Database and Distributed Computing Foundations of Blockchains

Sujaya Maiyya Victor Zakhary Mohammad Javad Amiri
Divyakant Agrawal Amr El Abbadi
Department of Computer Science, University of California, Santa Barbara
Santa Barbara, California
{victorzakhary,sujaya-maiyya,amiri,agrawal,amr}@cs.ucsb.edu

ABSTRACT

The uprise of Bitcoin and other peer-to-peer cryptocurrencies has opened many interesting and challenging problems in cryptography, distributed systems, and databases. The main underlying data structure is blockchain, a scalable fully replicated structure that is shared among all participants and guarantees a consistent view of all user transactions by all participants in the system. In this tutorial, we discuss the basic protocols used in blockchain, and elaborate on its main advantages and limitations. To overcome these limitations, we provide the necessary distributed systems background in managing large scale fully replicated ledgers, using Byzantine Agreement protocols to solve the consensus problem. Finally, we expound on some of the most recent proposals to design scalable and efficient blockchains in both permissionless and permissioned settings. The focus of the tutorial is on the distributed systems and database aspects of the recent innovations in blockchains.

KEYWORDS

Permissionless Blockchain, Permissioned Blockchain, Distributed Consensus, Byzantine Faults

ACM Reference Format:
Sujaya Maiyya Victor Zakhary Mohammad Javad Amiri, Divyakant Agrawal Amr El Abbadi. 2019. Database and Distributed Computing Foundations of Blockchains. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299869.3314030

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3314030

1 INTRODUCTION

Bitcoin [26] is considered the first successful global scale peer-to-peer cryptocurrency. The Bitcoin protocol explained by the *mysterious Nakamoto* allows financial transactions to be transacted among participants without the need for a trusted third party, e.g., banks, credit card companies, or PayPal. Bitcoin eliminates the need for such a trusted third party by replacing it with a distributed ledger that is fully replicated among all participants in the cryptocurrency system. This distributed ledger is referred to as *blockchain*.

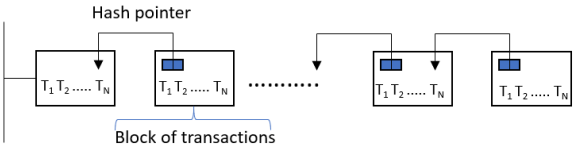

Figure 1: Blockchain is a chain of blocks of transactions linked by hash pointers.

Blockchain, as shown in Figure 1, is a secure linked list of blocks containing financial transactions that occur in the system and linked by hash pointers. The main challenge that Bitcoin addresses is to maintain a consistent view of this replicated blockchain in a secure and fault-tolerant manner in a *permissionless* setting and in the presence of malicious participants. Unlike *permissioned* settings where all the participants in the system are known *a priori*, a permissionless setting allows participants to freely join and leave the system without maintaining any global knowledge of the number of participants. To address these challenges, Bitcoin builds on foundations developed over the last few decades from diverse fields [27], but primarily from the fields of **cryptography** [5, 30], **distributed systems** [8, 20, 21] and **data management** [6, 24, 33] as illustrated in Figure 2. Figure 2 shows the space of techniques that are used in Bitcoin to implement a permissionless decentralized payments. Digital signatures and hashing are the cryptographic foundations that are used in Bitcoin to support distributed transactions stored in a ledger that is replicated across globally distributed sites in the presence of malicious faults.

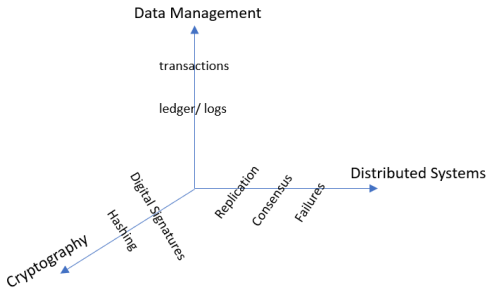

Figure 2: The space of techniques that are used to implement permissionless decentralized cryptosystems.

In spite of its current success, Bitcoin suffers from scalability performance limitations represented by the number of transactions executed per second, especially when compared with commercially successful On Line Transaction Processing (OLTP) financial systems, such as credit card companies. Bitcoin uses a notion of *miners* who need to perform a computationally challenging *Proof of Work (PoW)* puzzle before they can add any block of transactions to the replicated blockchain. Since the PoW puzzle is computationally hard, very few miners can successfully solve the puzzle, and hence a successful miner can add a block to the blockchain and be guaranteed, with very high probability, to be unique. Many concerns have been raised about the wasted massive energy requirements to *mine* one Bitcoin block. In addition, the difficulty of Bitcoin's Proof of Work (PoW) discourages *miners* from mining independently and pushes them to form few powerful *mining cartels*. Concentrating most of the mining power in few hands, or pools, risks to derail the whole idea of decentralization.

This mining approach to determine the process eligible to add a new block to the block chain is in contrast to the distributed systems approach, that has been promoting the use of Byzantine Agreement or consensus, which is efficient and more egalitarian. In fact, consensus protocols such as Paxos have been quite successful in recent years in laying the foundations of large global scale data management system. Unfortunately, Paxos has many limitations, especially from a global cryptocurrency point of view, including the requirement of a permissioned setting, and that participants can only fail by crashing. An alternative to Paxos that tolerates malicious failures is Practical Byzantine Fault-Tolerance (PBFT) [8]. Although it tolerates malicious failures, PBFT still requires a permissioned setting, and requires a large number of message exchanges, hence does not scale to the large number of participants expected in modern day permissionless cryptocurrencies. Recently, the security and distributed systems communities have been aggressively exploring alternative scalable solutions. These systems attempt to solve the Bitcoin protocol shortcomings in permissionless, e.g., BitcoinNG [13], Byzcoin [18], Elastico [22], and Algorand [15], or permissioned, e.g., Tendermint [19], Hyperledger Fabric [4], and Quorum [25], settings using efficient and practical solutions that integrate cryptographic and consensus mechanisms.

In this tutorial, our goal is to present to the database community an in-depth understanding of state-of-the-art solutions for efficient scalable blockchains. We progress towards this goal by starting from a detailed description of the protocols and techniques underlying the design of Bitcoin. Since most recent innovations in blockchain design depend critically on consensus protocols in malicious settings, we outline the basic foundations of distributed fault-tolerant consensus protocols. This is followed by a discussion of the most recent proposals to solve the blockchain design problem using scalable fault-tolerant solutions.

2 TUTORIAL OUTLINE

2.1 Background

Consensus and *Byzatine Agreement (BA)* are well studied problems that were first proposed by Lamport, Shostak and Pease in 1982 [21]. Any solution to the BA problem tries to reach agreement among a well defined set of processes on a single value. The distributed systems community has extensively explored this problem in both synchronous and asynchronous systems. In an asynchronous system, the Fisher Lynch and Patterson (FLP) impossibility result states that consensus is not guaranteed to terminate in the presence of even a single crash failure [14]. This led to many BA protocols for synchronous systems, where the lower bound requires that the number of maliciously faulty processes is at most one third of the total number of processes. Synchronous BA protocols require multiple rounds of communication and extensive message passing. On the other hand, several efficient asynchronous BA protocols have been developed based on Lamport's Paxos protocol [20]. The main challenge such asynchronous protocols face is that they do not guarantee termination. However, many systems have been designed that depend on Paxos, and have been practically successful [7, 9]. Paxos, however, assumes that processes may only fail by crashing. In a cryptocurrency setting, processes may act in a malicious manner. In 1999, Castro and Liskov proposed the Practical Byzantine Fault Tolerance (PBFT) algorithm [8], which is similar to Paxos in that it uses a small number of message rounds and assumes an asynchronous system. However, it tolerates malicious faults. During the tutorial, we will provide a general overview of the consensus problem and a high level description of various BA protocols. In particular, we will provide a detailed description of PBFT, given its

particular relevance to many recent cryptocurrency proposals, as discussed in the next section. We will also highlight the main advantages and limitation of these distributed BA protocols.

2.2 Bitcoin and Nakamoto's Consensus

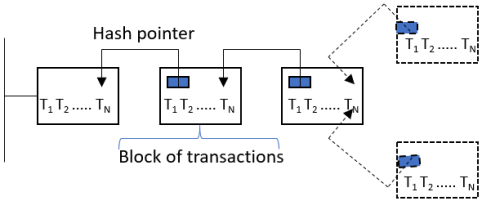

Figure 3: A fork in the blockchain.

The problem of *double spending* is one of the main challenges of the Bitcoin protocol. Double spending happens when a coin owner signs *two* concurrent transactions trying to spend the same coin twice. This *concurrency* anomaly can easily be prevented if transactions are serialized [6]. Bitcoin relies on a network of *miners* to achieve serializability. Every financial transaction is broadcast to all sites/miners in the system. Each miner receives these digitally signed transactions and groups them into a new block. A miner "mines" to add this block to the blockchain but before doing that, these transactions need to be verified. The verification process ensures that the transactions in a new block are neither conflicting with each other nor conflicting with other transactions in any preceding block in the current blockchain.

To add a block to the blockchain, miners need to perform a computationally challenging *Proof of Work (PoW)* puzzle before they can add their block of transactions to the replicated blockchain. Since the PoW puzzle is computationally hard, very few miners can successfully solve the puzzle, hence a successful miner can add a block to the blockchain and be guaranteed, with very high probability to be unique. Serializability is ensured by adding verified blocks one at a time. After a block is added, the miner who "mined" this block broadcasts it to all other miners. Miners are incentivized to accept the first received block, add it to their copy of the blockchain, and immediately start mining for the next block.

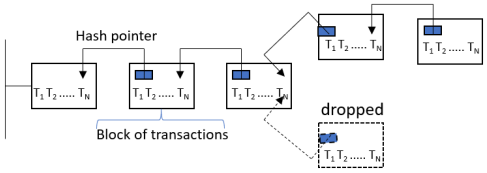

Figure 4: Miners join the longest chain to resolve forks.

The difficulty of PoW aims to minimize the probability that two miners solve the puzzle at the same time. However, with low probability, more than one miner can concurrently reach a solution to the puzzle causing a fork in the blockchain as shown in Figure 3. Forks are a violation of serializability and cause double spending as transactions in the two added blocks can have conflicts with each other. Forks divide the miners network into two groups and each group independently mines for the next block. Once a block is added to either of the fork branches, miners in both groups join the longest chain and drop the other branch of the fork as shown in Figure 4. Transactions in the dropped block are considered *aborted* and need to be resubmitted to the network again. As blocks can be dropped after being added to the blockchain, transactions should not be considered *committed* unless their blocks are buried deep in the blockchain, typically a depth of 6 blocks. A block that is buried in the chain for more than 6 blocks is guaranteed with very high probability not to be dropped, and hence transactions in that block will probably not be aborted. However, if 51% of the mining power maliciously collude, they can redo the whole blockchain risking the safety of the network and causing committed transactions to be dropped even if they were deeply buried in the blockchain. This is widely known as the *51% attack* [10].

2.3 Enhancing Permissionless Blockchain Performance

This section discusses some recent state-of-the-art proposals in permissionless crytocurrency, *i.e.*, any server can attach (or detach) itself from the network of servers that are mining the currency. Bitcoin executes 7 transactions per second [11] whereas trusted, centralized systems such as Visa execute thousands of transactions per second [11]. The recent works are focused on tackling the performance limitation posed by Bitcoin.

BitcoinNG [13] separates the blocks in the chain into *key-blocks*, which is created using proof-of-work and *micro-blocks*, which contains transactions chosen by the latest key-block miner. BitcoinNG increases the throughput of Bitcoin by allowing the key-block miner to be a *leader* who then can publish many micro-blocks, until the next key-block is mined. But allowing one miner to be a leader, even for a brief interval, presents many concerns. ByzCoin [18] identifies that issue and instead of having a single miner act as a leader, forms a dynamically changing group of *trustees*, who collectively decide on the micro-blocks. Each time a key-block is mined, the trustee group changes by one member. The trustees run traditional PBFT [8] to obtain consensus on the next micro-block and use Collective Signing (CoSi) [31] to collectively sign the chosen block. Elastico [22] is another distributed ledger solution with a key idea of splitting all the

servers in the system into smaller sized groups called *committees*, each of which is responsible for processing a subset or a *shard* of transactions. Every committee runs classical PBFT to agree on a set of transactions; these transactions are sent to a special committee called the *final committee*, which then aggregates the transactions obtained from different committees and runs another round of PBFT to make a global, final decision on the next block that is appended to the blockchain.

After discussing some of the distributed ledger solutions that use PBFT as a way to achieve Byzantine consensus, we now explore Algorand [15] which proposes a novel Byzantine Agreement protocol (BA*). As with the previously discussed approaches, Algorand also uses smaller committees to obtain consensus. It uses a *cryptographic sortition* method to randomize committee member selection. Each committee then uses BA*, a priority based protocol, to choose the next block to be appended to the chain. With this new byzantine agreement protocol, Algorand attains throughput 125x of Bitcoin's throughput.

The overall takeaway from the above protocols is that they all strive towards avoiding forks of the blockchain by reaching *finality* in consensus on the new block to be appended. To reach immediate commitment, all these protocols significantly reduce the size of the consensus group. Each protocol uses different cryptographic techniques to guarantee randomness in committee formation, and overall, achieve better throughput than Bitcoin. During the tutorial, we highlight the main advantages and limitations of each of these approaches.

2.4 Permissioned Blockchains

A *permissioned* blockchain consists of a set of known, identified participants but which do not fully trust each other. Since the participants are known and identified, permissioned blockchains can benefit from many techniques developed in the area of distributed computing over decades for reaching consensus, replicating state, and broadcasting transactions. This section discusses some recent state-of-the-art permissioned blockchains.

Permissioned blockchains mainly follow an *order-execute* architecture where a set of peers (might be all of them) validates the transactions, agrees on a total order for the transactions (using a consensus protocol), puts them into blocks and multicasts them to all the peers. Each peer then validates the block, executes the transactions, and updates the ledger.

Since the participants of a blockchain are known *a priori*, depending on the trust model among the participants, a Byzantine fault-tolerant protocol, e.g., PBFT [8], or a crash fault-tolerant protocol, e.g., Paxos [20], is used to reach agreement on a total order of the transactions.

Tendermint [19] is a permissioned blockchain that differs from the original PBFT in two ways. First, only a subset of nodes, called *validators*, participate in the consensus protocol. To become validators, nodes have to lock their coins and once a validator is found to be dishonest, it would be punished. Each validator has voting power equal to the amount of the locked coins (in contrast to PBFT where nodes have equal voting power). Second, Tendermint uses the idea of leader rotation where the leader is changed after every block. This technique is first introduced by the Spinning protocol [32] in the domain of Byzantine fault-tolerant consensus.

Blockchains execute programmable transaction logic in the form of *smart contracts* [1]. Smart contracts are written in domain specific languages, e.g. Solidity, to ensure deterministic execution. In the order-execute architecture, every "smart contract" runs on all nodes which results in confidentiality issues. Smart contracts include the logic of applications and it might be desired to restrict access to such contracts. While cryptographic techniques are used to achieve confidentiality, the considerable overhead of such techniques makes them impractical. Furthermore the sequential execution of transactions on all nodes reduces the blockchain performance in terms of throughput and latency.

In contrast to the order-execute architecture, Hyperledger Fabric [4] employs the *execute-order* architecture by switching the order of execution and ordering. Execute-order architecture is first introduced in Eve [17] in the context of Byzantine fault-tolerant SMR. In Hyperledger peers (*endorsers*) execute transactions concurrently and then a separate set of peers, called *orderers*, orders the transactions using a consensus protocol. Hyperledger is extensible as it supports modular consensus protocols. Different consensus protocols can be plugged to the platform depending on the trust model among the participants. In addition, since the execution is the first phase, Hyperledger supports non-deterministic execution. As a result, general-purpose programming languages, e.g. Java, can be used to write smart contracts.

While Fabric guarantees the confidentiality of smart contracts by executing each transaction on a specified subset of peers and increases the performance of blockchains by executing the transactions in parallel (instead of sequentially as the order-execute paradigm does), it performs poorly on workloads with high-contention, i.e., many *conflicting transactions* in a block, due to its high abort rate. To solve this problem in Parblockchain [3], a new paradigm, called *OXII*, consisting of ordering and execution phases is presented. In the ordering phase, a separate set of nodes establishes agreement on the order of the transactions of different applications, constructs the blocks of transactions, and also generates a *dependency graph* for the transactions within a block. A dependency graph, on the one hand, gives a partial order based on the conflicts between transactions, and, on

the other hand, enables higher concurrency by allowing the parallel execution of non-conflicting transactions. Then, the nodes of each application execute the transactions of the corresponding application following the dependency graph. As long as the partial order of transactions in the dependency graph is preserved, the transactions of different applications can be executed in parallel.

Finally, the increase in the number of permissioned blockchain platforms motivated Dinh et al. [12] to develop BlockBench, an evaluation framework for analyzing private blockchains. Blockbench is considered as the first benchmarking framework that implements different workloads to evaluate and compare private blockchains based on throughput, latency, scalability, and fault-tolerance.

2.5 Off-Chain and Cross-Chain Solutions

Another way to address the scalability limitations of permissionless blockchains is to execute transactions off-chain. Lightening network [29] allows untrusted peers to open direct payment channels where peers execute off-chain micro-payments without committing every micro-payment transaction to the underlying blockchain. A payment channel is comprised of a contract that is committed to the blockchain and holds the funds of the transacting peers. Micro-payments use signatures to verify that peers agree on a transaction. In addition, micro-payments use **hashlocks** and **timelocks** to ensure that a malicious or uncooperative participant cannot take advantage of a conforming participant. The tutorial presentation covers the lightening network protocol in details.

The wide adoption of permissionless open blockchain networks by both industry (e.g., Bitcoin [26], Ethereum [34], etc) and academia (e.g., Bzycoin [18], Elastico [22], BitcoinNG [13], Algorand [15], etc) suggests the importance of developing protocols and infrastructures that support peer-to-peer atomic cross-chain transactions. The permissionless blockchain ecosystem requires infrastructure enablers and protocols that allow users to be able to atomically exchange tokens and resources without depending on centralized intermediaries such as exchanges. A two-party atomic cross-chain swap protocol was originally proposed by Nolen [2, 28] and generalized by Herlihy [16] to process multi-party atomic cross-chain swaps. Both Nolan's protocol and its generalization by Herlihy use smart contracts, hashlocks and timelocks to achieve atomic cross-chain swaps. The tutorial presentation discusses the details of the atomic cross-chain swap protocol by both Nolen and Herlihy.

3 TUTORIAL INFORMATION

This is a **three hours** tutorial targeting researchers, designers, and practitioners interested in large scale transaction support in both permissionless and permissioned blockchain consensus-based research. The **target audience** with basic background about distributed consensus should benefit the most from this tutorial. For the general audience and newcomers, the tutorial explains the design space of distributed consensus and blockchains. This tutorial differs from previous tutorials on the same topic in database conferences, especially C. Mohan [23], where he explicitly states that the scope of his tutorial "is general in nature without getting into the nitty gritty of, e.g., cryptographic algorithms or the distributed consensus protocols". Furthermore, Mohan's tutorial "only discuss(es) permissioned/private blockchains and not permissionless ones". This tutorial, in contrast, focuses on both *permissionless* and *permissioned* blockchains and discusses the distributed protocols and their interaction with user transactions in detail. The cryptographic algorithms will be specified and their properties will be discussed to help understand the distributed systems and database implications.

This tutorial was presented in VLDB 2018 in Rio de Janeiro. The tutorial differs from its last iteration in taking a holistic view of advancements made in the Blockchain community. Starting with the traditional consensus protocols, the tutorial delves into the details of Nakamoto consensus and discusses many academic works in both permissionless and permissioned blockchains. The tutorial then presents solutions such as Atomic Swap and Lightning networks that solve the challenges stemming from the rise of multiple blockchain systems.

4 BIOGRAPHICAL SKETCHES

Sujaya Maiyya is a PhD student at the University of California at Santa Barbara. Her current research work is targeted towards building consensus and commitment protocols in a byzantine environment and to integrate these protocols into large scale data management systems.

Victor Zakhary is a PhD student at the University of California at Santa Barbara. His current research work is in the areas of data placement for geo-replicated databases and for distributed caching services to achieve low access latency and dynamically handle data access skewness.

Mohammad Javad Amiri is a PhD student at the University of California at Santa Barbara. His current work spans research topics such as Distributed systems, large-scale data management, and business processes management.

Divyakant Agrawal is a Professor of Computer Science at the University of California at Santa Barbara. His current interests are in the area of scalable data management and data analysis in cloud computing environments, security and privacy of data in the cloud, and scalable analytics over big data. Prof. Agrawal is an ACM Distinguished Scientist (2010),

an ACM Fellow (2012), an IEEE Fellow (2012), and an AAAS Fellow (2016).

Amr El Abbadi is a Professor of Computer Science at the University of California, Santa Barbara. Prof. El Abbadi is an ACM Fellow, AAAS Fellow, and IEEE Fellow. He was Chair of the Computer Science Department at UCSB from 2007 to 2011. He has served as a journal editor for several database journals and has been Program Chair for multiple database and distributed systems conferences. Most recently Prof. El Abbadi was the co-recipient of the Test of Time Award at EDBT/ICDT 2015. He has published over 300 articles in databases and distributed systems and has supervised over 30 PhD students.

5 ACKNOWLEDGEMENT

This work is funded by NSF grants CNS-1703560 and CNS-1649469.

REFERENCES

[1] [n. d.]. Ethereum blockchain app platform. https://www.ethereum.org. ([n. d.]).
[2] 2018. Atomic cross-chain trading. https://en.bitcoin.it/wiki/Atomic_cross-chain_trading. (2018).
[3] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. 2019. ParBlockchain: Leveraging Transaction Parallelism in Permissioned Blockchain Systems. *arXiv preprint arXiv:1902.01457* (2019).
[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *arXiv preprint arXiv:1801.10228* (2018).
[5] Adam Back. 2002. Hashcash-a denial of service counter-measure. (2002).
[6] Philip A Bernstein, Vassos Hadzilacos, and Nathan Goodman. 1987. Concurrency control and recovery in database systems. (1987).
[7] Mike Burrows. 2006. The Chubby lock service for loosely-coupled distributed systems. In *The 7th symposium on Operating systems design and implementation (OSDI)*. USENIX Association, 335–350.
[8] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.
[9] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. 2008. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)* 26, 2 (2008), 4.
[10] Mauro Conti, Chhagan Lal, Sushmita Ruj, et al. 2017. A survey on security and privacy issues of bitcoin. *arXiv preprint arXiv:1706.00916* (2017).
[11] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, et al. 2016. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*. Springer, 106–125.
[12] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. Blockbench: A framework for analyzing private blockchains. In *ACM International Conference on Management of Data*. ACM, 1085–1100.
[13] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-NG: A Scalable Blockchain Protocol.. In *NSDI*. 45–59.
[14] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.
[15] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *The 26th Symposium on Operating Systems Principles*. ACM, 51–68.
[16] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, 245–254.
[17] Manos Kapritsos, Yang Wang, Vivien Quema, Allen Clement, Lorenzo Alvisi, Mike Dahlin, et al. 2012. All about Eve: Execute-Verify Replication for Multi-Core Servers.. In *OSDI*, Vol. 12. 237–250.
[18] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. 2016. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium*. 279–296.
[19] Jae Kwon. 2014. Tendermint: Consensus without mining. (2014).
[20] Leslie Lamport et al. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.
[21] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.
[22] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 17–30.
[23] C Mohan. 2017. Tutorial: blockchains and databases. *PVLDB* 10, 12 (2017), 2000–2001.
[24] C Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. 1992. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems (TODS)* 17, 1 (1992), 94–162.
[25] JP Morgan. 2016. Quorum whitepaper. *En línia]. Available: https://github. com/jpmorganchase/quorumdocs/blob/master/Quorum% 20Whitepaper% 20v0* 1 (2016).
[26] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[27] Arvind Narayanan and Jeremy Clark. 2017. Bitcoin's academic pedigree. *Commun. ACM* 60, 12 (2017), 36–45.
[28] Tier Nolan. 2013. Alt chains and atomic transfers. https://bitcointalk.org/index.php?topic=193281.msg2224949/msg2224949. (2013).
[29] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. *See https://lightning. network/lightning-network-paper. pdf* (2016).
[30] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.
[31] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Keeping authorities" honest or bust" with decentralized witness cosigning. In *IEEE Symposium on Security and Privacy (SP)*. Ieee, 526–545.
[32] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2009. Spin one's wheels? Byzantine fault tolerance with a spinning primary. In *28th IEEE International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 135–144.
[33] Gerhard Weikum and Gottfried Vossen. 2001. *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. Elsevier.
[34] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.