National Security Use-Cases." Public Safety Canada, October 18, 2018.
- Montreal Police Pension Fund (ABRPPVM). "Blockchain Technology: Landscape & Future Directions." Invited speaker, September 22, 2018.
- BMO 13th Annual Real Estate Conference. "Blockchain Applications & Real-Estate." Panel, BMO Capital Markets. September 20, 2018.
- Blockchain Technology Symposium (BTS). "Blockchain Nuances: Lessons from Fintech use-cases." Invited talk, Fields Institute. September 18, 2018.
- GoSec. "Blockchain Technologies: Landscape and Future Directions." August 29, 2018.
- StartupFest. "Democracy Enhancing Technologies." CryptoFest. July 10, 2018.
- Finteq. "Blockchain Nuances" Keynote, Desjardins Labs & UQAR, June 20, 2018.
- The Walrus LIVE. "The Future of Money" Panel Discussion with David Tax (TD) and Susan Prince (CBC). June 14, 2018.
- BMO ThinkSeries. "Blockchain Technologies: Landscape and Future Directions." June 12, 2018.
- Autorite des marches financiers (AMF). "Crypto Primer II." June 11, 2018.
- Canada Pension Plan Investment Board (CPPIB). "Blockchain Technologies." June 1, 2018.
- Security Revolution. "Blockchain Primer." SERENE-RISC, May 31, 2018.
- "Blockchain Technologies: Landscape and Future Directions." True North Science Bootcamp. May 25, 2018.

- Anticipating Future Trends and Managing Risks Program. "Blockchain Technologies: Landscape and Future Directions," HEC Paris and Concordia. May 10, 2018.
- Autorite des marches financiers (AMF). "Crypto Primer I." May 1, 2018.
- GC Blockchain Day. "Ledgers Past, Present and Future." Treasury Board Secretariat of Canada. April 23, 2018.
- "Workplace 2020." Management Consulting Club, Concordia. Panel. April 8, 2018.
- "Blockchain Technologies: Landscape and Future Directions." Canadian National Railway (CN). February 8, 2018.
- Kenneth Woods Portfolio Management Program. "Cryptocurrencies: An Investable Asset?" John Molson School of Business. January 23, 2018.
- "Provisions: Privacy-Preserving Proofs of Solvency." Newcastle University. December 7, 2017.
- "Democracy Enhancing Technologies: From Theory to Practice." CSDC Speaker Series. McGill, September 15, 2017.
- Hydro-Québec Symposium 3i. "Bitcoin & Blockchains: Landscape and Future Directions." Invited Speaker, Montreal,
- Privacy, Security and Trust (PST). "Bitcoin & Blockchains: Landscape and Future Directions." Keynote, Calgary, Aug 28, 2017.
- Metropolis 2017. "The Bitcoin & Blockchain Technology Landscape." June 28, 2017.
- Blockchain Meetup. "Zero Knowledge." District 3. May 4, 2017.
- Canada Music Week. "Blockchains: Smart Contracts and Media-Driven Crypto Currencies" Panel discussion, April 19, 2017.
- District 3. "The Future of Blockchain." Panel discussion, December 8, 2016.
- Symposium on Foundations & Practice of Security. "The Bitcoin & Blockchain Technology Landscape." Keynote presentation. Université Laval, October 26, 2016.
- Online Voting Roundtable: Electoral Futures in Canada. "Blockchain and Voting: Assessment & Critique." Invited Speaker, University of Ottawa. September 26, 2016.
- P2P Financial Systems Workshop. "Blockchain nuances." Keynote presentation. UCL, September 8, 2016.
- Bank of Canada. "Bitcoin & Blockchains: Part 2." July 14, 2016.
- Anti-phishing working group (APWG) eCrime 2016. "Bitcoin: an impartial assessment of its use and potential for cybercrime." May 31, 2016.
- C.D. Howe. "Blockchain Technologies and the Future of Finance." May 30, 2016.
- ASIMM Colloque RSI. "Bitcoin & Blockchains: Tutorial," May 12, 2016.
- Bank of Canada. "Bitcoin & Blockchains: Landscape and Future Directions," May 11, 2016.
- National Research Council (NRC), "Security Training Course," Mar 22, 2016.
- MIT Bitcoin Expo. "Blockchain-based voting: potential and limitations," MIT, Mar 6, 2016.
- Bitcoin and Cryptocurrency Research Conference. "Altcoins," Center for Information Technology Policy (CITP), Princeton University, March 27, 2014.

- USENIX Summit on Hot Topics in Security (HotSec 2013). “Eroding Trust and the CA Debacle,” August 13, 2013.
- CIISE Distinguished Seminar. “How to Carbon Date Digital Information,” Concordia University, March 8, 2012.
- MITACS Digital Security Seminar Series. “Panic Passwords and their Applications,” Carleton University, January 27, 2011.
- CACR Cryptography Seminar. “The First Governmental Election with a Voter Verifiable Tally: Experiences using Scantegrity II at Takoma Park,” University of Waterloo, February 5, 2010.
- CACR Cryptography Seminar. “Selections: An Internet Voting System with Over-the-shoulder Coercion Resistance,” University of Waterloo, December 3, 2010
- Information Technology and Innovation Foundation (ITIF) Forum: Future of Voting. “Panel Discussion,” Longworth House Office Building, Washington, D.C. March 6, 2008.
- CACR Cryptography Seminar. “Combating Adverse Selection in Anonymity Networks,” University of Waterloo, October 17, 2007.

Expert Testimony & Public Interest Consultations

- Elections Quebec. “Internet Voting,” Citizen Jury. Nov 2, 2019.
- House of Commons, Standing Committee on Finance. Testimony: Statutory Review of the Proceeds of Crime and Terrorist Financing Act. March 27, 2018.
- Investissement Quebec. Bitcoin & Blockchains: Landscape and Future Directions. January 15, 2018.
- Government of Canada (GC) Digital Target State Architecture and Direction. Blockchain working group. August 2017 — April 2018.
- Karina Gould, Minister of Democratic Institutions (House of Commons, Canada). CDSC roundtable. August 30, 2017.
- Autorité des marchés financiers (AMF). “Blockchain nuances.” March 29, 2017.
- Royal Canadian Mounted Police (RCMP). Bitcoin brainstorming session (#2). Participant in roundtable. September 28, 2016.
- Royal Canadian Mounted Police (RCMP). Bitcoin brainstorming session. Participant in roundtable. July 5, 2016.
- Formation régionale de la Cour du Québec. “Bitcoin: Introduction & Implications,” May 9, 2015.
- 2013–2014 City of Toronto. Subject Matter Expert on Internet Voting Security and Cryptography (RFP No. 3405-13-3197).
- Senate of Canada, Standing Committee on Banking, Trade and Commerce. Testimony: Study on the use of digital currency. April 3, 2014.
- City of Edmonton: Citizen Jury on Internet Voting. “Security Risks Related to Internet Voting,” Centre for Public Involvement/University of Alberta, November 23–25, 2012.

Press & Media (Selected)

- “What is Worldcoin and what does it mean for our privacy?” *Context.news (Thomson Reuters Foundation)*, June 7, 2023.
- “Clarity, please.” *CBA/ABC National*, Nov 14, 2022
- “Deception, exploited workers, and cash handouts: How Worldcoin recruited its first half a million test users.” *MIT Technology Review*, April 6, 2022.
- “It’s a first, Bitcoin is now legal tender in one country.” *CBC Radio*, Sep 23, 2021.
- “New kid on the blockchain: the young people using crypto for good.” *DAZED*, Jul 22, 2021.
- “Digital currencies bring new options for financial privacy.” *Hill Times*, May 5, 2021.
- “Satoshi & Company: The 10 Most Important Scientific White Papers In Development Of Cryptocurrencies.” *Forbes*, Feb 13, 2021.
- “Contact tracing segment.” *The Aaron Rand Show, CJAD 800*, May 26, 2020.
- “Are we ready for an app that trades privacy for more freedom?” *Montreal Gazette*, May 25, 2020.
- “Chaînes de blocs: dompter la décentralisation de l’informatique.” *Le Devoir*, Mar 2, 2020.
- “Academic: All Undergrads Should Learn About Bitcoin & Blockchain.” *Cryptonews*, Dec 22, 2019.
- “Why Quebec is betting big on Bitcoin.” *Pivot Magazine (CPA Canada)*, Jan 8, 2019.
- “Banks Claim They’re Building Blockchains. They’re Not.” *Investopedia*, July 13, 2018.
- “The evolution of cryptojacking.” *CryptoInsider*, March 20, 2018.
- “The Ethics Of Cryptojacking: Rampant Malware Or Ad-Free Internet?” *CoinTelegraph*, March 16, 2018.
- “One of the Biggest Coinhive Users Made \$7.69 In 3 Months.” *Motherboard*, March 14, 2018.
- “Attack Or Business Opportunity?: Academics Question Ethics Of Coinhive Cryptojacking.” *CoinTelegraph*, March 10, 2018.
- “How much should I regret not buying Bitcoin?” *Gizmodo*, January 29, 2018.
- Interview on Bitcoin regulation. *CBC Radio One*, December 5, 2017.
- “How blockchain-based payment is changing the cannabis industry,” *IBM thinkLeaders*, June 21, 2017.
- “Ottawa explores potential of ‘blockchain,’ billed as next-generation Internet tech.” *Toronto Star*, Feb 28, 2017.
- “Block the vote: Could Blockchain Technology Cybersecure Elections?” *Forbes*, Aug 30, 2016.
- “He’s Bitcoin’s Creator, He Says, but Skeptics Pounce on His Claim,” *New York Times*, May 2, 2016.
- “Logged out, but still out there,” *Globe and Mail*, Feb 19, 2016.
- “Princeton University releases first draft of bitcoin textbook,” *CoinDesk*, Feb 10, 2016.
- “The top 10 cryptocurrency research papers of 2015,” *CoinDesk*, Dec 27, 2015.

- “Canada’s Internet Voting Problem,” *SC Magazine*, Feb 2015 issue.
- “Latest Internet voting reports show failures across the board,” *Al Jazeera America*, Feb 8, 2015
- “How Block Chain Technology Could Usher in Digital Democracy,” *CoinDesk*, June 16, 2014.
- “Can Bitcoin Help Predict the Future?,” *CoinDesk*, May 24, 2014.
- “Heartbleed and sentinels of the net,” *Montreal Gazette*, Apr 21, 2014.
- “PROFESSOR: There Is A Big, Gaping Flaw In The New Satoshi Study,” *Business Insider*, Mar 28, 2014.
- “2014 Federal Budget Calls Bitcoin A Terrorist, Crime ‘Risk’,” *Huffington Post*, Feb 12, 2014.
- “Bitcoin: How its core technology will change the world,” *New Scientist*, Feb 5, 2014.
- “More than money, bitcoin’s real value lies in its algorithms,” *InfoWorld*, Jan 12, 2014.
- “U. researchers develop Bitcoin prediction market,” *Daily Princetonian*, Jan 5, 2014.
- “This Princeton professor is building a Bitcoin-inspired prediction market,” *The Verge*, Nov 29, 2013
- “Montreal’s Bitcoin Embassy bridges gap between digital currency and real world,” *Montreal Gazette*, Nov 29, 2013.
- “Bitcoin online currency gets new job in web security,” *New Scientist*, Jan 11, 2012.
- “Secure, verifiable voting: Cryptography, invisible ink, and other voting magic,” *Imprint*, Nov 6, 2009.
- “Scantegrity: Voters Test New Transparent Voting System,” *Huffington Post*, Nov 5, 2009.
- “Maryland Voters Test New Cryptographic Voting System,” *Wired News*, Nov 4, 2009.
- “Voters try out new security system,” *UW Daily Bulletin*, Nov 3, 2009.
- “E-voting system lets voters verify their ballots are counted,” *Computerworld*, Nov 3, 2009.
- “First Test for Election Cryptography,” *Technology Review*, Nov 2, 2009.
- “Mock election tests new voting system,” *Gazette.net*, April 15, 2009.
- “Geek the Vote 2012: What Election Tech Will Look like 4 Years From Now,” *Popular Mechanics*, Nov 4, 2008.
- “Canadian voting machine technology enters American political scene,” *CBC.ca*, Oct 28, 2008.
- “New Voter Counter System Uses Encrypted Codes, Invisible Ink,” *Voice of America*, Oct 24, 2008.
- “A Really Secret Ballot,” *The Economist*, Oct 22, 2008.
- “Class voting hacks prompt call for better audits,” *MSNBC*, Oct 20, 2008.
- “Clean Elections,” *Communications of the ACM*, October 2008.
- “Protecting Your Vote With Invisible Ink,” *Discover Magazine*, Oct 2008.
- “Flawless Vote Counts,” *Technology Review*, Sept/Oct 2008.
- “Shift Back to Paper Ballots Sparks Disagreement,” *Morning Edition*, Mar 7, 2008.
- “Down for the Count,” *ACM netWorker*, Mar 2008.

- “The future of voting IT,” *Government Computer News*, Mar 10, 2008.
- “A Damaging Paper Chase In Voting,” *Washington Post*, Sept 8, 2007.
- “Punchscan Wins VoComp 2007,” *As It Happens (CBC)*, August 23, 2007.
- “US/Canada Team Wins Voting Competition,” *Threat Level (Wired)*, July 19, 2007.
- “Electronic Democracy,” *Digital Planet (BBC)*, Jan 29, 2007.
- “Making Every E-vote Count,” *IEEE Spectrum*, Jan 2007.

Concordia Promotional Activities

- Thinking Out Loud. “Bitcoin & Cryptocurrency,” Podcast, Episode 14. 27 Feb 2018.
- “Back to the future — reclaiming the internet” Distinguished Alumni Speaker Series with Fay Arjomandi. September 22, 2018.
- “This is Concordia. Now. “Bitcoin and cryptocurrency.” Conversation with Alan Shepherd. April 11, 2018.
- “X EXPLAINED: What you need to know about internet cookies.” Concordia Video. March 29, 2018.
- This Is Concordia. Now. “Jeremy Clark talks Bitcoin and cryptocurrency.” Conversation with Sudha Krishnan (CBC Montreal). February 22, 2018.
- Next-Gen. Now. “The Campaign for Concordia.” Promotional video with on-screen interview. November 24, 2017.
- Capstone Magazine. “Cyberattacks: everything you need to know.” Fall 2016.
- Concordia Alumni Association. “Everyone knows your birthday: How secure is your password Hint: not very!” New York City, May 16, 2017.
- Thinking Out Loud. “One Vote,” The Futurecast podcast, Episode 4. April 12, 2017.
- Next-gen. Now. “My Name is Jeremy Clark.” Website feature. March 1, 2017.
- Concordia University Magazine. “Guardians of the IT galaxy.” February 9, 2017.
- Thinking Out Loud. “Connecting your tech future,” conversation with Nora Young (CBC), Concordia University. March 1, 2016.
- Breakfast Talk. “Heartbleed & other CIISE Research,” Concordia University. May 6, 2014.

Highly Qualified Personnel

HQP Job Placement


Sector	Organization
Blockchain Industry	ConsenSys Diligence, Offchain Labs, Trail of Bits, Quantstamp, BitAccess, Ether Capital
Faculty	Carleton University, Boise State University
PDFs	UQAM
Industry	KPMG, Deloitte, Morgan Stanley
Government	National Defence

Post-Doctoral










Name	State	Dates	Research Topic	Papers	Co-Supervisor
Elizabeth Stobert	PDF 🎓	2018/W-2018/F	Usable security	C24	

PhD

Name	State	Dates	Research Topic	Papers	Co-Supervisor
Reza Rahimian	PhD	2018/F-Part Time	Financial technology	C37	
Mahsa Moosavi	PhD	2018/S-	Layer-2 blockchain technology	C30, C35, J08, C44	
Didem Demirag	PhD 🎓	2018/W-2022/F	“Moving Multiparty Computation Forward for the Real World”	C33, J08, C40, C43, C47, J10	
Shayan Eskandari	PhD	2017/F-	Blockchain technology	C24, C27, C29, C32, C35, C37, C42, J09	
Pratyusha Bhattacharya	PhD	2017/S-	Smart Grid Security		M. Debbabi (CIISE)
Nan Yang	PhD 🎓	2014/S-2020/F	“Non-Local Contamination in Cryptography”	C28, C39	C. Crépeau (McGill)

Name	State	Dates	Research Topic	Papers	Co-Supervisor
Gaby Dagher	PhD 	2013/F - 2015/F	"Toward secure and privacy-preserving data sharing and integration"	C26, J06	B. Fung (McGill)

MASc

Name	State	Dates	Research Topic	Papers	Co-Supervisor
Youwei Deng	MASc	2023/W-	Zero Knowledge Proofs		
Sina Pilehchiha	MASc 	2021/S-2022/F	"Improving Reproducibility in Smart Contract Research"		A.G. Aghdam (ECE)
Mahdi Nejadgholi	MASc 	2019/F-2022/S	"Nullification, a coercion-resistance add-on for e-voting protocols"	C39, C46	
Mehdi Salehi	MASc 	2020/W-2022/W	"An Analysis of Upgradeability, Oracles, and Stablecoins in the Ethereum Blockchain"	C41, C42, C45	M. Mannan (CIISE)
Corentin Thomasset	MASc 	2019/F-2020/S	"SERENIoT : Politiques de sécurité collaboratives pour maisons connectées"		D. Barrera (Carleton), J. Fernandez (Polytechnique)
Chidinma Okoye	MASc 	2016/S - 2017/F	"New applications of blockchain technology to voting and lending"	C31	
Mahsa Moosavi	MASc 	2015/F - 2018/W	"Rethinking Certificate Authorities: Understanding and decentralizing domain validation"	C30, C35, J08, C44	
Michael Colburn	MASc 	2014/F - 2018/S	"Short-Lived Signatures"		
Abhimanyu Khanna	MASc 	2014/F - 2017/S	"Towards Usable and Fine-grained Security for HTTPS with Middleboxes"		M. Mannan (CIISE)
Shayan Eskandari	MASc 	2013/F - 2016/W	"Real world deployability and usability of Bitcoin"	C24, C27, C29, C32, C35, C37, C42, J09	W. Hamou-Lhadj (ECE)

Supervised Graduate Projects (ENGR 6991)

Year	Students
2023	Mohammad Zawad Tahmeed
2019	Abhinav Kumar
2018	Jinumol James, Laleh Alimadadi, Rupesh Gawde, Brindha Shree, Isreal Tei, Saad Ahmen (MIAE: ENGR 6971)
2017	Temitiope Adetula, Shahab Odagar
2016	Ejiro Mary, Ogor Umukoro, Omoye Obazele
2015	S. Sandisha
2014	Paemka-Ojugbana Judah Chukwuma, Manish Megnath

Teaching

Courses Taught

Year/Term	Course	Class Size	Evaluation*
2022/4	INSE 6615: Blockchain Technology		
2022/4	INSE 6150: Security Evaluation Methodologies		
2022/2	INSE 6150: Security Evaluation Methodologies	70	1.72
2021/4	INSE 6630: Recent Developments in Info. Systems Security	67	N/A
2021/4	INSE 6150: Security Evaluation Methodologies	68	N/A
2021/2	INSE 6150: Security Evaluation Methodologies	49	N/A
2020/1	INSE 6150: Security Evaluation Methodologies	78	N/A
2018/4	INSE 6150: Security Evaluation Methodologies	92	1.20
2018/4	COMP 249: Object Oriented Programming II	109	1.73
2018/2	INSE 6630: Recent Developments in Info. Systems Security	53	1.19
2018/2	COMP 352: Algorithms and Data Structures	68	1.57
2017/4	INSE 6150: Security Evaluation Methodologies	88	1.69
2017/2	INSE 6110: Foundations of Cryptography	79	1.22
2017/2	INSE 6630: Recent Developments in Info. Systems Security	35	1.71
2016/4	INSE 6150: Security Evaluation Methodologies	59	1.13
2016/2	INSE 6150: Security Evaluation Methodologies	63	1.09
2016/2	INSE 6110: Foundations of Cryptography	79	1.32
2015/4	COMP 249: Object Oriented Programming II	50	1.44
2015/4	INSE 6150: Security Evaluation Methodologies	86	1.15
2015/2	INSE 6110: Foundations of Cryptography	76	1.24
2014/4	COMP 249: Object Oriented Programming II	93	1.81
2014/4	INSE 6150: Security Evaluation Methodologies	86	1.41
2014/2	INSE 6110: Foundations of Cryptography	69	1.55
2013/4	INSE 6150: Security Evaluation Methodologies	46	1.73
2013/2	INSE 6110: Foundations of Cryptography	21	1.11

- *Evaluation is for Question 20: "Overall, the professor is an effective teacher." Score is from 1.00 (best) to 5.00 (worst).*
- *Evaluations were suspended by the university from 2020-2021 due to COVID19*

Teaching Awards

- Teaching Excellence Award, Junior Faculty, ENCS, Concordia University, 2017.

External Lectures (Selected)

- "Decentralized finance (DeFi)," Faculty of Law, University of Ottawa. 22 Mar 2021.
- "Improving usability and trust for moving Bitcoin adoption forward," MAS.S65 - Blockchain Technologies, Massachusetts Institute of Technology (MIT). Guest lecture, 4 Nov 2015.
- "History of cryptocurrencies," Bitcoin and Cryptocurrency Technologies, Princeton University. Guest lecture, Online: Coursera, recorded in Sep 2015.
- COMP 4109: Applied Cryptography, Carleton University. Course, Winter 2013.

Service to University

University Committees

Leaves: Parental 2019-2020; Sabbatical 2020-2021

Year	Committee
2022-	GCS Elections Committee (Chair)
2022-	Concordia University Faculty Tribunal Pool
2021-	GCS Faculty Council
2018-2019	Concordia University Faculty Tribunal Pool
2018-2019	ENCS Blended/Online Pedagogy Committee
2017-2019	ENCS Elections Committee
2015-2019	CIISE PR/Website [Co-Chair]
2013-2019	CIISE Seminar Committee
2014–2016	Concordia University Faculty Tribunal Pool
2014–2015	CIISE Website Committee (merged with PR above)
2013–2015	CIISE PR Committee (merged with Website above)

Graduate Student Committees

Year	Occurrences				
	MASc Defence	PhD Comp.	PhD Proposal	PhD Seminar	PhD Defence
2022	1	2	1		1
2021	3	1	1	1	1
2020	1	1		1	1
2019			2	3	3
2018		3	1		2
2013-2017	6	6	3	4	2

External PhD Examiner

- Md Mamunur Rashid Akand, University of Calgary, 2023
- Farimah Ramezan Poursafaei, McGill, 2022
- Patrick McCorry, Newcastle University, UK, 2017
- Giulia Alberini, McGill, 2015
- Jérôme Dossogne, Université libre de Bruxelles, Belgium, 2015

Service to Academia

Program Chairs

Year	Conference
2024	Financial Cryptography and Data Security 2024 (FC)
2022	Blockchain Technology Symposium (BTS)
2019	Workshop on Advances in Secure Electronic Voting (VOTING)
2018	Workshop on Advances in Secure Electronic Voting (VOTING)
2017	The Smart Cybersecurity Network: Spring 2017 Workshop (SERENE-RISC)
2016	Workshop on Bitcoin and Blockchain Research (BITCOIN)

General Chairs

Year	Conference
2024	Blockchain Technology Symposium (BTS)
2020	Privacy Enhancing Technologies Symposium (PETS)

Advisory Boards

Year	Journal
2019—	Privacy Enhancing Technologies Symposium (PETS)

Editorial Boards

Year	Journal
2013—2015	USENIX Journal of Election Technologies (USENIX JETS)

Program Committees (Selected)

Year	Conference
2023	ACM Computer and Communications Security (CCS)
2023	Workshop on Decentralized Finance (DeFi)
2023	Financial Cryptography and Data Security (FC)

Year	Conference
2022	Workshop on Privacy in the Electronic Society (WPES)
2022	Sixth International Joint Conference on Electronic Voting (E-VOTE-ID)
2022	Workshop on Advances in Secure Electronic Voting (VOTING)
2022	Workshop on Decentralized Finance (DeFi)
2022	Financial Cryptography and Data Security (FC)
2021	IEEE Security & Privacy on the Blockchain (IEEE S&B)
2021	Financial Cryptography and Data Security (FC)
2021	Sixth International Joint Conference on Electronic Voting (E-VOTE-ID)
2021	Workshop on Advances in Secure Electronic Voting (VOTING)
2020	Financial Cryptography and Data Security (FC)
2020	IEEE Security & Privacy on the Blockchain (IEEE S&B)
2019	Financial Cryptography and Data Security (FC)
2019	IEEE Security & Privacy on the Blockchain (IEEE S&B)
2018	APWG Symposium on Electronic Crime Research (eCrime)
2018	Symposium on Usable Privacy & Security (SOUPS)
2018	IEEE Security & Privacy on the Blockchain (IEEE S&B)
2018	Workshop on Bitcoin Research (BITCOIN)
2018	Financial Cryptography and Data Security (FC)
2017	APWG Symposium on Electronic Crime Research (eCrime)
2017	Workshop on Advances in Secure Electronic Voting (VOTING)
2017	Workshop on Bitcoin Research (BITCOIN)
2017	Financial Cryptography and Data Security (FC)
2016	RSA Conference: Cryptographer's Track (CT-RSA)
2016	ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)
2016	IEEE Advanced and Trusted Computing (Bitcoin track)
2016	Workshop on Advances in Secure Electronic Voting (VOTING)
2016	Workshop on Bitcoin Research (BITCOIN)
2016	Financial Cryptography and Data Security (FC)

Year	Conference
2015	International Conference on E-Voting and Identity (VoteID)
2015	Workshop on Bitcoin Research (BITCOIN)
2014	Annual Computer Security Applications Conference (ACSAC)
2014	Conference on Privacy, Security and Trust (PST) – Privacy Theme.
2014	Workshop on Bitcoin Research (BITCOIN)
2013	International Conference on E-Voting and Identity (VoteID)
2012	USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)
2011	USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)

Journals (Most Recent Year / Selected)

Most Recent Year	Journal / Conference
2023	IEEE Security and Privacy Magazine
2022	IEEE Transactions on Information Forensics and Security (TIFS)
2021	Bank for International Settlements (BIS) Working Paper Series
2021	IEEE Transactions on Dependable Secure Computing (TDSC)
2021	Communications of the ACM

Reviews for Funding Agencies (Most Recent Year / Selected)

Most Recent Year	Agency
2023	Israel Science Foundation (ISF)
2023	Natural Sciences and Engineering Research Council of Canada (NSERC)
2022	Social Sciences and Humanities Research Council of Canada (SSHRC)
2020	MITACS
2019	Fonds de Recherche du Québec – Nature et technologies (FRQNT)
2019	Alberta Innovates

Most Recent Year	Agency
2017	Office of the Privacy Commissioner