Doctoral theses at NTNU, 2022:241

Doctoral thesis

Mayank Raikwar

Cryptography for Innovative Blockchain Services

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Mayank Raikwar

Cryptography for Innovative Blockchain Services

Thesis for the Degree of Philosophiae Doctor

Trondheim, August 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

© Mayank Raikwar

ISBN 978-82-326-5180-1 (printed ver.)
ISBN 978-82-326-6420-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2022:241

Printed by NTNU Grafisk senter

Abstract

Since the advent of Bitcoin, scientific interest in its underlying core technology *Blockchain* has been thriving. Much work has been carried out for blockchain use cases in different industrial areas such as healthcare, Internet of Things (IoT), supply chain, and decentralized finances, to name a few. The research addressed some generic and well-defined topics adapted as challenges for the blockchain, such as its security, privacy, scalability, and fairness. To solve these challenges, a plethora of research employed cryptography as its founding basis.

The thesis aims to address the challenges mentioned above. The starting point of the thesis is to investigate and scrutinize cryptographic primitives, schemes, and protocols that are or can be used to solve some of the issues identified in blockchain and improve the current state-of-the-art designs. The thesis consists of four main topics. The first topic addresses how to construct an energy-efficient, fair consensus mechanism. For that, two novel consensus mechanisms are presented in the thesis. The second topic covers the use of client puzzles for Denial of Service (DoS) attack mitigation. The thesis includes two works about DoS attacks; the first work presents a new construction of a client puzzle scheme; the second work studies the DoS attack in the blockchain ecosystem and proposes a few mitigation techniques, including a client puzzle scheme. The third topic in the thesis assesses the privacy of cryptocurrency systems. Two papers contribute to this topic: the first paper offers a novel construction of a privacy-preserving cryptocurrency system and models the system's security; the second paper employs the work developed in the first paper to construct a general security model for the existing privacy-preserving cryptocurrency systems. The fourth topic is about decentralized randomness beacon protocols. These protocols are essential to generate publicly verifiable, trusted randomness used in consensus mechanisms and smart contracts. Two papers contribute to this topic: the first paper presents a systematization of knowledge of existing decentralized randomness beacon protocols; the second paper proposes a new protocol for randomness generation using blockchain as a bulletin board.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree Philosophiae Doctor (PhD) at NTNU, Norwegian University of Science and Technology. The presented work was carried out at the Department of Information Security and Communication Technology (IIK), Trondheim, in the period from February 2019 - May 2022, under the supervision of Professor Danilo Gligoroski and the co-supervision of Professor Kristian Gjøsteen, Professor Colin Boyd and Associate Professor Katina Kralevska.

Acknowledgements

First, I must thank my supervisor Danilo Gligoroski for his guidance and encouragement throughout my PhD and for giving me enough freedom to explore my research direction. I would like to thank Kristian Gjøsteen for his guidance, insightful feedback, and fruitful discussions. I would next like to thank Colin Boyd for being a constant source of encouragement, especially during the dark days of the pandemic. Many thanks to Katina Kralevska for her useful advice and help during the initial days of PhD study.

I am fortunate to have constant support from Sushmita Ruj, who introduced me to the research world, both on a work and personal level. I am incredibly grateful to her for helping me climb the ladder of academia.

I would like to thank my co-authors Kristian Gjøsteen and Shuang Wu for the collaboration. The collaboration made me gain an improved understanding of some fundamentals of cryptography. I would also acknowledge all the NaCl group members for making the research environment enjoyable and delicious with weekly cakes. I also appreciate the effort and support from the members of the DT-Blockchain project.

Many thanks to Mona Nordune, Pål Sæther, Poul Heegaard, and Maria Sofie for making the department very welcoming, lively, and energetic. Mona deserves special credit for being an incredibly helpful and supportive person throughout my PhD stint.

I would like to acknowledge my office mates, Befe and Ali, with whom I spent the majority of my PhD life. From being office mates to flatmates, we became great friends in determination.

My list of friends to thank is long, so please bear with me. I would start by thanking my dearest friends in Trondheim, with whom I shared many laughs while enjoying a delicious homemade meal: Shuang, Faiga, Mattia, Sonu, Jabir Ali, George. These people made my PhD life much easier by showing their constant care and support.

In the department, I have been lucky to have made many friends with whom many enjoyable hours have been spent procrastinating by the coffee machine: Murad, Lise, Stas, Jonathan, Bor, Julie, Sruti, Sahana, Lea.

I would like to thank my Norwegian friends outside the department. Thanks to Hans Olav and Dag for being very welcoming when I arrived in Norway. They have always been a great company to be with. Many thanks to Tjerand for being a great friend and a fast-running buddy. Special thanks to Tom and Mr. Roland for always being present for me and being a great source of inspiration and encouragement.

I would like to thank my friends back in India: Kamlesh, Deep, Ashutosh, Gaurav, Diwakar, Amitesh. I also thank my friends back in Singapore: Nishant, Bali, Suman, Gaurav, and Arko. Even though being geographically distant,

these friends have always been cheering me up and made my PhD journey exhilarating.

My final thanks are reserved for my family: my parents, my sisters Chitra, Akanksha, and my little cutie pie Mini. Their unconditional love, support, and faith have shaped me into the person I am today.

Finally, over the last 2 odd years, the world has been ravaged by COVID-19. I consider myself incredibly privileged to be around a great group of friends, colleagues, and family. I would like to thank all of them, and I apologize if I have missed anybody.

Contents

List of Figures

List of Tables

List of Acronyms

ASIC Application-Specific Integrated Circuit.

aSVC Aggregatable Subvector Commitment.

BFT Byzantine Fault Tolerant.

CRS Common Reference String.

DDoS Distributed Denial of Service.

DoS Denial of Service.

DRB Decentralized Randomness Beacon.

ECDSA Elliptic Curve Digital Signature Algorithm.

EdDSA Edwards-curve Digital Signature Algorithm.

EUF-CMA Existential Unforgeability under Adaptive Chosen Message Attack.

FBA Federated Byzantine Agreement.

GPU Graphic Processing Unit.

IoT Internet of Things.

MAC Message Authentication Code.

NIZK Non-Interactive Zero-Knowledge.

PIR Private Information Retrieval.

PoA Proof of Authority.

PoS Proof of Stake.

PoW Proof of Work.

PPoS Private Proof of Stake.

PPT Probabilistic Polynomial-Time.

ProgPoW	Programmatic Proof-of-Work.
SoK	Systematization of Knowledge.
UC	Universal Composition.
UTXO	Unspent Transaction Output.
VDF	Verifiable Delay Function.
VRF	Verifiable Random Function.
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.

Part I

Summary

Chapter 1

Introduction

More than a decade ago, academia and industry witnessed the birth of new disruptive technology: *Blockchain*. Blockchain, in simple words, is a distributed ledger managed by a peer-to-peer network collectively adhering to some consensus protocol. Bitcoin [1] was the first application that introduced blockchain technology by creating a decentralized environment for a cryptocurrency, where the participants can buy or exchange goods with digital money. Since the advent of Bitcoin, there has been an avalanche of cryptocurrencies that all together built a financial market worth around $1.4 Trillion (as of 13 May 2022) [2].

Analyzing the underlying fundaments of blockchain leads to cryptography that makes the blockchain reliable and immutable. The origin of the idea of using cryptography for secure transactions traces back to the 1990s with David Chaum's eCash system [3]. The eCash system was used for payments regarding internet shopping, online money transfers, and access to online services. The negative aspect of eCash was that it was operated and controlled by a trusted third party that hurdled the broader acceptance of the eCash system.

The development of several cryptographic ideas related to financial transactions also started after 1990. Dwork and Naor proposed the use of computation-expensive functions to combat junk emails in 1992 [4]. Later, other proposals using computation-expensive function were proposed [5, 6]. A cryptography practitioner and an implementor known under the pseudonym "Satoshi Nakamoto" also embraced the idea of employing a computation-expensive function and presented the first Bitcoin blockchain [1].

Besides computation-expensive cryptographic function, blockchain utilizes additional cryptographic concepts such as public-key cryptography (digital signatures) and public key management. The core segment of blockchain is its consensus mechanism, and for Bitcoin, it is known as *Proof of Work*. In practice, it was proven that it could be implemented successfully [7], having as evidence the monumental success and the rise of Bitcoin as a dominant cryptocurrency. Nonetheless, many other cryptographic primitives have been proposed and applied to design consensus mechanisms to improve blockchain efficiency, security, and reliability. Blockchain research has gained pace in recent years since the evolution of cryptographic primitives such as Zero-knowledge Proofs [8, 9, 10] and Verifiable Delay Function (VDF) [11, 12, 13], improving efficiency, throughput, and security.

The main principle under which the source code of Bitcoin was published was *Open Source*. Consequently, it provoked thousands of programmers and cryptography practitioners to launch alternative cryptocurrencies, each of them claiming some potential improvements. But the disruption continued further, and blockchain technology was heralded both by academia and industry as a game-changer in the financial sector due to the evolution of cryptocurrencies to a broad range of applications, including the Internet of Things (IoT) [14], supply-chain [15], insurance industries [16] and healthcare [17].

From a development feedback loop perspective, we can now see that blockchain technology has stimulated considerable research in constructing new distributed consensus mechanisms for privacy-preserving cryptocurrencies [18, 19, 20]. More concretely, recent research trends put more emphasis on the privacy aspects of modern cryptocurrencies. Alternatively, there has been an immense amount of work toward improving the scalability of the blockchain. We can say that nowadays, there is a growing need for having an ideal balance between privacy and scalability in the blockchain.

Although blockchain is envisioned as a promising and powerful technology, it still encounters many research challenges apart from privacy and scalability. We can name a few: permanent improvement of the blockchain security, key management, analysis of new attacks of the blockchain components and attacks on the whole blockchain ecosystems, smart contracts management, and incremental introduction of new cryptographic features in the existing blockchains. These challenges arise due to the underlying network structure, consensus mechanisms, and the cryptographic schemes used within the blockchains. Many cryptographic concepts such as signature schemes, zero-knowledge proofs, and commitment protocols are scrutinized and applied to overcome the existing challenges in blockchain and find enhanced solutions. As the field of cryptography is vast, it opens many directions to explore and discover its applicability to construct new solutions or improve the existing solutions in the blockchain domain.

1.1 Motivation

The concept of decentralization with blockchain technology brings an incredible number of research questions and various practical applications. For example, besides the traditional questions of security and privacy, the concept of decentralization brought another challenge. Namely, as the decentralization removes the trusted authorities and is an alternative way for monetary transactions, the growth of the blockchain platforms faced the following problem: the need to process an immense number of transactions on these platforms. Thus, a new research challenge appeared: blockchain scalability.

The research community continuously contributes to solving the challenges connected with the blockchain's security, privacy, and scalability. These challenges persist both for cryptocurrencies and the blockchain protocols for transactions of assets. However, apart from these challenges, we observed that the fairness in blockchain protocols was somewhat disregarded [21]. Fairness in blockchain protocols typically refers to the fair contribution of each participant. Fair contribution refers to that the participants will get a fair chance to participate and produce new blocks in blockchain irrespective of their available resources (e.g., computational power, stake). Otherwise, a participant with more resources will have an advantage in producing new blocks and subsequently earning the block incentive. This scenario will bring the blockchain system towards centralization as participants with more resources will be controlling the system by adding new blocks in the blockchain.

The motivation for the research conducted and presented in this thesis comes from the aforementioned challenges. To address them, the thesis first aims to scrutinize the cryptographic primitives that have been employed or have the potential to be utilized in the blockchain. Then, the thesis seeks to comprehend the definition of fairness in Proof of Stake (PoS) consensus [22] context and to construct new consensus mechanisms to provide better fairness.

The second motivation comes from the increasing intensity and frequency of Denial of Service (DoS) attacks not only on the blockchain ecosystems but also on the internet industries. Due to the enormous potential of the cryptocurrency market, the severity of the DoS attacks poses a great concern. Therefore, the thesis presents an investigation of the DoS attacks on the blockchain ecosystems and constructs a general puzzle scheme for DoS mitigation.

The third motivation for the thesis focuses on the privacy and scalability of cryptocurrency systems. Since the introduction of Layer-2 protocols [23], there has been a tremendous amount of work to solve the scalability of the blockchain. Layer-2 protocols, in simple words, provide a way to move a vast amount of transactions off-chain and put a succinct, verifiable update on-chain about these transactions. In this way, the blockchain processes more transactions and achieves scalability. Even though there are many Layer-2 protocols, e.g., payment channel networks [24], sidechains [25], and commit-chain [26], the privacy of off-chain transactions and the users' privacy have not been addressed. The thesis presents a construction that achieves privacy on top of a scalable solution. The construction is further used to create a general framework to assess the security of existing privacy-preserving cryptocurrency systems.

Most of the constructions e.g., fair PoS consensus mechanism, and puzzle scheme for DoS mitigation presented in the thesis require secure and verifiable randomness. Randomness is a crucial element needed in many applications such as secure parameter generations in cryptographic protocols [27], byzantine fault-tolerant protocols [28], E-voting [29], privacy-preserving messaging services [30], online gaming [31], blockchain, and smart contracts [32]. Decentralized Randomness Beacon (DRB) protocols model public, distributed, reliable, trusted, and verifiable randomness generation. In recent years, there have been numerous constructions of DRB protocols using different cryptographic primitives [28, 33, 34, 35, 36, 37, 38]. Nevertheless, there is no systematization of these DRB protocols, which motivates the thesis to present a Systematization of Knowledge (SoK) of existing DRB protocols. Further, the thesis also describes a new class of DRB protocol called competitive DRB and offers a novel construction of DRB protocol under the new class.

1.2 Research Questions

The objective of the thesis is to study state-of-the-art cryptography in the blockchain. That means identifying existing research problems in the blockchain domain and investigating novel solutions to the research problems. The following research questions have been identified and accounted for in this thesis:

- **RQ1** How can we provide fairness to the participants in the blockchain ecosystem, particularly with a proposal of a novel consensus mechanism?
- **RQ2** How can we improve the security in the blockchain ecosystem concerning resistance to DoS attack?
- **RQ3** How can we achieve and assess privacy in cryptocurrency systems?
- **RQ4** How can we provide or improve scalability in the blockchain protocols?

1.3 Thesis Structure

The thesis is a collection of papers. This thesis is composed of mainly two parts. Part I presents a comprehensive summary of the thesis and provides the overall motivation, objective, and main results from the thesis work. It comprises four chapters. Chapter 2 presents the necessary background knowledge required for the thesis and the related works. Chapter 3 provides a brief overview of the research contributions of the thesis and also answers the research question defined in Section 1.2. Finally, Chapter 4 presents the concluding remarks and future research directions.

Part II includes 9 contributing papers where 7 are published, 1 is accepted (to be presented), and 1 is submitted for a peer-reviewed publication. In addition to the main two parts, Part III presents the abstracts of the 5 related secondary papers that are not included in the thesis. The secondary papers do not directly answer the research questions but emphasize more the topics such as cryptographic primitives in blockchain, the relation between databases and blockchain, and aggregation in the blockchain ecosystem.

Chapter 2

Background and Related Works

This chapter aims to present the underlying background knowledge and most important related works relevant to the thesis. Thus, this chapter explains blockchain technology and the significant challenges faced by the technology. First, we describe what blockchain is and what are its components. One of the core parts of blockchain is its consensus mechanism. Henceforth, further, we discuss consensus mechanisms in detail. Additionally, we present the major issues in the blockchain, which are security, privacy, scalability, and fairness. The works included in the thesis employ a few cryptographic primitives to address some of the significant issues in the blockchain. We describe these cryptographic primitives in the last part of the chapter.

2.1 Blockchain

Blockchain was first described in a 2008 white paper entitled "*Bitcoin: A Peer-to-Peer Electronic Cash System*" as the underlying core technology. Blockchain is a distributed ledger that maintains a continuously growing list of records confirmed by the participating nodes operating over a Peer-to-Peer network. A record refers to a block of valid transactions kept in the public ledger and shared among participating nodes. Hence, the nodes participating in the blockchain have a local copy of the ledger.

Blockchain technology is arguably considered a disruptive technology, and it has evolved rapidly in the past decade. The success of blockchain is traced back to the financial success of the Bitcoin cryptocurrency, which subsequently provoked the appearance of thousands of alternative cryptocurrencies. The rationale for the success of blockchain is its set of unique features such as immutability, transparency, distributed and trusted ledger without any central party, and secure smart contracts.

Figure 2.1 (Fig. 2 in [39]) depicts the structure of a blockchain together with the block format. There are mainly two fundamental cryptographic building blocks in the blockchain: 1) Hash Function and 2) Digital Signature. Blockchain relies on these building blocks for its security. In the following subsections, first, we illustrate these two main constituents in detail. Further, we present a brief overview of Merkle tree, ledger, and blockchain types.

2.1.1 Hash Function

Hash functions are split into two classes: keyed hash function and un-keyed hash function [40]. Keyed hash functions are primarily used to construct Message Authentication Code (MAC) and due to their limited use in blockchain, keyed

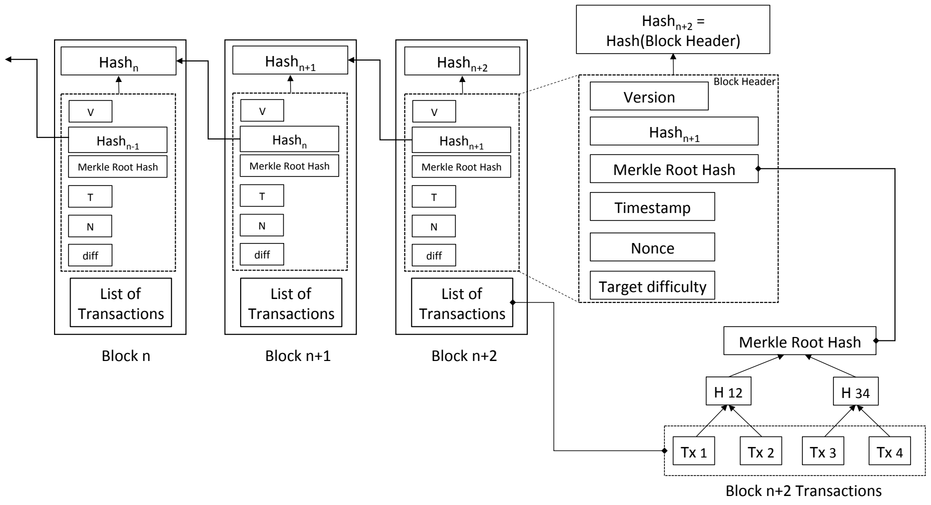

Figure 2.1: Blockchain data structure [39].

hash functions are not explored in this thesis. Following, we treat a hash function as an un-keyed hash function used in blockchain.

Hash function in the blockchain is used for various purposes, including address generation, solving a cryptographic puzzle (e.g., in Bitcoin), and shortening the public addresses. A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is an algorithm that maps an arbitrary finite size input to a fixed size output. A cryptographic hash function should have the following security properties:

1. *Preimage Resistance* implies that hash function is not invertible, means given an output y it is hard to find an input x such that $H(x) = y$.

2. *Second Preimage Resistance* infers that given an input x, it is hard to find x' where $x \neq x'$, such that $H(x) = H(x')$.

3. *Collision Resistance* implies that it is hard to find a pair of input x, x' mapping to the same hash value, which means $H(x) = H(x')$.

A hash function having properties 1 and 2 is called a one-way hash function, whereas a hash function having both properties 2 and 3 is called a collision-resistant hash function [40]. The most popular cryptographic hash function used in blockchains is SHA-2 [41] (especially the variant SHA256 - a variant that produces outputs of 256 bits) which is so far collision-resistant.

In the blockchain, the most common way to use hash functions is in the form of a mode of operation which includes several invocations of the same or different hash functions. For example, in Proof of Work (PoW) of Bitcoin, SHA256 is used twice, and that construction is called SHA256d, i.e.,

$$\mathrm{SHA256d}(message) = \mathrm{SHA256}(\mathrm{SHA256}(message)). \tag{2.1}$$

2.1.2 Digital Signature

A digital signature is based on public-key cryptography to produce shortcodes called a signature using the private key of a user, and the verification of the signature is done using the public key of the user.

Digital signatures are primarily used to verify the authenticity of blockchain transactions. By providing a signature over a transaction, a user proves that he is authorized to spend the fund of the transaction while preventing other users from spending those funds. Hence, a signature provides source authentication, transaction integrity, and non-repudiation of the sender. A signature scheme consists of the following Probabilistic Polynomial-Time (PPT) algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: A key generation algorithm takes the security parameter λ and generates a private key and corresponding public key pair (pk, sk).
- $\mathsf{Sign}(m, sk)$: A signing algorithm takes a message m, the secret key sk and produces a signature σ as output.
- $\mathsf{Verify}(m, \sigma, pk)$: Given a message m, signature σ and a public key pk, a verification algorithm outputs true or false.

For every (pk, sk) pair generated by $\mathsf{KeyGen}(1^\lambda)$, for all $m \in \{0,1\}^*$, and for some negligible function negl, following holds:

$$\Pr[\mathsf{Verify}(m, \mathsf{Sign}(m, sk), pk) = \mathsf{true}] = 1 - \mathsf{negl}(\lambda)$$

The security of a digital signature is given as Existential Unforgeability under Adaptive Chosen Message Attack (EUF-CMA), which states that an adversary having access to a signing oracle cannot forge a valid signature on a new message of his choice.

Signature schemes have gained a lot of attention due to the rapid development of blockchain technology. The most common signature schemes in blockchain are Elliptic Curve Digital Signature Algorithm (ECDSA) [42] and Edwards-curve Digital Signature Algorithm (EdDSA) [43]. The security of both signatures lies under the hardness assumption of the elliptic curve version of the discrete logarithm problem. ECDSA is built on general elliptic curve cryptography and is used in many blockchains, including Bitcoin and Ethereum [44]. However, EdDSA works on a variant of Schnorr signature [45] based on twisted Edwards curves and used in blockchains such as Monero [46].

2.1.3 Merkle Tree

Merkle tree is used as an accumulator in the blockchain. An accumulator commits to a collection of values as a succinct digest. Merkle tree commits to a set of transactions within a block and results in a single element that can be used to prove the membership of transactions within the block. Merkle tree uses constant storage in a block of the blockchain. Merkle tree is the most commonly used accumulator in cryptocurrencies (e.g., Bitcoin) due to its space and time-efficient data structure for set membership.

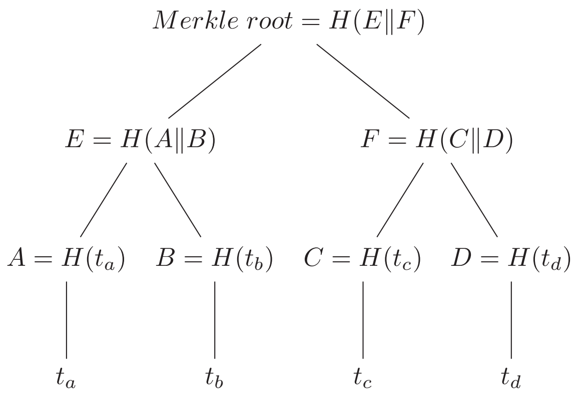

Figure 2.2: Merkle tree of transactions.

Merkle trees are also known as Binary hash trees, in which each leaf node is labeled with the cryptographic hash of a data block (e.g., transaction), and each non-leaf node is labeled with the cryptographic hash of its child nodes' labels as depicted in Figure 2.2. Bitcoin and Ethereum both use this technique (although in different forms of implementation of Merkle trees) so that the blocks only contain the Merkle tree root of the transactions/states.

2.1.4 Ledger

The ledger is a common term used when referring to a blockchain. In general, ledgers are used in accounting, where ledgers record all the incoming and outgoing transactions, and once added, these transactions can not be removed. In a similar way, as blockchain is a read and append-only data store, it is not possible to remove an entry from it under the security assumption. There are two main ledger models for the blockchains.

1. *UTXO Model* In an Unspent Transaction Output (UTXO) model, transactions are transfers of value from the previous transaction outputs to new unspent outputs inductively (e.g., in Bitcoin). New unspent transaction outputs are called UTXO, which are used as inputs to new transactions spending them.

2. *Account-based Model* In an account-based model, each public address is considered an account (e.g., in Ethereum). Each account has a balance associated with it. A transaction in an account-based model transfers value from one account to another.

2.1.5 Blockchain Type

Blockchains can be classified depending on the implementation design, administration rules, data availability, and access privileges. The type differs based on the academic and administrative points of view. From an academic view, blockchains have been classified as "public" and "private". However, in an administrative view, blockchains are called "permissioned" and "permissionless". Nevertheless, these terms are used interchangeably in most blockchain studies and applications. Even though the classification of blockchains is not very distinctly specified in the literature, we can still categorize blockchains by coupling public, private, permissioned, and permissionless.

1. *Permissionless Public*: In a permissionless public blockchain, anyone can join/leave the blockchain network and can participate in consensus at any time. Everyone has read and write access to the blockchain. Consequently, it equips minimum trust among the participants, but it still accomplishes maximum transparency. Most of the cryptocurrencies and blockchain platforms are permissionless public, e.g., Bitcoin [1], Zerocash [47], and Monero [46].
2. *Permissioned Public*: In a permissioned public blockchain, everyone can read the blockchain state at any time, but to participate in consensus and to write the data in the blockchain, there are certain permissions/privileges associated with the participants provided by the network administrator. Having permissions makes the system not entirely decentralized in a particular perspective. In this type of blockchain, once a participant has some privileges, based on that, it can become a validator as well. Examples of permissioned public blockchains are Ripple [48] and EOS [49].
3. *Permissionless Private*: These types of blockchains allow different organizations to collaborate without sharing their information publicly. Due to the permissionless property, anyone can join or leave the blockchain network at any time, which other nodes acknowledge. In a smart contract-based permissionless private blockchain, the read and write permissions for smart contract data can also be defined in the smart contract. Some permissionless private blockchains use the Federated Byzantine Agreement (FBA) as a consensus protocol. LTO [50] is an example of a permissionless private blockchain that creates a "live contract" on the network.
4. *Permissioned Private*: These blockchains are predominantly used in organizations where data/information is stored in the blockchain with permissioned access control by members of the organization. The network administrator or some membership authority provides the membership in the network. The network administrator also provides read and write access to the data. Hyperledger fabric [51], Multichain [52] are examples of permissioned private blockchains.

2.2 Consensus Mechanism

The area of classical consensus is well-established and spans decades [53, 54]. The distributed system community has extensively studied and explored consensus, but the field is revitalized due to being a core part of the blockchain.

In the blockchain, participating nodes collectively adhere to a predetermined set of rules to reach an agreement to ensure the distributed ledger's consistency. This set of rules is determined by a mechanism called consensus. It is the critical component of the blockchain. The first consensus mechanism used in the blockchain is PoW, implicitly given in Bitcoin. As consensus is a core component of blockchain, it is a high-volume, high-churn area of research. Since the introduction of PoW (from Bitcoin), a surfeit of consensus mechanisms has been constructed [55].

Depending upon the network architecture and the type of blockchain, a set of participants, participate in the consensus protocol. The consensus protocol determines this set; it can be a set of all the participants or a set of selected participants through some selection criteria. As a result of the consensus, a leader is elected from the set of participants responsible for creating the new block and adding it to the ledger in a particular execution of the consensus protocol. The leader is elected based on a leader election mechanism, such as using a PoW puzzle competition or executing a Verifiable Random Function (VRF) [28]. The agreement on the new block created by the leader is reached through a voting process. This process can be explicit (by multiple voting rounds) or implicit (by checking the correctness of the leader election process).

2.2.1 Consensus Properties

- *Safety* This property ensures that once an honest participant accepts a new block in its blockchain, the probability of the block being reverted becomes negligible as the chain continues to grow.
- *Liveness* It ensures that the blockchain will continue to progress with the addition of new blocks by honest participants.
- *Fairness* It ensures that each participant gets a fair and equal chance to keep the system alive by contributing towards appending new blocks and obtaining rewards for it.

Safety and Liveness are the two main properties of a consensus protocol. Fairness is not usually justified in most consensus protocols; however, recent works emphasize the fairness of their consensus protocol [21, 56].

On the other hand, fairness is a crucial property needed in consensus. By the fairness property, every participant should have an equal chance of being selected as a leader of the consensus irrespective of available resources or stake to the participant. The reward mechanism of a consensus should also be fair to the participant, which means the effort made by the participants in the consensus should not go to waste. The effort should be rewarded to enforce the participants' positive behavior and make the system work.

Apart from consensus, fairness is a vital property in the blockchain ecosystem. It should also be assessed on the users with differing computational resources in a blockchain protocol. The randomness generation in the blockchain, e.g., randomness for leader election in consensus, should also be a fair process.

2.2.2 Proof of Work

Bitcoin employs Proof of Work (PoW) as its consensus mechanism. The idea behind PoW came from *Hashcash Protocol* [6]. Hashcash proposes a spam-prevention mechanism for public databases. In the continuation, Dwork and Noar [57] first presented an academic treatment of PoW. The main idea of PoW is that someone can verify that a third party has spent computational effort to solve a hash-based computation-intensive puzzle. After the introduction of PoW, the concept was later adapted for securing digital money through the idea of a "reusable proof of work" using the SHA-256 hashing algorithm.

The Hashcash Protocol presents the general idea of PoW; therefore, we represent the Hashcash Protocol as follows. Suppose an email client wants to send an email to an email server. In the beginning, the client and the server both agree on a cryptographic hash function H, which maps an input string to an n-bit output string. Then, the email server sends a challenge string c to the client. Now the client has to find a string x such that $H(c||x)$ starts with k zeros. Since H has pseudorandom outputs, the probability of success in a single trial is

$$\frac{2^{n-k}}{2^n} = \frac{1}{2^k}$$

Here x corresponding to c is considered as PoW, and the process of finding the solution x is difficult. Thus, PoW is difficult to generate but easy to verify.

The process of creating a new block of transactions through solving the PoW puzzle is called mining. The participant who solves the puzzle first is called a *miner*. The mining process involves collecting all the necessary information needed for the PoW puzzle, such as a set of valid transactions, Merkle root of the set, current difficulty, version number (Ver), and previous block hash ($HashPrevBlock$).

If we look at the Bitcoin PoW puzzle, we can see that a miner has to find a $Nonce$ (similar to the Hashcash protocol) to create the next block in the blockchain. The puzzle is as follows:

$$\text{SHA256d}(Ver||HashPrevBlock||\ldots||Nonce) \leq T \tag{2.2}$$

where T is the 256-bit target value.

In a decentralized system with several participants, each participant puts their computational effort into solving the PoW puzzle. Although a single participant is selected as a leader who solves the puzzle first, gets rewarded, and extends the blockchain, as a sum, a substantial computational effort is wasted for each block addition through the effort made by all the participants. Computational PoW efforts by the participants in the blockchain resulted in trivial and massive parallelization approaches, first using Graphic Processing Units (GPUs) and later Application-Specific Integrated Circuit (ASIC) devices, bringing, as a consequence, a huge waste of energy and computational power.

2.2.3 Energy Consumption

PoW is an integral driving mechanism for many modern cryptocurrencies and blockchains, including major blockchains like Bitcoin and Ethereum. Despite their success in the financial market, their substantial energy consumption raises many questions about their use. The issue of enormous energy waste poses a significant hurdle for the adoption of public blockchain use cases.

The energy debate has been going on since Bitcoin came into the limelight, and following Bitcoin, the cryptocurrency market started growing with many modern cryptocurrencies. In 2014, O'Dwyer and Malone conducted a study about the energy consumption of Bitcoin. The conclusion of the study was that "the energy used by Bitcoin mining is comparable to Irish national energy consumption" [58]. Since then, many studies have been conducted, and many research articles have been published about the energy consumption of cryptocurrency mining [59, 60, 61].

The energy debate about PoW-based blockchains again became mainstream in 2021 when China cracked down on almost the majority of cryptocurrency mining within a short amount of time. Consequently, Tesla publicly reversed its decision about accepting Bitcoin as a payment. According to The Cambridge Bitcoin Electricity Consumption Index (CBECI), Bitcoin energy consumption is estimated at 150 terawatt-hours of electricity per year [1]. Additionally, miners in public blockchain networks also frequently upgrade their mining chips to stay economically competitive, which results in a significant amount of electronic waste.

On the other hand, the supportive community of cryptocurrencies, including Bitcoin developers, claims that energy waste is analogous to the energy expenditure of financial institutions such as banks. The Bitcoin developers argue that energy consumption is hard to calculate for financial systems such as banks, and it is arguably much more complex and inefficient than PoW-based cryptocurrencies. Moreover, critics also raise a question about the negative impact of blockchain on climate change due to colossal energy waste. On the contrary, the Data-driven EnviroLab (DDL) and Open Earth Foundation (OEF) claim that blockchain has the potential to improve climate action accounting [62].

Potential Solutions There are a few solutions to the energy challenges of cryptocurrencies:

- Shifting towards renewable electricity sources can be beneficial not only for PoW-based cryptocurrency mining but can also push the sector towards more sustainable energy sources.

- Using more energy-efficient consensus mechanisms such as Proof of Stake can significantly reduce the energy consumption of public blockchains.

[1] https://ccaf.io/cbeci/index

2.2.4 ASIC Resistance

ASICs are specialized hardware devices designed to serve a specific use case, such as performing a specific computing task fast and efficiently. In the case of Bitcoin mining, ASIC implementation mines the blocks several orders of magnitude faster than ordinary CPUs and GPUs by efficiently running SHA-256. Since mining involves multiple attempts to solve a complex cryptographic puzzle, an ASIC device to speed up the mining performs as many attempts as possible (i.e., as many hashing functions per second). Companies such as Bitmain (bitmain.com) sell these ASIC devices ranging from a few hundred dollars to over a thousand.

An ASIC-resistant cryptocurrency configures its mining algorithm so that the use of ASIC devices for mining cannot bring any significant advantage compared to traditional GPU mining. After an increasing trend of designing ASIC devices for SHA256d implementation (especially for Bitcoin mining), starting from 2011, there were few initiatives to design PoW-based cryptocurrencies that include ASIC-resistant hash functions. In this direction of work, first, Litecoin [63] (a fork of Bitcoin) makes use of *Scrypt* [64] - a memory-intensive compilation of use of the HMAC [65]. The idea was that *Scrypt* would be impractical to implement in ASIC, but later ASIC devices for Litecoin mining were offered by Bitmain company. Later, with the idea of ASIC-resistant PoW-based cryptocurrency, QuarkCoin [66] and Darkcoin [67] were also introduced. These cryptocurrencies employ a chain of different hash functions (hashing algorithms) where each hash is calculated and then submitted to the next algorithm in the chain. However, commercial ASIC devices came into the market after a while.

Nevertheless, continuous development is required to make a cryptocurrency ASIC-resistant. It is due to the fact that ASIC manufacturers are constantly producing new ASIC devices that bypass the ASIC resistance of specific cryptocurrencies. The growing production of ASIC devices creates friction between ASIC miners and the cryptocurrency community, which inspires the creation of novel proposals in blockchain consensus. Moreover, blockchain protocols employing other methods of consensus, such as Proof of Stake (PoS) [68], Proof of Authority (PoA) [69], and Programmatic Proof-of-Work (ProgPoW) [70], are ASIC-resistant by design.

2.2.5 Proof of Stake

Proof of Stake (PoS) consensus mechanism was introduced to overcome the problems of PoW. Compared to PoW-based blockchains, where any participant can be a miner who can try to solve a cryptographic puzzle, in PoS-based blockchains, participants have to lock some initial stake in order to participate in consensus. These participants are called validators. The blockchain keeps track of validators who have put aside some stake. The idea behind PoS is to randomly select a validator as a block proposer based on the stake available to the validators. Thus, the probability of being selected as a block proposer depends on the number of stakes available to the participant. PoS is based on the assumption that honest participants own the majority of the stake.

These PoS protocols can be categorized into Chain-based and Byzantine Fault Tolerant (BFT)-based. Chain-based follows the longest chain rule of PoW protocol for the selection of the right chain. BFT-based protocols require voting from the participants and assume that 2/3 of the stakes are held by honest participants. Chain-based PoS protocols simulate the PoW leader election. There are only a few implementations of chain-based PoS protocols which includes Ouroboros [71] and its descendants [72, 73]. Nevertheless, BFT-based protocols have proven mathematical properties. Therefore, there have been many constructions of BFT-based protocols such as Algorand [28], Tendermint [74], and Casper [75].

Another categorization of PoS protocol can be: a) slot-based and b) committee-based. In a slot-based mechanism (e.g., Ouroboros Praos), winning the lottery means being able to create a new block for a slot (consensus round). However, in contrast, winning the lottery in a committee-based mechanism (e.g., Algorand) can encompass different roles, e.g., proposing a new block or voting on a proposed block.

PoS emerged to tackle the energy waste problem of PoW, but it suffers from inherent privacy issues. In PoS, the identity of the selected validator is disclosed in order to verify the proof of selection, and the stake of this validator can be deducted by frequency analysis. Thus, the privacy of validators' identities and their associated stake is essential. Henceforth, to impose privacy in PoS consensus, Private Proof of Stake (PPoS) mechanisms [18, 19] are constructed.

Although the aforementioned PoS protocols solve the huge energy waste problem of PoW, the problem in current PoS protocols is fairness to the participants. A validator with more stakes in the system always has the probability of being richer by being selected as the block proposer and earning the reward. On the contrary, a validator with significantly less stake might not get even a single chance of being selected as a block proposer. Moreover, BFT-based PoS schemes incur high communication costs to reach an agreement. Therefore, one of the fundamental goals of the research presented in the thesis is to design new consensus mechanisms with the following properties:

- Better fairness to the validators;
- Less communication complexity.

2.3 Security

The basic security of blockchain stems from the underlying cryptography and its implementation. Blockchain is constructed with the aim to achieve several inherent security attributes, such as tamper-resistant, consistency, and resistance to attacks, e.g., double-spending attacks. In the case of digital cash, starting from Bitcoin, there were mainly three security guarantees: a) one cannot trivially mint the cash; b) one cannot forge a valid payment; c) one cannot spend the same cash more than once.

Security of blockchain is a broad research area. Much research has been conducted to study the security properties of blockchain [76, 77]. So far, the research studies on the security of blockchain stress on mainly two things: 1) exploring different attack vectors on the blockchain systems; 2) proposing new methods to countermeasure the subsets of attacks. In this section, we first explain the following security attributes (components in general), and further, we briefly discuss the security properties in the blockchain.

1. *Confidentiality* It is a set of rules that limits access to information.
2. *Integrity* It is the assurance that the information is trustworthy and accurate.
3. *Availability* It is a guarantee of reliable access to information by authorized people.

In the blockchain context, the term information used in the above context has multiple meanings; it can be a transaction, a block, smart contract data, or other valuable data. Following, we broadly organize the main security properties of the blockchain.

- *Consistency* In the blockchain context, consistency refers to the property that all the nodes in the system should have the same ledger at the same time. Consistency is a somewhat controversial topic. As many states that blockchain systems such as Bitcoin only have eventual consistency, which is a weak consistency. Eventual consistency means that each node in the blockchain system gets consistent eventually.
- *Tamper-resistance* Tamper-resistance in the blockchain refers to the property that the transaction information stored in a block cannot be tampered with during and after block generation. There are two main ways to tamper with the transactions: 1) The block proposer (e.g., a miner in Bitcoin) can attempt to tamper with the transaction information; 2) An adversary can attempt to tamper with the transaction information in the blockchain. Nevertheless, the employed digital signature on each transaction and the use of the hash function to connect the blocks make it impossible to tamper with the transaction data without the network knowing about it; hence, the above attempts are elegantly prevented.
- *Resistance to double-spending attack* Double-spending attack is specific to blockchain-based cryptocurrency systems. The attack is to spend a coin more than once in simple terms. The double-spending attack is a general security concern in digital payment systems because digital information can be reproduced easily. In cryptocurrency systems, signing transactions using a digital signature and public verification of transactions with a majority consensus can prevent double-spending attacks.

Given the generic security requirements, the following apply to the blockchain:

1. *Confidentiality of transaction data* In the majority of financial institutes, users wish to have minimal disclosure of their transaction details.
2. *Integrity of transactions* The transaction details must be trustworthy and should not be tampered with.
3. *Availability of system and data* The users in the blockchain network should be able to access the blockchain information, i.e., transaction data, at any time.

Apart from the above security requirement and properties, there are many real attacks on blockchain systems. The most common attacks are selfish mining, eclipse, Denial of Service (DoS), and Sybil attacks [76]. DoS attack is one of the most severe attacks in blockchain, especially in cryptocurrency markets. Therefore, following, we present a brief overview of the DoS attack.

2.3.1 Denial of Service Attack

A Denial of Service (DoS) attack targets to disrupt the availability of the network, application, or server and thwarts legitimate requests from taking place. For a DoS attack to follow, the attacker has to send more requests than the victim server can handle. These requests can be legitimate or bogus. The DoS attack exhausts the server's resources, such as CPU, memory, or network. Due to blockchain's various configurations and decentralized features, many attacks are preventable. Regardless, DoS attacks, especially its distributed variant Distributed Denial of Service (DDoS), are still prominent attacks on cryptocurrencies and blockchain-based applications.

Due to the increasing intensity and frequency of DoS attacks, it is pondered as one of the biggest and most severe threats to the Internet industry. One of the strong DoS attacks was mounted on a DNS server in October 2016, which manifested in a cut of access to major websites, including PayPal, Netflix, and Twitter, for several hours [78]. The spectrum of DoS attacks can vary from DNS services, cloud providers, and IoT devices to the cryptocurrency and blockchain markets. Nowadays, the cryptocurrency market is a popular target of DoS attacks, with the motivation of ransom, stealing funds, or business competition. In the past, many works [79, 80, 81] regarding the detection and prevention of DoS attacks have been carried out. Moreover, DoS/DDoS solutions based on blockchain are an emerging area of research.

Definition 2.3.1. (DoS): Let a server $\mathcal{S}$ be given, with the available resources $R_1, R_2, \ldots, R_n$ (R_i can be bandwidth, memory, CPU etc.). Let $\mathcal{A}$ or a set of $\{\mathcal{A}_j\}$ are an attacker or a set of attackers, and let the set $\{\mathcal{U}_k\}$ represents the legitimate users. A DoS attack on server $\mathcal{S}$ is expressed by a set of probabilities for successful resource-depletion $\{P_{R_1}, P_{R_2}, \ldots, P_{R_n}\}$. The total probability for a success of a DoS attack is then a probability the server $\mathcal{S}$ to refuse legitimate transactions from a user u, where $u \in \{\mathcal{U}_k\}$ and is modeled as the probability of blocking the legitimate traffic in at least one of the resources:

$$P_{DoS} = 1 - (1 - P_{R_1})(1 - P_{R_2}) \ldots\ldots (1 - P_{R_n}) \tag{2.3}$$

Note that the situation when attacker(s) exhausts at least one resource R_i implies the attack probability is $P_{R_i} = 1$, which from equation (2.3) further leads to $P_{DoS} = 1$.

DoS attacks can be categorized based on network and application layers or volume and protocol attacks. Network-level DoS attacks aim to overload the server's bandwidth or cause CPU usage issues. However, application-level DoS attacks concentrate on applications, websites, or online services.

The most notable mitigation schemes for DoS are client puzzles. In a client puzzle scheme, a client has to prove legitimate intentions to the server by solving a puzzle. In the majority of the client puzzle schemes, the puzzle is provided by the server, which makes the scheme interactive in nature. A client puzzle scheme can be CPU-bound (such as in Hashcash Protocol), memory-bound, or network-bound. There are many constructions of client puzzle schemes, but most of the existing schemes do not offer asymptotically efficient verification and public verifiability of the puzzle solution.

The research presented in the thesis focuses on the following subjects with respect to DoS mitigation:

– Construction of a non-interactive client puzzle scheme with desirable properties, such as asymptotically efficient verification and public verifiability;

– Investigation of the DoS attack and its mitigation approaches in the entities of the blockchain ecosystem.

2.4 Privacy

Privacy is a huge concern in blockchain systems. Popular systems like Bitcoin and Ethereum do not have privacy out of the box. All the transaction data is recorded in the clear on the blockchain, allowing anyone to infer the detail of the transactions. Although these systems do not reveal real identities, the random addresses can be easily linked to their real owner [82]. This raises an issue about disclosing sensitive data in the transactions, such as health data. Therefore, privacy issues should be addressed in the blockchain.

Lack of privacy also affects fairness as users in blockchain become susceptible to front-running attacks [83]. That means, having all the transaction details public, an adversarial user races to have his transaction confirmed first.

Privacy, in general, can be categorized as privacy of data or privacy of users (anonymity). Data privacy in blockchain refers to the confidentiality of all the data or sensitive data stored in the blocks. Anonymity refers to the privacy of a user's identity. Usually, pseudonymity is employed in cryptocurrencies, where pseudonymity refers to the state of disguised identity.

Data privacy research has been proliferating over the past decade. Several academic and industrial initiatives have been brought in recent years. So far, research shows the risk of privacy leakage due to inference attacks through which sensitive transaction data can be obtained, and the true identity of the users can be inferred from the pseudonym. Hence, privacy is a significant challenge to be addressed in blockchain and blockchain-based applications involving sensitive information. Following, we describe how privacy is/has been achieved in the blockchain.

In the blockchain context, the notion of privacy varies from the privacy of transaction amounts [47, 32, 20] and transacting parties [84, 85] to the privacy of embedded functional calls in a smart contract [86]. Different solutions have been proposed to achieve meaningful privacy notions in the blockchain. Several of these solutions employ privacy-oriented cryptographic techniques such as zero-knowledge proofs [47], ring signature [46], homomorphic encryption [87], and mixing techniques [88, 89] to achieve different forms of privacy. Financial systems zkLedger [90], Solidus [91], and RSCoin [92] also achieve privacy in their transactions, but banks regulate the supply of funds, and a blockchain is used to make transactions. A detailed overview of these systems can be found in the work of Almashaqbeh and Solomon [93].

The introduction of Zcash [47] and Monero [46] intrigued the crypto community to design other privacy-focused cryptocurrency systems. Zcash and Monero are Bitcoin-like blockchains. Both rely on recording every transaction in the history to perform further transactions following the UTXO model. Systems like Hawk [32] and Zexe [86] use a zk-SNARK proof system to generate privacy-preserving transactions.

Most of the new designs of privacy-preserving systems are built on the top of the smart contract. The idea is to build private smart contracts that allow for arbitrary computations on the blockchain while keeping the inputs and outputs secret. Zether [20], zKay [94] and Kachina [95] are such systems built on Ethereum. These systems establish privacy-preserving smart contracts on Ethereum to provide confidential payment or confidential data. zKay [94] defines its privacy model by defining a language zKay for writing smart contracts with private data. Moreover, Kachina [95] provides a unified security model based on the Universal Composition (UC) model for deploying privacy-preserving and general-purpose smart contracts. The interest in building private smart contracts is further extended to support function privacy, which means even the computation itself is hidden.

The majority of privacy-preserving systems offer some meaningful privacy. Still, many of these systems, such as Hawk [32] and Monero [46], do not have a security model to assess the security properties and privacy of the systems. A few of these systems, such as Zcash [47] and Zether [20], do have their security model, but their model cannot be utilized to check the security of other privacy-preserving systems. Therefore, in order to assess the security of most of the privacy-preserving systems, the present thesis provides:

- A general security framework to assess the privacy of existing privacy-preserving systems;

- Security definitions for privacy-preserving systems.

2.5 Scalability

Due to the continuously growing size of the blockchain, scalability is becoming a challenging issue. The scalability of a blockchain depends on many factors, such as the underlying consensus mechanism and network structure. Scalability is one of the critical issues for the widespread adoption of blockchain technology for different use cases. Moreover, improving the scalability (transaction throughput) of a blockchain can also improve the energy efficiency per transaction. Current blockchains, including Bitcoin and Ethereum, are not suitable for daily financial transactions due to the limited throughput.

The main challenge when trying to improve the scalability of a blockchain is simultaneously preserving the other properties of the blockchain, especially security and decentralization. Many potential solutions have been proposed to improve the scalability of public blockchains pertaining to high decentralization and security. These solutions include Layer-2 solutions [23], sharding mechanism [96], and advanced Layer-1 solutions.

Layer-2 solutions move the large amount of data and computation off-chain and record a summary of the transactions on-chain. Constructions such as Plasma [97], NOCUST [98], and ZK-Rollup [99] follow a similar model, and an off-chain untrusted operator puts an abbreviated update for the transactions on-chain. Plasma/NOCUST claim to decrease the transaction cost nearly to zero. However, these systems send fewer data to the blockchain, but they suffer from the problem of mass exit and long waiting time in case of withdraw.

Sharding, in general, splits a big data set into multiple small data sets. To improve the scalability of blockchain by sharding, instead of having a large monolithic blockchain, the blockchain network can have multiple small interconnected blockchains. The hypothesis of sharding in the blockchain is to split the processing of transactions among the smaller group of nodes called shards. These shards work in parallel and improve the throughput and performance while incurring less computation, communication, and storage cost. The main feature of sharding is scaling the blockchain without making an individual blockchain significantly larger. The sharding approach also functions with Layer-2 solutions to achieve even greater scalability. Some examples of blockchains utilizing the sharding approach for scalability are OmniLedger [100], Chainspace [101], and RapidChain [102].

A few Layer-1 blockchains with smart contract functionality scale their transaction throughput by employing Proof of stake and a mix of other techniques. Blockchains such as Algorand [28], Solana [103], and Tezos [104] fall under this category. For instance, Solana applies Proof of history with parallelization to scale the throughput. In contrast, these approaches may bring the risk of less decentralization.

Layer-2 solutions are the most popular scalable solutions at present. However, these scalable solutions do not provide privacy of the off-chain data, which is crucial concerning the privacy of the user's balance and transaction amount. Therefore, there is a need for a scalable solution with the privacy of off-chain data, which is addressed in the present thesis.

2.6 Cryptographic Primitives Used in the Thesis

This section demonstrates a few important cryptographic concepts used in the thesis. The section covers only those cryptographic primitives which have been used in novel constructions presented in the contributed works in the thesis. Following, we present a brief overview of these primitives; however, detailed information about these primitives can be found in the papers attached to the thesis.

2.6.1 Multi-Signature

A multi-signature scheme allows a group of n signers to jointly produce a single short signature σ on a message m. This short signature σ convinces a verifier that all n signers signed the message m. Given the message m and the set of public keys of all the n signers, the verification of σ can be publicly performed. For practical purposes, the size of the produced signature in a multi-signature scheme should be close to the regular signature size and independent of the number of signers in the scheme. While dealing with multi-signature, the rogue-key attack must be taken into account. In a rogue-key attack, an adversary first crafts public keys correlated with the public keys of honest signing parties. Using these crafted public keys, the adversary forges a multi-signature on a message of his choice. Security against a rogue-key attack is achieved by requiring proof of possession (knowledge) of the corresponding secret key from each signer of the scheme.

Multi-signatures are helpful in reducing the size of the blockchain. Maxwell et al. constructed a Schnorr-based multi-signature scheme [105] called MuSig, which was built on the work of Bellare and Neven [106], and provably secure in the plain public-key model (where users do not need to prove the knowledge of their secret key). Boneh et al. constructed a new multi-signature scheme [107] which is useful not only for reducing Bitcoin blockchain size but also for use in other settings where multi-signatures are needed. Their scheme is pairing-based and derived from BLS signature scheme [108]. It supports public key aggregation and security in a plain public-key model. BLS multi-signature scheme requires bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ along with a full-domain hash function for signing process $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. BLS multi-signature scheme works as follows:

- $\mathsf{KeyGen}(1^\lambda)$: For a user, given a security parameter λ, choose random $sk \overset{\$}{\leftarrow} \mathbb{Z}_q$, compute $pk \leftarrow g_2^{sk} \in \mathbb{G}_2$, the user's keypair is (pk, sk).
- $\mathsf{Sign}(sk, M)$: For a user, given the secret key sk and a message $M \in \{0,1\}^*$, signature on M is $\sigma \leftarrow H(M)^{sk} \in \mathbb{G}_1$.
- $\mathsf{Verify}(pk, M, \sigma)$: Given a user's public key pk, a message M, accept the signature σ, if $e(\sigma, g_2) = e(H(M), pk)$.

- $\mathsf{SigAgg}(\sigma_1, \sigma_1, \ldots, \sigma_n, M)$: Given n signatures $\sigma_1, \sigma_1, \ldots, \sigma_n$ on message M by n users, the procedure for signature aggregation of n signatures works as: $\sigma_{agg} \leftarrow \prod_{i=1}^{n} \sigma_i$. The aggregate signature is $\sigma_{agg} \in \mathbb{G}_1$.
- $\mathsf{AggVerify}(\sigma_{agg}, M, (pk_1, pk_2, \ldots, pk_n))$: To verify the aggregate signature σ_{agg}, given the original message M and the n public keys $pk_1, pk_2, \ldots, pk_n$ for all n users, the verifier checks if:
$$\begin{aligned} e(\sigma_{agg}, g_2) &\stackrel{?}{=} e(H(M), pk_1)e(H(M), pk_2)\ldots e(H(M), pk_n) \\ &\stackrel{?}{=} e(H(M), \prod_{i=1}^{n} pk_i) \stackrel{?}{=} e(H(M), apk) \end{aligned}$$
If the equation holds, the verifier "Accept" the signature, else "Reject". In the above equation, $apk \in \mathbb{G}_2$ and stands for *aggregate public key.*

2.6.2 Zero-Knowledge Proofs

Zero-knowledge proofs are a growing research area in cryptography. The use of zero-knowledge proof has been found in many areas of cryptography, including blockchain. In zero-knowledge proofs [109], two parties, a prover and a verifier, participate. First, the prover asserts some statement and proves its validity to the verifier without revealing any other (secret/witness) information except the statement. Thus, a zero-knowledge proof proves the statement as 'transfer of an asset is valid' without revealing anything about the asset. Zero-knowledge protocols (protocols employing zero-knowledge proofs) are valuable cryptographic protocols for achieving secrecy in applications. Many variants of zero-knowledge proofs have been introduced with differences in interactive, non-interactive proofs, size of the proof, transparent and trusted setup, Common Reference String (CRS) model, etc.

A Non-Interactive Zero-Knowledge (NIZK) argument for a NP relation R includes three polynomial time algorithms ($\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}$) defined as follows:

- $\mathsf{Setup}(1^\lambda)$: takes as input the security parameter λ, outputs the the common reference σ.
- $\mathsf{Prove}(\sigma, x, w)$: takes as input the common reference σ, a statement x, a witness w for the statement, and outputs an argument π.
- $\mathsf{Verify}(\sigma, \pi, x)$: takes as input the common reference σ, a statement x and an argument π for the statement, outputs 1 if verifier accepts the argument, otherwise outputs 0 rejecting the argument.

One of the variants, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK), reduces the complexity and the proof size, which is an intriguing topic to explore. Many blockchains, including Zerocash [47] and Quorum [110], apply the zk-SNARK concept for transaction privacy, anonymity, and unlinkability. Adequate theoretical background on zk-SNARK is presented in [8, 9], and a practical point of view is explored here [111].

2.6.3 Verifiable Random Function

A Verifiable Random Function (VRF) is a pseudorandom function that produces output along with proof of correctness of the output. The output of VRF cannot be generated by two different inputs that ensure the collision resistance property of VRF. A VRF has three algorithms as follows:

- $\mathsf{KeyGen}(r)$: On input value r, the key generation algorithm generates a secret key sk and a verification key vk.
- $\mathsf{Eval}(sk, M)$: The evaluation algorithm produces pseudorandom output O and the corresponding proof π on input sk and a message M.
- $\mathsf{Verify}(vk, M, O, \pi)$: The verification algorithm outputs 1 if and only if the output produced by the evaluation algorithm is O and it is verified by the proof π given the verification key vk and the message M.

VRFs became quite popular due to their properties (Uniqueness, Collision resistance, Pseudorandomness, and Unpredictability) and their usefulness in blockchains. VRF has many use cases, e.g., blind auctions, DNS denial of existence, and cryptographic sortition [68]. Many blockchain platforms, such as Algorand [68], Ouroboros [112], and Coda [113], started using cryptographic sortition in their consensus mechanism, which brought VRF to the limelight. VRF has also been written in Solidity to be used in the Ethereum blockchain. VRF is used in leader election and consensus algorithms of the blockchain.

2.6.4 Verifiable Delay Function

A Verifiable Delay Function (VDF) is a recent cryptographic primitive, and the research on this particular topic is getting intense interest from academia and industry. Many collaborative efforts have been made to design and implement production-grade VDFs in software and hardware. A VDF is a paramount tool to add a delay in decentralized applications. It is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ defined formally by Boneh et al. [11] that takes a prescribed minimum number of steps to compute and is exponentially easy to verify. That means the VDF function can not be parallelized. The output of the VDF function is unique and sound; hence an adversary has a negligible chance of randomly guessing the correct output. A VDF is defined as a tuple of the following algorithms:

- $\mathsf{Setup}(1^\lambda, T)$: It is a randomized algorithm that takes security parameter λ, time parameter T and outputs public parameter pp.
- $\mathsf{Eval}(\mathsf{pp}, x, T)$: The evaluation algorithm takes public parameter pp, input value $x \in \mathcal{X}$ and time parameter T, returns an output value $y \in \mathcal{Y}$ together with a proof π. The algorithm may use random coins to generate the proof π but not for the computation of output y.
- $\mathsf{Verify}(\mathsf{pp}, x, y, \pi, T)$: The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

After the introduction of VDF by Boneh et al., two efficient constructions were presented; Wesolowski [13] and Pietrzak [12] schemes that use a group of unknown order to construct their VDF. Both schemes differ in verification time and proof size. Henceforth, based on the desired property, these VDF schemes can be applied in promising applications such as random beacon, proof of replication, resource-efficient blockchains, and computational time-stamping in proof of stake consensus systems. A client puzzle scheme can also utilize VDF to mitigate the Denial of Service attacks.

2.6.5 Commitment Scheme

A commitment scheme provides an excellent way to hide a value while maintaining its proof of commitment. It can be defined as a digital analog of a sealed envelope. The commitment scheme was introduced in 1988 by Gilles Brassard, David Chaum, and Claude Crepeau [114]. A commitment scheme is a two-phase process between two parties, sender S and receiver R. The two phases are called Commit and Reveal.

- Commit: In the commit phase, the sender S commits to a value x by creating a commitment $\mathsf{Com}(x, r)$ for a uniformly random value r, and sends it to the receiver R.
- Reveal: In the reveal phase, the sender S opens the committed value $\mathsf{Com}(x, r)$ by sharing r and x with the receiver R, who checks that the sender did not cheat.

A commitment scheme is an important building block for other cryptographic primitives such as zero-knowledge proofs and verifiable secret sharing. It is also used as a building block in different blockchain applications. The commonly used commitment scheme in the blockchain is Pedersen commitment [115]. Pedersen commitment provides computational binding and unconditional hiding properties based on the discrete logarithm problem. Pedersen commitment is used in different contexts of blockchains, e.g., Zerocoin [116] to bind a serial number s to a Zerocoin z and to construct RingCT 2.0 [117]. Recent findings depict an enormous interest in the field of another kind of commitment scheme, which is subvector commitments. Moreover, Aggregatable Subvector Commitment (aSVC) are gaining much attention for more exploration, especially in blockchain use cases [118, 119].

Chapter 3

Contributions

3.1 Research Contributions

Paper	Title. Author List. Conference/Journal
A	**SoK of Used Cryptography in Blockchain** [39] M. Raikwar, D. Gligoroski, and K. Kralevska IEEE Access, vol. 7, pp. 148550-148575, 2019
B	**Meshwork Ledger, its Consensus and Reward Mechanisms** [120] M. Raikwar, D. Gligoroski 13th International Conference on COMmunication Systems and NETworkS (COMSNETS), IEEE, 2021, pp. 290-298
C	**R3V: Robust Round Robin VDF-based Consensus** [121] M. Raikwar, D. Gligoroski 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), IEEE, 2021, pp. 81-88
D	**Non-Interactive VDF Client Puzzle for DoS Mitigation** [122] M. Raikwar, D. Gligoroski European Interdisciplinary Cybersecurity Conference (EICC), ACM, 2021, pp. 32-38
E	**DoS Attacks on Blockchain Ecosystem** [123] M. Raikwar, D. Gligoroski FPDAPP@Euro-Par, 4th International Workshop on Future Perspective of Decentralized Applications, LNCS, 2021
F	**PriBank: Confidential Blockchain Scaling Using Short Commit-and-Proof NIZK Argument** [124] K. Gjøsteen, M. Raikwar, S. Wu In Cryptographers' Track at the RSA Conference (CT-RSA), Springer, Cham, 2022, pp. 589-619
G	**Security Model for Privacy-preserving Blockchain-based Cryptocurrency Systems** [125] K. Gjøsteen, M. Raikwar, S. Wu Submitted to 13th Conference on Security and Cryptography for Networks (SCN), 2022
H	**SoK: Decentralized Randomness Beacon Protocols** [126] M. Raikwar, D. Gligoroski Accepted in 27th Australasian Conference on Information Security and Privacy (ACISP), LNCS, 2022
I	**Competitive Decentralized Randomness Beacon Protocols** [127] M. Raikwar BSCI@ASIACCS, 4th ACM International Symposium on Blockchain and Secure Critical Infrastructure, ACM, 2022, pp. 83-94

Table 3.1: List of publications included in the thesis.

The author of the presented thesis contributed to 14 scientific publications. Table 3.1 outlines the list of 9 publications (Papers A-I) as the primary contributions to the thesis. Table 3.2 presents a list of 5 publications (Papers J-M) as secondary contributions which are not included in the thesis. The order in which the publications appear in the Table 3.1 is not necessarily chronological but rather in relation to the research questions described in Section 1.2. The presented order provides a natural flow in the exposition of the thesis.

Note: Paper F and G follow the cryptography convention of authors' ordering, where the authors' names appear in alphabetical order of their last name. Besides, Paper I receives the *Best Student Paper Award (Runner-up)*.

Paper	Title. Author List. Conference/Journal
J	**Trends in Development of Databases and Blockchain [128]** M. Raikwar, D. Gligoroski, and G. Velinov Seventh International Conference on Software Defined Systems (SDS), IEEE, 2020, pp. 177-182
K	**Aggregation in Blockchain Ecosystem [129]** M. Raikwar, D. Gligoroski Eighth International Conference on Software Defined Systems (SDS), IEEE, 2021, pp. 1-6
L	**Efficient Novel Privacy Preserving PoS Protocol Proof-of-concept with Algorand [130]** K. Stevenson, O. Skoglund, M. Raikwar, and D. Gligoroski 3rd Blockchain and Internet of Things Conference (BIOTC 2021), ACM, 2021, pp. 44-52
M	**Databases fit for blockchain technology: A complete overview** J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, D. Gligoroski Submitted to Elsevier Journal on Blockchain: Research and Applications, 2022 (in Journal Revision)
N	**Cryptographic Primitives in Blockchain** M. Raikwar, S. Wu Book chapter In : S. Kanhere, M. Conti, and S. Ruj, (eds) "Blockchains - A Handbook on Fundamentals, Platforms, and Applications", Springer, 2022 (to appear)

Table 3.2: List of publications not included in the thesis.

3.2 Summary of Results Contributing to the Thesis

This section presents a summary of the papers included in the thesis. Contributions of each paper are discussed with respective state-of-the-art results and presented in accordance to the research questions.

Paper A: SoK of Used Cryptography in Blockchain
M. Raikwar, D. Gligoroski, and K. Kralevska
IEEE Access, vol. 7, pp. 148550-148575, 2019

This paper is the first step for Systematization of Knowledge (SoK) that gives a complete picture of the existing cryptographic concepts that have been deployed or have the potential to be deployed in the blockchain. In this paper, we thoroughly review and systematize all the cryptographic concepts used in blockchain. As the underlying fundaments of blockchain are cryptographic concepts, it opens a broad spectrum to explore their applicability in the blockchain. To strategically review and investigate the cryptographic concepts, we designed inclusion and exclusion criteria. Keeping that in mind, we reviewed more than 25 cryptographic concepts, some of these concepts have already been used in blockchain, and some of these can be employed in blockchain to provide reliable and secure decentralized solutions. We also include possible instantiations of these cryptographic concepts in the blockchain domain. Last but not least, we explicitly postulate 21 challenging research problems that cryptographers interested in blockchain can work on.

The research problems proposed in the paper got attention from the cryptographers and some of the research problems have been taken into account and further being solved. An example of such an incident is related to the research problem about Private Information Retrieval (PIR) of transaction information without revealing the transaction itself which is proposed in our SoK paper. The research problem have been approached and solved by Sasidharan and Viterbo [131] by employing coded sharding with Reed-Solomon codes [132].

This SoK paper also discusses the main challenges in the blockchain. From these challenges, we chose a few main challenges such as security, privacy, and scalability to work on. Apart from the challenges, we also selected and studied a few cryptographic primitives such as Verifiable Delay Function (VDF), Zero-knowledge Proof, and Aggregate Signature. These primitives are further employed to address some of the challenges in blockchain and presented results are published and included in the thesis.

Paper B: Meshwork Ledger, its Consensus and Reward Mechanisms
M. Raikwar, D. Gligoroski
13th International Conference on COMmunication Systems and NETworkS (COMSNETS), IEEE, 2021, pp. 290-298

This paper proposes a new permissioned public blockchain ledger concept called "Meshwork Ledger" with its consensus and reward mechanisms. Meshwork has two primary entities validator and client nodes. Validator nodes are the major players for reaching the consensus in the ledger. Meshwork is a network of coequal client nodes that contribute to the endorsement of the transactions by providing digital signatures to a validator node that collects them in an aggregate signature scheme. Therefore, the main component of the consensus algorithm is an aggregate multi-signature scheme.

The core idea of the consensus is to race for the maximum number of signatures (approvals) on a block from the mesh clients, in order to append the block to the blockchain. The essential sustainability component of the ledger is the reward mechanism for the Meshwork client nodes. The prime design objective is the coequality of all client nodes, meaning there is no advantage for getting rewards if the client is an early adopter, if the client has collected a significant stake of rewards or if the client just joined the Meshwork. We adapt the commit-chain [26] as a suitable off-chain payment solution for rewarding the Meshwork client nodes in a cost-efficient manner.

We provide the security analysis of the Meshwork ledger and also present the feasibility of the signature scheme by implementing it. We also discuss the consensus properties of the Meshwork ledger which are essential parts of consensus to be explored and analyzed. Compared with other blockchain consensus algorithms, the Meshwork consensus algorithm is faster, significantly more energy-efficient and scalable. The scalability is achieved with the off-chain reward mechanism and the grouping of client nodes with the validator nodes. Nevertheless, a detailed analysis is needed for the scalability. In conclusion, Meshwork ledger presents a simple and scalable approach to achieve faster agreement while providing fairness to the participating nodes.

Paper C: R3V: Robust Round Robin VDF-based Consensus
M. Raikwar, D. Gligoroski
3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), IEEE, 2021, pp. 81-88

This paper proposes a novel consensus protocol "R3V". The proposed consensus also focuses on fairness, similar to Paper B. However, the idea of the protocol is quite different but relatively simple. Each round of consensus consists of two steps: In the first step, a set of eligible candidates are chosen based on a robust round-robin method according to their age; in the second step, these eligible candidates compete with each other by solving a VDF-based puzzle. The one who solves the puzzle first becomes the leader and proposes a new block. We offer different methods to generate verifiable identities for the stakeholders. The identities are enrolled in the blockchain, which provides the age norm needed for the consensus. Compared with the other PoS protocols, our protocol shows better resilience against the most common attacks on PoS protocols.

Although Proof of Stake (PoS)-based consensus provides a better mechanism than Proof of Work (PoW) consensus for extending the blockchain without significant energy waste, most PoS protocols do not offer better fairness for the stakeholders participating in the consensus. Due to the use of randomness to elect a leader candidate in PoS consensus, the consensus becomes weaker and suffers from attacks, e.g., long-range attacks and block withholding attacks. Moreover, these protocols suffer from high communication complexity for selecting a leader candidate in each consensus round. Our R3V consensus solves all these mentioned problems in PoS consensus to some extent and provides low energy consumption, less communication complexity, and better fairness. Nevertheless, due to the round-robin selection of candidates, the consensus lacks adequate resistance against DoS attack.

Paper D: Non-Interactive VDF Client Puzzle for DoS Mitigation
M. Raikwar, D. Gligoroski
European Interdisciplinary Cybersecurity Conference (EICC), ACM, 2021, pp. 32-38

The motivation for this paper originates from Paper B and C. Although Paper B and C propose efficient and fair consensus mechanisms, the concern of the DoS attack on the participating nodes still hinders the availability and efficiency of the consensus. Such instances are the DoS attack on the Meshwork validator node of the Meshwork consensus or the DoS attack on eligible candidates in the R3V consensus. Henceforward, to solve the issue of DoS attack in a general context, we explore DoS mitigation methods and propose a novel approach for it in Paper D.

This paper proposes a non-interactive VDF client puzzle scheme. Client puzzles [4] are proposed to mitigate DoS attacks by requiring clients to prove legitimate intentions. Since its introduction, there have been several constructions of client puzzles. Nevertheless, most of the existing client puzzles are interactive, where a server constructs a puzzle for a client request and asks the client to solve it before giving access to a resource.

Additionally, most existing client puzzles do not provide desirable properties such as fairness, non-parallelizability, or non-interactivity. In this work, our proposed non-interactive client puzzle achieves all these desired properties through VDF. In a non-interactive puzzle, the client generates a puzzle and sends its solution along with the puzzle to access a server resource. We present different methods to generate verifiable client puzzles to prevent puzzle forgery and attacks from the client-side. Further, we exhibit a transformation of the client puzzle into a DoS-resistant protocol. We also demonstrate the applicability of the DoS-resistant protocol in different contexts of the blockchain ecosystem.

Paper E: DoS Attacks on Blockchain Ecosystem
M. Raikwar, D. Gligoroski
FPDAPP@Euro-Par, 4th International Workshop on Future Perspective of Decentralized Applications, LNCS, 2021

DoS attack is not only a severe concern in participating nodes in consensus, such as in Paper B and C, but it can also be mounted in other entities in the blockchain ecosystem. The financial potential of the cryptocurrency market attracts more adversarial issues and makes itself a prevalent target of DoS attack. Due to the immense growth of the cryptocurrency market, the frequency and intensity of DoS attacks have been rapidly increasing. Therefore, this paper offers the first thorough systematic investigation of DoS attacks in the blockchain ecosystem.

This paper identifies ten entities in the blockchain ecosystem where DoS attack happens or can happen. We present DoS mitigation approaches where some of the approaches have already been employed. We also devise a new

DoS mitigation scheme based on VDF, which we call a VDF client puzzle. The presented VDF client puzzle scheme is interactive and feasible to be applied for DoS mitigation based on the experimental results. Additionally, we suggest the appropriate DoS mitigate scheme for each of the ten identified entities. In addition to the interactive scheme, the paper mentions the client puzzle scheme constructed in Paper D for the DoS mitigation in the blockchain ecosystem.

Paper F: PriBank: Confidential Blockchain Scaling Using Short Commit-and-Proof NIZK Argument
K. Gjøsteen, M. Raikwar, S. Wu
In Cryptographers' Track at the RSA Conference (CT-RSA), Springer, Cham, 2022, pp. 589-619

This paper offers PriBank, a novel construction of a privacy-preserving cryptocurrency system that implements privacy on top of Layer-2 scaling solutions. A privacy-preserving cryptocurrency system facilitates privacy for the users of the system, which involves the privacy of user balances and transaction values. Layer-2 scaling solutions provide scalability to cryptocurrency systems. These Layer-2 solutions move the expensive computations off-chain and later commit the abbreviated transaction data on-chain. However, these scalable solutions do not have the privacy of the off-chain data, which involves users' balances and transaction data. Our PriBank achieves privacy of the off-chain data on top of scaling solutions. To construct PriBank, we propose an efficient Commit-and-Prove short NIZK argument for quadratic arithmetic programs.

The Commit-and-Prove short NIZK argument is built on top of the existing zero-knowledge proof scheme: Bulletproofs [133]. It allows a prover to commit to an arbitrary set of witnesses by Pedersen commitments [115] before proving, which may be of independent interest. We also compare PriBank with the existing privacy and scalability solutions. We define transaction indistinguishability and overdraft-safety properties to assess the security of the PriBank. Further, we present a detailed security analysis for PriBank preserving these properties. We also conducted experiments to check the performance of the PriBank.

Paper G: Security Model for Privacy-preserving Blockchain-based Cryptocurrency Systems
K. Gjøsteen, M. Raikwar, S. Wu
Submitted to 13th Conference on Security and Cryptography for Networks (SCN 2022)

This paper employs the PriBank model of Paper F to construct a general model for assessing the security of privacy-preserving cryptocurrency systems. Privacy-preserving blockchain-based cryptocurrency systems such as Zcash [47] and Monero [46] have become quite popular for confidential payments. The confidential payment differs in providing confidentiality to users, transactions, or both. These payment systems also differ in their designs and constituent cryptography. Since the introduction of the most notable privacy-preserving cryptocurrencies, Zcash and Monero, there has been a thriving interest in constructing different privacy-preserving cryptocurrency systems with improved security and additional features. Although there are many new constructions of these systems, many of these systems lack their security models, which makes it hard to prove the security properties of these systems.

In this paper, we present a general model to assess the security of privacy-preserving cryptocurrency systems. We propose a framework for a privacy-preserving blockchain-based bank PBB. We illustrate the security properties of the framework and present the security experiments for each of the properties. Further, using the security properties of the PBB framework, we analyse the security of Zcash and Monero. Our analysis confirms that the PBB framework can be employed to formalize the security of other privacy-preserving cryptocurrency systems.

Paper H: SoK: Decentralized Randomness Beacon Protocols
M. Raikwar, D. Gligoroski
27th Australasian Conference on Information Security and Privacy (ACISP), 2022

The motivation for this paper originates from all the papers described above, as verifiable randomness is needed in most of the cryptographic protocols defined in the above papers. For instance, verifiable trusted randomness is an essential element in client puzzle construction in Paper D and E; it is also needed in the construction of the Commit-and-Prove short NIZK argument in PriBank in Paper F. Therefore, this paper presents a Systematization of Knowledge (SoK) of Decentralized Randomness Beacon (DRB) protocols that intend to structure the multi-faced body of research on DRB protocols. DRB protocols provide a continuous and reliable source of randomness. Due to its need in modern cryptography, such as blockchain and cryptocurrency, there has been a thriving interest in constructing DRB protocols. DRB protocols have many applications, including byzantine fault-tolerant (BFT) protocols, anonymous messaging services, blockchain, and smart contracts. Although there have been several constructions of DRB protocols, there is no systematization of these protocols. Therefore, in this paper, we systematize and scrutinize the existing DRB protocols.

In this SoK, we give a standard definition and requirements for DRB protocols such as Unpredictability, Bias-resistance, Availability (or Liveness), and Public Verifiability. We categorize the DRB protocols into interactive and non-interactive DRB protocols according to the nature of interactivity among DRB participants. In this paper, we examine the current challenges of DRB protocols, including complexity, scalability, and performance. We highlight these challenges and provide a few possible solutions along with intriguing research problems. These research problems can be taken into account and can push the cryptographers to unravel them and enrich the domain of DRB protocols.

Paper I: Competitive Decentralized Randomness Beacon Protocols
M. Raikwar
BSCI@ASIACCS, The Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure, ACM, 2022

This paper offers the general construction of competitive DRB protocols. Pertaining the knowledge from paper H, this paper highlights a new classification of DRB protocols as collaborative and competitive. This classification was first introduced in RANDCHAIN [38]; however, to the best of our knowledge, there is no other existing work that constructs competitive DRB except RANDCHAIN. In a collaborative DRB protocol, participants of the DRB collaborate on their local entropy to compute global randomness (beacon output). On the contrary, in a competitive DRB protocol, participants compete to generate global randomness. The global randomness is posted on a blockchain in a competitive DRB (e.g., RANDCHAIN). The idea behind constructing a competitive DRB protocol follows from the R3V consensus in Paper C.

Although RANDCHAIN is the first competitive DRB, a few challenges still exist related to fairness and blockchain-oriented attacks. We propose a general construction of competitive DRB protocols to overcome these problems. The basic idea of our competitive DRB protocol is a committee selection strategy followed by a puzzle-based competition among committee members. We define a few committee selection strategies, and each strategy provides a different level of fairness to the DRB participants. As a result, our competitive DRB protocol has linear communication complexity, improved fairness, and better scalability compared to state-of-the-art DRB protocols.

This paper receives the *Best Student Paper Award (Runner-up)* at the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure, as part of ACM AsiaCCS 2022.

3.3 Contributions toward Research Questions

Following, we present the contributions toward the research questions **RQ** defined in Section 1.2. We represent a contribution as an answer **ANS** to a research question **RQ**.

- **ANS1** Paper B, Paper C, and Paper I answer RQ1.
  Paper B and Paper C construct new consensus protocols involving a fair selection of consensus participants in each consensus round. Paper I presents novel competitive decentralized randomness beacon protocols involving a fair committee selection mechanism for the randomness generation process.

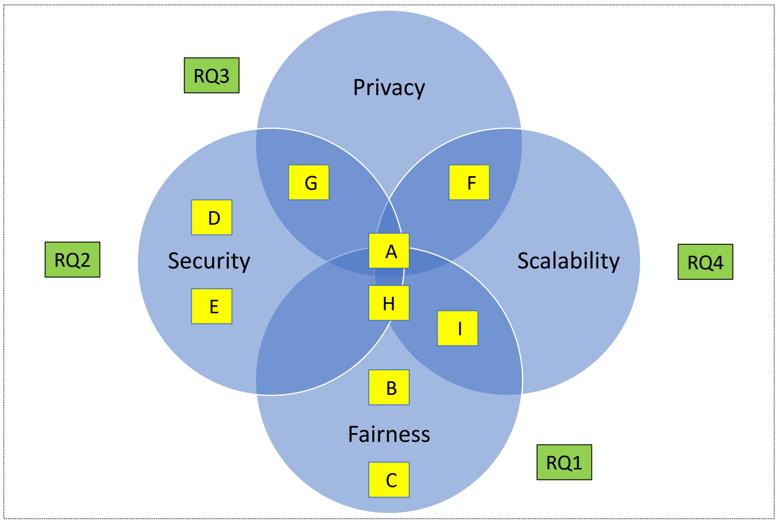

Figure 3.1: Overview of the papers included in the thesis.

- **ANS2** Paper D and Paper E answer RQ2.
  Paper D designs a DoS-resistant protocol and showcases its applicability in different contexts. Paper E presents a few techniques to mitigate DoS attacks in possible avenues of the blockchain ecosystem.

- **ANS3** Paper F and Paper G answer RQ3.
  Paper F proposes a privacy-preserving cryptocurrency system on top of scalable solutions. Moreover, Paper F also constructs a security model for the system and analyzes the security of the system. Paper G presents a general framework to assess the security of privacy-preserving cryptocurrencies.

- **ANS4** Paper F, Paper H, and Paper I address RQ4.
  Paper F considers an underlying scalable method to build a private bank on top of it. Paper H briefly introduces the solutions to improve scalability in randomness beacon protocols. Paper I defines a puzzle-based competition among participants without any setup assumption to generate randomness, and that makes the protocol scalable.

In addition, Paper A gives a brief overview of underlying cryptographic primitives in the blockchain. Paper A also addresses some research questions, especially RQ2, RQ3, and RQ4. Although Paper A does not provide direct answers to the questions, it provides a brief description of security, privacy, and scalability in the blockchain. Furthermore, Paper A presents several interesting research problems, and some of these research problems have been taken into consideration and solved by researchers and practitioners. Figure 3.1 depicts an overview of the papers included in the thesis. The papers are presented in yellow boxes, and the research questions are represented in green boxes.

The figure presents the relation between the papers and also shows which papers address which research question. Furthermore, it also illustrates which topic each paper covers, along with answering the research question. For instance, Paper F answers research question RQ3, but it also emphasizes scalability property in the blockchain. Another instance is Paper I, which answers RQ3 but it also provides scalability in terms of the number of participants in a DRB protocol.

Chapter 4
Conclusion

This chapter presents a perspective of the research contributions of the thesis. The chapter explains the findings of the research during the PhD and further examines the limitations and possibilities for future research.

4.1 Concluding Remarks

The thesis has presented a viewpoint on the challenges and development of the blockchain ecosystem. Blockchain research has been relatively advanced due to the effort put in by researchers in academia or industry. Advancements in blockchain research have been carried out in different communities, such as networking, cryptography, and business. The thesis has sought to present novel constructions from cryptography to solve existing challenges or improve upon existing schemes.

The thesis also furnishes several research problems presented in the two SoK papers and other auxiliary papers. These research problems can be of independent interest. Additionally, researchers have evaluated and solved some of these research problems, e.g., [131].

Altogether, the thesis does not only solve some of the existing problems in the blockchain ecosystem but also delivers several exciting research problems to investigate further and solve.

The thesis as a whole shows the following key findings **KF** in cryptography and blockchain research presented in the contributed papers.

- **KF1** Fairness has different meanings in blockchain protocols and can be instantiated differently.

- **KF2** Denial of Service attack is a serious concern on the internet, including in the blockchain ecosystem, and can be mitigated using client puzzles.

- **KF3** It is possible to achieve privacy together with scalability in cryptocurrency systems. Additionally, a general model can be constructed to assess the security of privacy-preserving cryptocurrency systems.

- **KF4** Randomness can be generated using different cryptographic primitives among a collaborative set of participants. Consequently, it can also be constructed by a competitive set of participants.

4.2 Future Research Directions

The thesis has contributed to solving some of the existing challenges in the blockchain. The works presented in the thesis suggest various research directions. Some of the vital research proposals can be described as follows:

- The consensus protocols presented in Paper B and C are well-suited for permissioned public blockchains. However, for a permissionless public blockchain, further research is required in order to maintain similar efficiency with better scalability. Although Paper B and C present a security analysis of the consensus, it would be interesting to perform a formal verification to check the correctness of the protocols.
- The client puzzle scheme designed in Paper D exhibits better properties compared to state-of-the-art client puzzle schemes except for costly verification time. An effort can be made to improve the verification complexity of the client puzzle. Another compelling area can be implementing the designed client puzzle in one of the possible avenues of DoS attack in the blockchain ecosystem.
- Paper F implements privacy on the top of a scalable cryptocurrency system. Regardless, each update from off-chain to on-chain incurs a high cost and a high on-chain verification cost. Therefore, a future research focus can be to reduce these associated costs and improve the overall performance of the system.
- Paper H systematizes the knowledge of decentralized randomness beacon protocols. It also lists a few interesting research problems. These research problems can be intriguing to look into and construct efficient solutions. Paper I presents a theoretical layout of constructing competitive DRB protocols. Implementing a competitive DRB protocol and conducting thorough experimentation to check the protocol's robustness would be fulfilling.

References

[1] Nakamoto, S. *Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.* 2009.

[2] CoinMarketCap. *Total Market Capitalization.* https://coinmarketcap.com. [Online; accessed 13-May-2022].

[3] Chaum, D. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology.* Ed. by Chaum, D., Rivest, R. L., and Sherman, A. T. Boston, MA, 1983, pp. 199–203.

[4] Dwork, C. and Naor, M. "Pricing via processing or combatting junk mail". In: *CRYPTO 92, Annual International Cryptology Conference.* Springer. 1992, pp. 139–147.

[5] Rivest, R. L., Shamir, A., and Wagner, D. A. *Time-lock Puzzles and Timed-release Crypto.* Tech. rep. Cambridge, MA, USA, 1996.

[6] Back, A. "The Hashcash Proof-of-Work Function". In: *Draft-Hashcash-back-00, Internet-Draft Created,(Jun. 2003)* (2003).

[7] Bitcoin. *Bitcoin Core integration/staging tree.* https://github.com/bitcoin/bitcoin.

[8] Groth, J. "Short non-interactive zero-knowledge proofs". In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2010, pp. 341–358.

[9] Groth, J. "On the size of pairing-based non-interactive arguments". In: *Annual international conference on the theory and applications of cryptographic techniques.* Springer. 2016, pp. 305–326.

[10] Bootle, J., Cerulli, A., Chaidos, P., and Groth, J. "Efficient zero-knowledge proof systems". In: *Foundations of security analysis and design VIII.* 2016, pp. 1–31.

[11] Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. "Verifiable Delay Functions". In: *Advances in Cryptology – CRYPTO 2018.* Ed. by Shacham, H. and Boldyreva, A. Cham, 2018, pp. 757–788.

[12] Pietrzak, K. "Simple Verifiable Delay Functions". In: *10th Innovations in Theoretical Computer Science Conference (ITCS 2019).* Ed. by Blum, A. Vol. 124. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany, 2018, 60:1–60:15.

[13] Wesolowski, B. "Efficient Verifiable Delay Functions". In: *Advances in Cryptology – EUROCRYPT 2019.* Ed. by Ishai, Y. and Rijmen, V. Cham, 2019, pp. 379–407.

[14] Novo, O. “Blockchain meets IoT: An architecture for scalable access management in IoT”. In: *IEEE internet of things journal* vol. 5, no. 2 (2018), pp. 1184–1195.

[15] Azzi, R., Chamoun, R. K., and Sokhn, M. “The power of a blockchain-based supply chain”. In: *Computers & industrial engineering* vol. 135 (2019), pp. 582–592.

[16] Raikwar, M., Mazumdar, S., Ruj, S., Gupta, S. S., Chattopadhyay, A., and Lam, K.-Y. “A blockchain framework for insurance processes”. In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2018, pp. 1–4.

[17] Hasselgren, A., Kralevska, K., Gligoroski, D., Pedersen, S. A., and Faxvaag, A. “Blockchain in healthcare and health sciences—A scoping review”. In: *International Journal of Medical Informatics* vol. 134 (2020), p. 104040.

[18] Ganesh, C., Orlandi, C., and Tschudi, D. “Proof-of-stake protocols for privacy-aware blockchains”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 690–719.

[19] Kerber, T., Kiayias, A., Kohlweiss, M., and Zikas, V. “Ouroboros crypsinous: Privacy-preserving proof-of-stake”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 157–174.

[20] Bünz, B., Agrawal, S., Zamani, M., and Boneh, D. “Zether: Towards privacy in a smart contract world”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 423–443.

[21] Huang, Y., Tang, J., Cong, Q., Lim, A., and Xu, J. “Do the rich get richer? Fairness analysis for blockchain incentives”. In: *Proceedings of the 2021 International Conference on Management of Data*. 2021, pp. 790–803.

[22] King, S. and Nadal, S. “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake”. In: *self-published paper, August* vol. 19, no. 1 (2012).

[23] Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., and Gervais, A. “Sok: Layer-two blockchain protocols”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 201–226.

[24] Poon, J. and Dryja, T. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[25] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., and Wuille, P. “Enabling blockchain innovations with pegged sidechains”. In: *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* vol. 72 (2014).

[26] Khalil, R., Zamyatin, A., Felley, G., Moreno-Sanchez, P., and Gervais, A. “Commit-chains: Secure, scalable off-chain payments”. In: *Cryptology ePrint Archive* (2018).

[27] Lenstra, A. K. and Wesolowski, B. “A random zoo: sloth, unicorn, and trx.” In: *IACR Cryptol. ePrint Arch.* vol. 2015 (2015), p. 366.

[28] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. “Algorand: Scaling byzantine agreements for cryptocurrencies”. In: *Proceedings of the 26th Symposium on Operating Systems Principles.* 2017, pp. 51–68.

[29] Adida, B. “Helios: Web-based Open-Audit Voting.” In: *USENIX security symposium.* Vol. 17. 2008, pp. 335–348.

[30] Goel, S., Robson, M., Polte, M., and Sirer, E. *Herbivore: A scalable and efficient protocol for anonymous communication.* Tech. rep. Cornell University, 2003.

[31] Bonneau, J., Clark, J., and Goldfeder, S. “On Bitcoin as a public randomness source.” In: *IACR Cryptol. ePrint Arch.* vol. 2015 (2015), p. 1015.

[32] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts”. In: *2016 IEEE symposium on security and privacy (SP).* IEEE. 2016, pp. 839–858.

[33] Cascudo, I. and David, B. “Albatross: publicly attestable batched randomness based on secret sharing”. In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2020, pp. 311–341.

[34] Bhat, A., Shrestha, N., Kate, A., and Nayak, K. “RandPiper-Reconfiguration-Friendly Random Beacons with Quadratic Communication.” In: *IACR Cryptol. ePrint Arch.* vol. 2020 (2020), p. 1590.

[35] Das, S., Krishnan, V., Isaac, I. M., and Ren, L. “SPURT: Scalable Distributed Randomness Beacon with Transparent Setup.” In: *IACR Cryptol. ePrint Arch.* vol. 2021 (2021), p. 100.

[36] Hanke, T., Movahedi, M., and Williams, D. “Dfinity technology overview series, consensus system”. In: *arXiv preprint arXiv:1805.04548* (2018).

[37] Schindler, P., Judmayer, A., Hittmeir, M., Stifter, N., and Weippl, E. “Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness”. In: *IACR Cryptol. ePrint Arch.* vol. 2020 (2020), p. 942.

[38] Han, R., Lin, H., and Yu, J. “RandChain: A Scalable and Fair Decentralised Randomness Beacon”. In: *Cryptology ePrint Archive* (2020).

[39] Raikwar, M., Gligoroski, D., and Kralevska, K. “SoK of used cryptography in blockchain”. In: *IEEE Access* vol. 7 (2019), pp. 148550–148575.

[40] Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. *Handbook of applied cryptography.* 2018.

[41] Gallagher, P. and Director, A. “Secure hash standard (SHS)”. In: *FIPS PUB* vol. 180 (1995), p. 183.

[42] Johnson, D., Menezes, A., and Vanstone, S. “The Elliptic Curve Digital Signature Algorithm (ECDSA)”. In: *International Journal of Information Security* vol. 1, no. 1 (2001), pp. 36–63.

[43] Josefsson, S. and Liusvaara, I. “Edwards-curve digital signature algorithm (EdDSA)”. In: *Internet Research Task Force, Crypto Forum Research Group, RFC*. Vol. 8032. 2017.

[44] Wood, G. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Yellow Paper. 2014.

[45] Schnorr, C. P. “Efficient Identification and Signatures for Smart Cards”. In: *Advances in Cryptology — CRYPTO’ 89 Proceedings*. Ed. by Brassard, G. New York, NY, 1990, pp. 239–252.

[46] The Monero Project. *Monero*. 2014.

[47] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. “Zerocash: Decentralized anonymous payments from bitcoin”. In: *2014 IEEE Symposium on Security and Privacy*. IEEE. 2014, pp. 459–474.

[48] Arthur Britto David Schwartz, R. F. *Ripple*. 2012.

[49] IO, E. “EOS. IO technical white paper”. In: *EOS. IO (accessed 18 December 2017) https://github.com/EOSIO/Documentation* (2017).

[50] LTO Network. *Blockchain for Decentralized Workflows*. 2014.

[51] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S. W., and Yellick, J. “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”. In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys ’18. New York, NY, USA, 2018, 30:1–30:15.

[52] Greenspan, G. *MultiChain Private Blockchain*. https://www.multichain.com/download/MultiChain-White-Paper.pdf. 2015.

[53] Lamport, L. “The part-time parliament”. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 277–317.

[54] Garay, J. and Kiayias, A. “Sok: A consensus taxonomy in the blockchain era”. In: *Cryptographers’ track at the RSA conference*. Springer. 2020, pp. 284–318.

[55] Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., Wen, Y., and Kim, D. I. “A survey on consensus mechanisms and mining strategy management in blockchain networks”. In: *Ieee Access* vol. 7 (2019), pp. 22328–22370.

[56] Liu, J., Li, W., Karame, G. O., and Asokan, N. “Toward fairness of cryptocurrency payments”. In: *IEEE Security & Privacy* vol. 16, no. 3 (2018), pp. 81–89.

[57] Dwork, C. and Naor, M. “Pricing via processing or combatting junk mail”. In: *CRYPTO 92, Annual International Cryptology Conference*. Springer. 1992, pp. 139–147.

[58] O’Dwyer, K. J. and Malone, D. “Bitcoin mining and its energy footprint”. In: (2014).

[59] Li, J., Li, N., Peng, J., Cui, H., and Wu, Z. “Energy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies”. In: *Energy* vol. 168 (2019), pp. 160–168.

[60] Gallersdörfer, U., Klaaßen, L., and Stoll, C. “Energy consumption of cryptocurrencies beyond bitcoin”. In: *Joule* vol. 4, no. 9 (2020), pp. 1843–1846.

[61] Huynh, A. N. Q., Duong, D., Burggraf, T., Luong, H. T. T., and Bui, N. H. “Energy consumption and Bitcoin market”. In: *Asia-Pacific Financial Markets* vol. 29, no. 1 (2022), pp. 79–93.

[62] Schletz, M. *Blockchain energy consumption: Debunking the misperceptions of Bitcoin’s and blockchain’s climate impact.* https://datadrivenlab.org. 2021.

[63] Lee, C. *Litecoin.* 2011.

[64] Percival, C. *Stronger key derivation via sequential memory-hard functions.* 2009.

[65] Krawczyk, H., Bellare, M., and Canetti, R. *HMAC: Keyed-hashing for message authentication.* Tech. rep. 1997.

[66] Buterin, V. “QuarkCoin: Noble Intentions, Wrong Approach”. In: *Bitcoin Magazine* (Dec. 2013).

[67] Duffield, E. and Diaz, D. *Dash: A payments-focused cryptocurrency.* Whitepaper, https://github.com/dashpay/dash/wiki/Whitepaper. 2018.

[68] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. “Algorand: Scaling Byzantine Agreements for Cryptocurrencies”. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. SOSP ’17. Shanghai, China, 2017, pp. 51–68.

[69] AplaProject. *Proof of Authority (POA).* 2018.

[70] Colvin, G., Lanfranchi, A., Carter, M., and IfDefElse. *ProgPoW, a Programmatic Proof-of-Work.* 2018.

[71] Kiayias, A., Russell, A., David, B., and Oliynykov, R. “Ouroboros: A provably secure proof-of-stake blockchain protocol”. In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

[72] David, B., Gaži, P., Kiayias, A., and Russell, A. “Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 66–98.

[73] Badertscher, C., Gaži, P., Kiayias, A., Russell, A., and Zikas, V. "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 913–930.

[74] Buchman, E. *Tendermint: Byzantine fault tolerance in the age of blockchains*. 2016.

[75] Buterin, V. and Griffith, V. "Casper the friendly finality gadget". In: *arXiv preprint arXiv:1710.09437* (2017).

[76] Conti, M., Kumar, E. S., Lal, C., and Ruj, S. "A survey on security and privacy issues of bitcoin". In: *IEEE Communications Surveys & Tutorials* vol. 20, no. 4 (2018), pp. 3416–3452.

[77] Zhang, R., Xue, R., and Liu, L. "Security and privacy on blockchain". In: *ACM Computing Surveys (CSUR)* vol. 52, no. 3 (2019), pp. 1–34.

[78] Woolf, N. "DDoS attack that disrupted internet was largest of its kind in history, experts say". In: *The Guardian* vol. 26 (2016).

[79] Gupta, B. and Badve, O. P. "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment". In: *Neural Computing and Applications* vol. 28, no. 12 (2017), pp. 3655–3682.

[80] Gasti, P., Tsudik, G., Uzun, E., and Zhang, L. "DoS and DDoS in Named Data Networking". In: *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. 2013, pp. 1–7.

[81] Douligeris, C. and Mitrokotsa, A. "DDoS attacks and defense mechanisms: classification and state-of-the-art". In: *Computer Networks* vol. 44, no. 5 (2004), pp. 643–666.

[82] Ron, D. and Shamir, A. "Quantitative analysis of the full bitcoin transaction graph". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 6–24.

[83] Eskandari, S., Moosavi, S., and Clark, J. "Sok: Transparent dishonesty: front-running attacks on blockchain". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2019, pp. 170–189.

[84] Diamond, B. E. "Many-out-of-Many Proofs and Applications to Anonymous Zether". In: *2021 2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA, 2021.

[85] Fauzi, P., Meiklejohn, S., Mercer, R., and Orlandi, C. "Quisquis: A new design for anonymous cryptocurrencies". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 649–678.

[86] Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., and Wu, H. "ZEXE: Enabling Decentralized Private Computation". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 947–964.

[87] Zyskind, G., Nathan, O., et al. “Decentralizing privacy: Using blockchain to protect personal data”. In: *2015 IEEE Security and Privacy Workshops*. IEEE. 2015, pp. 180–184.

[88] Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J. A., and Felten, E. W. “Mixcoin: Anonymity for bitcoin with accountable mixes”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 486–504.

[89] Maxwell, G. “CoinJoin: Bitcoin privacy for the real world”. In: *Post on Bitcoin forum*.

[90] Narula, N., Vasquez, W., and Virza, M. “zkledger: Privacy-preserving auditing for distributed ledgers”. In: *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 2018, pp. 65–80.

[91] Cecchetti, E., Zhang, F., Ji, Y., Kosba, A., Juels, A., and Shi, E. “Solidus: Confidential distributed ledger transactions via PVORM”. In: *Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 701–717.

[92] Danezis, G. and Meiklejohn, S. “Centrally banked cryptocurrencies”. In: *arXiv preprint arXiv:1505.06895* (2015).

[93] Almashaqbeh, G. and Solomon, R. *SoK: Privacy-Preserving Computing in the Blockchain Era*. Cryptology ePrint Archive, Report 2021/727. https://ia.cr/2021/727. 2021.

[94] Steffen, S., Bichsel, B., Gersbach, M., Melchior, N., Tsankov, P., and Vechev, M. “zkay: Specifying and enforcing data privacy in smart contracts”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 1759–1776.

[95] Kerber, T., Kiayias, A., and Kohlweiss, M. “KACHINA–Foundations of Private Smart Contracts”. In: *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. IEEE. 2021, pp. 1–16.

[96] Wang, G., Shi, Z. J., Nixon, M., and Han, S. “Sok: Sharding on blockchain”. In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019, pp. 41–61.

[97] Poon, J. and Buterin, V. “Plasma: Scalable autonomous smart contracts”. In: *White paper* (2017).

[98] Khalil, R., Zamyatin, A., Felley, G., Moreno-Sanchez, P., and Gervais, A. “Commit-chains: Secure, scalable off-chain payments”. In: *Cryptology ePrint Archive, Report 2018/642* (2018).

[99] Gluchowski, A. *Zk rollup: scaling with zero-knowledge proofs*. Matter Labs. 2019.

[100] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. “Omniledger: A secure, scale-out, decentralized ledger via sharding”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 583–598.

[101] Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., and Danezis, G. “Chainspace: A sharded smart contracts platform”. In: *arXiv preprint arXiv:1708.03778* (2017).

[102] Zamani, M., Movahedi, M., and Raykova, M. “Rapidchain: Scaling blockchain via full sharding”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 931–948.

[103] Yakovenko, A. “Solana: A new architecture for a high performance blockchain v0. 8.13”. In: *Whitepaper* (2018).

[104] Goodman, L. “Tezos—a self-amending crypto-ledger White paper”. In: *URL: https://www. tezos. com/static/papers/white paper. pdf* (2014).

[105] Maxwell, G., Poelstra, A., Seurin, Y., and Wuille, P. “Simple schnorr multi-signatures with applications to bitcoin”. In: *Designs, Codes and Cryptography* vol. 87, no. 9 (2019), pp. 2139–2164.

[106] Bellare, M. and Neven, G. “Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS ’06. Alexandria, Virginia, USA, 2006, pp. 390–399.

[107] Boneh, D., Drijvers, M., and Neven, G. “Compact Multi-signatures for Smaller Blockchains”. In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Peyrin, T. and Galbraith, S. Cham, 2018, pp. 435–464.

[108] Boneh, D., Lynn, B., and Shacham, H. “Short Signatures from the Weil Pairing”. In: *Advances in Cryptology — ASIACRYPT 2001*. Ed. by Boyd, C. Berlin, Heidelberg, 2001, pp. 514–532.

[109] Goldreich, O. and Oren, Y. “Definitions and properties of zero-knowledge proof systems”. In: *Journal of Cryptology* vol. 7, no. 1 (1994), pp. 1–32.

[110] Morgan, J. P. *Quorum*. 2016.

[111] Eberhardt, J. and Tai, S. “Zokrates-scalable privacy-preserving off-chain computations”. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE. 2018, pp. 1084–1091.

[112] Kiayias, A., Russell, A., David, B., and Oliynykov, R. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”. In: *Advances in Cryptology – CRYPTO 2017*. Ed. by Katz, J. and Shacham, H. Cham, 2017, pp. 357–388.

[113] Meckler, I. and Shapiro, E. “Coda: Decentralized cryptocurrency at scale”. In: *O (1) Labs whitepaper. May* vol. 10 (2018), p. 4.

[114] Brassard, G., Chaum, D., and Crépeau, C. “Minimum disclosure proofs of knowledge”. In: *Journal of Computer and System Sciences* vol. 37, no. 2 (1988), pp. 156–189.

[115] Pedersen, T. P. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *Advances in Cryptology — CRYPTO ’91*. Ed. by Feigenbaum, J. Berlin, Heidelberg, 1992, pp. 129–140.

[116] Miers, I., Garman, C., Green, M., and Rubin, A. D. “Zerocoin: Anonymous Distributed E-Cash from Bitcoin”. In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 397–411.

[117] Sun, S.-F., Au, M. H., Liu, J. K., and Yuen, T. H. “RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero”. In: *Computer Security – ESORICS 2017*. Ed. by Foley, S. N., Gollmann, D., and Snekkenes, E. Cham, 2017, pp. 456–474.

[118] Tomescu, A., Abraham, I., Buterin, V., Drake, J., Feist, D., and Khovratovich, D. “Aggregatable subvector commitments for stateless cryptocurrencies”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2020, pp. 45–64.

[119] Campanelli, M., Fiore, D., Greco, N., Kolonelos, D., and Nizzardo, L. “Incrementally aggregatable vector commitments and applications to verifiable decentralized storage”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 3–35.

[120] Raikwar, M. and Gligoroski, D. “The Meshwork Ledger, its Consensus and Reward Mechanisms”. In: *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2021, pp. 290–298.

[121] Raikwar, M. and Gligoroski, D. “R3V: Robust Round Robin VDF-based Consensus”. In: *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE. 2021, pp. 81–88.

[122] Raikwar, M. and Gligoroski, D. “Non-Interactive VDF Client Puzzle for DoS Mitigation”. In: *European Interdisciplinary Cybersecurity Conference*. 2021, pp. 32–38.

[123] Raikwar, M. and Gligoroski, D. *DoS Attacks on Blockchain Ecosystem*. 2022. arXiv: 2205.13322 [cs.CR].

[124] Gjøsteen, K., Raikwar, M., and Wu, S. “PriBank: Confidential Blockchain Scaling Using Short Commit-and-Proof NIZK Argument”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2022, pp. 589–619.

[125] Gjøsteen, K., Raikwar, M., and Wu, S. “Security Model for Privacy-preserving Blockchain-based Cryptocurrency Systems”. In: *Submission*. 2022.

[126] Raikwar, M. and Gligoroski, D. *SoK: Decentralized Randomness Beacon Protocols*. 2022. arXiv: 2205.13333 [cs.CR].

[127] Raikwar, M. "Competitive Decentralized Randomness Beacon Protocols". In: *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure*. BSCI '22. Nagasaki, Japan, 2022, pp. 83–94.

[128] Raikwar, M., Gligoroski, D., and Velinov, G. "Trends in development of databases and blockchain". In: *2020 Seventh International Conference on Software Defined Systems (SDS)*. IEEE. 2020, pp. 177–182.

[129] Raikwar, M. and Gligoroski, D. "Aggregation in Blockchain Ecosystem". In: *2021 Eighth International Conference on Software Defined Systems (SDS)*. IEEE. 2021, pp. 1–6.

[130] Stevenson, K., Skoglund, O., Raikwar, M., and Gligoroski, D. "Efficient Novel Privacy Preserving PoS Protocol Proof-of-concept with Algorand". In: *2021 3rd Blockchain and Internet of Things Conference*. 2021, pp. 44–52.

[131] Sasidharan, B. and Viterbo, E. "Private Data Access in Blockchain Systems Employing Coded Sharding". In: *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2021, pp. 2684–2689.

[132] Wicker, S. B. and Bhargava, V. K. *Reed-Solomon codes and their applications*. 1999.

[133] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. "Bulletproofs: Short proofs for confidential transactions and more". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 315–334.

Part II

Included Papers

Paper A

SoK of Used Cryptography in Blockchain

M. Raikwar, D. Gligoroski, and K. Kralevska

Published in IEEE Access Vol. 7, 2019

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 29, 2019, accepted October 3, 2019, date of publication October 11, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946983

SoK of Used Cryptography in Blockchain

MAYANK RAIKWAR, DANILO GLIGOROSKI, AND KATINA KRALEVSKA
Department of Information Security and Communication Technologies, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway
Corresponding author: Mayank Raikwar (mayank.raikwar@ntnu.no)

This work was supported by the NTNU Research Project under Grant 81771158.

ABSTRACT The underlying fundaments of blockchain are cryptography and cryptographic concepts that provide reliable and secure decentralized solutions. Although many recent papers study the use-cases of blockchain in different industrial areas, such as finance, health care, legal relations, IoT, information security, and consensus building systems, only few studies scrutinize the cryptographic concepts used in blockchain. To the best of our knowledge, there is no Systematization of Knowledge (SoK) that gives a complete picture of the existing cryptographic concepts which have been deployed or have the potential to be deployed in blockchain. In this paper, we thoroughly review and systematize all cryptographic concepts which are already used in blockchain. Additionally, we give a list of cryptographic concepts which have not yet been applied but have big potentials to improve the current blockchain solutions. We also include possible instantiations of these cryptographic concepts in the blockchain domain. Last but not least, we explicitly postulate 21 challenging problems that cryptographers interested in blockchain can work on.

INDEX TERMS Blockchain, cryptography, hash function, proof-of-work, consensus, signature, encryption, zero-knowledge proofs, access control, accumulator.

I. INTRODUCTION

Blockchain, a distributed ledger managed by a peer-to-peer network collectively adhering to some consensus protocol, is arguably considered as a new and disruptive technology. Both academia and industry are profoundly affected by new solutions to some old problems which are based on this new technology. The success of the blockchain concept is ultimately connected with the financial success of Bitcoin [1] that was developed just one decade ago, and the subsequent avalanche of more than 2140 other crypto-currencies that all together built a financial market worth around $285 billion (as of 16 June 2019) [2].

We can trace the origins of the ideas to use cryptography for secure and private transactions for paying access to databases, paying for services such as online games, transferring money over the Internet, Internet shopping and other commercial activities back in 1990's with David Chaum's eCash system [3]. One of the negative aspects of eCash was that it was a centralized system, controlled by a trusted third party. Another hurdle for a broader acceptance of eCash was the fact that it was covered by a long list of patented algorithms – something that is considered as a big obstacle to acceptance among the crypto community.

In parallel, in 1990's we saw the development of several cryptographic ideas not directly connected but somehow still related to the ideas of using cryptography in financial transactions. We mention some of them such as the proposal on how to combat junk email [4] by Dwork and Naor that was published in 1992, and which used computationally expensive functions. Then in 1996, there was a proposal for time-lock cryptographic puzzles [5] by Rivest, Shamir, and Wagner by using RSA based CPU expensive computations. At the end of 90's and early 2000's several patent free cryptographic concepts were proposed, implemented and released as open source projects by an online movement and a community of cryptographers and programmers known as "Cypherpunks" [6]. Those cryptographic concepts and implementations include Adam Back's "hash-cash" proposal for a currency based on the hardness of finding partial hash collisions [7], Wei Dai's "b-money" [8] and Nick Szabo's[1] "Bitgold" proposal [9]. These concepts have been the basis of the Satoshi Nakamoto's decentralized cryptocurrency, nowadays known as Bitcoin [1], [10]. As a recognition of their pioneering activities in the decentralized cryptocurrencies, Ethereum [11] – the second most popular

The associate editor coordinating the review of this manuscript and approving it for publication was Yunlong Cai.

[1]Nick Szabo was also part of the eCash development team in late 90's.

cryptocurrency – named the three of its denominations as "Wei", "Szabo" and "Finney" [12].[2]

The underlying core technology in Bitcoin is blockchain. Blockchain is a distributed ledger maintaining a continuously growing list of data records that are confirmed by all of the participating nodes. The data is recorded in this public ledger in a form of blocks of valid transactions, and this public ledger is shared and available to all nodes.

Blockchain is envisioned as a promising and powerful technology but it still encounters many research challenges. Some of the main challenges are constant improvement of its security and privacy, key management, scalability, analysis of new attacks, smart contract management, and incremental introduction of new cryptographic features in existing blockchains. These challenges arise due to the network structure and the underlying consensus mechanisms and cryptographic schemes used within the blockchains. To overcome these challenges and to find enhanced solutions, many of the cryptographic concepts such as signature schemes, zero-knowledge proofs, and commitment protocols are scrutinized and applied. As cryptography is a vast research field, there is always a scope to find new cryptographic schemes in order to improve the solutions in blockchain.

The majority of the ongoing research in Blockchain focuses on finding and identifying improvements to the current processes and routines, mostly in industries that rely on intermediaries, including banking, finance, real estate, insurance, legal system procedures, and healthcare. The study on business innovation through blockchain [14] presents some blockchain enabled business applications and their instantiations. These blockchain enabled applications still need a proper way for selecting the cryptographic technique employed in their respective solution in order to meet the business requirements. Not only these blockchain applications but also the research community will benefit from an overview in a form of systematization of the current state of knowledge of all available cryptographic concepts which have been applied or can be applied in existing and future blockchain solutions. To the best of our knowledge, this is the first systematization of knowledge that gives a complete picture of the existing cryptographic concepts related to blockchain. We have tried to depict most of the cryptographic concepts in the blockchain domain. Although there are various works about specific cryptographic concepts used in blockchain, there are only few works which merge all these atomic works and present them in a single paper. Most of the review and survey works such as [15], [16] discuss security, privacy, consensus or other challenges in blockchain. A recent work of Wang et al. [17] gives a comprehensive analysis of cryptographic primitives in blockchain. Their analysis presents the functionality and the usage of these primitives in blockchain. However, the analysis is based only on existing cryptocurrencies and it lacks many of the cryptographic protocols which are used in blockchain.

A. OUR CONTRIBUTION

In this study, we classify cryptographic concepts based on their use in blockchain.[3] We have divided them into two categories: 1. Concepts which are well used in blockchain, and 2. Concepts which are promising but not yet implemented in blockchain. This categorization does not have a clear boundary. We classify some cryptographic concepts as promising ones, and that requires further research and scrutiny in order to be deployed in blockchain. As a result, the following points are the main contributions of our Systematization of Knowledge (SoK) paper:

- We provide a description of cryptographic concepts which have been applied in the blockchain field. We also include instantiation of these concepts in blockchain.
- We provide a list of cryptographic concepts which are rarely used or have not been used in blockchain but they have the potential to be applied in this field. These concepts open many possible research directions and they can be examined in different blockchain applications.
- We identified 21 research challenges that we formulate as *Research Problem*. Some of them are rephrased research challenges already published in the literature and some of them are newly formulated research problems.

In this study, we do not claim that we have exhausted all of the cryptographic concepts which are employed in blockchain, but we have tried to cover the concepts which we felt are propitious for the blockchain domain. We also describe each cryptographic concept along with its associated properties and its instantiation in the blockchain field. Additionally, in order to give one unified presentation about blockchain, we give a brief explanation about:

- Enabling concepts of blockchain such as hash function, consensus protocol, network architecture.
- Layered architecture of blockchain and emphasis on some of the major challenges associated with blockchain.

B. ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows. Section II presents the research methodology. Section III explains the main pillars of blockchain such as hash functions, consensus mechanisms, network infrastructure and types of blockchain. Section IV gives an overview of some critical challenges faced by existing blockchains. Section V reviews already used cryptographic concepts in blockchain and presents the basic idea of each cryptographic concept with available instantiation in blockchain. Section VI presents cryptographic concepts which have not been employed or implemented in blockchain yet, but look very promising for

[2]Hal Finney was a cypherpunk and the receiver of the first Bitcoin transaction of 10 Bitcoins from the anonymous Satoshi Nakamoto [13].

[3]A continuously updated version of cryptographic concepts is available on this github repository http://bit.do/fchb5

blockchain. Finally, Section VII concludes this SoK and gives possible future work directions.

II. RESEARCH METHODOLOGY

To perform a systematization of knowledge of the existing cryptographic concepts related to blockchain, we established and followed a methodology that we explain in this Section. Since the invention of Bitcoin, there has been a growing interest in blockchain from both academia and industry. The number of publications in the blockchain field has been rapidly increasing in recent years. Not all of these publications are research works; some of these works discuss different use-cases of blockchain. Therefore, to review these many papers in the blockchain field, we pursued a research methodology which defines the inclusion criteria, a search strategy to search for respective publications and a data collection mechanism to accumulate the relevant publications. The collected data is later processed based on inclusion and exclusion criteria. The publications which meet the inclusion criteria go through one final step of quality assessment. Once a publication passes the quality assessment, it is included in our systematization.

We use keyword search to make the first selection of potentially relevant scientific publications. For the keyword search, we typed keywords such as <*cryptographic concept name*> <*in blockchain*> or <*use of*> <*cryptographic concept name*> <*in blockchain*>. We use Google Scholar as our primary source to search for the relevant literature, but as Google Scholar does not exhaust all of the available literature, we also searched in databases such as: 1) IACR eprint archive, 2) IEEE Xplore, 3) ACM Digital Library, 4) ScienceDirect, and 5) Springer Link.

The inclusion criteria for this study is based on the following questions:

- Is the elaborated cryptographic concept useful in blockchain? The usefulness of the cryptographic concept is measured as whether we achieve some essential properties in blockchain by using the concept or whether the cryptographic concept can be beneficial for some use-case compared to an already implemented concept.
- Which properties can be achieved by using the cryptographic concept in blockchain?
- Is there any instantiation of the cryptographic concept in a blockchain study or application? If not, is there any potential?

The criteria for excluding a paper is:

- Informal literature discussing some cryptographic concepts in blockchain.
- Literature which claims on using a cryptographic concept but it does not give any guarantees about the feasibility and prospects of a potential implementation.

The quality of the papers that meet the inclusion criteria is assessed. For quality assessment, we apply the following questions:

- Is the cryptographic concept implemented in blockchain? If not, is it possible to implement it and will it be more efficient than the existing solution?
- Is there any security analysis or does the implemented concept rely on another underlying platform?
- Are the fundamental concept and its related properties adequately described?

III. SUPPORTING AND ENABLING CONCEPTS OF BLOCKCHAIN

As previously mentioned, blockchain is a way to encapsulate transactions in the form of blocks where blocks are linked through the cryptographic hash, hence forming a chain of blocks. Figure 1 shows the basic blockchain structure. Each block in the blockchain contains a block header and a representation of the transaction. For instance, in Figure 1, each block consists of its hash, the hash of the previous block, a timestamp and some other block fields (e.g., version, nonce). This depends from the block design. Merkle root hash represents the set of transactions in the Merkle tree, and this representation of transactions varies according to the design of the blockchain implementation. Figure 2 depicts the Bitcoin blockchain data structure showing in details the block format.

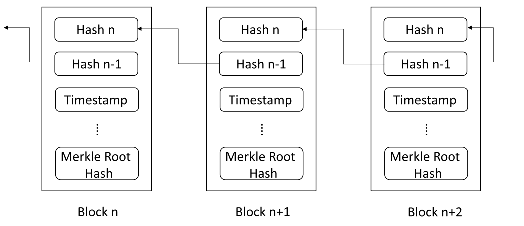

FIGURE 1. **Basic blockchain structure.**

Blockchain relies on different constituents which serve different purposes. In this Section, we give an overview of the main underlying concepts used to build a blockchain. A detailed technical explanation of all these concepts is out of the scope of this paper, but we have tried to cover the essentials of their functionality.

A. CRYPTOGRAPHIC HASH FUNCTION

A hash function H is a function which takes an input of an arbitrary size and maps it to a fixed size output. Cryptographic hash functions have some additional properties such as: **a**) *collision resistance* - it is hard to find two inputs a and b such that $H(a) = H(b)$; **b**) *preimage resistance* - for a given output y it is hard to find an input a such that $H(a) = y$; and **c**) *second preimage resistance* - for a given input a and output $y = H(a)$ it is hard to find a second input b such that $H(b) = y$. Readers interested in an extensive cover of the field of cryptographic hash functions are referred to [18].

Cryptographic hash functions in blockchain are used for various purposes such as:

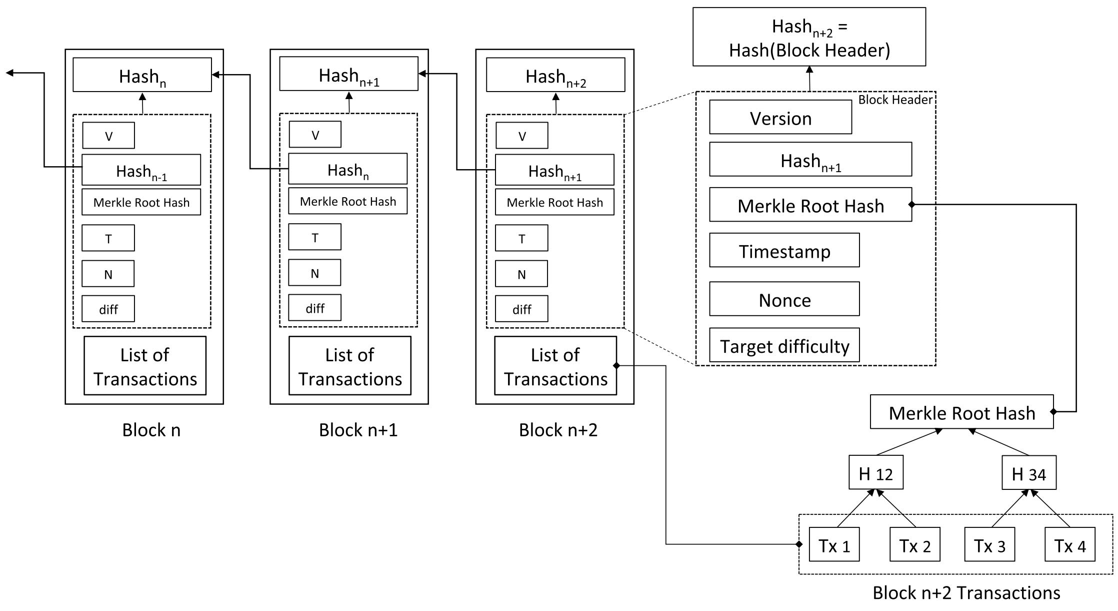

FIGURE 2. **Blockchain data structure with block format.**

1) solving cryptographic puzzles (the Proof of Work (PoW) in Bitcoin [1]);
2) address generation (for public and private keys);
3) shortening the size of the public addresses;
4) message digests in signatures.

The most popular cryptographic hash functions used in blockchains are SHA-2 [19] (especially the variant SHA256 - a variant that produces outputs of 256 bits), and some of the well analyzed hash functions from the NIST SHA-3 competition and standardization that went to the later stages of that process (final 5 proposals or some of the 14 proposals from the second phase [20]). Some of the existing blockchain designs such as IOTA constructed their own "homebrewed" cryptographic hash function called Curl-P, that was received very critically and negatively by the crypto community [21], [22].

A typical way how cryptographic hash functions are used in blockchain designs is in a form of a mode of operation, i.e., a combination of several invocations of a same or different hash functions. For example, in Bitcoin [1], SHA256 is used twice and that construction is called SHA256d, i.e.,

$$\text{SHA256d}(message) = \text{SHA256}(\text{SHA256}(message)). \quad (1)$$

Mining is a process of creating a new block of transactions through solving a cryptographic puzzle, and the participant who solves the puzzle first is called a *miner of the block*. If we look at the Bitcoin PoW puzzle, we can see that a miner has to find a *Nonce* (similar to Hashcash protocol [7] that we discuss in the next subsection) to create the next block in the blockchain. The puzzle looks like this:

$$\text{SHA256d}(Ver||HashPrevBlock||\ldots||Nonce) \leq T \quad (2)$$

where T is 256-bit target value.

Looking into the fraction of SHA256d outputs that are less than the target value T for different values of T in Table 1 helps us to understand why mining is hard in PoW. Namely, the probability of finding a nonce that will cause the whole block to have a hash that is less than the target value is

$$Pr[\text{SHA256d}(Block) \leq T] \approx \frac{T}{2^{256}}. \quad (3)$$

TABLE 1. **Fraction of SHA256d outputs with respective target value.**

Target Value T	Fraction of SHA256d outputs $\leq T$
$0\text{x}7\,\underbrace{\mathtt{t}_1\mathtt{t}_2\mathtt{t}_3\mathtt{t}_4\ldots\mathtt{t}_{62}\mathtt{t}_{63}}_{63\,times}$	$\frac{1}{2}$
$0\text{x}0\,\underbrace{\mathtt{t}_1\mathtt{t}_2\mathtt{t}_3\mathtt{t}_4\ldots\mathtt{t}_{62}\mathtt{t}_{63}}_{63\,times}$	$\frac{1}{16}$
$0\text{x}\,\underbrace{00\ldots00}_{16\,times}\underbrace{\mathtt{t}_1\mathtt{t}_2\mathtt{t}_3\mathtt{t}_4\ldots\mathtt{t}_{47}\mathtt{t}_{48}}_{48\,times}$	$\frac{1}{2^{64}}$

We next discuss the research and innovative activities in the area of cryptographic hash functions that were either remotely or directly connected and inspired by the trends in blockchain.

Several years after the launch of the Bitcoin and its source code being published as an open source on Github, blockchain designers started to clone and fork its basic code, and started to introduce different variants and innovations. One of the earliest forks from 2011 that is still popular nowadays is Litecoin [23]. The basic idea by the Litecoin design was to use a different hash function for its proof of work puzzles. The motivation came from the fact that even in 2011 there were trends to build specialized application-specific integrated circuit (ASIC) hardware implementations of SHA256d that will mine the blocks several orders of magnitude faster than ordinary CPUs and GPUs. Instead of SHA256d, Litecoin uses *Scrypt* [24] - a memory-intensive compilation of use of the HMAC [25] construction instantiated with SHA256 and use of the stream cipher Salsa20/8 [26]. The idea was that the use of *Scrypt* will be impractical to implement it in ASIC, thus, giving chances of individual owners of regular computers and GPUs to become a significant mining community. While with no doubts we can say that Litecoin is a very successful alternative cryptocurrency, we can for sure claim that its initial goal to be ASIC resistant blockchain design was not successful. Nowadays, you can find commercial products for Litecoin hardware mining.[4]

Actually, we can say that the 10 years of history of blockchain, in general, and cryptocurrencies, in particular, is a history of failed attempts to construct a sustainable blockchain that will prevent the appearance of profitable ASIC miners that can mine the blocks with hash computing rates that are several orders of magnitude higher than the ordinary users of CPUs and GPUs. In that short history, we can mention Ethash used in Ethereum [11] for which there are now commercially available ASIC miners by at least two companies. In 2013, QuarkCoin [27] introduced the idea of using a chain of six hash functions (five SHA-3 finalists BLAKE, Grøstl, JH, Keccak and Skein [28]) and the second round hash function Blue Midnight Wish [29]. One of the motivations behind the QuarkCoin PoW function was to be more ASIC resistant than SHA256d. The cascading idea of QuarkCoin was later extended to a cascade of eleven hash functions in Darkcoin (later renamed DASH [30]). Needless to say, nowadays there are commercially available ASIC miners for X11 as well.

The frictions between ASIC miners and the cryptocurrency community seem to remain to the present days, and are somewhat evolving and inspiring novel proposals in blockchain protocols. The latest is the Programmatic Proof-of-Work (ProgPoW) initiative for Ethereum blockchain ecosystem that aims to make ASIC mining less efficient and to give some advantages to graphics processing units (GPU) mining [31].

B. CONSENSUS MECHANISMS

Consensus is the key component of blockchain to synchronize or update the ledger by reaching an agreement among the participants. In order to maintain the ledger in a decentralized way, many consensus mechanisms have been proposed. The first introduction of the use of a consensus mechanism in blockchain is implicitly given by Bitcoin. Bitcoin uses Proof of Work (PoW) mechanism as consensus where the idea came from *Hashcash Protocol* [7]. The objective of Hashcash was to prevent spam in public databases. The *Hashcash Protocol* is as follows. Suppose an email client wants to send an email to an email server. In the beginning, the client and the server both agree on a cryptographic hash function H which maps an input string to an n length output string. Then, the email server sends a challenge string c to the client. Now the client has to find a string x such that $H(c||x)$ starts with k zeros. Since H has pseudorandom outputs, the probability of success in a single trial is

$$\frac{2^{n-k}}{2^n} = \frac{1}{2^k}.$$

Here x corresponding to c is considered as PoW and the process of finding that x is called mining. PoW is difficult to generate but easy to verify.

Many literature studies on consensus mechanisms, for instance, the survey by Wang et al. [16] and "SoK: Consensus in the age of blockchains" [32], have been carried out in the past few years. Since consensus mechanisms have already been thoroughly studied in the literature, in this paper, we present the basic idea about how consensus mechanisms work and their classification.

In a consensus protocol, depending on the network architecture and blockchain type, some or all of the participants take part and maintain the ledger by adding a block consisting of transactions to their ledger. However, the creation of a new block to be added to the ledger is performed by a participant who is known as a leader of the consensus protocol in that particular execution. This leader is elected by different mechanisms of leader election process, and some of these mechanisms are given in Table 2.

TABLE 2. Leader election in consensus protocols.

Leader Election Criteria	Reference Protocols
PoW Puzzle Competition	Bitcoin-NG [33], Casper [34], Proof of Stake velocity [35]
Verifiable Random Function	Tendermint [36], Algorand [37], Secure Proof of Stake [38]
Trusted Random Function	Proof of luck [39], Proof of elapsed time [40]
Modified Preimage Search	Snow White [41]
Sub-network of Masternodes / Validator nodes	Darkcoin and DASH [42], Libra [43]

After the leader is elected and the new block is created in order to achieve consensus or agreement on this block, two types of voting mechanisms are followed: *explicit* and *implicit*. In explicit voting, multiple rounds of voting occur and then based on the votes, consensus is reached. However, in implicit voting, the new block created by the leader is accepted by others who implicitly vote for the new block

[4]One such a product that can compute 580 billion Scrypt hashes per second, is offered by the company Bitmain and is called "Antminer L3++". As of the time of writing this article, this product was advertised at https://shop.bitmain.com/ for a price of $213.00 and for a 10 days delivery (2 June 2019).

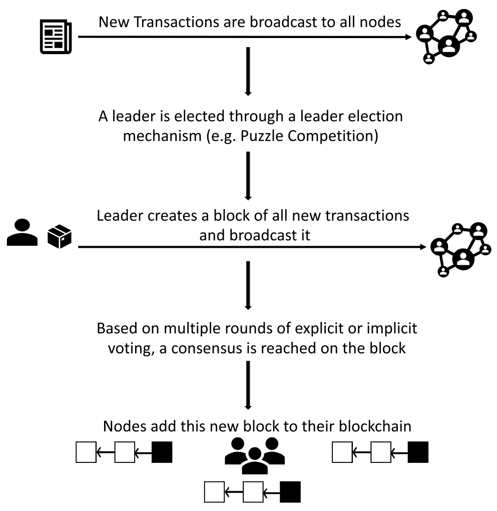
New Transactions are broadcast to all nodes
A leader is elected through a leader election mechanism (e.g. Puzzle Competition)
Leader creates a block of all new transactions and broadcast it
Based on multiple rounds of explicit or implicit voting, a consensus is reached on the block
Nodes add this new block to their blockchain

FIGURE 3. Blockchain consensus scenario.

and add it to their ledgers. A leader election through PoW puzzle competition (e.g., PoW puzzle 2 in Bitcoin) followed by an implicit voting to reach an agreement is also called "Nakamoto Consensus".

Consensus mechanisms also determine the performance of the blockchain network in terms of consensus finality, throughput, scalability, and robustness against various attacks. In some manner, consensus orchestrates the state of the programs executed in the blockchain network nodes by providing a runtime environment to collectively verify the same program and hence reach to a finality. There is no exact classification of consensus mechanisms, but in general they can be classified as consensus protocols with proof of concept and consensus protocols with byzantine fault-tolerant replication. These consensus protocols can be chosen based on the blockchain network and type. Most of the proof of concept consensus protocols are used in permissionless blockchains. There are many proof of concept schemes which have been proposed and implemented, e.g., Proof of Work (PoW) [44], Proof of Stake (PoS) [45], Equihash [46], having Masternodes in Dash [42], etc. As described in Section III-A, in PoW puzzle based consensus protocols, miners try to solve the cryptographic puzzle by mining and these miners are also responsible for verification of the transactions, and an incentive (*reward*) is given to the first miner who solves the puzzle.

In case of a permissionless network, as there is no authentication and no proper synchronization, the underlying consensus algorithm should be able to handle the synchronization problem, scale well and mitigate different attacks in order to maintain canonical blockchain state in P2P network. To solve this synchronization issue, most of the blockchains use "Longest chain rule" to have a consistent canonical state of blockchain in this P2P blockchain network. On the contrary, in the permissioned blockchain, as there are restrictions and privileges associated with the peers, there is a strict control on the synchronization among the peers. Byzantine fault-tolerant protocols are usually adopted in permissioned blockchains to provide consensus properties such as validity, agreement, and termination. Practical Byzantine Fault Tolerant (PBFT) [47], Proof of Elapsed Time [40], Ripple consensus [48] are some of the consensus protocols used in permissioned blockchains. Recently, Facebook launched its own global cryptocurrency Libra [43] which works as a permissioned blockchain and provides users to do transactions with nearly zero fee. Libra blockchain comes with a new programming language Move and a new consensus protocol called LibraBFT.

1) MINING, POOL MINING AND INCENTIVE MECHANISMS

In Proof of Work based blockchains, the addition of new transactions in the blockchain is performed by the mining process. In the Bitcoin mining process, a puzzle is solved by computing many hashes repeatedly (Equation 2) by putting different values for the nonce to satisfy the condition. When a miner successfully solves the puzzle first among all of the miners, it gets a monetary incentive for solving the puzzle. Because of this incentive process, all consensus nodes or miners follow the rules of the blockchain state transition during the puzzle competition. Mining is a resource-intensive process where the main resources are computational power and memory. Mining can be performed either by a solo miner or by a group of miners, called a mining pool, who collectively try to solve the puzzle. Mining pools may operate on different mining techniques and incentive mechanisms. These incentive mechanisms can vary based on the used mining technique or the decision of the pool operator. Reference [16] gives a brief idea about the mining strategy management in blockchain networks, while reference [49] provides a strategic study of mining through stochastic games. Different incentive mechanisms are proposed and tested in blockchains. Reference [50] analyzes Bitcoin pooled mining reward systems, and a reward system based on information propagation in blockchain network is presented in [51].

C. NETWORK INFRASTRUCTURE

Blockchain is maintained by a peer-to-peer (P2P) network. P2P network is an overlay network which is built on the top of the Internet. This P2P blockchain network can be modeled as structured, unstructured or hybrid based on several parameters such as the consensus mechanism and the type of blockchain. Regardless of the representation of the network, a blockchain network should quickly disseminate the newly generated block so that the global view of the blockchain remains consistent. Consequently, a synchronization protocol is needed, but a routing protocol might or might not be needed. A traditional P2P network uses a routing protocol to route the information through multihop; however, in many blockchains (e.g., Bitcoin), routing is not required because a peer can get information through at most one hop, so no routing table is maintained.

Almost all cryptocurrencies and blockchains such as Bitcoin [1], Ethereum [11], Litecoin [23] use unstructured P2P network where the idea is to have equal privileges for all of the nodes and to create an egalitarian network. A P2P network can follow flat or hierarchical organization for building a random graph among the peers. This graph is not fully connected, but in order to receive all of the communication and to maintain the ledger, each peer maintains a list of peer addresses. Thus, if any peer propagates a message in the network, eventually all peers receive it through their available connections. In an unstructured network, techniques like flooding and random walk are used to make new connections with the peers. In the unstructured network, peers can leave and join at any time. This can be exploited by an adversary that can join and see the messages floating in the network and can further do source spoofing, reordering or injecting of messages.

Blockchain can also use structured P2P network where nodes are organized in a specific topology and thus finding any resource/information becomes easier. In this structured P2P network, an identifier is assigned to each node to route the messages in a more accessible way. Each node also maintains a routing table. A structured P2P network maintains a distributed hash table (DHT) where (key, value) pairs are stored corresponding to the peers which help in the resource discovery. Ethereum has started the adoption of structured P2P network by using Kademlia protocol [60]. However, most of the blockchain networks are unstructured, and moreover, if the blockchain is public where no restriction to join or leave the network is enforced, then many possible attacks can happen. Thus, the security of blockchain depends heavily on the network architecture. A propagation delay or a synchronization problem in a P2P network can affect the consensus protocol of blockchain, leading to a non-consistent global view in blockchain. In addition to these problems, an adversary can cause several attacks in a P2P network, where few of the main attacks are as following:

- Netsplit (Eclipse) attack: An adversary monopolizes all of the connections of a node and splits that node from the entire network. Further, the node cannot participate in consensus or validation protocol and this causes inconsistency in the network [61].
- Routing attack: A set of participants are isolated from the blockchain network by the adversary and thus the block propagation is delayed in the network [62].
- Distributed Denial-Of-Service (DDOS) attack: An adversary exhausts the network resources and targets honest nodes so that honest nodes do not get the services or information which they are supposed to receive [63], [64].

D. TYPES OF BLOCKCHAIN

Blockchains can be classified depending on the implementation design, administration rules, data availability, and access privileges. From an academic point of view, they have been classified as "public" and "private". While from the administrative point of view, they are described as "permissioned" and "permissionless". Nevertheless, these terms are used interchangeably in most of the blockchain studies and applications in industries, which is not the correct way to use these terms. Even though the classification of blockchains is not very clearly specified in the literature, we can still classify blockchains by coupling public, private, permissioned and permissionless.

1) *Permissionless Public:* In this type of blockchain, anyone can join or leave the network at any time and participate in consensus as well to maintain the ledger. Everyone also has read and write access to the blockchain. Thus, it provides minimum trust among the participants, but it still achieves maximum transparency. Most of the cryptocurrencies and blockchain platforms are permissionless public, e.g., Bitcoin [1], Zerocash [52] and Monero [53].
2) *Permissioned Public:* This type of blockchain allows everyone to read the blockchain state and data, but in order to write the data and take part in consensus, there are permissions/privileges associated with the participants provided by the network administrator which in a certain way makes the system not fully decentralized. In this type of blockchain once a participant has some privileges, based on that it can become a validator as well. Examples for permissioned public blockchain are Ripple [54], EOS [55] and the newest Libra [43].
3) *Permissionless Private:* This type of a blockchain allows organizations to collaborate without the need of sharing information publicly. Being permissionless, allows anyone to join or leave the blockchain at any time, which is also acknowledged by other nodes as well. The smart contracts on these networks also define who is allowed to read the contract and the related data, not only just who is allowed to perform the actions. Some permissionless private blockchains use Federated byzantine agreement as a consensus protocol. LTO [56] network is an example of a permissionless private blockchain which creates "live contract" on the network.
4) *Permissioned Private:* These blockchains are mostly used in organizations where data/information is stored in the blockchain with permissioned access control by members of the organization. The membership in the network is provided by the network administrator or some membership authority. Read and write access to the data is also provided by the network administrator. Hyperledger fabric [57], Monax [58], Multichain [59] are examples of permissioned private blockchains.

Table 3 proffers a clear picture of the classification of blockchains with associated advantages, challenges and application domains. However, in general, permissionless public blockchains are commonly referred to as public blockchains and permissioned private blockchains are

TABLE 3. Blockchain classification.

Blockchain Type	Application Domain	Anonymity	Scalability	Challenges	References
Permissionless Public	Decentralized P2P Networks	High	Low	Privacy, Scalability	Bitcoin [1], Zerocash [52], Monero [53]
Permissioned Public	Decentralized Organizations	High	Moderate	Privacy, Centralization	Ripple [54], EOS [55]
Permissionless Private	Intra-Organization Networks	Moderate	Moderate	Consensus, Scalability	LTO [56]
Permissioned Private	Organizational restricted ledgers	Low	High	Consensus, Centralization	Hyperledger fabric [57], Monax [58], Multichain [59]

TABLE 4. Layered architecture of blockchain.

	Confidentiality	Integrity	Availability	Data Privacy	Anonymity
Smart Contract	Encryption	MAC	–	Data Privacy Preserving Computation	Identity Privacy Preserving Computation
Transaction	–	Signature Scheme	Access Structure of Transactions	Zero-Knowledge Proofs, Mixing Techniques	Zero-Knowledge Proofs
Consensus	–		Consensus	Access Control	Blind or Ring Signature
Network	Encryption	MAC	Protocols e.g. Gossip	–	IP Anonymity e.g. TOR
Database			Access Control	Access Control	–

referred to as fully private blockchains. A combination of permissioned public and permissionless private makes "consortium blockchain" which is also called a federated blockchain. A consortium blockchain is neither completely public nor completely private, and it makes blockchain as partially decentralized. In consortium blockchain, the consensus is reached by a selected group of participants. Nowadays most of the organizations have embraced consortium blockchains for their blockchain enabled solutions.

IV. CHALLENGES IN BLOCKCHAIN

Blockchain as an emerging technology comes with many challenges. In order to solve these challenges, various solutions have been proposed and implemented in the blockchain. The proliferation of cryptocurrencies across multiple payment systems brings many risks in social, economic and technical terms. Blockchain encounters many challenges due to network architecture, underlying consensus protocol and applied cryptographic primitives. Some of these major challenges are security and privacy associated with blockchain, scalability of blockchain, and resource consumption (computational power, memory, network bandwidth). An insightful analysis on the research perspectives and challenges for bitcoin and other cryptocurrencies [65] has been presented in the past and gives a nice overview of scalability, security, privacy and consensus of cryptocurrencies.

We can summarize our discussion in Section III-B, in a form of generic research problems and research challenges in the area of blockchain consensus mechanisms as follows. Construct a new blockchain consensus mechanism that is better than the existing ones from the following perspectives:

1) Less energy consumption;
2) More efficient consensus achievements;
3) Better security than the existing consensus mechanisms.

However, further in the paper when we identify a more concrete and focused research challenge, we formulate it in a form of a Research Problem. For example, from the discussion given in the III-A we can formulate the following:
Research Problem 1: Construct sustainable blockchain systems that have one of the following properties:

1) They are provably resistant to give mining advantages to ASIC miners as opposite to GPU and CPU miners;
2) They are provably resistant to give mining advantages to ASIC and GPU miners as opposite to CPU miners.

If we observe the blockchain as a layered architecture, we can identify the challenges that occur in each layer. Table 4 shows blockchain as a stack of five layers. These five layers serve the following purposes:

- **Smart contract layer** processes contract data and send the result data to the transaction layer.
- **Transaction layer** creates the transactions and sends those to consensus layer.
- **Consensus layer** runs the consensus algorithm and adds the transactions to the block.
- **Network layer** deals with all P2P communication among blockchain nodes.
- **Database layer** stores the blockchain data in a respective database used by respective blockchain platform.

Table 4 gives a glimpse of blockchain layered architecture and also mentions some of the cryptographic techniques to achieve properties like security and privacy. In Table 4, the first column defines the layers of blockchain, and the first row illustrates the properties which can be accomplished in the different layer using different cryptographic techniques. Thus to understand, each cell corresponds to the deployed cryptographic method to attain the property in the corresponding column in the respective blockchain layer (corresponding row). For example, encryption can be used to achieve confidentiality in smart contract layer, Message Authentication Code (MAC) can be used to achieve integrity in the network layer of blockchain. Table 4 names few of the techniques used in the blockchain but there are more available cryptographic techniques which can be employed in blockchain. "–" in Table 4 represents that the corresponding property for the corresponding layer does not make much sense. Some of the significant challenges of blockchain are as follows.

A. SECURITY AND PRIVACY

For any blockchain, a key evaluation parameter is how well the security and privacy conditions meet the requirement of the blockchain. Analyzing the security and privacy issues of blockchain is a broad research area, and some studies have been conducted in this area. Here we do not cover those details, instead we only define these terms. Security is defined as three components: confidentiality, integrity, and availability. In a generic context, (i) confidentiality is a set of rules that limits access to information, (ii) integrity is the assurance that the information is trustworthy and accurate, and (iii) availability is a guarantee of reliable access to the information by authorized people. However, in case of blockchain, the term *Information* used in the above context can have multiple meanings such as data in the database, smart contract data or transactions. Privacy can be defined as data privacy and user privacy (anonymity). Table 4 includes some cryptographic mechanisms for achieving security and privacy of information subjected to different blockchain layers.

In the light of recent increased number of incidents with the security of the different layers of blockchain platforms and the theft of millions of dollars worth cryptocurrencies, we formulate the following research problem.

Research Problem 2: Construct a penetration testing tool irrespective of the blockchain platform to test the security and privacy requirements for each layer of any blockchain platform.

B. SCALABILITY ISSUES

The size[5] of blockchain is continuously growing, and scalability is becoming a big problem in the blockchain domain. Scalability depends on the underlying consensus, network synchronization and architecture. To scale the blockchain, the computational power and the bandwidth capabilities should be high for each node in the blockchain, which is practically infeasible. Most of the current blockchains grant limited scalability.

One proposal how to address the scalability problems of the blockchain ledger is so called: "SPV, Simplified Payment Verification" [66]. It verifies if particular transactions are valid but without downloading the entire ledger. This method is used by some wallet and lightweight Bitcoin clients, and its security was first analyzed in [67]. Another proposal to achieve high scalability is to use erasure codes in blockchain by encoding validated blocks into small number of coded blocks. A recent work [68] proposes the use of fountain codes (a class of erasure codes) to reduce the storage cost of blockchain by the order of magnitude and hence achieving high scalability. Applying other types of erasure codes for distributed storage, such as regenerating codes [69], [70], locally repairable codes [71], [72] or a combination of both types of codes [73], [74], may reduce even further the storage and communication costs.

Another issue in connection with the scalability is the issue of the interoperability. Namely, it is a fact that the number of different public ledgers is increasing rapidly. While some sort of a rudimentary interoperability has been implemented in cryptocurrencies exchange platforms [75], the risks and insecurities with these platforms are vast and well documented [76].

Research Problem 3: Construct a new blockchain mechanism that periodically prunes its distributed ledger (reduces its size), producing a fresh but equivalent ledger, while provably keeping correct state of all assets that are subject of the ledger transactions.

Research Problem 4: Construct secure protocols for blockchain interoperability.

A recent reference [77] strongly supports our research problem 3 since it admits that Ethereum blockchain is almost full now and hence the scalability is a big bottleneck.

C. FORKING

A blockchain fork is essentially caused when two miners find a block at almost the same time due to a software update or versioning. In a blockchain network, each device or computer is considered as "a full node" which runs software to keep the blockchain secure by verifying the ledger. The software is updated to adjust some parameters and to install new features in the blockchain. This updated software may not be compatible with the old software. Consequently, the old nodes which have not updated their software and the new nodes which have performed a software update can cause a fork in the blockchain when they create new blocks. There are two types of forks: one which is not compatible with previous software version, called a hard fork, and another one which is compatible with the previous version (backward-compatible), called a soft fork. A hard fork happens when there is a significant change in the software such as change of block parameters or change of consensus mechanism. In the case of Ethereum, a hard fork will occur when it will migrate from Proof of Work to Proof of Stake. One example of a soft

[5]https://bitinfocharts.com gives most of the statistics (including size) of popular cryptocurrencies.

fork is Segregated Witness (SegWit) which was implemented in Bitcoin by changing the transaction format. Recently, privacy coin Beam [78] (an implementation of Mimblewimble privacy protocol) conducted its first hard fork away from ASICS. Figure 4 depicts a blockchain forking scenario where the correct chain can be any of these two forked chains depending on the case of the hard or soft fork.

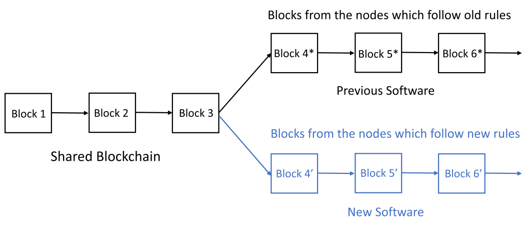

FIGURE 4. Blockchain forking.

Research Problem 5: Construct Forking-free consensus mechanism for permissionless public blockchain.

D. THROUGHPUT

It is a measure of the number of blocks appended in blockchain per second which effectively means the number of transactions processed per second. Throughput depends on many factors such as underlying consensus algorithm, number of nodes participating in consensus, network structure, node behavior, block parameters and the complexity of the contract (in case of smart contract supported blockchains). The complexity of a smart contract depends on whether the programming language of the blockchain is turing-complete or not. However, regarding turing-completeness of blockchains [79], there is always a division between the blockchain community. Considering these primary factors, attaining high throughput is a bit hard in blockchain. However, for value-asset blockchains to achieve high throughput, the size of the transaction can be reduced by excluding some information from the transaction and the throughput can be increased by increasing the block size and the bandwidth of the network till a certain level.

The number of transactions per second was recognized as a serious problem in Bitcoin network. While in the peak holiday period Visa and MasterCard can handle up to 50,000 transactions per second worldwide, the Bitcoin network can handle just 7 transactions. One proposal how to address this scalability issue is the "The Bitcoin Lightning Network" [80]. It is a network that handles instantly the Bitcoin transactions off the main ledger. It establishes a network of micropayment channels that addresses the malleability by using Bitcoin multi-signatures 2-of-2. Special nodes are needed for these micropayment networks and as of June 2019, there were around 4,500 nodes. The first financial transaction via the Lightning network was reported in January 2018. Litecoin decided to follow the Bitcoin Lightning network, and as of March 2019 there were more than 1000 registered nodes that handle the micropayments for that alternative cryptocurrency. Many other solutions were proposed to solve the scalability issue, similar to the Lightning off-chain computation and off-chain state channels, such as Sharding [81], Plasma [82], Liquid [83] and the recent Channel Factories [84].

As the Lightning network has gained popularity, new research challenges emerge as explained in [85], and here we rephrase one of their research challenges.

Research Problem 6: *[85]:* Develop scalable protocols that will perform multi-hop payment-channel and path-based transactions with strong privacy guarantees even against an adversary that has network-level control.

Addressing Problem 6, many works have been done in the past but all those works are mostly compatible with Bitcoin or Ethereum blockchain. Recent works [86], [87] on multi-hop payment channel provide value privacy and security but only for Bitcoin-compatible blockchains. Instead of supporting only payments like Lightning network, there are off-chain state channels, like Celer Network [88], which support general state updates while providing significant improvement in terms of cost and finality.

Research Problem 7: Develop fully functional state channel with strong security and privacy guarantee.

E. ENERGY CONSUMPTION

The mining process of blockchain (e.g., bitcoin mining) consumes a lot of energy. Most of the PoW puzzle based consensus protocols waste a huge amount of energy.[6] Many alternative consensus algorithms are introduced which use less energy than Bitcoin's PoW such as PoS [45], Equihash [46], and PBFT [47]. Energy is also consumed during communication over the network. Some cryptographic mechanisms also consume high energy so the selection of a proper cryptographic mechanism should be based not only on the memory requirement and the computational load but also on the amount of energy consumption. The use of blockchain should be energy efficient and to fulfill that 1) PoS-like consensus should be used and 2) proper energy management techniques should be utilized, for example in the case of Internet-of-Things (IoT).

F. INFRASTRUCTURE DEPENDENCIES

The blockchain infrastructure is built with several elements of network protocols, cryptographic concepts, and mining hardware. All these elements depend on each other in some sense. If we look into the layered architecture of blockchain in Table 4, each layer is dependent on its upper and lower layers for some input/output. Thus, there are many infrastructure dependencies in blockchain. For instance, the data from the smart contract layer is an input to the transaction layer that outputs actual transactions; the data from the consensus layer

[6]https://digiconomist.net/bitcoin-energy-consumption depicts Bitcoin energy consumption index charts in TWh per year. It also shows the energy consumption per country.

results in an input to the network layer through a communication protocol; and the data from the network layer data is sent to the database through database storage management. These dependencies must be taken into account while building a comprehensive blockchain framework for any use case; otherwise, some of the blockchain functionalities will not be fulfilled.

From the blockchain infrastructure perspective, we have to mention here one evolving and enabling technology that will be very important in the next decade: 5G. 5G will connect hundreds of billions of IoT devices, but that vast number of devices can be governed securely only by strong decentralized mechanisms offered by the blockchain technologies [89], [90]. We formulate this debate as the following.
Research Problem 8: Construct efficient, scalable, inexpensive and sustainable blockchain systems capable to handle and securely manage up to billions of IoT devices connected via the 5G network infrastructure.

V. OVERVIEW OF USED CRYPTOGRAPHIC CONCEPTS IN BLOCKCHAIN

From the cryptographic point of view, many of the cryptographic techniques have already been exhibited and heavily employed in various blockchain platforms and blockchain use-cases [17]. As the spectrum of the cryptographic concepts is vast, there is always a scope to dig out some of the existing cryptographic schemes and use them in blockchain services.

In Table 5 we give a comprehensive summary of all cryptographic concepts that we will cover in this and in the next Section. It serves as a handy overview and quick reference table for our systematization of the cryptographic knowledge used in blockchain.

Following are some of the cryptographic concepts which have already been well analyzed and implemented in blockchain.

A. SIGNATURE SCHEME

A standard digital signature is a mathematical scheme based on public-key cryptography that aims to produce short codes called signatures of digital messages by the use of a private key, and where those signatures are verifiable by the use of the corresponding public key. In this context, digital signatures guard against tampering and forgeries in digital messages.

A signature scheme is used in blockchain to sign the transaction, hence, authenticating the intended sender and providing transaction integrity as well as non-repudiation of the sender. Many of the signature schemes are widely accepted to employ integrity and anonymity in blockchain. The digital signature is one of the most important cryptographic primitives that makes blockchain to be publicly verifiable and with achievable consensus. Signature schemes are used in almost every blockchain. Figure 5 represents a general example about how a blockchain user (signer) creates a digitally signed transaction or block using his private key. Moreover, figure 6 shows how other blockchain nodes (verifier) verify whether the signature on the transaction or block is valid or not using the signer's public key. Blockchain applies different signature schemes to provide additional features like privacy, anonymity, and unlinkability. Signature scheme can also be applied to generate constant size signature using signature aggregation. Schnorr Signatures are a form of signature aggregation and it has been used in Bitcoin instead of P2SH [125] for scalability [126]. Some of the signature schemes applied in blockchain are:

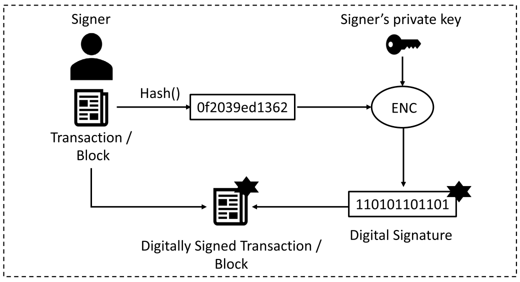

FIGURE 5. Signing process of blockchain transaction/block.

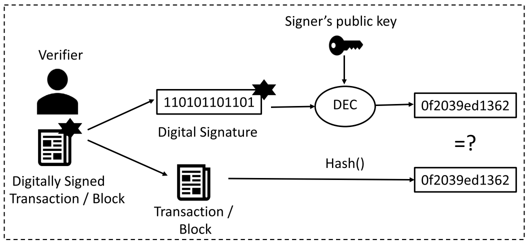

FIGURE 6. Verification of digitally signed transaction/block.

1) Multi-Signature: In a multi-signature scheme, a group of users signs a single message. In a blockchain network, when a transaction requires a signature from a group of participants, it might be advantageous to use a multi-signature scheme. Blockchain platforms such as Openchain [127] and MultiChain [59] support $M-$of$-N$ multi-signature scheme which reduces the risk of theft by tolerating compromise of up to M-1 cryptographic keys. Boneh et al. also designed compact multi-signatures for smaller blockchains [128].
2) Blind Signature: In this scheme [129], signatures are employed in privacy-related protocols where the signer and the message authors (transaction in case of blockchain) are different parties. Blind signatures are used to provide unlinkability and anonymity of the transaction. In a blockchain setup, a blind signature might be helpful to provide anonymity and unlinkability where the transacting party and the signing party are different. Blind signatures have been used in BlindCoin [130] distributed mixing network to provide the unlinkability of transactions. Blind signatures are

TABLE 5. Summary of Cryptographic Concepts in Blockchain.

Cryptographic Concept	**Properties**	**Instantiation (Reference)**
Access Control	Data privacy	Hyperledger Fabric [57], FairAccess [91], [92]
Accumulator	Provides Membership Proofs, Anonymity	Batching Techniques for Accumulators in Blockchain [93]
Aggregate Signature	Fast Signature Verification	Tested in Bitcoin [94]
Commitment Scheme	Non-Repudiation	Used in Bulltеproof [95] and in Monero [53], [96]
Decentralised Authorization	Data Privacy	BlendCAC [97], WAVE [98]
Encryption Scheme	Confidentiality and Anonymity	Kadena [99], Hyperledger Fabric [57], Tendermint [36]
Identity Based Encryption	No Public Key Distribution Infrastructure	BAVP [100], BLIC [101]
Incremental Cryptography	Efficiency Improvement	Kadena [99]
Lightweight Cryptography	Fast, less Memory/Energy Consumption	LSB [102], EVCE [103]
Obfuscation	Privacy	Tested in Bitcoin [104]
Oblivious RAM	Confidentiality and Integrity	Solidus [105], EVORAM [106]
Oblivious Transfer	Data Privacy	Searchain [107], [108]
Post-Quantum Cryptography	Quantum Resistant	Post-Quantum Blockchain [109], [110], [111]
Private Information Retrieval	Data Privacy	Private Blockchain Queries from PIR [85]
Proof of Retrievability	Cloud Data Recovery	Permacoin [112], Retricoin [113], Storj [114]
Secret Sharing	Data privacy	SHARVOT [115], Wanchain [116]
Secure Multiparty Computation	Privacy of Peers and Smart Contract	Enigma [117], Hawk [118], Wanchain [116]
Signature Scheme	Integrity and Authentication	In Every Blockchain e.g. Multichain [59], CryptoNote [119]
Verifiable Delay Function	Less Parallelism, Fast Verification	Chia Network [120]
Verifiable Random Function	Verifiable Pseudorandom Output	Algorand [37], Ouroboros Praos [121], Dfinity [122]
White-Box Cryptography	Data Privacy	Runtime Self-Protection in Blockchain Ledger [123]
Zero-Knowledge Proof	User and Data privacy	Zerocoin [124], Zerocash [52]

also tested in Bitcoin to provide the anonymity for the Bitcoin on-chain and off-chain transactions [131].

3) Ring Signature: This scheme [132] uses a protocol where a signature is created on a message by any member of a group on behalf of the group while preserving the identity of the individual signer of the signature. Ring signatures are used to achieve anonymity of the signing party in the blockchain network. CryptoNote [119] technology uses a ring signature scheme to create untraceable payments in the cryptocurrencies. A trustless tumbling platform [133] also uses ring signature for anonymity.
4) Threshold Signature: This signature scheme is a (t, n) threshold signature where n parties receive a share of the secret key to create the signature and t out of n parties create a signature over any message. As the parties directly construct the signature from the shares, the key is never revealed in the entire scheme. Threshold signature can be helpful to provide anonymity in the blockchain network. CoinParty [134] uses a threshold signature scheme for multi-party mixing of Bitcoins. A recent work about coin mixer, ShareLock [135], uses threshold ECDSA (Elliptic Curve Digital Signature Algorithm [136]) to provide privacy-enhancing solution for cryptocurrencies. However threshold ECDSA signatures are complex due to the intricacies of the signing algorithm. Other signature schemes, such as EdDSA (Edwards-curve Digital Signature Algorithm [137]) using the Edwards25519 curve, are efficient threshold signatures. Libra [43] blockchain applies this EdDSA during new account address generation.

While digital signatures produced with the keys used in Public Key Infrastructure (PKI) are well legally regulated and can be used in different types of legal disputes, it is a big challenge how to achieve similar regulations with all types of digital signatures used in the existing blockchain solutions. Additionally, in the physical world if an asset is stolen (for example an expensive car, or an expensive watch), it can be traced back to its legal owner.

Research Problem 9: Develop security protocols that merge the existing standardized and legalized PKI systems with some of the developed blockchain systems.

Research Problem 10: Design an anti-theft blockchain system, i.e., a system that guarantees a return of stolen assets back to their legitimate owners.

Regarding Research Problem 10, recently the Vault proposal was re-launched. Its purpose is to shield the bitcoin wallet from theft without the need for hard forking [138]. However, for other blockchain systems, no such proposal or solution exists.

B. ZERO-KNOWLEDGE PROOFS

In Zero-knowledge proofs [139], two parties, a prover and a verifier, participate. First, the prover asserts some statement and proves its validity to the verifier without revealing any other information except the statement. Thus, a zero-knowledge proof proves the statement as 'transfer of an asset is valid' without revealing anything about the asset. Zero-knowledge protocols are extremely useful cryptographic protocols for achieving secrecy in the applications. They can be used to provide the confidentiality of an asset (transaction data) in the blockchain while keeping the asset in the blockchain. Some of the public blockchains use zero-knowledge proofs such as Zerocoin [124] or Zerocash [52] for untraceable and unlinkable transactions. Zerocoin is a decentralized mix and extension to Bitcoin for providing anonymity and unlinkability of transactions by applying zero-knowledge proofs. In Zerocoin protocol, a user who has Bitcoins can generate an equal value of Zerocoins without the need of any third party mixing set. A user can spend his/her Bitcoin by 1) producing a secure commitment (i.e., Zerocoin), 2) recording it in the blockchain, and 3) broadcasting a transaction and a zero-knowledge proof for the respective Zerocoin. Hence, other users can validate the Zerocoin recorded in the blockchain and verify the transaction along with the proof. Here zero-knowledge proof protects the linking of Zerocoin to a user, yet Zerocoin is a costly protocol due to its high complexity and large proof size.

To reduce the complexity and the proof size, a variant of zero-knowledge proof known as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) [140] is used by Zerocash protocol. zk-SNARK hides the information about the amount and the receiver address in a transaction. The main idea of zk-SNARK is any computational condition can be represented by an arithmetic circuit, which takes some data as input and gives *true* or *false* in response. zk-SNARK reduces the proof size and the computational effort compared to the basic zero-knowledge proofs. An enterprise-focused version of Ethereum, Quorum blockchain platform [141] also uses zk-SNARK for transaction privacy and anonymity. Figure 7 illustrates an interactive protocol of zero-knowledge where the prover has a statement, and he/she wants to prove that the statement is correct without revealing any information related to the statement. In the interactive protocol, the verifier asks many questions related to the statement and the prover answers these questions in such a way where the prover proves the statement and does not reveal any necessary information.

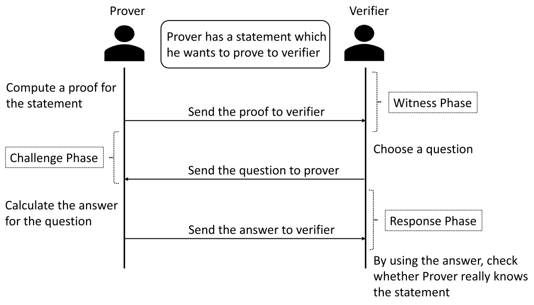
Prover
Verifier
Prover has a statement which he wants to prove to verifier
Compute a proof for the statement
Send the proof to verifier
Witness Phase
Choose a question
Challenge Phase
Send the question to prover
Calculate the answer for the question
Response Phase
Send the answer to verifier
By using the answer, check whether Prover really knows the statement

FIGURE 7. An interactive zero-knowledge protocol.

C. ACCESS CONTROL

It is a selective restriction on information or resource based on some policy or criteria. These mechanisms [142] can be enforced to put a restriction or access in the blockchain. The access can be a read/write access or an access to participate in a blockchain protocol. There are many different access control mechanisms such as role-based, attribute-based, organizational-based access control which can be used in blockchain. Recent incidents show security breaches and data theft from certain blockchain platforms, which can be tackled and prevented by access control. The privacy of data can be ensured in blockchains by using access control [91], [92]. Nowadays, access control techniques are profoundly used in blockchain based medical applications [143] or blockchains for the insurance industry where the data is sensitive information that must be accessible to only trusted and authorized parties. There are different types of access control mechanisms which can be utilized in blockchain applications.

1) Role-based Access Control (RBAC): RBAC is an approach for restricting the system view to the users of the system according to their roles in the system. Thus, it can be applied in a blockchain framework where access is provided according to the user roles. RBAC is used in a blockchain based solution for healthcare [144]. A simple example depicted in Figure 8 describes the role-based access control in a private healthcare blockchain. Based on the role, each entity in the blockchain system has its own access rights. A Patient can ask for his personal medical data, however only the Doctor associated with the patient can enter or modify the patient's health record in the blockchain. A Research Company on the other hand can ask for patients' data for any disease for research purpose.
2) Attribute-based Access Control (ABAC): In ABAC, the access control rules are based on the attribute structure. These attributes can be user specific, environment-specific or object specific. For example, in a blockchain setup for the insurance industry, 'department' could be an attribute through which the access of the blockchain data is restricted, which means the claims handling department would have a different

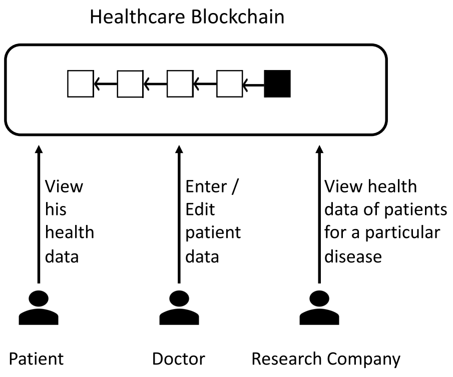
Healthcare Blockchain
View his health data
Enter / Edit patient data
View health data of patients for a particular disease
Patient
Doctor
Research Company

FIGURE 8. Role-based access control in healthcare blockchain.

view of the blockchain compared to the audit department. ABAC can be used in a fair access blockchain model [91] by keeping attributes in policy.

3) Organization-based Access Control (OrBAC): OrBAC is one of the richest access control models. OrBAC consists of three entities (subject, action, object) which define that some subject has the permission to realize some action on some object. OrBAC has already been used in blockchain for IoT in a fair access blockchain model [91] and in dynamic access control model on blockchain [145].

Other access control mechanisms such as context-based access control and capability-based access control (proposed in blockchain solutions for autonomous vehicles, smart cities, IoT [146]) can also be useful for different blockchain solutions.

D. ENCRYPTION SCHEME

It is a process of encoding a piece of information by which only authorized parties can access it. It can be used to achieve confidentiality of blockchain data by encrypting it. There are many encryption schemes which can be used in blockchain. Symmetric-key Encryption is used in Hyperledger fabric for confidentiality of smart contract [57] and Blockchain for Smart Home [147]. Although searching and computation over an encrypted data is a big challenge, there are many existing techniques which can be used for that purpose. Some of these techniques such as searchable encryption for searching on encrypted data in the cloud is already used in permissioned blockchain [148], and for computation over encrypted data, fully homomorphic encryption and functional encryption can also be utilized in blockchain. Monero cryptocurrency [53] uses (half) additive homomorphic encryption together with range proof techniques, yet supporting only value transactions.

In order to assure simultaneously confidentiality and authenticity of data, an authenticated encryption can be used in blockchain. In authenticated encryption, two peers establish a connection, they both share their public keys and compute the shared secret which is used as the symmetric key for the authenticated encryption algorithm. The recently finished cryptographic competition CAESAR [149] has identified a portfolio of six ciphers for authenticated encryption. So far, as of this writing (June 2019), none of those ciphers has been deployed in some blockchain system.

Broadcast encryption can be used in blockchain to provide the anonymity of blockchain receiver nodes. [150] gives a proposal to use for Availability and Accountability for IoT by blockchain. It has as every user in the group receives the encrypted message, although only users with the correct permission or key can decrypt it.

E. SECURE MULTI-PARTY COMPUTATION (SMPC)

Secure Multi-party Computation enables parties to act together in a way that no single party has an access to all of the data, and hence no one can leak any secret information. The main idea of SMPC scheme is to jointly compute a function by parties over their inputs without disclosing their inputs. For example, a group of people can compute the average salary of the group without disclosing their actual individual salaries. The blockchain platform Enigma [117] leverages the concept of SMPC to achieve strong privacy. In Enigma platform, a blockchain network is combined with SMPC network, where the blockchain network contains the hashes and SMPC network contains the data corresponding to those hashes which split is among different nodes. For each node, the view over SMPC network differs as everyone has a different piece of information. Specifically, each node contains a random piece of data, and no single party ever has access to the entire data.

A blockchain model Hawk [118] for privacy-preserving smart contracts also specifies the use of SMPC to minimize the trust in the generation of common reference string in SNARK proof used in the model. SMPC can also be exercised for private data storage in a decentralized system, such as Keep [151]. Keep provides a privacy-focused storage solution for Ethereum. In this system, network nodes collaborate to provide secure decentralized data containers, called keeps, which can be accessed from smart contracts on Ethereum.

An application of SMPC can also be seen in the Wanchain [116] Cross-Chain network. Figure 9 reflects the SMPC idea in cross-chain transfer model. In Wanchain network, if user A wants to send an asset (say ETH) from one blockchain (say Ethereum blockchain) to user B on Wanchain blockchain, then at first the asset value is locked in an account on its original chain using smart contract. This locked account holds control of the funds. The equivalent token WETH is sent to another user B of the Wanchain network. When user B wants to convert its WETH to ETH, the locked amount is released from the locked account and sent to user B, and the equivalent portion of WETH is burned. These locking and unlocking of asset value (ETH) happen using SMPC. Wanchain has a concept of Storeman nodes

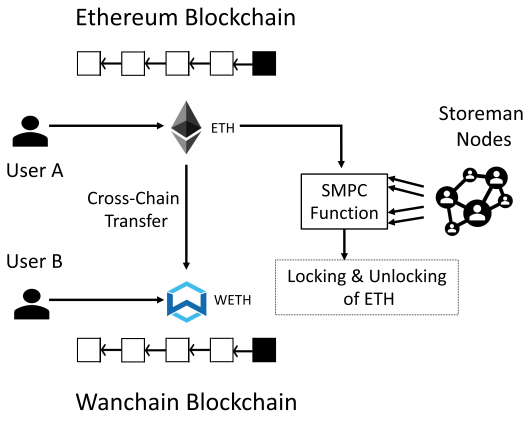

FIGURE 9. Cross-Chain transfer mechanism of blockchain using SMPC.

which work together and perform locking and unlocking of account. These Storeman nodes jointly work together to create public and private key pair of the related locked account. This shared account private key is scattered among the Storeman nodes as pieces of the key. To unlock the account, M out of N ($M \leq N$) Storeman nodes contribute their shares of the private key to generate the signature using MPC jointly.

F. SECRET SHARING

In this concept, a secret is divided into multiple parts among the participants, and it is reconstructed by using a minimum number of parts. These parts are called shares and they are unique for each participant. Secret sharing is used to secure sensitive information. Secret sharing scheme is advantageous in SMPC for distributing the shares among parties. Shamir's secret sharing [152] is already being used to distribute transaction data, without a significant loss in data integrity in blockchain [153]. Decentralized Autonomous Organizations (DAO) can take advantage of secret sharing by distributing the shares of information among the system nodes rather than storing full information in each node. Secret sharing in DAO can be practiced in consensus where each participating node stores a set of shares of the system state rather than storing full system state. These shares are points on polynomials which make up part of the state.

Secret sharing schemes are also used in different off-chain and on-chain bitcoin wallets to safeguard the private keys of the crypto holders. For example, suppose an organization wants to store its bitcoin with a single master private key. In that case, secret sharing scheme helps to store the same key among multiple people. A simple example of this scenario will be sharing a bitcoin wallet key among three people by distributing the shares of the key. These individual shares do not convey any information about the actual key. However, any 2 of 3 people can reconstruct the key using their shares as presented in Figure 10. Secret sharing schemes can also benefit blockchain by storing secret information in a decentralized way so that unauthorized parties cannot access it. Secret sharing is used in blockchain for different purposes such as secret share-based fair and secure voting protocol (SHARVOT) [115] and new cryptocurrency based on mini blockchain [154].

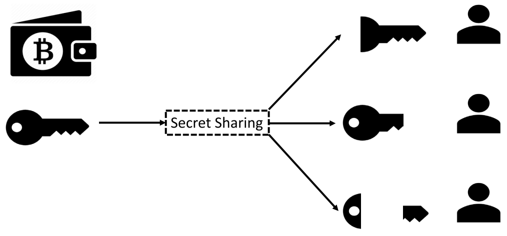

FIGURE 10. Secret-Sharing-Scheme 2-of-3 for a cryptocurrency wallet private key.

G. COMMITMENT SCHEME

A commitment scheme is a digital analog of a sealed envelop. It is a two-phase game between two parties where the phases are commit and open. Commit phase involves hiding and binding of a secret by the first party and send it to the second party; while open is to prove that the first party did not cheat the second party in the commit phase. Therefore, a commitment scheme satisfies the aforementioned two security properties: *hiding* and *binding*. Hiding ensures that the receiver cannot see the message before the open phase, while binding ensures that the sender cannot change the message after the commit phase. The following example shows a binding commitment:

1) Pick a secret value s to commit from 0 to $p-1$ where p is a large prime number;
2) Calculate the value $c = g^s \mod p$;
3) Publish the value c as a commitment.

In the above example, the binding property follows as it is infeasible for the sender to find any other value y which gives the same c. Here finding the value s from known c, p and g is a computationally hard problem of discrete logarithm but any party can verify the commitment value c if s is provided. There are many commitment schemes such as Pedersen commitment [155] and elliptic curve Pedersen commitment. Zerocoin [124] uses Pedersen commitment to bind a serial number s to Zerocoin z. The commitment c is given as follows:

$$c = g^s h^z \mod p.$$

Here g, h, and p are known to everyone, and the user chooses s, z and computes and publishes the commitment c. These s, z cannot be computed from c even if one is provided. As a consequence, in Zerocoin when the serial number s is published, the user can prove his/her ownership by providing z. Pedersen commitment has also been used to build blockchain-oriented range proof system, Bulletproof [95] and its elliptic curve version is also successfully implemented in Monero [53], [96]. A switch commitment scheme is designed for confidential transactions in blockchain [156].

H. ACCUMULATOR

An accumulator is a one-way function which gives a membership proof without revealing individual identity in the underlying set. This can be used in blockchain to build other cryptographic primitives such as commitment, ring signatures, and zero-knowledge proofs. Merkle tree, used in many cryptocurrencies, fits under a more comprehensive class of cryptographic accumulators which is space and time efficient data structure to test for set membership. Figure 11 shows how blockchain transactions are represented in the Merkle tree, and the Merkle root is stored in the block structure of the blockchain. Non-Merkle accumulators are classified as RSA accumulators and elliptic curve accumulators.

In Zerocoin [124], an accumulator A is computed by the network overall coin commitments $(c_1, c_2, \ldots, c_n)$ along

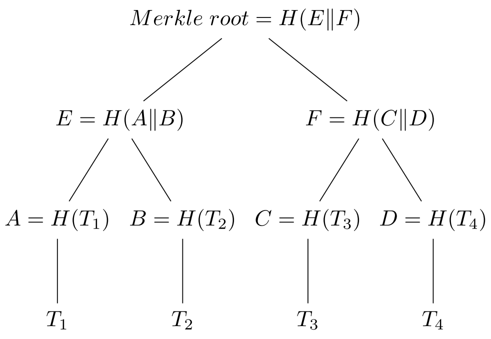

FIGURE 11. Merkle tree of blockchain transactions.

with the appropriate membership witnesses for each item in the set. The witness w is computed by the accumulation coins with the exception of one. In this way, during Zerocoin spend transaction, a user proves the knowledge of one coin by using that witness. This witness w and accumulator A are publicly verifiable without any trusted third party. Accumulator A in Zerocoin is defined as:

$$A = u^{c_1\, c_2\, c_3\, \ldots\, c\, \ldots\, c_n} \mod N,$$

where the integers A, u and N are known to everyone. The coin c is a Pedersen commitment of a coin serial number s and the random number z. Zerocoin uses Random Number Generator (RNG) to generate different s and z to find c using Pedersen commitment until c is prime. The witness w of a coin c is defined as the accumulation of all coins with the exception of c:

$$w = u^{c_1\, c_2\, c_3\, \ldots\, c_n} \mod N.$$

Accumulators can also be employed for range proofs in blockchain. Accumulators are used in [93] to design a stateless blockchain where in order to participate in consensus, the node only needs a constant amount of storage.

I. OBLIVIOUS TRANSFER (OT)

Oblivious Transfer is a two-party protocol between a sender S and a receiver R. The general type of oblivious transfer is k-out-of-n oblivious transfer $\binom{n}{k}$-OT, where $k < n$, in which S holds n messages and R retrieves simultaneously k of them without letting S know about which k out of n messages R received. Oblivious transfer is introduced by Rabin [157] in which a sender sends a message to a receiver with probability $\frac{1}{2}$. The protocol is called as $\frac{1}{2}$-OT, and it is as following:

1) Sender S chooses two large primes p, q and computes $N = pq$ and then the sender generates RSA public key (e, N) such that e is relatively prime to $(p-1)(q-1)$.
2) S computes cipher text c over message M as $c = E_{(e,N)}(M) = M^e \mod N$ and sends e, N, c to receiver R.
3) R chooses a random $x \in \mathbb{Z}_{N^*}$ and sends $a = x^2 \mod N$ to S.
4) S computes four square roots of $a \mod N$ and chooses one of the roots y at random and sends it to R.
5) R checks whether $y^2 \equiv a \mod N$ and if $y \not\equiv \pm x \mod N$, then R will be able to factor N and, hence, be able to decrypt c to recover M.

$\frac{1}{2}$-OT is *complete* for secure multi-party computation. Oblivious transfer has been realized in secure multiparty computation to create private and verifiable smart contracts on blockchain [158]. Oblivious transfer can also be utilized for exchange of secrets, private information retrieval, and building protocols for signing contracts. There has been loads of work done in oblivious transfer, and some of these works have been applied in blockchains such as Searchain [107] and APDB [108] (for automated penalization of data breaches using crypto-augmented smart contracts).

J. OBLIVIOUS RAM (ORAM)

Oblivious RAM is a cryptographic protocol through which a client can safely store his/her data in an untrusted server. The client performs read and write operations remotely. ORAM hides the memory access pattern from the server as well as from outside entities accessing to that part of the data. Therefore, if a client performs two operations of equal length, then the polynomial-bounded adversarial server cannot distinguish between these operations. ORAM bestows freshness, confidentiality of data and integrity so it can be used in various blockchain use-cases and applications. Solidus [105], a protocol for confidential transactions on public blockchain, uses oblivious RAM. Solidus framework operates on a modest number of banks where each bank maintains a large number of user accounts. Solidus introduces a new primitive called Publicly Verifiable Oblivious RAM Machine (PVORM). Most of the previous usage of Oblivious RAM is performed by a single client to outsource storage. In Solidus, ORAM is used to store user account balances and uses PVORM to verify the valid transaction set of a bank. Oblivious RAM is also used in the client-server ORAM protocol [106], Externally Verifiable Oblivious RAM, where Ethereum's automated crypto-currency contracts adjudicate the disputes occurred due to the malicious server by penalizing the server.

K. PROOF OF RETRIEVABILITY (POR)

With the advent of cloud computing, a client might outsource his/her data to the cloud, but still, the client needs a guarantee that the old data has not been modified or deleted. This can be achieved by using proof of retrievability [159] which is an interactive mechanism between a client (verifier) and a server (prover) where the server provides a compact proof to the client that his/her data is intact and he/she can recover the data at any point of time. In this direction, to verify the proof, the client should be equipped with devices having some computational power and network access. This requirement hinders the large-scale adoption of POR by cloud users. To solve this issue, outsource proof of retrievability (OPOR) [160] is introduced where external auditors verify the POR with the cloud provider on behalf of the clients. OPOR protocol specification uses Bitcoin functionalities for the building blocks.

Permacoin [112] uses proof of retrievability. The primary goal of Permacoin is the distributed storage of archival data. As in Bitcoin's mining mechanism, the client continuously invests his/her computational power, and in addition to the computational power, his/her storage is invested. As a consequence, Permacoin requires storage overhead and high bandwidth consumption. To solve these issues, *Retricoin* [113] is proposed to repurpose the mining work in order to ensure the retrievability of a large file at any point of time. Retricoin also proposes a new algorithm for miners to mine collectively. Storj [114] also uses POR to prove the existence of a fresh copy of a shard on the storer side. As a result, POR can be employed in many cryptocurrencies and blockchain applications.

L. POST-QUANTUM CRYPTOGRAPHY

Recent advances in quantum computing pose a severe threat to classical cryptography, as most of the widely used cryptography is based on the hardness of some problem which can be efficiently solved using quantum computers. Thus, research in the Post-Quantum cryptography [161] has taken a massive leap. The security impact of breaking public key cryptography by quantum computers would be tremendous. Elliptic curve cryptography (ECC), which is an approach to public key cryptography, is mostly used in blockchain applications. Using a variant of Shor's algorithm [162], a quantum computer can easily forge an elliptic curve signature that underpins the security of each transaction in blockchain and so breaking of ECC will affect blockchain in terms of broken keys, hence, digital signatures.

Research in this field is in the rise to create Post-Quantum resistant digital signatures (BPQS) [163] which is a hash-based signature and uses one-time signature (OTS) schemes as a building block. OTS does not depend on any number-theoretic hard problem, and it requires only a secure cryptographic hash function, hence, it is not vulnerable to Shor's algorithm. BPQS has advantages like shorter signatures, faster key generations, and customizable property. Post-Quantum cryptography is also used to design Post-Quantum blockchain [109] using one-time signature chains or to create secure crypto-currency based on Post-Quantum blockchain [110].

For the quantum proof solutions, research is now focused on Lattice-based cryptography [164], multivariate cryptography [165], hash-based cryptography [161], and code-based cryptography [166]. Most of the developed primitives within these areas offer either signatures or public keys that are orders of magnitude bigger than the currently used ones, and that is really a hard research challenge that we formulate as:

Research Problem 11: Construct a new blockchain mechanism that has comparably efficient public key addresses and comparably small digital signatures as the currently used ones, but that is based on Post-Quantum cryptographic schemes.

M. LIGHTWEIGHT CRYPTOGRAPHY

Conventional cryptographic methods such as RSA and SHA256, work well on systems having reasonable memory and processing power, but these methods are not suitable for devices constrained with memory, physical size, and battery. Conventional cryptographic methods are challenging to implement in resource-constrained devices due to implementation size, large key size, throughput, speed, and energy consumption. Nevertheless, to solve these issues, lightweight cryptography has evolved. Lightweight cryptography targets sensor networks, embedded systems and other variety of resource-constrained devices such as IoT end nodes and RFID tags. Lightweight cryptography is simpler and faster than conventional cryptography but less secure (suffers from many attacks). In IoT, embedded devices having sensors are interconnected through a public or private network. As these are resource-constrained devices, lightweight cryptography solves the issues of communication, memory, and power consumption, but still lacks security. To provide better security, blockchain can be used in conjunction with the sensor network.

Reference [167] reinforces our point to use lightweight cryptography and blockchain for IoT devices to improve security (confidentiality and integrity of IoT device data). A lightweight scalable blockchain (LSB) [102] is also introduced to improve IoT security and privacy. LSB uses a lightweight hash function and lightweight consensus algorithm in order to achieve scalability, security, and privacy. Blockchain is also used to cater security in electric vehicles, cloud and edge computing [103] which use lightweight cryptographic primitives like lightweight symmetric key encryption.

N. VERIFIABLE RANDOM FUNCTION (VRF)

This cryptographic primitive [168] is a pseudorandom function which gives a public verifiable proof of its output based on public input and private key. In short, it maps inputs to verifiable pseudorandom outputs. VRFs can be used to provide deterministic precommitments which can be revealed later using proofs. VRFs are resistant to pre-image attacks unlike traditional digital signature. VRF is a triple of the following algorithms:

- $KeyGen(r) \rightarrow (VK, SK)$. Key generation algorithm generates verification key VK and secret key SK on random input r.
- $Eval(SK, M) \rightarrow (O, \pi)$. Evaluation algorithm takes secret key SK and message M as input and produces pseudorandom output string O and proof π.
- $Verify(VK, M, O, \pi) \rightarrow 0/1$. Verification algorithm takes input as verification key VK, message M, output string O, and proof π. It outputs 1 if and only if it verifies that O is

the output produced by the evaluation algorithm on input secret key SK and message M, otherwise it outputs 0.

In context of blockchain, many Proof of Stake blockchains use VRF to perform secret cryptographic sortition such that electing leader and committee as part of underlying consensus protocol. Proof of Stake blockchain protocols given in [169] use VRF to elect block proposers and voting committee members. Algorand [37] and Witnet network protocol [170] also employ VRF to conduct secret cryptographic sortition. Ouroboros Praos [121] uses VRF on current timestamp and nonce to determine whether a participant is eligible to issue a block. Dfinity [122] network is a decentralized cloud computing resource which uses VRF to generate stream of outputs over time. Thus, the usage of verifiable random function brings many advantages to be exploited in blockchain and opportunities for more research.

O. OBFUSCATION

Obfuscation is a way of transforming a program P into a "Black-box" equivalent of the program $Q = O(P)$ so that P and Q give the same output when the given inputs are same. It should be hard to find out the internal logic or structure of the program once it is obfuscated. Obfuscation aims to make reverse engineering difficult by making the program unintelligible while preserving its functionality. Finding a perfect black-box obfuscation is mathematically impossible. Along these lines, a weaker solution is to find an "Indistinguishability Obfuscation" so that one cannot determine whether the generated output is from the original program or the obfuscated program. A very simplified example for understanding the Indistinguishability Obfuscation, is the following: There are two equivalent programs $P = x * (y + z)$ and $P' = x * y + x * z$. They are obfuscated such that we have $O(P)$ and $O'(P')$. We say that the obfuscated programs O and O' are indistinguishable if on a received output o one cannot determine which of the programs O, O' gave that output.

Obfuscation can be applied for witness encryption, functional encryption, and restricted use of software. It can be applied in blockchain to turn smart contract into a black-box. An obfuscated smart contract can also possess a secret key to decrypt an encrypted input to the smart contract. As a result, publicly running contracts can possess secret data inside it by obfuscating the smart contract. Figure 12 depicts an obfuscated smart contract which stores the private key corresponding to a public key which is used to encrypt the transaction data. It is hard to get the corresponding private key because of the obfuscated smart contract.

One of the very first successful attempts to offer a very limited variant of obfuscation in Bitcoin was the standardization of the "Pay to script hash (P2SH) transactions" [125]. The amounts of Bitcoins in P2SH transactions are sent to a script hash instead of a public key hash. We say that it was a limited variant of obfuscation because in order to spend Bitcoins received via P2SH, the recipient must provide a script that matches the script hash. Still, the successful acceptance of the P2SH transactions without causing a hard fork in Bitcoin

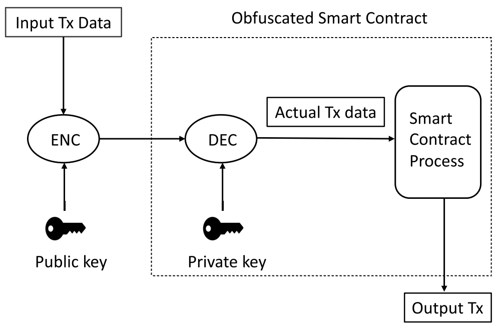

FIGURE 12. An example of smart contract obfuscation.

showed that there is an interest in obfuscation in Blockchain, and that subject is a viable research area.

Research on obfuscation in Bitcoin [104] has been conducted and can be compiled for other cryptocurrencies and blockchain applications. Obfuscation is also used in blockchain for power grid consumption [171] where noise is added to the user's electricity consumption data through obfuscation, and the electricity consumption data is divided into random and non-random obfuscated data.

As noted in [172] the definition and characteristics of some languages determine how easy is to obfuscate programs written in those languages. For example C, C++, Java and Perl are languages that offer easier program obfuscation. What about scripting languages used in Blockchain? We reformulate this question as a research problem:

Research Problem 12: Study the easiness/hardness of obfuscating programs written in the scripting languages used in the current blockchain systems. Study the feasibility of applying some of the developed obfuscation techniques in C, C++, Java and Perl for the blockchain scripting languages.

VI. PROMISING BUT YET NOT EMPLOYED CRYPTOGRAPHIC PRIMITIVES IN BLOCKCHAIN

This Section construes some cryptographic concepts which are promising candidates to be utilized in blockchain. These cryptographic concepts are not yet well-studied and fully applied in blockchain but constitute of some excellent properties which overlap with some desired properties of blockchain. Therefore, some use cases and blockchain services can benefit from these concepts. The included concepts in this Section have either not at all been studied for use in blockchain or have been studied but not implemented yet. We include references which show some initial ideas how to use these concepts in blockchain, but these references do not give any details about concrete implementation.

A. AGGREGATE SIGNATURE

An aggregate signature allows creating a single compact signature from k signatures on k distinct messages from k distinct signers. It provides faster verification as well as reduction in storage and bandwidth. As in blockchain, the requirement of storage and computation is high; aggregate signatures can be used for reduction in storage and computation. Aggregate signatures are the non-trivial generalization of multi-signatures (where all users sign the same message). There are two primary mechanisms of signature aggregation: general and sequential aggregation. In order to describe these mechanisms, assume a set of k users having public-private key pair (PK_i, SK_i) and user i wants to sign message M_i.

1) In general signature aggregation scheme, each user i (from the group of k users) creates signature σ_i on his/her message M_i. Now to create aggregate signature, anyone can run public aggregation algorithm to take all k signatures $\sigma_1, \sigma_2, \ldots, \sigma_k$ and compress them into a single signature σ.
2) In sequential signature aggregation scheme, user 1 signs M_1 to obtain σ_1; user 2 then combines σ_1 and M_2 to obtain σ_2; and so on. The final signature σ is generated by user k which binds M_k and the signature σ_{k-1}. Sequential signature aggregation can only take place during the signing process.

Techniques for aggregating signatures are known for a variety of signature schemes such as DSA, Schnorr, pairing-based, and lattice-based. Aggregate signature schemes should restrict any adversary from creating a valid aggregate signature on his/her own. Aggregate signatures have been proposed for Bitcoin [94], and they can be applied to other cryptocurrencies and blockchain designs.

Research Problem 13: Construct an efficient new signature scheme based on aggregate signatures, that is specifically tailored for blockchain transactions.

B. IDENTITY-BASED ENCRYPTION (IBE)

Identity-Based Encryption first proposed as idea in [173] and later realized as complete cryptographic primitive in [174], allows the encrypting party to use any known (or supposedly known) identity of any receiving party as its public key. Upon receiving the encrypted message, the receiving party asks a trusted third party "Private Key Generator (PKG)" to generate the corresponding private key. Then the receiver decrypts the message using the private key received by PKG. Nowadays, by using identity-based encryption, public keys can be generated using the social identities (Facebook, Twitter, LinkedIn).

There are many flavors and extensions of IBE such as Hierarchical IBE [175], Attribute-based encryption [176], Decentralized attribute-based encryption [177], Functional encryption [178] to name a few.

One of the specifics of IBE is that it replaced the role of the Public-Key Infrastructure with the trusted third party PKG. The presence of a trusted third party somehow defeats the purpose to use it in permissionless blockchain, but still there is a scope to use it in the distributed ledger. Namely, it seems that IBE can be used in permissioned blockchain network. In permissioned blockchain a consortium of trusted third parties that distribute the private keys to the users can take the role to be IBE PKG. Another variant could be a smart contract layer being responsible for the generation of public-private key pairs inside the PKG using IBE.

We identified that the use of IBE within blockchain has started in [100] as well as in supply chain management [101]. Still, there are a lot of challenges and opportunities for other blockchain applications and services.

Research Problem 14: Construct an IBE based (or IBE related) permissioned blockchain network.

C. VERIFIABLE DELAY FUNCTION (VDF)

Verifiable Delay Function (VDF) is a function $f : X \rightarrow Y$ which takes a prescribed number of sequential steps to compute; however, the output can be easily verifiable by anyone. This delay function prevents malicious miners from computing the random output, and it also provides a short proof which is used during the verification of the output along with previously generated public parameters. Boneh et al. described the concept of VDF [179] as well as illustrated the idea about how it can be applicable to blockchain. VDF can be efficiently used as a way to add a delay in decentralized applications. VDF can be used in the application of decentralized systems such as in leader election process of consensus mechanisms, constructing randomness beacons and proofs of replication.

Delay function was initially implemented in Ethereum prototype [180] where the main idea was verification of delay functions through smart contract by using a multi-round protocol. After this prototype implementation, the concept of verifiable delay function was proposed by Boneh et al. Nowadays several blockchain industries are trying to use VDF in their consensus mechanisms. Chia Network [120] which is open source blockchain is trying to use VDF in its "Proof of space and time" consensus mechanism. Ethereum is also trying to develop a pseudorandom number generator using VDF. In this way, VDF brings opportunities to dig deeper and to be applied in the blockchain domain.

Research Problem 15: *[181]:* Finding a post-quantum secure simple VDF for the use of blockchain.

D. PRIVATE INFORMATION RETRIEVAL (PIR)

It is a cryptographic primitive in which a client queries to a server and retrieves the corresponding response from the server without exposing query terms as well as response. It is a weaker version of 1-out-of-n oblivious transfer. It can facilitate private blockchain queries to fetch transaction data privately from blockchain. Accordingly, it can be used to find out whether a particular transaction has been appended in the blockchain or can be used to check the transactions associated with the set of public keys and find out the remaining balances. In addition, PIR can be helpful to query transaction data in simplified payment verification (SPV) clients without compromising privacy. PIR requires an adequate amount of processing, but in the future there might be efficient PIR techniques which can be implemented in blockchain. PIR has also been applied in distributed storage [182] which can be further investigated and adopted in blockchain.

Paper [85] sets several research problems in the area of blockchain transactions privacy and private information retrieval. We rephrase some of the research challenges postulated there:

Research Problem 16: *[85]:* Develop protocols where non-anonymous users can publish transactions that cannot be linked to their network addresses or to their other transactions.

Research Problem 17: *[85]:* Develop protocols where non-anonymous users can fetch details of specific transactions without revealing which transactions they seek.

Research Problem 18: *[85]:* Develop efficient and scalable protocols for anonymous publishing on permissioned blockchains, by combining the asynchronous Byzantine-tolerant consensus protocols for agreeing on transactions with the process of mixing users' announcements.

E. DECENTRALIZED AUTHORIZATION

Authorization and/or hiding sensitive data and actions are essential concepts of resource sharing in open and collaborative environments such as the Internet. Furthermore, in a decentralized form of authorization, parties have full control over their resources and authority to delegate it whether entirely or in part to other parties. An authorization system should provide only as little access to the users as possible to perform their jobs.

Traditional access control is a centralized authorization server which imposes a problem of single point of failure. The centralized authorization scheme has different methods of authorization such as access control list or role-based access.

In comparison, decentralized authorization is more efficient and easier in terms of time, resource and quality. A decentralized authorization system should be well administrated to give access privileges to the users. On the negative side, having in mind that the auditing is also a key component of authorization, in a decentralized manner, it is hard to efficiently implement it and to enforce it.

By using blockchain smart contract, some decentralized authorization systems have been designed, e.g., BlendCAC [97] and WAVE [98]. WAVE introduces an authorization layer for the name spaces and resources. Moreover, for the outside entities, a delegation of trust is used to obtain permission on a resource. Decentralized authorization and blockchain can be used to grow each other by combining one another in a specific way.

Research Problem 19: Construct a decentralized authorization protocol for permissioned blockchain that will provide access privileges as well as a delegation of these access to the users.

F. WHITE-BOX CRYPTOGRAPHY

White-box attack is a threat model where the attacker has full visibility of the internal data flow and can modify the data and code. White-box cryptography [183] aims to address the challenge of implementing a cryptographic algorithm in software in such a way that cryptographic assets remain secure even when subject to white-box attacks. A white-box cryptographic implementation must resist black-box (the attacker has access to only input and output of algorithm), grey-box (side-channel), and also white-box attacks. White-box cryptography is a way to implement cryptographic algorithms like RSA and AES so that the keys remain hidden all the time even during the execution. In some white-box implementations, the key is baked into the code and further concealed to use it in a cryptographic algorithm. In blockchain, it can be used to hide the private key inside the smart contract, and that key can be unlocked when smart contract executes and further it can be used to create a signature.

White-box cryptography can be orchestrated in blockchain to establish trust and privacy of assets. As in blockchain, key and seed secrets are a single point of compromise; these are the highly vulnerable and lucrative targets when stored in memory. To safely store the key, it can be obfuscated in white-box cryptography and further used for encryption/decryption. The implementation of white-box cryptography should be strong enough to facilitate the key storage in blockchain. It has been used in runtime self-protection in a trusted blockchain-inspired ledger [123] and can be promoted in other blockchain applications and services.

G. INCREMENTAL CRYPTOGRAPHY

The idea behind incremental cryptography [184] is if there is a modification to some document M to M', then the time to update the result upon modification of M should be "proportional" to the "amount of modification" done to M. Incremental cryptography can be used in incremental collision free-hashing or incremental digital signature. The initial idea proposed for incremental cryptography uses the example of a digital signature. The idea was to have a digital signature which is easy to update upon the modification of the underlying message. Suppose M is a message and σ is the corresponding signature. If M is changed to M' by adding/deleting any block, then the time to update the signature from σ to σ' should be "proportional" to the "amount of modification" done to get M' from M.

A proposal for construction of an incremental hash function based on SHA-3 is given in [185], and a private blockchain "Kadena" [99] proposes the use of either Merkle tree or incremental hashing for transaction verification. The concept of incremental hashing in Kadena blockchain is to update the distributed log among the blockchain nodes.

Research Problem 20: Construct a new blockchain mechanism that uses an incremental hash function for updates of the distributed ledger.

H. IDENTITY-BASED BROADCAST ENCRYPTION (IBBE)

IBBE scheme [186] can be considered as a generalization of identity-based encryption scheme (Section VI-B) where instead of having one receiver, there are multiple receivers. In broadcast encryption the users are recognized by their identities rather than by their public keys. In a multi-receiver setting, IBBE proves as a powerful method to provide data security and privacy. In this scheme, a sender broadcasts the encrypted message to an intended set of users called privilege set. There can be many privilege sets with different cardinalities. A revocable IBEE scheme [187] shows a scenario of IBEE in which the involved players are the key authority, revoked and non-revoked users. In this setting, the decryption key is updated through the release of a key update material by the key authority. These decryption keys are updated only for the non-revoked users. In this scheme, a membership is revoked for a user if he/she is found malicious or his/her keys are compromised. This RIBBE scheme is further implemented in Charm framework [188].

As blockchain is a multi-receiver setting, IBBE can be a propitious candidate to provide transaction data security and privacy. It can also be used in a permissioned blockchain to certify blocks of membership operation logs. RIBBE scheme as being very efficient in terms of computational complexity and communication can work efficiently as well in the case of blockchain.

Research Problem 21: Develop protocols to certify the blocks of membership operation logs in permissioned blockchain setting.

I. OTHER TECHNIQUES

1) *Message Authentication Code (MAC)*: It is a short piece of information (known as a tag) to authenticate a message which states that the message comes from the stated sender and it has not been changed. It can be used in blockchain to provide integrity of smart contracts or network data. A blockchain-based system for secure mutual authentication (BSeIn) [189] uses MAC for the authentication.
2) *Non-Interactive Witness Indistinguishability (NIWI)*: These are proof systems which are weaker variants of Non-Interactive zero-knowledge (NIZK) proofs. Witness Indistinguishable property states that the verifier cannot distinguish which witness is used to prove the statement by the prover, considering the case of existence of many witnesses. NIWI has been used to construct NIZK over POS based blockchain protocol [190] as well as recently, a new construction of publicly verifiable NIWI proofs from blockchain [191] is also proposed. Hence NIWI proofs bring a new direction to be exploit within the blockchain domain.
3) *Position-based Cryptography*: In this cryptographic protocol [192], the identity or the credentials of a party are derived from his/her geographical location. These credentials can be further used for position-based secure communication and position-based authentication. Position-based cryptography has not been applied in blockchain yet, but it looks promising.
4) *Elliptic Curve Diffie-Hellman Merkle (ECDHM) addresses*: These addresses [193] can be used to exchange messages privately in the blockchain. It can be used in blockchain for secure communication among parties. ECDHM address is shared between the sender and the receiver as secret shares, and they use this shared secret to derive anonymous transacting addresses of each other. This address may only be exposed once they have the share to construct these addresses. In this way, it can be used for the privacy of transaction data.
5) *Verifiable Secret Shuffle*: It is a variant of a zero-knowledge proofs (an honest-verifier zeroknowledge) proposed in [194]. An initial application of verifiable shuffles has been proposed as a mixing service for Ethereum [195].

VII. CONCLUSION

The goal of this work was to offer a systematic study of available cryptographic concepts and to identify different research directions and problems. Based on these reviewed concepts and associated properties, we hope that the paper will help cryptographers interested in blockchain to choose a challenging research problem and for practitioners to choose a suitable concept for their particular use case.

Current transitions to blockchain enabled solutions by different industries give rise to more research on this technology. Academic and industrial research is focused on making blockchain cost efficient in terms of computational power, memory requirements and security. Many existing cryptographic concepts have been embraced for blockchain use. This paper systematizes the current state-of-the-art knowledge of existing cryptographic concepts used in the blockchain. It also gives a brief description of the used cryptographic concept and points to the available blockchain models that are using that concept. The paper also identifies some concepts which have not yet been used in blockchain but can be beneficial if applied in the blockchain. Apart from existing cryptographic concepts, the paper also presents the basic building blocks of blockchain and how these building blocks are dependent on each other.

Table 5 summarizes all of the cryptographic concepts (used or with potentials to be used in blockchain) presented in this work.

REFERENCES

[1] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] CoinMarketCap. (May 2019). *Total Market Capitalization*. Accessed: Jun. 16, 2019. [Online]. Available: https://coinmarketcap.com/charts/
[3] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Boston, MA, USA: Springer, 1983, pp. 199–203.
[4] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. Annu. Int. Cryptol. Conf.* Springer, 1992, pp. 139–147.
[5] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TR-684, 1996.
[6] E. Hughes. (1993). *A Cypherpunk's Manifesto*. Accessed: Apr. 18, 2019. [Online]. Available: https://www.activism.net/cypherpunk/manifesto.html

[7] A. Back, *The Hashcash Proof-of-Work Function*, document Draft-Hashcash-back-00, Internet-Draft Created, Jun. 2003.

[8] W. Dai. (1998). *B-Money*. Accessed: Apr. 18, 2019. [Online]. Available: http://www.weidai.com/bmoney.txt

[9] N. Szabo. (2005). *Bit Gold*. Accessed: Apr. 18, 2019. [Online]. Available: https://unenumerated.blogspot.com/2005/12/bit-gold.html

[10] N. Satoshi. (Jul. 2010). *RE: They Want to Delete the Wikipedia Article*. Accessed: Apr. 18, 2019. [Online]. Available: https://bitcointalk.org/index.php?topic=342.msg4508#msg4508

[11] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum, Yellow Paper 1e18248, 2014.

[12] Ether Foundation. (Jan. 2016). *The Ether Denominations are Called Finney, Szabo, and Wei. What/Who are These Named After?* Accessed: Apr. 30, 2019. [Online]. Available: https://ethereum.stackexchange.com/questions/253/

[13] H. Finney. (Mar. 2013). *Bitcoin and Me (Hal Finney)*. Accessed: Apr. 30, 2019. [Online]. Available: https://bitcointalk.org/index.php?topic=155054.0

[14] V. Morabito, *Business Innovation Through Blockchain*. Cham, Switzerland: Springer, 2017.

[15] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.

[16] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," 2018, *arXiv:1805.02707*. [Online]. Available: https://arxiv.org/abs/1805.02707

[17] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *J. Netw. Comput. Appl.*, vol. 127, pp. 43–58, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451830362X

[18] B. Preneel, "The state of cryptographic hash functions," in *School organized by the European Educational Forum*. Berlin, Germany: Springer, 1998, pp. 158–182.

[19] P. Gallagher and A. Director, "Secure hash standard (SHS)," *FIPS PUB*, vol. 180, p. 183, Mar. 1995.

[20] A. Regenscheid, R. Perlner, S.-J. Chang, J. Kelsey, M. Nandi, and S. Paul, "Status report on the first round of the SHA-3 cryptographic hash algorithm competition," Inf. Technol. Lab., Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 7620, 2009.

[21] E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza, and T. Dryja, "Cryptanalysis of curl-P and other attacks on the IOTA cryptocurrency," in *Proc. IACR Cryptol. ePrint Arch.*, 2019, p. 344.

[22] E. Heilman, N. Narula, T. Dryja, and M. Virza, "Iota vulnerability report: Cryptanalysis of the curl hash function enabling practical signature forgery attacks on the iota cryptocurrency," Tech. Rep., 2017.

[23] C. Lee. (2011). *Litecoin*. [Online]. Available: https://litecoin.org

[24] C. Percival, "Stronger key derivation via sequential memory-hard functions," BSDCan, Ottawa, ON, Canada, Tech. Rep., 2009.

[25] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," Netw. Work. Group RFC, Tech. Rep., 1997.

[26] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New Stream Cipher Designs*. Berlin, Germany: Springer, 2008, pp. 84–97.

[27] V. Buterin, *QuarkCoin: Noble Intentions, Wrong Approach*. Nashville, TN, USA: Bitcoin Magazine, Dec. 2013. Accessed: Jun. 3, 2019.

[28] M. S. Turan, R. A. Perlner, L. E. Bassham, W. E. Burr, D. H. Chang, S.-J. Chang, M. J. Dworkin, J. M. Kelsey, S. Paul, and R. C. Peralta, "Status report on the second round of the SHA-3 cryptographic hash algorithm competition," NIST Interagency, Gaithersburg, MD, USA, Tech. Rep. 7764, 2011.

[29] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, and J. Amundsen, "Cryptographic hash function blue midnight wish," in *Proc. 1st Int. Workshop Secur. Commun. Netw.*, May 2009, pp. 1–8.

[30] E. Duffield and D. Diaz. (2018). *Dash: A Payments-Focused Cryptocurrency*. Accessed: Jun. 3, 2019. [Online]. Available: https://github.com/dashpay/dash/wiki/Whitepaper

[31] Open Source Community at Github. (2018). *ProgPoW—A Programmatic Proof of Work*. Accessed: Jun. 3, 2019. [Online]. Available: https://github.com/ifdefelse/ProgPOW

[32] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," 2017, *arXiv:1711.03936*. [Online]. Available: https://arxiv.org/abs/1711.03936

[33] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. NSDI*, 2016, pp. 45–59.

[34] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017, *arXiv:1710.09437*. [Online]. Available: https://arxiv.org/abs/1710.09437

[35] L. Ren, "Proof of stake velocity: Building the social currency of the digital age," White Paper, 2014, pp. 1–13. [Online]. Available: http://reddcoin.com

[36] J. Kwon. (2014). *Tendermint: Consensus Without Mining*. [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf

[37] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Oper. Syst. Princ. (SOSP)*, New York, NY, USA, 2017, pp. 51–68. doi: 10.1145/3132747.3132757.

[38] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov, "A provably secure proof-of-stake blockchain protocol," in *Proc. IACR Cryptol. ePrint Arch.*, 2016, p. 889.

[39] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proc. 1st Workshop Syst. Softw. Trusted Execution (SysTEX)*, 2016, pp. 2:1–2:6. doi: 10.1145/3007788.3007790.

[40] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (PoET)," in *Stabilization, Safety, and Security of Distributed Systems*, P. Spirakis and P. Tsigas, Eds. Cham, Switzerland: Springer, 2017, pp. 282–297.

[41] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake," in *Proc. IACR Cryptol. ePrint Arch.*, 2016, p. 919.

[42] E. Duffield, H. Schinzel, and F. Gutierrez. (2014). Transaction Locking and Masternode Consensus: A Mechanism for Mitigating Double Spending Attacks. CryptoPapers.info. Accessed: Jun. 3, 2019. [Online]. Available: https://cryptopapers.info/assets/pdf/instasend.pdf

[43] Libra Association. (Jun. 2019). *The Libra Blockchain*. Accessed: Jun. 24, 2019. [Online]. Available: https://developers.libra.org/docs/assets/papers/the-libra-blockchain.pdf

[44] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology—EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Germany: Springer, 2015, pp. 281–310.

[45] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2016, pp. 142–157.

[46] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric proof-of-work based on the generalized birthday problem," *Ledger J.*, vol. 2, pp. 1–30, Apr. 2017.

[47] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.

[48] D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," Ripple Labs, San Francisco, CA, USA, White Paper 5, 2014.

[49] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *Proc. ACM Conf. Econ. Comput. (EC)*, New York, NY, USA, 2016, pp. 365–382. doi: 10.1145/2940716.2940773.

[50] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," 2011, *arXiv:1112.4980*. [Online]. Available: https://arxiv.org/abs/1112.4980

[51] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On bitcoin and red balloons," in *Proc. 13th ACM Conf. Electron. Commerce (EC)*, New York, NY, USA, 2012, pp. 56–73. doi: 10.1145/2229012.2229022.

[52] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 459–474.

[53] The Monero Project. (2014). *Monero*. [Online]. Available: https://web.getmonero.org

[54] R. F. A. Britto and D. Schwartz. (2012). *Ripple*. [Online]. Available: https://ripple.com

[55] EOS. IO. (2017). *EOS. IO Technical White Paper*. Accessed: Dec. 18, 2017. [Online]. Available: https://github.com/EOSIO/Documentation

[56] LTO Network. (2014). *Blockchain for Decentralized Workflows*. [Online]. Available: https://www.lto.network

[57] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, New York, NY, USA, 2018, pp. 30:1–30:15.

[58] (2014). *Monax*. [Online]. Available: https://monax.io/

[59] G. Greenspan. (2015). *MultiChain Private Blockchain*. [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf

[60] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Germany: Springer, 2002, pp. 53–65.

[61] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp. (USENIX Secur.)*, Washington, DC, USA, 2015, pp. 129–144. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman

[62] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 375–392.

[63] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004. doi: 10.1145/997150.997156.

[64] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the bitcoin ecosystem," in *Financial Cryptography and Data Security*, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Berlin, Germany: Springer, 2014, pp. 57–71.

[65] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 104–121.

[66] Bitcoin. (2012). *SPV, Simplified Payment Verification*. Accessed: Jun. 8, 2019. [Online]. Available: https://bitcoin.org/en/glossary/simplified-payment-verification

[67] R. Skudnov. (2012). *Bitcoin Clients*. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/47166/Skudnov_Rostislav.pdf

[68] S. Kadhe, J. Chung, and K. Ramchandran, "SeF: A secure fountain architecture for slashing storage costs in blockchains," 2019, *arXiv:1906.12140*. [Online]. Available: https://arxiv.org/abs/1906.12140

[69] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[70] K. Kralevska, D. Gligoroski, R. E. Jensen, and H. Øverby, "Hashtag erasure codes: From theory to practice," *IEEE Trans. Big Data*, vol. 4, no. 4, pp. 516–529, Dec. 2018.

[71] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Aug. 2012.

[72] K. Kralevska, D. Gligoroski, and H. Øverby, "Balanced locally repairable codes," in *Proc. Int. Sym. Turbo Codes Iterative Inf. Process. (ISTC)*, Sep. 2016, pp. 280–284.

[73] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[74] D. Gligoroski, K. Kralevska, R. E. Jensen, and P. Simonsen, "Repair duality with locally repairable and locally regenerating codes," in *Proc. IEEE 15th Int. Conf. Dependable, Autonomic Secure Comput., 15th Int. Conf. Pervasive Intell. Comput., 3rd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Nov. 2017, pp. 979–984.

[75] L. H. White, "The market for cryptocurrencies," *Cato J.*, vol. 35, no. 2, p. 383, 2015.

[76] B. McLannahan, "Bitcoin exchange MT GOX files for bankruptcy protection," *Financial Times*, vol. 28, Feb. 2014.

[77] M. Huillet. (Aug. 2019). *Vitalik Buterin Talks Scalability: Ethereum Blockchain is Almost Full*. [Online]. Available: https://cointelegraph.com/news/vitalik-buterin-talks-scalability-ethereum-blockchain-is-almost-full

[78] Beam Development Team. *Beam*. 2019. [Online]. Available: https://www.beam.mw

[79] T. Rolfe. (Feb. 2019). *Turing Completeness and Smart Contract Security*. [Online]. Available: https://medium.com/kadena-io/turing-completeness-and-smart-contract-security-67e4c41704c

[80] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. Accessed: Jun. 8, 2019. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

[81] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2016, pp. 17–30. doi: 10.1145/2976749.2978389.

[82] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," White Paper, 2017, pp. 1–47. [Online]. Available: http://plasma.io

[83] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, and J. Timón, and P. Wuille. (2014). *Enabling Blockchain Innovations With Pegged Sidechains*. [Online]. Available: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains

[84] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of bitcoin micropayment channel networks," *Roy. Soc. Open Sci.*, vol. 5, no. 8, 2018, Art. no. 180089.

[85] R. Henry, A. Herzberg, and A. Kate, "Blockchain access privacy: Challenges and directions," *IEEE Security Privacy*, vol. 16, no. 4, pp. 38–45, Jul./Aug. 2018.

[86] C. Egger, P. Moreno-Sanchez, and M. Maffei, "Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks," in *Proc. Cryptol. ePrint Arch.*, 2019, pp. 1–27. [Online]. Available: https://eprint.iacr.org/2019/583

[87] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. NDSS*, 2019, pp. 1–30.

[88] M. Dong, Q. Liang, X. Li, and J. Liu, "Celer network: Bring Internet scale to every blockchain," 2018, *arXiv:1810.00037*. [Online]. Available: https://arxiv.org/abs/1810.00037

[89] N. Kshetri, "5G in E-commerce activities," *IEEE IT Prof.*, vol. 20, no. 4, pp. 73–77, Jul. 2018.

[90] R. H. N. J. Dewey and R. Plasencia, "Blockchain and 5G-enabled Internet of Things (IoT) will redefine supply chains and trade finance," in *Proc. Secured Lender*, Jan/Feb. 2018, pp. 43–45.

[91] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1748

[92] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*, Á. Rocha, M. Serrhini, and C. Felgueiras, Eds. Cham, Switzerland: Springer, 2017, pp. 523–533.

[93] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to IOPs and stateless blockchains," Cryptol. ePrint Arch., Tech. Rep. 2018/1188, 2018.

[94] Y. Zhao, "Aggregation of gamma-signatures and applications to bitcoin," Cryptol. ePrint Arch., Tech. Rep. 2018/414, 2018. [Online]. Available: https://eprint.iacr.org/2018/414

[95] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 315–334. [Online]. Available: https://ieeecomputersociety.org/10.1109/SP.2018.00020

[96] G. Maxwell and A. Poelstra. (2015). *Borromean Ring Signatures*. Accessed: Jun. 8, 2019. [Online]. Available: https://raw.githubusercontent.com/Blockstream/borromean_paper/master/borromean_draft_0.01_34241bb.pdf

[97] R. Xu, Y. Chen, E. Blasch, and G. Chen, "BlendCAC: A blockchain-enabled decentralized capability-based access control for iots," 2018, *arXiv:1804.09267*. [Online]. Available: https://arxiv.org/abs/1804.09267

[98] M. P. Andersen, J. Kolb, K. Chen, G. Fierro, D. E. Culler, and R. A. Popa, "Wave: A decentralized authorization system for iot via blockchain smart contracts," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2017-234, Dec. 2017. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-234.html

[99] W. Martino, "Kadena: The first scalable, high performance private blockchain," Kadena, Okinawa, Japan, Tech. Rep., 2016.

[100] S. Wei, S. Li, P. Liu, and M. Liu, "BAVP: Blockchain-based access verification protocol in LEO constellation using IBE keys," *Secur. Commun. Netw.*, vol. 2018, pp. 1–14, May 2018.

[101] S. Bose, M. Raikwar, D. Mukhopadhyay, A. Chattopadhyay, and K.-Y. Lam, "BLIC: A blockchain protocol for manufacturing and supply chain management of ICS," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul./Aug. 2018, pp. 1326–1335.

[102] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: A lightweight scalable blockchain for IoT security and privacy," 2017, *arXiv:1712.02969*. [Online]. Available: https://arxiv.org/abs/1712.02969

[103] H. Liu, Y. Zhang, and T. Yang, ''Blockchain-enabled security in electric vehicles cloud and edge computing,'' *IEEE Netw.*, vol. 32, no. 3, pp. 78–83, May 2018.
[104] A. Narayanan and M. Möser, ''Obfuscation in bitcoin: Techniques and politics,'' 2017, *arXiv:1706.05432*. [Online]. Available: https://arxiv.org/abs/1706.05432
[105] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi, ''Solidus: Confidential distributed ledger transactions via PVORM,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2017, pp. 701–717. doi: 10.1145/3133956.3134010.
[106] J. Gancher, A. Groce, and A. Ledger, ''Externally verifiable oblivious ram,'' *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 2, pp. 149–171, 2017. [Online]. Available: https://content.sciendo.com/view/journals/popets/2017/2/article-p149.xml
[107] P. Jiang, F. Guo, K. Liang, J. Lai, and Q. Wen, ''Searchain: Blockchain-based private keyword search in decentralized storage,'' *Future Gener. Comput. Syst.*, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17318630
[108] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, ''Automated penalization of data breaches using crypto-augmented smart contracts,'' Cryptol. ePrint Arch., Tech. Rep. 2018/1050, 2018. [Online]. Available: https://eprint.iacr.org/2018/1050
[109] W. van der Linde, P. Schwabe, A. Hülsing, and Y. Yarom, ''Post-quantum blockchain using one-time signature chains,'' Radboud Univ., Nijmegen, The Netherlands, Tech. Rep., 2018.
[110] Y.-L. Gao, X.-B. Chen, Y.-L. Chen, Y. Sun, X.-X. Niu, and Y.-X. Yang, ''A secure cryptocurrency scheme based on post-quantum blockchain,'' *IEEE Access*, vol. 6, pp. 27205–27213, 2018.
[111] D. Aggarwal, G. K. Brennen, T. Lee, M. Santha, and M. Tomamichel, ''Quantum attacks on bitcoin, and how to protect against them,'' 2017, *arXiv:1710.10377*. [Online]. Available: https://arxiv.org/abs/1710.10377
[112] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, ''Permacoin: Repurposing bitcoin work for data preservation,'' in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2014, pp. 475–490. [Online]. Available: https://ieeecomputersociety.org/10.1109/SP.2014.37
[113] B. Sengupta, S. Bag, S. Ruj, and K. Sakurai, ''Retricoin: Bitcoin based on compact proofs of retrievability,'' in *Proc. 17th Int. Conf. Distrib. Comput. Netw. (ICDCN)*, New York, NY, USA, 2016, pp. 14:1–14:10. doi: 10.1145/2833312.2833317.
[114] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, ''Storj a peer-to-peer cloud storage network,'' Storj Labs, Atlanta, GA, USA, Tech. Rep., 2014.
[115] S. Bartolucci, P. Bernat, and D. Joseph, ''SHARVOT: Secret SHARE-based VOTing on the blockchain,'' 2018, *arXiv:1803.04861*. [Online]. Available: https://arxiv.org/abs/1803.04861
[116] (2018). *Wanchain*. [Online]. Available: https://www.wanchain.org
[117] G. Zyskind, O. Nathan, and A. Pentland, ''Enigma: Decentralized computation platform with guaranteed privacy,'' 2015, *arXiv:1506.03471*. [Online]. Available: https://arxiv.org/abs/1506.03471
[118] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, ''Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,'' in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858.
[119] N. van Saberhagen. (2013). *Cryptonote*. [Online]. Available: https://cryptonote.org/whitepaper.pdf
[120] B. Cohen. (2017). *Chia Network*. [Online]. Available: https://www.chia.net
[121] B. David, P. Gaži, A. Kiayias, and A. Russell, ''Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain,'' in *Advances in Cryptology—EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham, Switzerland: Springer, 2018, pp. 66–98.
[122] T. Hanke, M. Movahedi, and D. Williams, ''DFINITY technology overview series, consensus system,'' 2018, *arXiv:1805.04548*. [Online]. Available: https://arxiv.org/abs/1805.04548
[123] C. Liem, E. AbdAllah, C. Okoye, J. O'Connor, M. S. Ul Alam, and S. Janes, ''Runtime self-protection in a trusted blockchain-inspired ledger,'' in *Proc. ESCAR Eur.*, Nov. 2017, pp. 1–10.
[124] I. Miers, C. Garman, M. Green, and A. D. Rubin, ''Zerocoin: Anonymous distributed e-cash from bitcoin,'' in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 397–411.
[125] Bitcoin. (2012). *Pay to Script Hash*. Accessed: Jun. 8, 2019. [Online]. Available: https://en.bitcoin.it/wiki/Pay_to_script_hash
[126] C. Coverdale. (2018). *Scaling Bitcoin: Schnorr Signatures*. [Online]. Available: https://bitcointechtalk.com/scaling-bitcoin-schnorr-signatures-abe3b5c275d1
[127] F. Charlon. *Openchain*. [Online]. Available: https://www.openchain.org/
[128] D. Boneh, M. Drijvers, and G. Neven, ''Compact multi-signatures for smaller blockchains,'' in *Advances in Cryptology—ASIACRYPT 2018*, T. Peyrin and S. Galbraith, Eds. Cham, Switzerland: Springer, 2018, pp. 435–464.
[129] D. Chaum, *Blind Signature System*. Boston, MA, USA: Springer, 1984, p. 153.
[130] L. Valenta and B. Rowan, ''Blindcoin: Blinded, accountable mixes for bitcoin,'' in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Germany: Springer, 2015, pp. 112–126.
[131] E. Heilman, F. Baldimtsi, and S. Goldberg, ''Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions,'' in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 43–60.
[132] F. Zhang and K. Kim, ''Id-based blind signature and ring signature from pairings,'' in *Advances in Cryptology—ASIACRYPT 2002*, Y. Zheng, Ed. Berlin, Germany: Springer, 2002, pp. 533–547.
[133] S. Meiklejohn and R. Mercer, ''Möbius: Trustless tumbling for transaction privacy,'' *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 2, pp. 105–121, 2018.
[134] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, ''CoinParty: Secure multi-party mixing of bitcoins,'' in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, New York, NY, USA, 2015, pp. 75–86.
[135] O. Shlomovits and I. A. Seres, ''ShareLock: Mixing for cryptocurrencies from multiparty ECDSA,'' Cryptol. ePrint Arch., Tech. Rep. 2019/563, 2019. [Online]. Available: https://eprint.iacr.org/2019/563
[136] D. Johnson, A. Menezes, and S. Vanstone, ''The elliptic curve digital signature algorithm (ECDSA),'' *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001. doi: 10.1007/s102070100002.
[137] S. Josefsson and I. Liusvaara, *Edwards-Curve Digital Signature Algorithm (EDDSA)*, document RFC 8032, Internet Research Task Force, Crypto Forum Research Group, 2017.
[138] B. Dale. (Aug. 2019). *The Vaul is Back: Coder Revives Plan to Shield Bitcoin Wallets From Theft*. [Online]. Available: https://www.coindesk.com/the-vault-is-back-bitcoin-coder-to-revive-plan-to-shield-wallets-from-theft
[139] O. Goldreich and Y. Oren, ''Definitions and properties of zero-knowledge proof systems,'' *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, Dec. 1994. doi: 10.1007/BF00195207.
[140] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, ''Succinct non-interactive zero knowledge for a von neumann architecture,'' in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, San Diego, CA, USA, 2014, pp. 781–796. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson
[141] J. P. Morgan. (2016). *Quorum*. [Online]. Available: https://github.com/jpmorganchase/quorum
[142] R. S. Sandhu and P. Samarati, ''Access control: Principle and practice,'' *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, Sep. 1994.
[143] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, ''MedRec: Using blockchain for medical data access and permission management,'' in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.
[144] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, ''Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control,'' *J. Med. Syst.*, vol. 40, no. 10, p. 218, Aug. 2016. doi: 10.1007/s10916-016-0574-6.
[145] A. Outchakoucht, J. P. Leroy, and H. Es-Samaali, ''Dynamic access control policy based on blockchain and machine learning for the Internet of Things,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 7, pp. 417–424, 2017.
[146] S. H. Hashemi, F. Faghri, and R. H. Campbell, ''Decentralized user-centric access control using pubsub over blockchain,'' 2017, *arXiv:1710.00110*. [Online]. Available: https://arxiv.org/abs/1710.00110
[147] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, ''Blockchain for IoT security and privacy: The case study of a smart home,'' in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 618–623.
[148] S. Tahir and M. Rajarajan, ''Privacy-preserving searchable encryption framework for permissioned blockchain networks,'' in *Proc. IEEE Proc. iThings, GreenCom, CPSCom SmartData*, Jul./Aug. 2018, pp. 1628–1633.
[149] D. J. Bernstein. (2014). *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. [Online]. Available: https://competitions.cr.yp.to/caesar.html

[150] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, ''Towards better availability and accountability for IoT updates by means of a blockchain,'' in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS PW)*, Apr. 2017, pp. 50–58.

[151] M. Luongo and C. Pon, ''The keep network: A privacy layer for public blockchains,'' Keep Netw., Tech. Rep., 2018. [Online]. Available: https://keep.network/whitepaper

[152] A. Shamir, ''How to share a secret,'' *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. doi: 10.1145/359168.359176.

[153] R. K. Raman and L. R. Varshney, ''Distributed storage meets secret sharing on the blockchain,'' in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–6.

[154] B. F. França, ''Homomorphic mini-blockchain scheme,'' Tech. Rep., 2015.

[155] T. P. Pedersen, ''Non-interactive and information-theoretic secure verifiable secret sharing,'' in *Advances in Cryptology—CRYPTO'91*, J. Feigenbaum, Ed. Berlin, Germany: Springer, 1992, pp. 129–140.

[156] T. Ruffing and G. Malavolta, ''Switch commitments: A safety switch for confidential transactions,'' in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 170–181.

[157] M. O. Rabin, ''How to exchange secrets with oblivious transfer,'' in *Proc. IACR Cryptol. ePrint Arch.*, 2005, p. 187.

[158] D. C. Sánchez, ''Raziel: Private and verifiable smart contracts on blockchains,'' 2018, *arXiv:1807.09484*. [Online]. Available: https://arxiv.org/abs/1807.09484

[159] A. Juels and B. S. Kaliski, Jr., ''PORs: Proofs of retrievability for large files,'' in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2007, pp. 584–597. doi: 10.1145/1315245.1315317.

[160] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, ''Outsourced proofs of retrievability,'' in *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2014, pp. 831–843. doi: 10.1145/2660267.2660310.

[161] D. J. Bernstein, *Introduction to Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 1–14.

[162] A. Ekert and R. Jozsa, ''Quantum computation and shor's factoring algorithm,'' *Rev. Mod. Phys.*, vol. 68, no. 3, p. 733, 1996.

[163] K. Chalkias, J. Brown, M. Hearn, T. Lillehagen, I. Nitto, and T. Schroeter, ''Blockchained post-quantum signatures,'' in *Proc. IACR Cryptol. ePrint Arch.*, 2018, p. 658.

[164] O. Regev, ''Lattice-based cryptography,'' in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2006, pp. 131–141.

[165] J. Ding and B.-Y. Yang, ''Multivariate public key cryptography,'' in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 193–241.

[166] R. Overbeck and N. Sendrier, ''Code-based cryptography,'' in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 95–145.

[167] M. A. Khan and K. Salah, ''IoT security: Review, blockchain solutions, and open challenges,'' *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17315765

[168] S. Micali, M. Rabin, and S. Vadhan, ''Verifiable random functions,'' in *Proc. 40th Annu. Symp. Found. Comput. Sci.*, Oct. 1999, pp. 120–130.

[169] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, ''Securing proof-of-stake blockchain protocols,'' in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham, Switzerland: Springer, 2017, pp. 297–315.

[170] A. S. de Pedro, D. Levi, and L. I. Cuende, ''Witnet: A decentralized oracle network protocol,'' 2017, *arXiv:1711.09756*. [Online]. Available: https://arxiv.org/abs/1711.09756

[171] Z. Guan, G. Si, X. Zhang, L. Wu, N. Guizani, X. Du, and Y. Ma, ''Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities,'' *IEEE Commun. Mag.*, vol. 56, no. 7, pp. 82–88, Jul. 2018.

[172] A. Binstock. (2003). *Obfuscation: Cloaking Your Code From Prying Eyes*. [Online]. Available: https://web.archive.org/web/20080420165109/ and http://www.devx.com/microsoftISV/Article/11351

[173] A. Shamir, ''Identity-based cryptosystems and signature schemes,'' in *Adv. Cryptol.*, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer, 1985, pp. 47–53.

[174] D. Boneh and M. Franklin, ''Identity-based encryption from the Weil pairing,'' in *Advances in Cryptology—CRYPTO 2001*, J. Kilian, Ed. Berlin, Germany: Springer, 2001, pp. 213–229.

[175] D. Boneh, X. Boyen, and E.-J. Goh, ''Hierarchical identity based encryption with constant size ciphertext,'' in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2005, pp. 440–456.

[176] V. Goyal, O. Pandey, A. Sahai, and B. Waters, ''Attribute-based encryption for fine-grained access control of encrypted data,'' in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[177] A. Lewko and B. Waters, ''Decentralizing attribute-based encryption,'' in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2011, pp. 568–588.

[178] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee, ''Functional encryption: New perspectives and lower bounds,'' in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2013, pp. 500–518.

[179] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, ''Verifiable delay functions,'' in *Advances in Cryptology—CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds. Cham, Switzerland: Springer, 2018, pp. 757–788.

[180] B. Bünz, S. Goldfeder, and J. Bonneau, ''Proofs-of-delay and randomness beacons in ethereum,'' in *Proc. IEEE Secur. Privacy Blockchain (IEEE S&B)*, Apr. 2017, pp. 1–11.

[181] D. Boneh, B. Bünz, and B. Fisch, ''A survey of two verifiable delay functions,'' in *Proc. IACR Cryptol. ePrint Arch.*, 2018, p. 712.

[182] S. Kumar, E. Rosnes, and A. G. I. Amat, ''Private information retrieval in distributed storage systems using an arbitrary linear code,'' in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1421–1425.

[183] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, ''White-box cryptography and an AES implementation,'' in *Proc. Int. Workshop Sel. Areas Cryptogr.* Berlin, Germany: Springer, 2002, pp. 250–270.

[184] M. Bellare, O. Goldreich, and S. Goldwasser, ''Incremental cryptography: The case of hashing and signing,'' in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1994, pp. 216–233.

[185] H. Mihajloska, D. Gligoroski, and S. Samardjiska, ''Reviving the idea of incremental cryptography for the zettabyte era use case: Incremental hash functions based on SHA-3,'' in *Proc. Int. Workshop Open Problems Netw. Secur.* Cham, Switzerland: Springer, 2015, pp. 97–111.

[186] C. Delerablée, ''Identity-based broadcast encryption with constant size ciphertexts and private keys,'' in *Advances in Cryptology—ASIACRYPT 2007*, K. Kurosawa, Ed. Berlin, Germany: Springer, 2007, pp. 200–215.

[187] A. Ge and P. Wei, ''Identity-based broadcast encryption with efficient revocation,'' Cryptol. ePrint Arch., Tech. Rep. 2019/038, 2019. [Online]. Available: https://eprint.iacr.org/2019/038

[188] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, ''Charm: A framework for rapidly prototyping cryptosystems,'' *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013. doi: 10.1007/s13389-013-0057-3.

[189] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, ''BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0,'' *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518301619

[190] R. Goyal and V. Goyal, ''Overcoming cryptographic impossibility results using blockchains,'' in *Theory of Cryptography*, Y. Kalai and L. Reyzin, Eds. Cham, Switzerland: Springer, 2017, pp. 529–561.

[191] A. Scafuro, L. Siniscalchi, and I. Visconti, ''Publicly verifiable proofs from blockchains,'' in *Public-Key Cryptography—PKC 2019*. Cham, Switzerland: Springer, 2019, pp. 374–401.

[192] N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky, ''Position based cryptography,'' in *Advances in Cryptology—CRYPTO 2009*, S. Halevi, Ed. Berlin, Germany: Springer, 2009, pp. 391–407.

[193] Notes on Bitcoin Privacy Technology, Open Bitcoin Privacy Project. (2019). *ECDHM Address*. [Online]. Available: http://wiki.openbitcoinprivacyproject.org/topics:ecdhm-address

[194] C. A. Neff, ''A verifiable secret shuffle and its application to e-voting,'' in *Proc. 8th ACM Conf. Comput. Commun. Secur.*, 2001, pp. 116–125.

[195] I. A. Seres, D. A. Nagy, C. Buckland, and P. Burcsi, ''MixEth: Efficient, trustless coin mixing service for ethereum,'' Cryptol. ePrint Arch., Tech. Rep. 2019/341, 2019. [Online]. Available: https://eprint.iacr.org/2019/341

MAYANK RAIKWAR was born in Uttar Pradesh, India, in 1994. He received the B.Tech. degree in computer science and engineering from Uttar Pradesh Technical University, in 2013, and the M.Tech. degree in computer science from the Indian Statistical Institute, India, in 2016. He is currently pursuing the Ph.D. degree with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU), since 2019. In 2017, he joined the Department of Computer Science, Nanyang Technological University, Singapore, as a Research Engineer. His research interests are in cryptography, blockchain, cryptocurrencies, and security.

DANILO GLIGOROSKI was born in Skopje, Republic of Macedonia, in 1967. He received the B.S. and M.S. degrees in applied mathematics from Ss Cyril and Methodius University in Skopje, and the Ph.D. degree in computer science from Ss Cyril and Methodius University in Skopje, in 1997.

From 1997 to 2008, he was an Assistant Professor with the Faculty of Natural Sciences, Skopje University. Since 2008, he has been a Professor of information security and cryptography with the Norwegian University of Science and Technology (NTNU). He is an author of more than 180 scientific publications and more than 10 inventions. His main research interests are in application of various algebraic structures in cryptography, information security, and coding theory.

KATINA KRALEVSKA was born in Skopje, Macedonia, in 1987. She received the B.Sc. and M.Sc. degrees in telecommunications from Ss. Cyril and Methodius University-Skopje, Macedonia, in 2010 and 2012, respectively, and the Ph.D. degree from the Norwegian University of Science and Technology (NTNU), in December 2016.

In 2017, she was a Postdoctoral Researcher with the Department of Information Security and Communication Technology, NTNU. In 2018, she became an Associate Professor with the same department. Since 2019, she has been the Deputy Head of the Department of Information Security and Communication Technology. Her research interests include coding theory, blockchain, and mobile and wireless communications. She is an author of more than 25 scientific publications and more than eight inventions.

Paper B

Meshwork Ledger, its Consensus and Reward Mechanisms

M. Raikwar, D. Gligoroski

Published in 13th International Conference on COMmunication Systems and NETworkS (COMSNETS), IEEE, 2021

The Meshwork Ledger, its Consensus and Reward Mechanisms

Mayank Raikwar*, Danilo Gligoroski*
*Norwegian University of Science and Technology (NTNU) Trondheim, Norway
Email: {mayank.raikwar,danilog}@ntnu.no

***Abstract*—We propose a new blockchain ledger concept called "Meshwork ledger" and a corresponding consensus algorithm. Meshwork is a network of coequal client nodes that contribute to the endorsement of the transactions by providing digital signatures to a validator node that collects them in an aggregate signature scheme. The essential sustainability component of the ledger is the reward mechanism for the meshwork client nodes. The prime design objective is the coequality of all client nodes, meaning there is no advantage for getting rewards if the client is an early adopter, if the client has collected a significant stake of rewards or if the client just joined the meshwork.**

The consensus algorithm is based on aggregate multi-signatures. A joint aggregate signature on a block of transactions is constructed with the signatures collected from the mesh client nodes in a multi-signature scheme. A signature provided on the block by a client node is acknowledged as approval. The core idea of the consensus is to race for the maximum number of signatures (approvals) on a block from the mesh clients, in order to append the block in the blockchain. The race in the consensus is among particular types of nodes called validator nodes that try to collect a maximum number of approvals (signatures) from the mesh clients. Clients that participated in the aggregate signature organized by the winner validator node get some reward as a small share of the total transaction fee. These reward transactions are performed in an off-chain manner, using a commit-chain.

Compared with other blockchain consensus algorithms, the meshwork consensus algorithm is faster, significantly more energy-efficient and scalable.

I. Introduction

Blockchain technology has evolved rapidly in the past decade. As a paradigm, it offers many unique features such as a distributed and trusted ledger of transactions without any need for a trusted third party, immutability of the ledger, pseudo-anonymous and anonymous transactions, smart and secure contracts, to name a few. The peers participating in the blockchain maintain the ledger, and each peer has a local copy of the ledger. These peers collectively adhere to some predetermined set of rules to ensure the consistency of the ledger. The mechanism to determine this set of rules is the core of the blockchain and is known as *consensus mechanism*. The first consensus mechanism proposed in the use of blockchain is Proof of Work (PoW) in Bitcoin [1].

Many consensus mechanisms have been introduced after Bitcoin PoW with their unique functionality. For example, there are numerous mechanisms that use different and complex compositions of cryptographic hash functions: QuarkCoin [2] using a chain of six hash functions, DASH [3] with eleven hash functions denoted as X11, and Verge [4] with even 17 hash functions (X17). The motives for their proposals were to offer PoW consensus protocols that will be fair for all users of the blockchain (not just only to early adopters, nor just to the powerful hardware miners). A general property of those consensus protocols is that they suffer from huge energy and computational power waste.

However, we can say that the short (slightly more than one decade) history of blockchain and cryptocurrencies [5], is a history of failed attempts to construct a sustainable blockchain that will address the issue of fairness by preventing the appearance of powerful hardware miners that can mine the blocks with hash computing rates that are several orders of magnitude higher than the ordinary users that would use just CPUs (and maybe GPUs).

The energy waste problem of PoW was addressed in several other consensus mechanisms such as Proof of Stake (PoS) [6], Proof of Stake Velocity (PoSV) [7], in mechanisms that use verifiable random functions such as in Algorand [8] and in the Secure Proof of Stake (SPoS) [9], in mechanisms that use trusted random functions such as in Proof of Luck (POL) [10], and in Proof of Elapsed Time (POET) [11]. Some of the proposed consensus protocols, to boost their energy efficiency, have also proposed the use of special nodes (master nodes in DASH [3] or validator nodes in Libra [12]).

Needless to say, the solutions addressing the energy problem of the original PoW consensus, did not offer solution for the fairness problem, since the owners of big stakes of coins would have significant advantage to get newly minted coins.

Since our Meshwork ledger heavily relies on aggregate multi-signatures, next, we describe a few works done on multi-signatures and their application to the blockchain. Boneh at el. [13] constructed a multi-signature scheme using BLS signatures, which provides public-key aggregation and security against rogue public-key attack. The scheme supports a fast verification of transactions and reduces the blockchain size. Algorand [8] presented a forward-secure multi-signature consensus scheme Pixel [14]. Pixel signatures reduce the storage, bandwidth, and verification costs in the blockchain. They integrated the Pixel signature with Algorand blockchain and showed a substantial reduction in the block size and the verification time. Another work of a multi-signature scheme in the blockchain is Decisional Diffie-Hellman based construction of multi-signatures [15].

The use of aggregate signature in consensus algorithms got much attention from academia and industry. Theta blockchain ledger protocol [16] adapted the concept of aggregate signatures in the BFT consensus algorithm and introduced a multi-level BFT consensus mechanism. They proposed an aggregated signature gossip protocol to reduce the communication complexity of the consensus. However, we notice that Theta blockchain ledger might suffer from a high signature verification cost when the number of guardian nodes in the system increases as more number of pairing operations are needed. PCHAIN [17] also introduced PDBFT2.0 [18] to reduce the communication complexity and hardware requirements in blockchain consensus using the aggregate signatures, but it also suffers from high signature verification cost. The other related works in this domain are [19], [20], [21], [22], [23], [24]. Another interesting related work is the new Schnorr multi-signature scheme with deterministic signing [25].

A. *Our Contribution*

We propose a new permissioned public blockchain ledger concept called "Meshwork ledger" with two design goals:

1) Its consensus protocol is fair to all ledger members i.e.; there exists a coequality for all client nodes: there is no advantage for getting rewards if the client is an early adopter, if the client has collected a significant stake of rewards, or if the client has just joined the meshwork;
2) Its consensus protocol is energy efficient, and there is no advantage if the client has powerful hardware or a single CPU or MCU just capable to produce digital signatures.

The sustainability of the "Meshwork ledger" is guaranteed by its reward mechanism, which is also designed to be fair to all meshwork client nodes. To achieve the design goals for the consensus algorithm, we use aggregate multi-signatures. The idea is to construct a joint aggregate signature on a block of transactions with the signatures collected from the mesh client nodes in a multi-signature scheme. One single signature on the block of transactions produced by a single client node is acknowledged as approval. Then, the main novel idea in our consensus proposal is to race for the maximum number of signatures (approvals) on a block of transactions from the mesh clients. The race in the consensus is among particular types of nodes called validator nodes that try to collect a maximum number of approvals (signatures) from the mesh clients. Clients that participated in the winning aggregate signature get some reward as a small and equal share of the total transaction fee. These reward transactions are nano-valued transaction. As there are many of these transactions, the transactions can be performed in off-chain manner. We reviewed all the possible off-chain transaction solutions and concluded that the best way to perform all these nano-value transaction for our model in a cost-effective manner is commit-chain [26].

Last but not least, we also provide complexity and security analysis of our consensus protocol. In the security analysis, we enlighten possible attack scenarios in the protocol and the prevention mechanism followed in the consensus protocol. We also conducted a few experiments for the robustness of the system.

II. Preliminaries

1) Bilinear Maps: $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative groups of prime order q with generators g_1, g_2 correspondingly. A map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ has the following properties:

- Bilinearity: $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p : e(u^a, v^b) = e(u, v)^{ab}$;
- Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_t}$.

We say the pair $(\mathbb{G}_1, \mathbb{G}_2)$ is a pair of bilinear groups iff the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$, and the bilinear map e are efficiently computable.

2) Multi-Signature Scheme: For individual transaction/block signing and verification we use the BLS signature scheme [27], and for the multi-signature scheme, we use the aggregate signature scheme proposed by Boneh et al [28]. In our case, all the signers sign the same message in the aggregate signature scheme while ensuring security. BLS signature scheme requires bilinear map e described as above alongwith a full-domain hash function for signing process $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. BLS signature scheme works as follows:

- **KeyGen:** For a user, choose random $sk \xleftarrow{\$} \mathbb{Z}_q$, compute $pk \leftarrow g_2^{sk} \in \mathbb{G}_2$, the user's keypair is (pk, sk).
- **Sign(sk, M):** For a user, given secret key sk and a message $M \in \{0,1\}^*$, signature on M is $\sigma \leftarrow H(M)^{sk} \in \mathbb{G}_1$.
- **Verify(pk, M, σ):** Given a user's public key pk, a message M, accept the signature σ, if $e(\sigma, g_2) = e(H(M), pk)$.

Signature Aggregation: Given n signatures $\sigma_1, \sigma_1, \ldots, \sigma_n$ on message M by n users, the procedure for signature aggregation of n signatures works as: $\sigma \leftarrow \prod_{i=1}^{n} \sigma_i$. The aggregate signature is $\sigma \in \mathbb{G}_1$.

Aggregate Verification: To verify the aggregate signature σ, given the original message M and the n public keys $pk_1, pk_2, \ldots, pk_n$ for all n users, the verifier checks if:

$$e(\sigma, g_2) \stackrel{?}{=} e(H(M), pk_1)e(H(M), pk_2) \ldots e(H(M), pk_n)$$
$$\stackrel{?}{=} e(H(M), \prod_{i=1}^{n} pk_i) \stackrel{?}{=} e(H(M), apk)$$

If the equation holds, the verifier "Accept" the signature, else "Reject". In the above equation, $apk \in \mathbb{G}_2$ and stands for *aggregate public key*.

This simple aggregation scheme suffers from **rogue public-key attack** where an attacker takes the public key pk of an honest user Alice, and constructs his public key pk^* as $g_2^{\alpha} \cdot (pk)^{-1}$ (where $\alpha \xleftarrow{\$} \mathbb{Z}_q$). Then, given a message $M \in \{0,1\}^*$, attacker presents an aggregate signature $\sigma := H(M)^{\alpha} \in \mathbb{G}_1$ claiming that he and Alice, both has signed the message M. However in reality, Alice did not sign the message M but the aggregate verification holds as

$$e(\sigma, g_2) = e(H(M)^{\alpha}, g_2) = e(H(M), g_2^{\alpha})$$
$$= e(H(M), pk \cdot pk^*)$$

Few defense mechanisms [29], [30] against this attack were proposed which require each user to prove the possession of the corresponding secret key (PoP). We also apply the PoP in Meshwork ledger for each party.

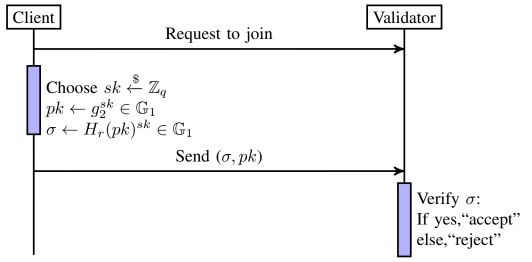
Client
Validator
Request to join
Choose $sk \xleftarrow{\$} \mathbb{Z}_q$
$pk \leftarrow g_2^{sk} \in \mathbb{G}_1$
$\sigma \leftarrow H_r(pk)^{sk} \in \mathbb{G}_1$
Send (σ, pk)
Verify σ:
If yes,"accept"
else,"reject"

Fig. 1. Client Registration Protocol

Why BLS Multi-Signature

- BLS Multi-signatures are non-interactive and easy to follow in nature, therefore, it is easier to perform the signature and key aggregation without any additional round.
- BLS signatures do not rely on random number generator and BLS signatures are single curve points, so its size is two times shorter than Schnorr or ECDSA signature [31].

III. MESHWORK LEDGER

A. *Entities in the Ledger*

In Meshwork ledger model, we have two primary entities client and validator node participating in the blockchain:

1) Client Node: Client nodes are the ones that invest a tiny computational power to sign a block, and for that gets rewarded. A client node can have a few connections with different validator nodes. For registration, client nodes have to go through strong authentication checks to prevent the Sybil attack [32], followed by key registration protocol using PoP with a validator node to prevent the rogue-key attack as depicted in Figure 1. Client nodes also execute transactions with other client and validator nodes in the system.

Client Registration Protocol It is an interactive protocol between a client node and a validator node as depicted in Figure 1. In this process, a client node registers itself by proving the knowledge of its secret key to a validator node by signing its public key. To completely prevent rogue public-key attack, the registration protocol uses a different hash function $H_r : \{0,1\}^* \to \mathbb{G}_2$ for creating signatures over the public key. This hash function can be constructed from hash function H using domain separation. Thus, the client generates a signature using H_r on its public key, and if verified, the validator stores the user's public key. The client node also has a public identifier associated with it, which is a hash of its public key. The specific hash function is global for all client nodes. Thus, rather than sending long public keys, these short identifiers are used for communication.

Note: The Client Registration Protocol is an important component for the overall security of the Meshwork ledger, but it is not an exclusive and non-replaceable component. Namely, our meshwork can use some pre-existing client registration protocols based on a national digital identity [33].

2) Validator Node: Validator nodes are the major players for reaching the consensus in the distributed ledger. Each validator node maintains its ledger of blocks. Validator nodes join the system according to the objective participation criteria, and these nodes have a stake in bootstrapping the blockchain system. Therefore, these nodes have to lock up some minimum amount of stake in the system for a period of time. Each validator node has a pair of public-private keys. The validator nodes publish their public key along with the Proof of Possession (PoP) of the corresponding secret key. Here, the PoP scheme is similar to the PoP used in Figure 1. Validator nodes can also perform transactions in the network. In a consensus round, validator nodes perform signature aggregation on the block, thus validator nodes invest resources towards achieving the consensus. Accordingly, each validator node should be equipped with enough computational power and storage space. There are fewer validator nodes than the client nodes, and each validator node maintains several client node connections. A validator node also has its publicly available list of connected client nodes along with their corresponding public keys and public Identifiers.

Validator Registration Protocol: To join the pool of validator nodes in the system, a new node has to pass a strong authentication check and also give proof about having enough stake and enough storage and computational power.

B. *Assumptions in the Meshwork ledger*

- The network is partially synchronous, which means the message transmission between two directly linked nodes arrives within a specified period.
- The majority of the validator nodes are honest for the security against byzantine fault and the local clock of each validator node is loosely synchronized.
- The blockchain should be account-based and smart-contract enabled e.g., Ethereum [34].

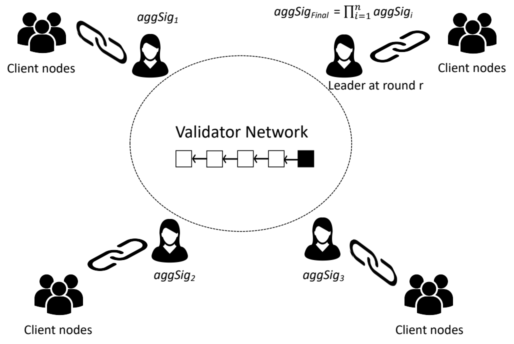

Fig. 2. Overview of Consensus at round r

Bootstrapping the System During the deployment of the system, a common genesis block is given to all the validator nodes. This genesis block specifies a number s denoting the minimum number of signatures required from each validator node during a consensus round. The value s is updated from time to time, depending on several factors including the number of participating entities in the system.

C. Consensus Algorithm

The consensus algorithm involves the active participation of validator nodes as well as client nodes of blockchain. The core idea of the algorithm is to collect the maximum number of signatures on a block from the nodes participating in consensus round r as depicted in Figure 2. The taxonomy of the symbols used in Meshwork ledger is listed in Table I. There are n validator nodes $V_1, V_2, \ldots, V_n$ and each validator node V_i has some client nodes $C_{i1}, C_{i2}, \ldots, C_{ij}$ (where $j \gg n$) connected to it. The detailed algorithm is as follows:

Symbols	Definition
$\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$	Multiplicative groups of order q
g_1, g_2	Generator of group $\mathbb{G}_1/\mathbb{G}_2$ respectively
e	Bilinear map
(PK, SK)	Public and secret key
H, H_r	Hash functions for signature and client key registration
V	Validator node
C	Client node
σ	A BLS signature
s	Number of required signatures from a validator node
$aggSig$	Aggregate signature
$aggPK$	Aggregate public key
B_k	k^{th} Block in the blockchain
τ_{block}	Time to wait to receive the new proposed block
τ_{agg}	Time to wait to receive all the aggregate signatures
$Tolerance$	Upper bound how many times a client node can send the same signature to multiple validator nodes

TABLE I
LIST OF SYMBOLS IN THE MODEL

- In a consensus round r, a validator node V_k (where $k \in [1, n]$) is elected as the leader of the round. The leader node (*block proposer*) collects different transactions, writes the recent value of s and prepares a block B_k. After the block preparation, block proposer V_k sends the block B_k to all validator nodes. Leader V_k is excluded from the task of contacting its client nodes in round r.
- After receiving the block B_k, each validator node V_i checks the minimum number of signatures s required for the block B_k from its side. The validator node V_i randomly selects a client node set $\{C_{im}\}$ (where $m \leq j$ but $m \geq s$). The validator node signs the block and creates a signature σ_{V_i}. The main goal of each validator node V_i is to collect at least s signatures from its connected client nodes.
- Each validator node V_i collects the signatures $\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im}$ on block B_k from its selected client nodes $C_{i1}, C_{i2}, \ldots, C_{im}$. Then V_i verifies all the individual signatures $\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im}$, and further creates aggregate signature $aggSig_i$ from the verified signatures including its own signature σ_{V_i}, and aggregate public key $aggPK_i$ from $PK_{i1}, PK_{i2}, \ldots, PK_{im}$ and PK_{V_i}. Then V_i sends $aggSig_i$, $aggPK_i$ and a list of client public key Identifiers L_i $(ID_{i1}, ID_{i2}, \ldots, ID_{im})$ to the block proposer V_k.
- Each validator node V_i also gives a *Proof of Inclusion* $P_{C_{il}}$ to each of its client nodes C_{il}. This proof corresponds that the client signature σ_{il} (for $l = 1 \ldots m$) has been included in the aggregated signature $aggSig_i$. This proof is a signature over the signature σ_{il} which is $P_{C_{il}} = Sign(sk_{V_i}, \sigma_{il})$ and verifiable using the public key pk_{V_i} of V_i.
- The leader V_k collects the received aggregate signatures $aggSig_1, aggSig_2, \ldots, aggSig_n$, along with aggregate public keys and identifiers from all the validator nodes in a *First come, First served* basis. In the beginning, the leader resets a variable $TotalAggSig$ that keeps track of the number of unique aggregated signatures for that round. The leader also checks the following conditions:
  1) The received aggregate signatures $aggSig_i$ (for $i = 1 \ldots n$, $i \neq k$) are valid.
  2) For each validator node V_i, check whether the total number of public key identifiers is $s_i \geq s$. If so, it updates the variable $TotalAggSig$:
  $$TotalAggSig \leftarrow TotalAggSig + Unique(s_i)$$
  where $Unique(s_i) \subseteq s_i$ is a subset of s_i that have not submitted its signatures to multiple validator nodes.
- The leader is also keeping track of how many aggregate signatures it collected so far:
  1) There should be at least $\frac{2}{3}$ of $(n-1)s$ unique signatures in all the aggregate signatures received by V_k i.e.,
  $$TotalAggSig \geq \tfrac{2}{3}(n-1)s$$
  2) Determine a list $\mathcal{D}$ of clients that submitted two or more signatures to two or more validator nodes.
- If all the above conditions satisfy, the leader V_k constructs final aggregate signature $aggSig$ from signatures $aggSig_1, aggSig_2, \ldots, aggSig_n$ and final aggregate public key $aggPK$ from public keys $aggPK_1, aggPK_2, \ldots, aggPK_n$ (where $n \neq k$). Finally, V_k constructs the final block $B_{round} = (B_k, aggSig, aggPK)$ by appending the final aggregate signature $aggSig$, final aggregate public key $aggPK$ to the proposed block B_k.
- Leader V_k also gives a *Proof of Inclusion* P_i (where $P_i = Sign(sk_{V_k}, aggSig_i)$) of the aggregate signature $aggSig_i$ to node V_i which confirms that the $aggSig_i$ is included in the final aggregate signature $aggSig$.
- Leader V_k attaches the block B_{round} to its blockchain and broadcasts the block B_{round} in the blockchain network. It also sends the list $\mathcal{D}$ to all the validator nodes. After receiving the block B_{round}, each node V_i verifies the block by verifying the final signature $aggSig$ using $aggPK$. If the block verifies, the node V_i attaches the block B_{round} to its blockchain. Each validator node also checks the received list $\mathcal{D}$ and keeps a record for clients that sent the same signatures to multiple validator nodes.

D. Reward System

One of the primary goals of this consensus algorithm is to consume significantly less amount of computational power. The total reward in each consensus round is the sum of the transaction fees associated with the transactions of the block. Then the reward is dispersed among the nodes who participated in the consensus on that block. The validator nodes invest more resources to get the signatures from their client nodes, and to aggregate the client signatures, so the validator nodes get more reward than the client nodes.

In each round, the leader node makes reward transactions to the validator nodes, which are recorded in the blockchain. These reward transactions are further distributed by each validator node to its client nodes that participated in the signing. Hence each validator node creates many client node transactions, and so the total client node transactions created by all the validator nodes are many in numbers.

Let say the total reward in a consensus round r is X, then the distribution of reward X will be as follows:

- Each validator node V_L and its client nodes $C_{L1}, \dots, C_{Lm}$ get the accumulated share of reward as $j = \frac{X}{n}$
- This share j is further distributed among a validator node V_L and its client nodes $C_{L1}, \dots, C_{Lm}$ in following way:
  1) Each validator node V_L gets share as $t * j$.
  2) Each client node C_{Li} gets share as $\frac{(1-t)*j}{m}$ (considering m client nodes connected to V_L participated in round r)

Here t is the reward distribution parameter.

Note: In the line of our design goal of the Meshwork ledger to be fair for early and late adopters that are meshwork clients as well as for the validator nodes, the incentives for validator nodes are also tweakable with the parameter t. This t is recalculated periodically and involves the energy and communication costs spend by the validator nodes. The process of including all client node reward transactions in the main blockchain is a severe bottleneck for the scalability. To tackle this, we adopted the off-chain solution, commit chain [26]. As the reward transactions for the client nodes are nano-value transactions, and the client nodes might not always be online, then to off-loading the transactions in off-chain, commit-chain fits as the viable option.

IV. Consensus in the Meshwork Ledger

Algorithm 1 illustrates the steps involved in the consensus. In this section, we describe all the consensus algorithm functions as $LeaderElection$, $CreateBlock$, $ClientSelection$, $Sign$ and $Verify$ in detail. It also defines other necessary factors of our Meshwork ledger.

A. Leader Election

In each consensus round, a validator node is chosen as a leader by the function $LeaderElection()$. The leader is responsible for the creation and finalization of the block. There are different mechanisms in different PoS blockchains for leader election. Many of the PoS blockchains [35] are using verifiable random function [36] for electing the leader. For our Meshwork ledger, we follow the leader election using an efficient robust round-robin selection technique [37] with some modification. In short, leader candidates are selected according to their age in a round-robin manner. After the candidates' selection, a set of endorsers give a quorum of confirmations for the leader candidates. The one having the majority of confirmation becomes the leader node. In our model, we follow the same idea with some modifications to prevent the malicious

Algorithm 1: Consensus Algorithm

```
Input  : $round, s, \{V_1, V_2, \dots, V_n\}$
Output: $B_{round}$, $\mathcal{D}$
1  $V_k \leftarrow LeaderElection(round, V_1, V_2, \dots, V_n)$;
2  $V_k$ runs $CreateBlock(tx_1, tx_2, \dots, tx_u)$ and output $B_k$;
3  $V_k$ sends block $B_k$ to other validator nodes and waits for
     $\tau_{agg}$ time to receive the required number of signatures;
4  $V_i$ actions:;
5  for each validator node $V_i$ from $V_1, V_2, \dots, V_n$, except $V_k$ do
6      run $ClientSelection(C_{i1}, C_{i2}, \dots, C_{im})$;
7      wait for $\tau_{block}$ time to receive new block $B_k$;
8      if receive $B_k$ then
9          $(aggSig_i, aggPK_i, \{ID_j\}_i) \leftarrow Sign(B_k, s)$;
10         send $(aggSig_i, aggPK_i, \{ID_j\}_i)$ to node $V_k$;
11     end
12 end
13 $V_k$ actions:;
14 $TotalAggSig \leftarrow 0$;
15 for tuples $(aggSig_i, aggPK_i, \{ID_j\}_i)$ received from the
     validator nodes $V_i$ do
16     · $Verify(aggSig_i, aggPK_i, B_k)$;
17     · Check if $s_i \geq s$, where $s_i$ is the number of client
         identifiers from $V_i$;
18     · Determine the number of unique signatures coming
         from $V_i$, $Unique(s_i)$;
19     · $TotalAggSig \leftarrow TotalAggSig + Unique(s_i)$;
20     if $TotalAggSig \geq \frac{2}{3}(n-1)s$ then
21         exit for;
22     end
23 end
24 $V_k$ prepares the final block
     $B_{round} = (B_k, aggSig, aggPK)$, appends it to its
     blockchain and sends to the other validator nodes;
25 $V_k$ also sends list of double signees $\mathcal{D}$ to all $V_i$;
```

leader or leader crash issue. We define a threshold value t_E for the number of endorsements and we select an expected number of leader candidates based on t_E. The value t_E varies and computed for each round. Hence, in our model, we have three sets of validator nodes after the leader election process:

1) *Leader validator node* is the deterministic leader candidate resulted from the robust round-robin technique.
2) *Backup validator nodes* are the validator nodes which passes the threshold criteria t_E of endorsements.
3) *Remaining validator nodes* are the nodes which are either not selected in the robust round-robin technique or did not pass the threshold t_E .

The leader validator node defined as V_k in Section III-C is responsible for proposing a block and collecting the required number of signatures from other validator nodes. Backup validator nodes $\{V_b\}$ and remaining validator nodes participate in consensus to get the aggregate signature on the proposed block by the main leader node. Backup validator nodes are also responsible for the following things:

- If the leader node crashes or if backup validator nodes do not receive any new block within τ_{block} time, then a backup validator node having the next highest quorum of

confirmations becomes the leader, announces itself as a leader, and executes the consensus protocol.
- Backup validator nodes continuously monitor the behavior of the leader node, and the leader is caught if it performs malicious activities. For example, if a leader node is malicious, it can give different blocks to different validator nodes but it will be caught by the backup validator nodes and later it will be penalized in the system.

B. Block Proposal and Client Selection

The leader of the consensus round collects the transactions from the client and validator nodes and prepares the block using $CreateBlock()$ function. The leader also appends value s, the minimum number of signatures required from each validator node, in the block. This current s should be the latest value for the next few consensus rounds. The other validator nodes wait for a definite amount of time for the new block to receive. A validator node might not receive the latest block in a specific amount of time due to some network-related problems like network congestion, etc.

After receiving the new block from the leader node, each validator node runs $ClientSelection()$ function. The strategy under this is to select at least s client nodes to sign the block is *Round Robin With a Reset* strategy. That means every validator node keeps track of how many rounds the clients were waiting to sign some block. Validator picks randomly s out of those clients that had the highest waiting time. Once a client is chosen to sign, its waiting time is reset to 0.

C. Block Signing and Verification

The validator nodes announce the new consensus round to its connected client nodes. The selected client nodes by a validator node sign the block and send it to the validator node. A validator node waits for a certain amount of time to receive at least the minimum number of required signatures from its client nodes. Then finally, the validator node verifies all the client signatures and prepares an aggregate signature from those, along with an aggregate public key using $Sign()$ function and sends the aggregate signature and key, along with the client node public identifiers to the leader node.

The leader node receives the different aggregate signatures along with other details from different validator nodes. The leader node first verifies all the aggregate signatures using the function $Verify()$. Further, the leader node checks whether these aggregate signatures qualify the criteria for the minimum number of required signatures for aggregation. If all the verification conditions meet, then the leader prepares the final block by appending other necessary details to the original block.

D. Race Conditions

The race in the Meshwork ledger is among validator nodes that are racing to collect at least s signatures and to be included in the leader signature aggregation that tries to collect at least $\frac{2}{3}(n-1)s$ unique signatures. We find that in comparison with the classical PoW consensus races, our race conditions have significantly less consumed energy.

The majority of 2/3 is a tweakable parameter and can be increased to 3/4 or some other value, but we do not recommend it to be less than 2/3.

E. Safety and Liveness of Consensus

In every consensus algorithm, safety and liveness are essential factors. These things majorly depend on the network synchronicity and the number of honest participants in the consensus. Particularly for our consensus:

- *Safety* in the blockchain context, ensures that the honest participants in consensus should work on the same blockchain. Hence, safety considers the past and take actions based on history. That means if an honest participant accepts a new block in its blockchain, then in the future, this block will always be in the blockchain of other users. In Meshwork ledger, a block will be in the blockchain if it gets on an average at least $\frac{2}{3}s$ signatures from each of the validator (Including its client nodes' signatures) in the blockchain. This argument implicitly points out that there must be at least $\frac{2}{3}$ honest participants during the consensus round in the model and hence, ensures safety in the model.
- *Liveness* ensures that the major participants will be in charge of keeping the system alive; hence it considers the future. That means the validator nodes will always make progress in the blockchain. From the duty of a leader node or backup validator nodes, a new block will always be created and added to the blockchain in a consensus round when at least $\frac{2}{3}$ participants in the round are honest.

F. Coequality of mesh clients

The coequality of every mesh client is ensured by the procedure $ClientSelection(C_{i1}, C_{i2}, \ldots, C_{im})$ which is invoked at line 6 of the algorithm. The core property of this function is that it selects at least s client nodes to sign the block in a *Round Robin With Reset* manner. In such a way, every mesh client that was once selected to contribute in an aggregate signature (and possibly get a reward) will have to wait for several rounds until other mesh clients from that validator node also get its fair share of contribution.

G. Parameters in the Meshwork Ledger

There are a few parameters in Meshwork ledger. Signature and timeout parameters play a vital role in reaching consensus and achieve safety and liveness properties. Details of these parameters are as follows:

- *Signature parameter "s":* The parameter s representing the minimum number of signatures required from each validator is updated from time to time (e.g., Weekly or monthly). This update also depends on the density of the network. If the number of participants (validator and client nodes) increases/decreases in a considerable amount, the parameter s is updated accordingly in a quick manner.
- *Timeout parameters:* In our consensus, we have a few timeout parameters. The parameter τ_{block} defines the time to wait for the proposed block to reach to a validator node in a consensus round. Another timeout parameter τ_{agg} establishes the time to expect by the leader node in a consensus

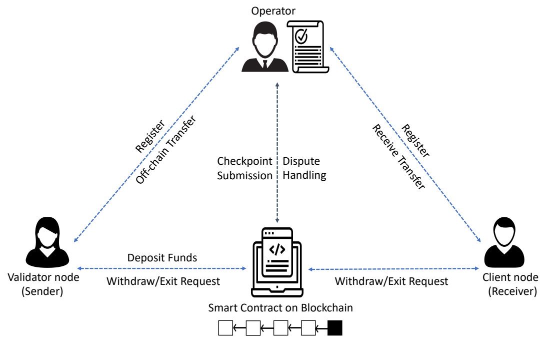

Fig. 3. Overview of Reward Mechanism through Commit-Chain

round to receive all the aggregate signatures from the other validator nodes. Both parameters should be reasonably and carefully decided using the Poisson distribution to assure the liveness of the system.

- *Reward distribution parameter "t":* It is decided in each consensus round based on the average number of client nodes connected to the validator nodes. In general, $0 > t \leq 0.2$. The leader node includes t along with the signature parameter s in the block proposal. Therefore, validator and client nodes can locally estimate the share they might receive after the successful consensus round.

V. Reward Mechanism using Commit Chain

Compared to traditional PoS models, our system has many nano-value reward transactions. Thus, the reward should be distributed cost-effectively, which should incur almost zero transaction fees. Hence we adopted a commit-chain distribution for the reward transfer. The reward transfer mechanism performed by commit-chain NOCUST requires an operator. The execution of the commit-chain protocol is performed in rounds, which are called eons. Each eon of the commit-chain will have many consensus rounds of the main blockchain protocol. Therefore, after completing each eon, the nodes participated in the consensus rounds within that eon will receive the sum of the rewards earned in those consensus rounds. All these nano-value reward transactions can be made zero-fee transactions reliably depending on the operator fee schedule. The correct execution of these transactions is enforced by the smart contracts of the main blockchain, but the transactions are performed on the commit chain. Following is the overview of the reward transaction mechanism using commit-chain:

Register All validator and client nodes create an account with the commit-chain operator via off-chain messages, hence register themselves to perform transactions on commit-chain.

Deposit After a consensus round, each validator node locks the amount of reward transaction (Originated from the leader validator node to other validator nodes) in the commit-chain.

Transfer To distribute the reward money to the client nodes, each validator node authorizes itself to the operator to debit its account and credit the client nodes' accounts.

Withdraw/Exit To withdraw the balance from the commit-chain or to exit the commit-chain, the validator or client nodes submit the off-chain request to the operator.

Checkpoint Constant size periodic checkpoints are used by the operator to commit the latest states of all the validator and client node accounts using a smart contract. This checkpoint is the root of the Merkle tree aggregating client and validator nodes state and balances. Each of the checkpoints requires an on-chain transaction.

Challenge/Response Challenge-response dispute mechanism is enforced by commit chain using the smart contract in case of operator misbehave.

The above operations are depicted in Figure 3. Moreover, a single operator for the off-chain solution becomes a central point of failure. Depending on the number of nodes participating in the blockchain, a few commit chains (less than the number of validator nodes) or watchtowers can be deployed to remove the single point of failure, but that will incur an extra cost. Hence in our model, the fairness of the reward mechanism depends on the fairness of the used commit-chain.

VI. Complexity Analysis of the Consensus

Communication Complexity In every consensus protocol, many iterations of communication are required to reach the final consensus and append the block in the blockchain. In Meshwork ledger, each validator node has to send the new proposed block to its connected client nodes. Therefore $O(sn)$ number of messages will be transmitted to propagate the block in the system. The same will apply for receiving the aggregate signatures from the validator nodes. So in total, the communication complexity of the network will be $O(sn)$.

Computational Complexity The signature process requires computation to create the signature and to verify them. On average, each validator node receives s signatures and verify them using $2s$ pairing computations. Nevertheless, in total, the number of pairing computations will be $n*2s = O(sn)$. After receiving all the aggregate signatures, the leader has to perform $2n$ pairing computations for verification. Hence in total, the computational complexity in terms of pairing operations in a consensus round will be $O(sn)$.

VII. Security

A. Security Model:

1) Malicious Validator Node: If a validator node V_m acts maliciously and does not include its client node C_{mj}'s signature in the aggregate signature $aggSig_m$, the client node can raise this issue in the blockchain network using its Proof of Inclusion $P_{C_{mj}}$. Therefore, the verification of $P_{C_{mj}}$ is performed by the other validator nodes. If the proof $P_{C_{mj}}$ is verified and the client node signature (Can be checked by client node public identifier $ID_{C_{mj}}$) is not found in the aggregate signature, then the validator nodes agree on penalizing the validator node V_m.

2) Malicious Client Node: A client node C_m can act maliciously by submitting the same signature σ_{C_m} multiple times in a consensus round to earn more reward. However, as in a consensus round, each validator node gets a list $\mathcal{D}$ from the leader specifying the public key identifiers of the client nodes that submitted signatures on the same block to validator nodes. The validator nodes at the final stage keep a clean local house by keeping a record for the number of double signatures incidents for its client nodes. If the number of incidents caused by a client node exceeds a $Tolerance$, the client node is permanently blocked in the system.

3) Security against Existential Forgery: In the Meshwork ledger, an adversary (a malicious validator) can try to forge a multi-signature on a block choosing some subset of its client nodes. The forgery of multi-signature can be reduced to the Computational Diffie-Hellman (CDH) Problem [28]. The forgery can be defined as; For n signers, an adversary must forge a multi-signature $\sigma \in \mathbb{G}_1$ on message M under the public keys $pk_1, \ldots, pk_n$. To see how forgery is equivalent to solving CDH problem, the following description justifies the point: Given randomly chosen $g, g^{sk_1}, h(= H(M))$, the adversary can randomly generate $(n-1)$ key pairs $(sk_2, pk_2) \ldots (sk_n, pk_n)$. Then adversary creates a multi-signature σ which should satisfy the the verification equation.

$$e(\sigma, g_2) = \prod_{i=1}^{n} e(h, pk_i) = \prod_{i=1}^{n} e(h^{sk_i}, g_2) = e(\prod_{i=1}^{n} h^{sk_i}, g_2)$$

which means $\sigma = \prod_{i=1}^{n} h^{sk_i}$,

From the above check for signature σ, if it passes that means adversary has computed h^{sk_1}, given randomly chosen g, g^{sk_1}, h which is equivalent to solving CDH problem.

B. Attacks

- Validator-Specific attacks: In many of the PoS based systems, the validator nodes try to collude and earn a bigger reward by increasing their total stake in the system. In our system, as the reward is equally distributed among the validator nodes, so in case of collusion, the validator nodes have to share the joint reward, and that will be less than the reward of any other validator node. Hence, any kind of collusion does not bring any advantage in terms of reward.
- Network-Level attacks: In case of a network partition (Eclipse attack), until the number of received aggregate signatures and client node identifiers satisfy the consensus rules to accept the block, the consensus is reached. The consensus will not be achieved in other scenarios of a network partition, and the system will go in recovery mode.
- Sybil attack: Sybil attack is prevented in the system using strong authentication checks for client and validator nodes.

VIII. Technical Details and Experiments

We conducted experiments to test the robustness and efficiency of the Meshwork ledger. The experiments were carried out on a MacBook Pro system having 2.3 GHz Intel Core i5 processor and 8 GB 2133 MHz memory. So far, we have designed a validator network which is connected with client nodes using Docker containers. Validator nodes perform the BLS signature aggregation. In our experiments,

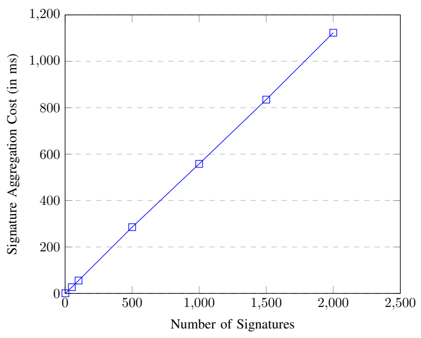

Fig. 4. BLS signature aggregation cost vs Number of signatures

we analyze the cost of signature aggregation and verification in each consensus round by varying the number of nodes participating in the consensus. The signature aggregation cost depends on the number of signatures to be aggregated, hence signature aggregation cost increase linearly with the number of nodes/signatures. Figure 4 depicts the implementation result of signature aggregation cost (in milliseconds). In contrast, the verification cost does not depend on the number of nodes, as we are verifying a single aggregate signature. An aggregated BLS signature verification requires the computation of two bilinear pairing operations. We are using ate pairing scheme in our implementation, and the verification cost of aggregate BLS signature is 3.54 ± 0.07 milliseconds. For storing the data in our account-based blockchain nodes, we use persistent key-value and fast database store leveldb [38] (see also [39]).

IX. Conclusion

We proposed the "Meshwork ledger" which establishes a network of coequal client nodes that contribute to the endorsement of the transactions by providing digital signatures to a validator node that collects them in an aggregate signature scheme. We also proposed a reward mechanism for the meshwork client nodes. That reward mechanism had a prime design objective to offer coequality for all client nodes. Our goal was to design a blockchain ledger where there is no advantage for getting rewards if the client is an early adopter, if the client has collected a significant stake of rewards or if the client just joined the meshwork.

The pillar design component in our consensus algorithm is the use of the aggregate multi-signatures. The core idea of the consensus is to race for the maximum number of signatures (approvals) on a block from the mesh clients, to append the block in the blockchain. The race in the consensus is among validator nodes that try to collect a maximum number of approvals (signatures) from the mesh clients. The future direction of work for our consensus algorithm is to perform a detailed scalability analysis, to evaluate the performance of consensus in a large network, and to compare it with existing consensus algorithms to advance its adoption in the practical world of implementation.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf," 2009.

[2] V. Buterin, "QuarkCoin: Noble Intentions, Wrong Approach," *Bitcoin Magazine*, Dec 2013, [Online; accessed 3-Jun-2019].

[3] E. Duffield and D. Diaz, "Dash: A payments-focused cryptocurrency," Whitepaper, https://github.com/dashpay/dash/wiki/Whitepaper, 2018, [Online; accessed 3-Jun-2019].

[4] CryptoRekt, "Official Verge Blackpaper 5.0," Blackpaper, https://tinyurl.com/y88sd7ze, Jan 2019, [Online; accessed 14-Jun-2020].

[5] M. Raikwar, D. Gligoroski, and K. Kralevska, "SoK of Used Cryptography in Blockchain," *IEEE Access*, vol. 7, pp. 148 550–148 575, 2019.

[6] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[7] L. Ren, "Proof of stake velocity: Building the social currency of the digital age," *Self-published white paper*, 2014.

[8] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 51–68.

[9] A. Kiayias, I. Konstantinou, A. Russell, B. David, and R. Oliynykov, "A Provably Secure Proof-of-Stake Blockchain Protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 889, 2016.

[10] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of Luck: An Efficient Blockchain Consensus Protocol," in *Proceedings of the 1st Workshop on System Software for Trusted Execution*, ser. SysTEX '16. ACM, 2016, pp. 2:1–2:6.

[11] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On Security Analysis of Proof-of-Elapsed-Time (PoET)," in *Stabilization, Safety, and Security of Distributed Systems*, P. Spirakis and P. Tsigas, Eds. Springer International Publishing, 2017, pp. 282–297.

[12] Libra Association, "The Libra Blockchain," June 2019. [Online]. Available: https://libra.org/en-US/

[13] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, Eds. Cham: Springer International Publishing, 2018, pp. 435–464.

[14] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," Cryptology ePrint Archive, Report 2019/514, 2019, https://eprint.iacr.org/2019/514.

[15] D.-P. Le, G. Yang, and A. Ghorbani, "Ddh-based multisignatures with public key aggregation." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 771, 2019.

[16] J. Long and R. Wei, "Scalable bft consensus mechanism through aggregated signature gossip," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 360–367.

[17] "Pchain," 2018. [Online]. Available: https://pchain.org

[18] Graytrain, "PDBFT2.0 — Pchain's Revolutionary Consensus Algorithm For Solving The Trilemma," November 2019. [Online]. Available: https://tinyurl.com/y58v6koa

[19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Designs, Codes and Cryptography*, vol. 87, no. 9, pp. 2139–2164, Sep 2019.

[20] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of mimblewimble," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 657–689.

[21] Y. Zhao, "Practical aggregate signature from general elliptic curves, and applications to blockchain," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, ser. Asia CCS '19. New York, NY, USA: ACM, 2019, pp. 529–538.

[22] G. Golan-Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D. Seredinschi, O. Tamir, and A. Tomescu, "SBFT: a scalable decentralized trust infrastructure for blockchains," *CoRR*, vol. abs/1804.01626, 2018.

[23] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 279–296.

[24] R. El Bansarkhani and J. Sturm, "An efficient lattice-based multisignature scheme with applications to bitcoins," in *Cryptology and Network Security*, S. Foresti and G. Persiano, Eds. Cham: Springer International Publishing, 2016, pp. 140–155.

[25] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille, "Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1717–1731. [Online]. Available: https://doi.org/10.1145/3372297.3417236

[26] R. Khalil, A. Gervais, and G. Felley, "Nocust-a securely scalable commit-chain," Cryptology ePrint Archive, Report 2018/642, Tech. Rep., 2018.

[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 514–532.

[28] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology — EUROCRYPT 2003*, E. Biham, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 416–432.

[29] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 465–485.

[30] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *Advances in Cryptology - EUROCRYPT 2007*, M. Naor, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 228–245.

[31] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[32] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[33] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Draft nist special publication 800-63-3 digital identity guidelines," *National Institute of Standards and Technology, Los Altos, CA*, 2017.

[34] V. Buterin *et al.*, "Ethereum: A next-generation smart contract and decentralized application platform," *URL https://github.com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper*, 2014.

[35] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham: Springer International Publishing, 2017, pp. 297–315.

[36] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, Oct 1999, pp. 120–130.

[37] M. Ahmed-Rengers and K. Kostiainen, "Don't mine, wait in line: Fair and efficient blockchain consensus with robust round robin," 2020.

[38] A. Dent, *Getting started with LevelDB*. Packt Publishing Ltd, 2013.

[39] M. Raikwar, D. Gligoroski, and G. Velinov, "Trends in development of databases and blockchain," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, 2020, pp. 177–182.

Paper C

R3V: Robust Round Robin VDF-based Consensus

M. Raikwar, D. Gligoroski

Published in 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), IEEE, 2021

R3V: Robust Round Robin VDF-based Consensus

Mayank Raikwar*, Danilo Gligoroski*
*Norwegian University of Science and Technology (NTNU) Trondheim, Norway
Email: {mayank.raikwar, danilog}@ntnu.no

***Abstract*—Proof of Stake (PoS) based consensus provides a better mechanism than Proof of Work (PoW) consensus for extending the blockchain without significant energy waste. Most of the PoS consensus protocols derive or use some randomness to elect a leader candidate. This makes the consensus weaker and attracts more attackers to mount different attacks, e.g., long-range attacks and block withholding attacks. In PoS consensus, having more stakes gives more chances to be a leader among participating stakeholders. Therefore, most PoS protocols do not provide better fairness for the stakeholders participating in the consensus protocol. Moreover, these protocols suffer from high communication complexity for selecting a leader candidate in each consensus round.**

In this work, we propose a novel consensus protocol "R3V" that selects a set of leader candidates in a round-robin manner according to age. Finally, these leader candidates compete to be the block leader by solving a Verifiable Delay Function (VDF) based puzzle. We propose different methods to generate verifiable identities for the stakeholders. The identities are enrolled in the blockchain, which provides the age norm needed for the consensus. Compared with the other PoS consensus protocols, our protocol shows better resilience against most of the common attacks on PoS protocols. Additionally, it proclaims low energy consumption, less communication complexity, and better fairness.

I. Introduction

Since the advent of blockchain [1], the scientific knowledge about the core of its underlying technology – its consensus mechanism – has been growing at a rapid pace. The initial consensus introduced by Bitcoin [1] is known as Proof of Work. The main idea of PoW consensus is solving computation-intensive hash-based hard puzzles. Many consensus protocols were introduced with a similar puzzle-based idea, but all these consensus protocols suffered from computational power and huge energy waste. To address and overcome the problems of PoW, another consensus mechanism, Proof of Stake, was introduced. PoS protocols randomly elect a block proposer from the set of participants based on the stake of each participant. There have been several constructions of different PoS protocols in the past [2].

These PoS protocols can be categorized into two types: Chain-based and BFT-based (Byzantine-Fault Tolerant). Chain-based follows the longest chain rule of PoW protocol for the selection of the right chain. BFT-based protocols require voting from the participants and assume that 2/3 of the stakes are held by honest participants. Chain-based PoS protocols simulate the PoW leader election. There are only a few implementations of chain-based PoS protocols which includes Ouroboros [3] and its descendants [4], [5]. Nevertheless, BFT-based protocols have proven mathematical properties. Therefore, there have been many constructions of BFT-based protocols such as Algorand [6], Tenderemint [7] and Casper [8].

In many PoS protocols, the randomness is generated either from the previous block or from randomness beacons. Therefore, these protocols suffer from long-range and grinding attacks and incur high communication costs. Algorand [6] and Ouroboros Praos [4] apply the functionality of verifiable random function in their leader election. However, both protocols suffer from selection bias. Some PoS protocols [9], [10] including Algorand, have a concept of committee selection in leader election protocols, which incur high communication complexity. Not only the complexity but also most of the PoS protocols do not provide fairness to consensus participants.

To achieve fairness, less communication cost, and less susceptibility to long-range and grinding attacks, we construct a round-robin-based consensus mechanism that leverages the functionality of verifiable delay function [11]. Our construction involves the creation and subsequent registration of long-term identities of consensus participants in the blockchain. The number of registered identities of a consensus participant is bounded by its stake. Our consensus protocol works in rounds, and in each consensus round, some of the oldest identities apply VDF on a common input. The one who finds the solution to VDF first is chosen as the leader of that round and proposes a new block.

Our consensus protocol involves two important constituents: first is the identity creation, and second is the applicability of VDF. We propose identity creation based on existing infrastructure such as Trusted Execution Environment (TEE), e.g., Intel Software Guard Extension (SGX). SGX has already been incorporated in a few consensus protocols, such as Proof of Elapsed Time (PoET) [12], Proof of Luck (PoL) [13], Proof of Queue (PoQ) [14], Proof of Trusted Execution Environments (TEEs) Stake (PoTS) [15]. PoET and PoL select leaders based on processors' random waiting time, and their security relies on the honest majority of TEE processors. However, PoQ and PoTS weigh consensus participants based on their stake. The drawback is that compromising a few high-stakes TEEs might collapse the whole system and halt the consensus. Most of these TEE-based consensuses do not provide fairness, while our consensus provides fairness to all consensus participants.

VDF has been applied in consensus protocols, but being a recent cryptographic primitive, it is not explored in much depth

978-1-6654-3924-4/21/$31.00 ©2021 IEEE

for use in new consensus protocols. The Chia network [16] uses VDF with its Proof of space protocol, where consensus participants evaluate VDF on a random integer and produce proof of access to its available disk space. The problem with their approach is that it can be biased, and grinding attacks can be performed. Another construction involves computing VDF on the last block information such as [17], [18]. Other consensus protocols concerning the use of VDF are Proof of Staked Hardware (PoSH) [19] and a Decentralised Autonomous Organisation (DAO) working as Random Number Generator (RNG) in Ethereum 2.0 (RANDAO) [20].

A. *Our Contribution*

We propose a new consensus protocol with the following novel contributions:

- Our consensus is a composition of round-robin selection mechanism of participants' identities with the verifiable delay function on a common input in the blockchain system.
- We present different ways to create long-term identities for consensus participants.
- We briefly analyze our consensus protocol concerning several possible attacks on PoS protocols. Moreover, we also describe the security properties of our consensus.
- We exhibit complexity analysis of consensus protocol and compare it with some of the existing protocols.

B. *Structure of the Paper*

The rest of the paper is described as follows. Section II presents the preliminaries needed to design the consensus protocol. Section III describes consensus elements comprehensively and further explains the whole consensus protocol. Section IV presents the feasibility of our consensus by defining all the security properties and shows its security and resistance against various PoS attacks. Section V demonstrates the complexity of the consensus protocol and also compares our consensus with existing consensus mechanisms. Finally, in Section VI, we conclude the paper and discuss possible ways to adapt and enhance this work.

II. Preliminaries

A. *Infrastructures for Identity Creation*

In our consensus, reliable identities are created from infrastructures such as a trusted execution environment (TEE) or trusted public certificates. A TEE is a secure area of the main processor that provides isolated execution of code and data. Therefore, with the provided security feature, reliable identities of stakeholders can be bootstrapped from TEE. The popular TEEs are Intel SGX and ARM TrustZone. Many of the PoS consensus protocols make use of Intel SGX as their TEE. We also demonstrate identity creation using Intel SGX. Moreover, a similar technique can be used to create identity using ARM TrustZone. TEE reliance on trusted hardware poses a challenge for its use in practical deployment in the blockchain. These challenges such as side-channel attacks [21], [22], Foreshadow [23] and Spectre [24] vulnerabilities should not impact the leader election in PoS consensus. Our R3V consensus provides a reasonable level of security even if an adversary compromises a significant number of stakeholders participating in the consensus. Trusted public certificates can also help in the creation of long-term identities. These public certificates should provide a method to check their correctness and integrity. These certificates can be national identity cards, bank credit cards, or passports.

B. *Verifiable Delay Function*

Verifiable delay function (VDF) is a cryptographic primitive proposed in 2018 by Boneh et al. [11]. A function $f : \mathcal{X} \to \mathcal{Y}$ is a VDF if given an input $x \in \mathcal{X}$, the computation of output $y \in \mathcal{Y}$ takes predefined number of steps T; additionally the verification of y is exponentially easy and efficient. Furthermore, the computation of y cannot be parallelized even if a polynomial number of processors are available.

Definition 1: (VDF): *A VDF is defined as a tuple of following algorithms:*

- Setup(λ, T): It is a randomized algorithm that takes security parameter λ, time parameter T and outputs public parameter pp.
- Eval(pp, x, T): The evaluation algorithm takes public paramater pp, input value $x \in \mathcal{X}$ and time parameter T, returns an output value $y \in \mathcal{Y}$ together with a proof π. The algorithm may use random coins to generate the proof π but not for the computation of output y.
- Verify$(\mathsf{pp}, x, y, \pi, T)$: The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

There have been two main constructions of VDF based on the modular exponentiations. These two main proposals are Wesolowski Scheme [25] and Pietrzak Scheme [26]. Both schemes evaluate an output value $y \leftarrow H(x)^{(2^T)} \mathsf{mod} N$, along with a proof π, given an input value x. Here $H : \mathcal{X} \to \mathbb{G}$ is an efficiently computable hash function, T is the number of squarings needed to compute the output, and N is an RSA modulus.

With the growing interest in VDF, Ephraim et al. [27] introduced a notion of Continuous Verifiable Delay Function (cVDF). They presented the construction by adapting Pietrzak Scheme. A VDF f can be a continuous VDF if it gives computation of function f on intermediate steps (i.e. $f(t)$ for $t < T$) along with an efficient proof π^t. The intermediate outputs of cVDF are publicly and continuously verifiable.

III. R3V Consensus

A. *Identity Establishment*

The identity of the stakeholders can be established by using different methods. These methods generate long-term and reliable identities which are recorded in the permissionless blockchain. Without loss of generality, we consider the public keys of stakeholders as their respective identities. The identity generation method should prevent an adversary from establishing multiple valid identities, thereby launching a Sybil attack.

Next, we define a few methods to generate reliable identities for the stakeholders in this consensus.

- *Using Trusted Execution Environment* Identities of consensus participants can be established using trusted execution environments (TEE) such as SGX. Intel SGX is defined as a set of instruction codes built into modern Intel CPU, which defines enclaves (a private memory region) in isolation, protected from outside processes. An entity A executing an enclave E has to be attested by a secure attestation service (Intel SGX), called Intel Attestation Service (IAS). This attestation convinces the entity A that the enclave E is running on an authentic SGX processor. An SGX platform has a local software called Quoting Enclave QE that performs local attestation with the enclave E. The complete attestation process is depicted as in Figure 1 where an SGX platform attests its enclave E to IAS through a remote verifier V. If the attestation is successful, then IAS sends the signed QUOTE. The REPORT prepared by E contains enclave's measurement and a USERDATA field containing application-specific parameters such as a hash of its public key. QUOTE structure prepared by QE signs the QUOTE using the processor-specific attestation key before forwarding it to IAS.

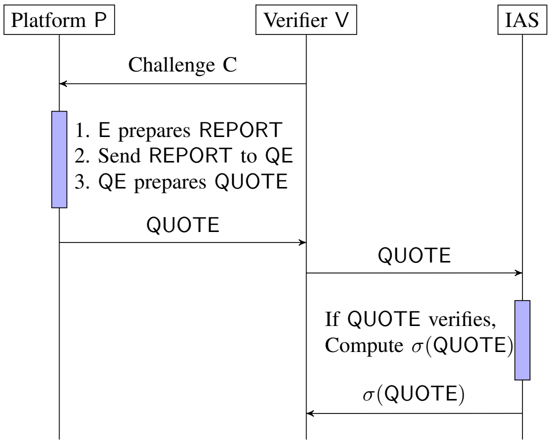

Fig. 1. SGX Remote Attestation

To bootstrap the blockchain, an untrusted entity GC (Genesis Creator) performs remote attestation with IAS and obtains an access credential c for IAS. Further, it chooses n number of platforms as initial blockchain members. Each platform P_i generates its public-private key pair (PK_i, SK_i) by installing enclave code. SK_i is sealed in the enclave, and PK_i is given to GC. Each platform P_i attests itself to IAS through GC as a remote verifier as depicted in Figure 1. The following operations take place for creating a genesis block.

1) For each platform P_i, QUOTE_i includes $H(PK_i)$; where H is a collision-resistance Hash function.
2) After successful attestation of P_i, IAS sends σ_i to GC, where σ_i is the signed QUOTE_i by IAS including a pseudonym X_i for P_i.
3) GC checks whether for each platform P_i, the returned hash matches in QUOTE_i and also checks if $X_i \neq X_j, \forall\, i \neq j$ where $i, j \in n$.
4) After performing all the above steps, GC creates the genesis block GB which includes $(\{PK_i, \sigma_i\}_{i=1}^{n}, C_e, h_{GB})$, where C_e is the enclave code which will be used as reference for future identity enrollments, h_{GB} is hash of all the contents included in the block.
5) GC sends the access credential c to all the attested enclaves that gets sealed in the attested enclaves.

New users seeking enrollment to the consensus are registered through the current members of the ledger (whose identities are already published in the ledger). Thereby, a new user having platform P_u with key pair (PK_u, SK_u) establishes and registers its identity by performing remote attestation with IAS through a current ledger member P_c. In this attestation process, all the steps are performed as defined previously. Now the QUOTE_u includes $H(PK_u||cr||h_{GB}||h_{CB})$, where *cr* is current round number of consensus, h_{CB} is hash of the latest block. After a successful remote attestation, P_c broadcasts an enrollment message $M = (PK_u, cr, h_{CB})$ to the network. Once M is included in a new block, the new identity PK_u gets established in the blockchain.

- *Using Mining Reward* Identities can also be created using mining. For each block creation, the winner participant P_w receives identity rewards R_w proportional to the stake rewards S_w. That means, $R_w \propto \lfloor \gamma . S_w \rfloor$, where $0 < \gamma \leq 1$. These identity rewards can be further used to enroll new identities in the blockchain. To control the number of identities in the system, a threshold T on the identity rewards can be placed for enrolling new identities. That means, once an existing member P_i of the ledger having its total identity rewards from previous j rounds $\sum_{j=l}^{r} R_i^j \geq \mathsf{T}$, then P_i can establish new identity either for itself or for a new user; here l is the latest round number where an enrollment of new identity is performed by P_i. The following description will provide a better understanding of identity creation.
  - Bootstrapping of the blockchain can be done using a preliminary PoW-Based phase. In this phase, a set of participants $\{P_1, \ldots, P_n\}$ having identities $\{PK_1, \ldots, PK_n\}$ and valid solutions $\{s_1, \ldots, s_n\}$ to PoW can run a distributed cryptographic protocol [28] to create a genesis block. The genesis block includes valid PoW solution of each party and the corresponding Identity rewards. These rewards are further used to enroll new identities in the ledger. Henceforth, the genesis block includes $(\{PK_i, s_i\}_{i=1}^{n}, h_{GB})$.
  - Enrollment of new identities can be performed by existing participants of the ledger. An existing participant P_i having identity PK_i and rewards R_i more than the threshold T, can create a new identity PK_u with the corresponding secret key SK_u. Then P_i broadcasts an

enrollment message $M = (PK_u, \pi_i, R, \sigma_i)$; where π_i is a proof of having sufficient reward that can be a set of hashes of previous blocks mined by P_i, R is the remaining identity reward of P_i, and σ_i is a signature over all the other information on M. Once the message M is included in a new block, the new identity PK_u gets confirmed in the ledger. P_i can sell the identity PK_u to new participants using a smart contract and in exchange for some stakes of the new participant.

- *Using Trusted Public Certificates* Reliable identities can be bootstrapped from trusted public certificates such as credit cards, national identity cards, Passport, etc. Genesis block of the blockchain can be created using multi-party computation among initial ledger participants. A new user P_u can generate its key-pair (PK_u, SK_u) using the public certificate PC_u, where PK_u is the identity of user P_u. Further, P_u broadcasts a message $M = (PK_u, \pi_u, \pi_u^{PoP}, \sigma_u)$; where π_u is a zero-knowledge proof of valid public certificate PC_u and correct generation of PK_u, π_u^{PoP} is a proof of possession of corresponding secret key SK_u, and σ_i is a signature over all the other information on M. Once the message M is included in a new block, the new identity PK_u gets confirmed in the ledger. Additionally, some interesting works [29], [30] can be utilized to create a zero-knowledge proof of identity.

There have been advancements to create decentralized identity by decentralized identity foundation (DIF) [31]. Once the DIF ecosystem would be mature, it can be directly applicable to create identities in our consensus protocol. Moreover, Self-sovereign identities (SSI) and verifiable credentials are also potential candidates for identity creation to be investigated and further integrated in our consensus.

B. VDF Puzzle

A VDF puzzle is constructed using the repeated squaring functionality of VDF. In what follows, we define the VDF-puzzle based on continuous VDF.

Definition 2: (VDF Puzzle): Given an input x, a function F, a constant α, a VDF-puzzle VP asks for a solution $\mathsf{S} = (t, s, \pi^s)$ such that

$$s = H(x)^{2^t} \tag{1}$$

$$F(s) \leq \alpha.\mathsf{M} \tag{2}$$

Here input x is a verifiable input for the puzzle, function $F : X \to Y$ is a one-way hash function with M as its maximum value, (t, s, π^s) is an intermediate output of cVDF on x, and H is a hash function used in cVDF acting as a random oracle. The value α is the puzzle difficulty and it is updated from time to time. Moreover, in the above VDF-puzzle, the output s is computed as in Equation 1 for an intermediate step t. Subsequently, s satisfies Equation 2, and a proof π^s of correct VDF computation for s is generated. As a result, (t, s, π^s) serves as a solution S to the VDF-puzzle VP.

In the above puzzle VP, the value t is unknown before the puzzle solution is found. As the puzzle is based on VDF, it follows the similar properties of VDF:

1) *Sequentiality* : The computation of the output s of the puzzle solution S on input x takes $O(t)$ sequential steps, even on parallel computers.
2) *Verifiability* : The verification of the puzzle solution S can be performed efficiently in $O(polylog(t))$ steps.

To apply the puzzle in the consensus process, each stakeholder's input x of the puzzle should be random and efficiently verifiable. Additionally, different inputs make the time needed to solve puzzle VP different for each stakeholder. Therefore, in the essence of generating different verifiable random inputs for each stakeholder, the functionality of Verifiable Random Function (VRF) [32] can be leveraged. Each stakeholder P computes the input x for its verifiable puzzle VP_P on the latest puzzle solution s_l as follows:

$$(x, \pi^x) \leftarrow VRF_{SK_P}(s_l) \tag{3}$$

C. Consensus Algorithm

In the R3V consensus protocol, the stakeholders operate in a partially synchronous network where the message transmission between two directly connected stakeholders occurs within a specific time-bound. Each stakeholder puts a minimum amount of stake in the blockchain in order to participate in the consensus. Each stakeholder maintains a list ReceiveNBlock of received new blocks in the current round, which is set to empty during the start of the round. R3V consensus algorithm runs in terms of rounds. On each round, a deterministic set of leader candidates among the stakeholders are selected according to their age. These chosen candidates try to solve the current VDF-puzzle. Once one of the leader candidates solves the puzzle, it creates a new block and broadcasts the new block.

Algorithm 1: Consensus Algorithm

```
Input  : $\mathcal{L}, PK_i, r, h_{r-1}, s_{r-1}$
Output: $B_r$
1  ReceiveNBlock ← $\emptyset$;
2  if $Eligible(\mathcal{L}, PK_i, r, h_{r-1})$ then
3      $(x_r, \pi_r^x) \leftarrow VRF_{SK_i}(s_{r-1})$;
4      $s_r \leftarrow H(x_r)^2$;
5      $t_r \leftarrow 1$;
6      while $F(s_r) > \alpha.\mathsf{M}$ do
7          $t_r \leftarrow t_r + 1$;
8          $s_r \leftarrow {s_r}^2$;
9      end
10     if ReceiveNBlock = $\emptyset$ then
11         Compute $\pi_r^s$;
12     end
13 end
14 $P_i$ prepares a new block $B_r$ for round $r$ with the header
       $(PK_i, x_r, \pi_r^x, t_r, s_r, \pi_r^s, \sigma_i)$;
15 $P_i$ broadcasts the block $B_r$
```

In each consensus round r, each stakeholder P_i with enrolled identity PK_i runs the above Consensus Algorithm 1 where first it checks whether the stakeholder is eligible to be a leader candidate. If a stakeholder is an eligible leader candidate, then it tries to find the solution to the current VDF-puzzle VP on its randomly generated input using Equation 3.

If a stakeholder finds the solution before anyone else, then it prepares a new block B_r. The stakeholder signs the content of the block, which includes the block header and the transaction lists. The signature is included in the block header. Finally, the block B_r is broadcast to the network. The other stakeholders (including leader candidates) receive the new block B_r and verify the solution for the puzzle VP and based on the verification, they accept or reject the block.

The main constituents of R3V consensus are the $Eligible$ predicate followed by VDF-puzzle. Section III-B describes the VDF-puzzle in detail. Hence, following we describe the $Eligible$ predicate in detail:

$Eligible(\mathcal{L}, PK_i, r, h_{r-1})$: This predicate checks whether a stakeholder P_i with enrolled identity PK_i is eligible to be a leader in round r. Here the robust round-robin part of R3V comes into the picture. In R3V consensus, each identity PK_i is attached with an age A_i. Here age A_i refers to the number of rounds since PK_i's latest message appeared in the ledger $\mathcal{L}$. This message can be either an identity enrollment message of PK_i or a newly created block B_r by PK_i.

Each stakeholder maintains a round-robin queue of stakeholders' identities. The queue is sorted in decreasing order based on the age of stakeholders' identities and gets updated after each consensus round. This queue can be formalized as a virtual queue containing all the registered identities of the ledger $\mathcal{L}$ as its elements. In that manner, the virtual queue of all the stakeholders should be the same.

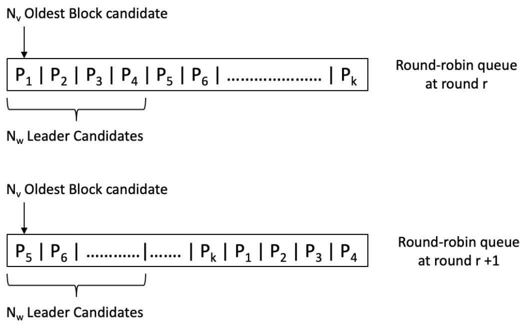
N_v Oldest Block candidate
P_1 | P_2 | P_3 | P_4 | P_5 | P_6 | .... | P_k
Round-robin queue at round r
N_w Leader Candidates
N_v Oldest Block candidate
P_5 | P_6 | ....|....... | P_k | P_1 | P_2 | P_3 | P_4
Round-robin queue at round r +1
N_w Leader Candidates

Fig. 2. Round-Robin Selection of Identities

In each consensus round, a window (set) N_w of stakeholders are chosen from the round-robin queue as leader candidates based on age. In the $Eligible$ predicate of round r, each stakeholder parses the ledger $\mathcal{L}$ using a threshold value N_v where the stakeholder selects the N_w oldest leader candidates from the current block (with block hash h_{r-1}) til the N_v oldest block. Then, the stakeholder with identity PK_i checks whether PK_i is in the selected N_w leader candidates. Further, all the selected N_w leader candidates compete to be the block leader by solving the current VDF-puzzle. Moreover, after the selection of N_w leader candidates, the age of all the identities in N_w is set to zero and further these identities are placed at the end of the round-robin queue of each stakeholder as depicted in Figure 2. The newly updated queue is used for the selection of leader candidates in the next consensus round $r + 1$. The updated queue confirms the fairness of the system by giving chance to the next oldest identities in round $r+1$. Additionally, the values N_w, N_v are adjusted/updated in the blockchain to make the deterministic selection process of identities fairer.

During a consensus round, each stakeholder including leader candidates continue checks it ReceiveNBlock list. Once a stakeholder P_i becomes the block leader of round r as in Algorithm 1, it broadcasts the new block B_r with the header $(PK_i, x_r, \pi_r^x, t_r, s_r, \pi_r^s, \sigma_i)$. The block B_r is added to the receiving stakeholders' ReceiveNBlock list. A receiving stakeholder P_j with identity PK_j performs the following verification checks.

- $Eligible(\mathcal{L}, PK_i, r, h_{r-1}) = 1$
- $VRF.\text{Verify}(PK_i, s_{r-1}, x_r, \pi_r^x) = 1$
- $VDF.\text{Verify}(pp, x_r, t_r, s_r, \pi_r^s) = 1$

If all the checks verifies, the stakeholder P_j with identity PK_j extends its blockchain by adding the new block B_r. If the stakeholder P_j is a leader candidate of the round r, it terminate the process of solving its VDF-puzzle after extending its blockchain with B_r.

Fork in R3V : In R3V consensus, a rare situation may arise where more than one leader candidates solve the puzzle almost at the same time and become the block leader. As the VDF-puzzle input for each leader candidate differs due to VRF, therefore, solving the VDF-puzzle around the same time becomes a less probable event. Moreover, the round-robin selection of N_w stakeholders in each consensus round also reduces the probability of forking the chain. Nevertheless, even if a fork happens, our consensus adopts the *longest chain* rule to choose between the forks where the branch with the most number of valid blocks is selected.

IV. Security Analysis

A. Analysis of Attacks

We present the analysis of R3V consensus based on the most common attacks on proof of stake consensus [33].

- Long-Range Attack: Long-range attacks are one of the most common attacks in Proof of stake blockchains. In this attack scenario, an adversary creates a new branch starting from a distant past (can be a genesis block) and tries to overtake the main public branch of the blockchain. This attack is also known as *Alternative history* or *History revision* attack. In R3V consensus, due to the round-robin selection of leader candidates for each consensus round, and due to the sequentiality of the VDF puzzle where mining a new block B_r requires solving a puzzle using the previous block puzzle output s_{r-1}, makes the probability of long-range attack negligible.
- Nothing-at-Stake Attack: Nothing-at-stake problem arises when a rational stakeholder publishes blocks on different forks. As there is no opportunity cost, the rational stakeholder can follow and extend both branches to maximize

its reward. However, in our consensus, this attack is only feasible when the rational stakeholder has created multiple successive identities and has become the leader in the consecutive rounds. The fair process of creating new identities and the fair chance of being the leader in a consensus round significantly reduce the probability of mounting a Nothing-at-Stake attack.

- Grinding Attack: In a grinding attack, the leader of the previous consensus rounds aims to gain extra advantage on the next consensus round by trying different block candidates (by changing the transactions or block header) and picking the most advantageous candidate block. To consecutively win, the leader of the previous consensus rounds iterate through many block candidates until it becomes the leader on the next round. In our consensus, VDF-puzzle VP for each consensus round is fully determined, and the leader of the previous round cannot perform a grinding attack just by trying different block candidates.
- Block Withholding Attack: In a block-withholding attack, an adversary mines new blocks in its forked branch of the blockchain without publishing the newly mined blocks in the blockchain network. Later adversary publishes its branch, and the other stakeholders revert their branch with the adversary's branch, resulting in more reward for the adversary. In the R3V consensus, the selection of leader candidates in each round is fair. Therefore, an adversary can only mount a block-withholding attack only if it is eligible as a leader in successive rounds and if it successfully solves all the VDF puzzles in each round. Nonetheless, the chances of doing so are very low.
- Sybil Attack: In a Sybil attack, an adversary creates multiple Sybil identities and tries to significantly influence the network based on the total voting power of Sybil identities. Most of the PoS consensus reduce the Sybil attack by requiring stakeholders to put a mandatory fixed amount of stake in the blockchain. In R3V consensus, to mount the Sybil attack, an adversary has to enroll multiple identities on blockchain and also put a small fixed amount of stake. Therefore, having no voting in R3V consensus and a fair identity enrollment process reduces the probability of Sybil attack significantly.
- Denial-of-service (DoS) Attack: In R3V consensus, the $Eligible$ predicate makes the selection of leader candidates very predictable. Therefore, an adversary can prepare the DoS attack in advance. However, the R3V consensus provides a fair chance to all leader candidates to solve the VDF-puzzle. Hence, by adjusting the window parameter N_w, the DoS attack can be made harder. Another approach to make the DoS attack harder is for each leader candidate to change its IP address in each round where it is selected as a leader. An interesting research problem can be to introduce the concept of heartbeats similar to Raft consensus [34] in case of leader faces a DoS attack and choosing another leader candidate as a leader.

B. *Security Properties*

A consensus protocol should achieve three security properties: *Persistence, Liveness* and *Fairness*.

- Persistence: It ensures that once an honest stakeholder accepts a new block in its blockchain, the probability of getting the block reverted is negligible as the chain continues to grow. In the R3V consensus, due to the deterministic nature of leader candidates selection, the probability of reverting a block is negligible. Therefore, R3V consensus is persistent.
- Liveness: It ensures that blockchain will continue to progress by adding new blocks and valid transactions submitted by an honest stakeholder will eventually be included in the blockchain. In each round of R3V consensus, a set of stakeholders try to solve VDF-puzzle, and the one solving the puzzle first extends the chain by creating a new block. Therefore, in the R3V consensus, a new block is added in each round, and hence R3V consensus achieves better liveness.
- Fairness: As blockchains rely on reward in the forms of transaction fees when a leader is elected, fairness is about how often a leader is rewarded. In the R3V consensus, each stakeholder gets a fair chance of creating a new block and getting the associated reward due to the robust round-robin leader candidate selection. Most PoS consensus algorithms lack fairness due to selection bias; however, R3V provides better fairness.

V. Consensus Analysis

A. *Complexity Analysis*

Regarding computation complexity, in each round of the R3V consensus protocol, only N_w leader candidates try to solve the VDF puzzle. Solving the puzzle involves computing the repeated squaring, so in total, the computation complexity is O($N_w t$), where t is the average number of steps performed by each leader candidate.

As R3V consensus does not involve any voting or complex endorsing procedures similar to BFT-based consensus; the communication complexity is similar to the communication complexity of chain-based protocols. Moreover, in the R3V consensus, a race on solving the VDF-puzzle is among N_w leader candidates out of total N stakeholders. Finally, the block leader who solves the VDF-puzzle first broadcasts the new block to the network. Thereby, the total communication complexity is linear with the size of the network, that is O(N).

B. *Comparison Analysis*

In Table I, we compare different consensus protocols with respect to some necessary attributes. The two columns about *Attacks* show whether it is easy to mount these attacks on the corresponding consensus protocol or not. *Randomness* shows whether the consensus protocol uses some kind of randomness for leader election. *Liveness* shows how early the valid transactions are included in the blockchain. *Deterministic Selection* manifest if the consensus selects a deterministic set

Consensus Scheme	Long-range Attack	Block-withholding Attack	Randomness	Liveness	Deterministic Selection	Communication Complexity	DoS Resistance
Dfinity [35]	Moderate	Hard	✓	Good	✗	Moderate	Less
RRR [10]	Moderate	Hard	✓	Good	✓	Moderate	Less
PoTS [15]	Hard	Hard	✓	Better	✗	Moderate	Moderate
NC-VDP [18]	Hard	Moderate	✗	Better	✗	Less	Moderate
Our Scheme	Hard	Hard	✗	Better	✓	Less	Less

TABLE I
COMPARISON TABLE OF DIFFERENT CONSENSUS SCHEMES

of participants for leader election in each round. The deterministic selection also affects the fairness of the consensus protocol. *Communication Complexity* indicates whether the consensus uses multiple rounds of communication to finalize the leader. *DoS Resistance* exhibits how resistant the consensus protocols are in case of a DoS attack. Our consensus protocol performs better in most of the listed attributes. One significant advantage of our consensus is its fairness to all the blockchain participants as every participant gets a fair chance to race in a consensus round due to the round-robin selection of identities.

In Table I, we use the terms 'Moderate', 'Hard', 'Good', 'Better', and 'Less' subjectively that justifies the evaluation and comparison of the respective attributes of consensus protocols. We are open to suggestions of other terms that quantitatively justify the comparison of attributes in our comparison table.

NC-VDP consensus scheme in Table I achieves almost similar performance as R3V consensus. However, in NC-VDP consensus, all the participants try to solve the VDF puzzle in each round. Hence, it increases the chances of mounting a block-withholding attack, as well as does not provide better fairness to all the participants in contrast to R3V consensus.

VI. Conclusion

In this paper, we proposed a consensus protocol that selects the leader candidates based on the age status in a round-robin manner, and further, a block leader is chosen using a VDF based puzzle. Our consensus does not depend on any kind of randomness; instead, it provides a deterministic selection of leader candidates with the resilience towards the long-range and block withholding attacks. Our consensus shows lower energy consumption, less communication complexity, and better fairness than most of the Proof of stake based consensus mechanisms.

Future Directions: There are several ways to extend this work. One interesting direction would be to find other ways for identity generation. The identity generation can also be made anonymous and publicly verifiable using the amazing features of zero-knowledge proofs. Not only the identity creation but also the anonymity of the leader candidate can be achieved using zero-knowledge proofs and anonymous verifiable random function similar to [36], [37]. Performing a formal verification of R3V consensus will be an interesting area for proving the correctness of the protocol. Furthermore, due to round-robin selection, our consensus lacks better DoS resistance; therefore, exploring the direction of providing better DoS-resistant to our consensus would be an interesting avenue to look into.

VII. Acknowledgement

We thank the anonymous reviewers for their valuable and constructive comments to improve the quality and readability of the paper.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf," 2009.

[2] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.

[3] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[4] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.

[5] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 913–930.

[6] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.

[7] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," 2016.

[8] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[9] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 931–948.

[10] M. Ahmed and K. Kostiainen, "Don't mine, wait in line: Fair and efficient blockchain consensus with robust round robin," *arXiv preprint arXiv:1804.07391*, 2018.

[11] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Advances in Cryptology – CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds. Cham: Springer International Publishing, 2018, pp. 757–788.

[12] H. Foundation, "Sawtooth lake, https://github.com/hyperledger/sawtooth-core," 2016.

[13] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *proceedings of the 1st Workshop on System Software for Trusted Execution*, 2016, pp. 1–6.

[14] G. D. Bashar, A. A. Avila, and G. G. Dagher, "Poq: A consensus protocol for private blockchains using intel sgx," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2020, pp. 141–160.

[15] S. Andreina, J.-M. Bohli, G. Karame, W. Li, and G. A. Marson, "Pots: A secure proof of tee-stake for permissionless blockchains," *IEEE Transactions on Services Computing*, 2020.

[16] B. Cohen, "Chia network," 2017. [Online]. Available: https://www.chia.net

[17] A. Biryukov and D. Feher, "Recon: Sybil-resistant consensus from reputation," *Pervasive and Mobile Computing*, vol. 61, p. 101109, 2020.

[18] J. Long and R. Wei, "Nakamoto consensus with verifiable delay puzzle," *arXiv preprint arXiv:1908.06394*, 2019.

[19] R. Khalil and N. Dulay, "Short paper: Posh proof of staked hardware consensus," *ePrint*, 2020.

[20] J. Drake, "Minimal vdf randomness beacon," *Ethereum Research*, 2018.

[21] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. A. Gunter, "Leaky cauldron on the dark land: Understanding memory side-channel hazards in sgx," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 2421–2434.

[22] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, "Truspy: Cache side-channel information leakage from the secure world on arm devices." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 980, 2016.

[23] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 991–1008.

[24] E. M. Koruyeh, K. N. Khasawneh, C. Song, and N. Abu-Ghazaleh, "Spectre returns! speculation attacks using the return stack buffer," in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[25] B. Wesolowski, "Efficient verifiable delay functions," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 379–407.

[26] K. Pietrzak, "Simple Verifiable Delay Functions," in *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), A. Blum, Ed., vol. 124. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 60:1–60:15. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/10153

[27] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass, "Continuous verifiable delay functions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 125–154.

[28] M. Andrychowicz and S. Dziembowski, "Pow-based distributed cryptography with no trusted setup," in *Annual Cryptology Conference*. Springer, 2015, pp. 379–399.

[29] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[30] D. Cerezo Sánchez, "Zero-knowledge proof-of-identity: Sybil-resistant, anonymous authentication on permissionless blockchains and incentive compatible, strictly dominant cryptocurrencies," *Anonymous Authentication on Permissionless Blockchains and Incentive Compatible, Strictly Dominant Cryptocurrencies (May 22, 2019)*, 2019.

[31] DIF, "Decentralized identity foundation," https://identity.foundation, Mar 2021, [Online; accessed 12-Mar-2021].

[32] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, 1999, pp. 120–130.

[33] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis, "A survey on long-range attacks for proof of stake protocols," *IEEE Access*, vol. 7, pp. 28 712–28 725, 2019.

[34] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[35] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548*, 2018.

[36] C. Ganesh, C. Orlandi, and D. Tschudi, "Proof-of-stake protocols for privacy-aware blockchains," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 690–719.

[37] F. Baldimtsi, V. Madathil, A. Scafuro, and L. Zhou, "Anonymous lottery in the proof-of-stake setting," in *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*. IEEE, 2020, pp. 318–333.

Paper D

Non-Interactive VDF Client Puzzle for DoS Mitigation

M. Raikwar, D. Gligoroski

Published in European Interdisciplinary Cybersecurity Conference (EICC), ACM, 2021

Non-Interactive VDF Client Puzzle for DoS Mitigation

Mayank Raikwar
Norwegian University of Science and Technology
Trondheim, Norway
mayank.raikwar@ntnu.no

Danilo Gligoroski
Norwegian University of Science and Technology
Trondheim, Norway
danilog@ntnu.no

ABSTRACT

Denial of Service (DoS) attacks pose a growing threat to network services. Client puzzles have been proposed to mitigate DoS attacks by requiring a client to prove legitimate intentions. Since its introduction, there have been several constructions of client puzzles. Nevertheless, most of the existing client puzzles are interactive, where a server constructs a puzzle for a client request and asks the client to solve it before giving access to a resource. Additionally, most existing client puzzles do not provide desirable properties such as fairness, non-parallelizability, or non-interactivity. In this work, we propose a non-interactive client puzzle that achieves all these desired properties through a verifiable delay function (VDF). In a non-interactive puzzle, the client generates a puzzle and sends its solution along with the puzzle to access a resource of the server. We present different methods to generate verifiable client puzzles to prevent puzzle forgery and attacks from the client side. Further, we exhibit a transformation of the client puzzle into a DoS-resistant protocol. We also demonstrate the applicability of the DoS-resistant protocol in different contexts of the blockchain ecosystem.

CCS CONCEPTS

• **Security and privacy → Denial-of-service attacks**; **Distributed systems security**; **Cryptography**.

KEYWORDS

Denial-of-service, Client Puzzles, Verifiable Delay Function, Random Beacon, Blockchain

ACM Reference Format:
Mayank Raikwar and Danilo Gligoroski. 2021. Non-Interactive VDF Client Puzzle for DoS Mitigation. In *European Interdisciplinary Cybersecurity Conference (EICC), November 10–11, 2021, Virtual Event, Romania.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3487405.3487406

1 INTRODUCTION

Client puzzles, initially introduced by Dwork and Naor [12], have been considered a valuable defense mechanism against DoS attacks [22]. In a client puzzle, a client has to provide a solution to the puzzle before being granted access to a resource by a server. These puzzles, that counter resource depletion DoS attacks, are moderately hard to solve (in terms of CPU or memory usage, or time required to solve) and are called *proofs of work*. They can be turned on and off based on necessity and the server's amount of requested traffic. Many client puzzle constructions have been proposed [1, 5], which are primarily dedicated to rigorously defining the client puzzles based on the required computation, memory to solve the puzzle.

The motivation for DoS attacks can vary from a business competition, hacktivism, ransom, to politics. The intensity and frequency of DoS attacks have been continuously increasing every year, and that is why the DoS attack is considered one of the biggest threats for the Internet industries. In October 2016, a DDoS attack was mounted on DNS servers that provide DNS services for the major websites, including Netflix, Twitter, and PayPal, which resulted in a cut off the access to these websites for several hours [37]. These DoS attacks have also been mounted on many different places ranging from cloud services, IoT devices to cryptocurrencies and blockchain. Many works have been carried out in the past [11, 17] to detect and prevent these attacks.

1.1 Related Work

In a general context, a client puzzle demands a client to commit some of its resources by solving the puzzle. These resource types can vary, and depending upon the resource type, different types of client puzzles have been proposed. Some of the important puzzles are: *CPU-bound puzzles* which are quantified by the number of CPU cycles to solve the puzzle, *Memory-bound puzzles* which are quantified by the number of memory access to solve the puzzle, and *Network-bound puzzles* which are evaluated by the time required to solve the puzzle bounded by the network latency.

The earlier introduction of the CPU-bound (computation bound) puzzle was proposed by Back [5] in his Hashcash system, which was based on the difficulty of inverting a hash function. Although most of the computation bound puzzles follow the same idea, these puzzles have a drawback regarding the mismatch in the processing power available to the clients over time [33] (fairness). Moreover, these puzzles' parallelizability nature allows the clients to solve the puzzle substantially faster by utilizing customized hardware. Therefore, Abadi et al. [1] introduced memory-bound puzzles, which led to many subsequent constructions of memory-bound puzzles [13]. However, these puzzles are still not fair for resource-constrained devices. The memory-bound puzzles depend on the number of memory accesses (or cache misses). Additionally, another variant of this puzzle, *memory-hard puzzles* requires a significant amount of memory. Conclusively, these puzzles cannot provide fairness to resource-constrained devices (e.g., IoT).

Time-lock puzzle [28] is also an interesting cryptographic puzzle that encapsulates messages for a precise amount of time. The solution to these puzzles can only be computed sequentially by performing a deterministic number of steps. Nonetheless, these

EICC, November 10–11, 2021, Virtual Event, Romania
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9049-1/21/11…$15.00
https://doi.org/10.1145/3487405.3487406

Scheme	Based On	Construction Cost	Verification Cost	Granularity	State-less	Deter-ministic	Publicly Verifiable	Non Parallel	Non Interactive
Client Puzzles [22]	Multiple Hash Inversion	1 hash	1 hash	Exponential	✓	✗	✗	✗	✗
Hashcash [5]	Single Hash Inversion	1 hash	1 hash	Exponential	✓	✗	✓	✗	✓
Hint-based [14]	Single Hash Inversion	1 hash	1 hash	Linear	✗	✗	✗	✗	✗
Trapdoor-RSA [16]	RSA Trapdoor Problem	3 mod mul 2 additions	1 comparison	Linear	✗	✗	✗	✗	✗
Trapdoor-DLP [16]	Discrete Log Trapdoor	2 mod mul 3 additions	1 comparison	Linear	✗	✗	✗	✗	✗
D-H Puzzle [35]	Diffie-Hellman Problem	1 mod exp	1 comparison	Linear	✓	✗	✗	✗	✗
Subset Sum [34]	Subset-Sum Problem	1 hash	1 comparison	Polynomial	✓	✓	✓	✗	✗
Guided-Tour [2]	Network-Tour Puzzle	1 hash	1 comparison	Linear	✓	✓	✗	✓	✗
Time-Lock [28]	On Repeated Squaring	2 mod mul	2 mod mul	Linear	✗	✓	✗	✓	✗
Y-M Puzzle [20]	Single Hash Inversion	1 hash	1 hash	Exponential	✓	✗	✓	✗	✓
Y-M Puzzle [21]	Modular Square Root	c hash	c hash 1 mod exp	Polynomial	✓	✗	✓	✓	✓
Our Scheme	VDF Based Puzzle	1 hash	2 mod exp	Linear	✓	✓	✓	✓	✓

Table 1: Comparison Matrix of Different Client Puzzle Schemes for DoS Resistance.
In the table, '✓' indicates that the feature is present, and '✗' indicates that the feature is not present in the corresponding scheme. The operation *mod mul* represents modular multiplication; *mod exp* represents modular exponentiation; and *comparison* represents memory look-up.

puzzles are defined in a private-key setting, and henceforth these are not publicly verifiable.

Most of the existing client puzzles do not offer asymptotically efficient verification and public verifiability. Boneh et al. [7] defined verifiable delay function (VDF) that provides efficient verification and public verifiability. They mentioned VDF as a *moderately hard* cryptographic function, which can be viewed as a client puzzle, but they didn't explicitly formulate a client puzzle based on VDF. Their VDF construction can be directly applied in interactive client puzzles. Therefore, we construct a non-interactive VDF-based puzzle for DoS prevention where the server verifies the puzzle. We explored different ways to construct verifiable puzzles.

Most of the fundamental interactive client puzzle schemes lack the authentication of packets with the puzzle parameters sent by the server. Therefore, a malicious server or an adversary pretending to be the server can send fraudulent puzzle parameters to a client, leading to a DoS attack on the client or a connection refusal to the genuine server. Nevertheless, with the non-interactivity feature, a client does not need to authenticate any parameter sent by a malicious server, preventing a DoS attack against legitimate clients from a bogus server. Furthermore, a client can compute and solve a puzzle in an offline manner. Hence, the client does not need to remain connected with the server, in contrast to most existing schemes where the client engages its resources and remains connected with the server while solving the puzzle in an online fashion. Additionally, our scheme does not suffer from a DoS attack on puzzle verification where a client floods the bogus puzzle solutions to the server. So our scheme prevents DoS attacks in both directions, from adversarial server to legitimate clients and from adversarial clients to legitimate server. In addition, our puzzle captures the following essential properties.

(1) *Asymmetry* Puzzle verification is easier than solving it.
(2) *Granularity* A server is allowed to finely adjust the puzzle parameters (e.g., difficulty parameter, solution time).
(3) *Unforgeability* An adversary cannot forge a valid puzzle.
(4) *Fairness* Evaluation of a puzzle does not depend on the clients' processing power (available hardware).
(5) *Stateless* A server does not store information for verification.
(6) *Deterministic* The number of operations required for solving a client puzzle is deterministic.

(7) *Non-parallelizability* A client cannot gain any advantage from having multiple machines to solve the puzzle.
(8) *Public Verifiability* The puzzle solution should be verifiable by anyone without having any trapdoor information.

Furthermore, we present a comparison of various client puzzle schemes for DoS resistance with the idiosyncratic features of client puzzles in Table 1. Our scheme achieves all the essential properties of a client puzzle compared to the existing schemes. The only drawback of our scheme is the high verification cost compared to the existing schemes. Yves and Martin [21]'s scheme achieves most of the properties, but increasing puzzle difficulty in their scheme makes the verification too expensive and solution size larger; hence hindering its applicability in complex networks (blockchain). Moreover, their scheme does not entirely solve the replay attack problem which our scheme solves. Our scheme can be particularly useful for the mitigation of resource depletion DoS attacks.

1.2 Our Contribution

- We propose a novel non-interactive VDF client puzzle scheme.
- We present a transformation of a VDF client puzzle scheme to a DoS-resistant protocol.
- We analyze the security of our scheme and show a few applications of our protocol in the blockchain ecosystem.

2 VERIFIABLE DELAY FUNCTION

A verifiable delay function is a recent cryptographic primitive defined in 2018 by Boneh et al. [7]. Formally, it can be described as a function $f : \mathcal{X} \longrightarrow \mathcal{Y}$ which takes a predefined number of steps T to compute the output $y \in \mathcal{Y}$, given an input $x \in \mathcal{X}$; furthermore, the verification of the output is exponentially easy. VDF produces a unique output that is efficiently and publicly verifiable. Given a polynomial number of processors, the computation of the function cannot be done in less than T time. Since the introduction of VDF, to date, there are two main follow-up constructions of VDF based on modular exponentiation. The two main proposals, Wesolowski Scheme [36] and Pietrzak Scheme [27] define VDF as a tuple of the following algorithms.

- Setup(λ, T): It is a randomized algorithm that takes security parameter λ, time parameter T and outputs public parameter $pp := (\mathbb{G}, N, H, T)$, where $\mathbb{G}$ is a finite abelian group of unknown order, N is an RSA modulus, and $H : \mathcal{X} \longrightarrow \mathbb{G}$ is a hash function.
- Eval(pp, x, T): The evaluation algorithm applies T squarings in $\mathbb{G}$ starting with $H(x)$ and outputs the value $y \longleftarrow H(x)^{(2^T)} \bmod N$, along with a proof π.
- Verify(pp, x, y, π, T): The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

The security of these schemes depends on the factorization of N; otherwise, the computation of y can be reduced to simpler modular exponentiation. In more details, group $\mathbb{G}$ is $(\mathbb{Z}/N\mathbb{Z})^{\times}/\{\pm 1\}$, where $N = p.q$ and p and q are large primes. N is the public key pk and the corresponding secret key sk is $\phi(N) = (p-1)(q-1)$. The secret key is available to the verifier of the scheme. Therefore, the evaluator (who runs the Eval algorithm) has to compute 2^T squaring, but the verifier can do the same evaluation with two exponentiations by first computing $a = 2^T \bmod sk$, followed by $H(x)^a \bmod N$.

The two schemes differ in the proof generation and especially in the verification of the output. Both schemes have their own strengths. Pietrzak's construction has better performance, but the bigger proof size creates a huge overhead for the network. In comparison, Wesolowski scheme has fast verification (2 exponentiation versus 2 $log_2 T$) and shorter proof size (1 group element versus $log_2 T$) compared to the Pietrzak scheme. Therefore, we adapt the Wesolowski scheme to construct a VDF-based client puzzle (especially in a blockchain context), owing to the construction of the DoS-resistant protocol using the puzzle. Furthermore, the downside of the Wesolowski scheme is the need for more group operations for proof construction, which is also beneficial for making the client spending more time solving the puzzle in DoS-resistant protocol.

Wesolowski scheme can be defined as: Given the public parameters as $pp := (\mathbb{G}, N, H, T)$, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ run the following protocol to prove $y \longleftarrow g^{(2^T)} \bmod N$, where $g = H(x)$ and x is an input to the prover $\mathcal{P}$:

(1) The verifier $\mathcal{V}$ uniformly samples a random prime $l \xleftarrow{\$} Primes(\lambda)$, where $Primes(\lambda)$ contains the first 2^λ primes. Then, the verifier $\mathcal{V}$ sends l to the prover $\mathcal{P}$.
(2) Given l from the verifier $\mathcal{V}$, the prover $\mathcal{P}$ computes q, r such that $2^T = ql + r$ where $0 \leq r < l$, and sends a proof $\pi \longleftarrow g^q$ to the verifier $\mathcal{V}$.
(3) The verifier $\mathcal{V}$ computes $r \longleftarrow 2^T \bmod l$ and checks the two conditions: 1) $\pi \in \mathbb{G}$, 2) $y = \pi^l g^r \bmod N \in \mathbb{G}$. If both conditions satisfy, the verifier $\mathcal{V}$ outputs accept.

3 DOS-RESISTANT PROTOCOL

3.1 Non-Interactive VDF Client Puzzle

In a non-interactive puzzle scheme, a client C creates and solves the puzzle for its request Req and sends the solution along with the puzzle to a server $\mathcal{S}$ for the verification. The public parameters pp are generated by the server $\mathcal{S}$ using VDF Setup algorithm. The difficulty parameter T is included in pp and can be modified by the server to make the puzzle hard. Our puzzle scheme is as follows:

- Compute(Req): Client runs this algorithm to construct value i.
- GenPuz(T, i, pp): Client runs this algorithm to create a puzzle p. Client computes $g = H(i)$, and sets the puzzle $p = g$.
- FindSol(i, p, pp): Client runs this algorithm to solve the puzzle p. Client does the following computations
  (1) Compute $y \longleftarrow p^{(2^T)} \bmod N$ by performing 2^T squarings.
  (2) Compute $l = H_{prime}(i + y)$, $H_{prime} : \{0, 1\}^* \longrightarrow Primes(2\lambda)$.
  (3) Compute proof $\pi = i^{\lfloor 2^T / l \rfloor}$.
  Client sets solution $s = (y, l, \pi)$ and sends (s, p, i) to the server.
- VerSol(i, p, s, pp): Server verifies i, computes $r \longleftarrow 2^T \bmod l$, and accepts if $p, y, s \in \mathbb{G}$, $y = \pi^l p^r \bmod N$, and $l = H_{prime}(i + y)$.

Here, Compute(Req), where $Req \xleftarrow{\$} \{0, 1\}^*$, is the main constituent that makes the scheme secure. If the client C has complete control over the computation of i, C cannot only do the forgery, but C can also perform the following two attacks.

- *Pre-computation Attack* In this attack, a client constructs several valid puzzles and corresponding solutions in advance. Later, the client floods the server with these valid puzzle requests. To prevent it, there should be a limit either on the number of valid puzzles an attacker can generate or on the validity of the puzzles.

- *Replay Attack* In this attack, a client reuses the same puzzle solution for several requests. To prevent it, a binding between a request and the puzzle is necessary so that each new request should force the client to construct and solve a new puzzle.

To address the above two attacks and the puzzle forgery in non-interactive puzzles, the Compute(Req) algorithm should generate verifiable inputs for the puzzle, which the server can easily verify. This verifiable input should have randomness so that each client puzzle would be different for each client request, and hence, each solution will differ too. We propose and describe few methods to construct verifiable input i in Section 4.

3.2 DoS-resistant Protocol from Non-Interactive VDF Client Puzzle

We construct a non-interactive DoS-resistant protocol by utilizing a non-interactive VDF client puzzle. Due to its non-interactive nature, there is no need for server authentication. The puzzle scheme can be employed when the server is facing the DoS attack. Therefore, to avoid the waste of CPU resources and time when the server is not suffering from DoS, clients can send only request without a puzzle solution. In the normal operation case, the server responds regularly, but the server drops the client request during a DoS attack and switches from normal to DoS-Defend-Mode. A client not receiving any reply has to solve a puzzle and provide the solution to the server before being served. The server identifies the event of DoS by network traffic monitoring and analysis. The server can create an alert to switch to DoS-Defend-Mode upon experiencing a legitimate traffic spike or an abnormal traffic load. Additionally, the server can also set a threshold on the number of requests it can serve at a time based on the incoming and drop requests.

Figure 1 depicts the non-interactive DoS-resistant protocol, where e represents the extra information needed for the server to verify the correctness of the puzzle generation.

```
Non-Interactive DoS-resistant Protocol
Client C                                   Server S

i ← Compute(Req)
p ← GenPuz(T, i, pp)
s ← FindSol(i, p, pp)
                 (s, p, i, e)
        ------------------------------>
                                           If VerSol(i, p, s, pp) = 0,
                                           Reject
                                           Else,
                                           Serve the client request
```

Figure 1: VDF-based Non-Interactive DoS-resistant Protocol

4 VERIFIABLE INPUT GENERATION

4.1 From Random Beacon Server

We follow the similar idea of Yves and Martin [20] scheme where client and server share a common source of randomness. This source is a random beacon server B located in LAN, which broadcasts beacon packets containing random values. These beacon packets are generated in advance for a longer time span, and the beacon server B periodically broadcasts these packets. Therefore, public availability and the periodically changing nature of beacon packets make the puzzle valid for a short period. All the clients in LAN get a signed fingerprint of these beacon packets by the server B for easy verification. To make the server B DoS-resistant, the beacon only provides outgoing traffic, and all the incoming traffic is simply dropped without inspecting them. After every δ time, the current random value r_B is replaced by a new random value $r_{B+\delta}$. A client receiving the current randomness r_B creates an input i for a VDF puzzle p for his request message Req. A server receiving the puzzle first verifies i using the current randomness obtained from the beacon server B. The idea of a beacon server can be embraced on Internet by utilizing multicast or unicast transmission. We refer the readers to [20] for more details on the random beacon server.

A client computes i as $i \leftarrow (Req||r_B)$. Subsequently, the client generates the puzzle p, corresponding solution s. Further, the client sends (Request : s, p, i, Req) to the server. After receiving the client's Request, the server verifies the received i with a constructed i' on its end using current randomness r_B and received message Req. Further, it verifies the hash of i with the received p to check the puzzle integrity. Finally, it checks the puzzle solution s and grants the client request Request based on the check.

4.2 From Verifiable Random Function

In this approach, we use the concept of verifiable random function (VRF). A VRF has three algorithms: 1) $KeyGen(r) \rightarrow (sk, vk)$ for key generation; 2) $Eval(sk, M) \rightarrow (O, \pi)$ for producing a pseudorandom output O and a proof π; 3) $Verify(vk, M, O, \pi) \rightarrow 0/1$ for verification of output O. To generate publicly verifiable input for the client puzzle, each client applies VRF on its request message Req. To produce different puzzles for each client request message, the client uses a value x. This x is publicly available unique information that is periodically changing and accessible to the server. For example, this information can be the current block hash in the bitcoin blockchain (which changes every 10 minutes).

For each client request message Req, the client C computes an intermediate input $j \leftarrow (Req||x)$. Further, the client applies on input j as $VRF.Eval(sk_C, j) \rightarrow (i, \pi_i)$ using its secret key sk_C and constructs a pseudorandom output i and proof of the output π_i. Now, the client C uses input i to construct the puzzle p and henceforth the corresponding solution s. Further, the client sends (Request : s, p, i, π_i, Req). The server first verifies the proof π_i for the correct generation of valid input i for the puzzle p. For the verification, the server first retrieves current publicly available information x and uses it along with the received request message Req from the client to computes j'. Further, the server runs $VRF.Verify(vk_C, j', i, \pi_i)$, and verifies the input for the generated client puzzle. Finally, the server checks the puzzle solution s and grants the client request Request based on the check.

4.3 From Distributed Random Beacon Protocols

In this approach, the client and server get a periodically changing common randomness from a random beacon protocol. We assume the existence of a random beacon protocol that produces continuous randomness at a regular rate that is unpredictable, publicly verifiable, and bias-resistant. This randomness is available to the client and server. We studied and reviewed several constructions of random beacon protocols that differ in their cryptographic constituents, complexity, and properties. Finally, we propose the following protocols that can be used for verifiable input generation.

HERB [9] protocol is based on additive homomorphic encryption for perpetual randomness generation but requires a trusted dealer or a distributed key generation (DKG). *HydRand* [31] is a random beacon protocol that employs a publicly verifiable secret sharing scheme to generate consecutive random values in a byzantine setting. *HydRand* does not require a trusted dealer but has high complexity similar to *HERB*. Furthermore, a recent construction, *RandRunner* [30] does not require any trusted dealer and shows better complexity by avoiding the necessity of byzantine fault-tolerant. *RandRunner* exploits the functionality of VDF and exhibits better performance, safety, and liveness. With the recent advancements and interests in constructing new distributed random beacon protocols [6], we will further investigate on finding the suitable randomn beacon protocol constructions for our protocol. We will give a detailed construction of our non-interactive puzzle using one of the distributed randomness beacon protocols with complete security proofs in the full version of the paper.

After receiving the current common randomness r_b from the beacon, the further computation of i, p, s is similar to the construction with the random beacon server in Section 4.1.

4.4 Discussion

- *Replay Protection* All the above-described verifiable input generation methods prevent the pre-computation attack and puzzle forgery. They prevent the replay attack to some extent due to the periodic change in random value from the beacon or in the publicly available information. A client (or a set of colluding clients) having a request message *Req* can mount the replay attack on the server by sending many identical requests $(\mathrm{Request} : \mathrm{s}, \mathrm{p}, \mathrm{i}, \mathrm{e})$ for the same puzzle p constructed using *Req*. These identical requests can be identified by a queue Q maintained by the server. The queue Q of length l, keeps the client request messages *Req*'s for a time period t. The server decides the length l based on the number of client requests it can serve in time period t. Each request message is deleted after spending time t in Q. The time t is decided based on the refresh time t_r of the current random value, and network delay t_d to receive/retrieve the current randomness; such that $t > t_r + t_d$. Moreover, with the change in the random value, the request messages linked to the previous random value are deleted from Q, making previous puzzles no longer valid.
- *Randomness Update* The randomness used to construct verifiable inputs must be updated after a certain time. This update depends on the frequency of randomness generation. The frequency should be kept at a certain rate but should be changed depending on the clients' statistics about the use of expired randomness. In our client puzzle scheme, clients who indulge in solving their client puzzles using the current randomness should always retrieve the updated randomness. However, as a downside, some unlucky clients can be caught during the randomness update, as they have been solving the puzzle using the randomness that expired during the update. Therefore, the server receiving the client puzzles from those clients will reject those puzzles, resulting in a wastage of clients' computational power.
The more frequent randomness update happens, the more secure the server is against the replay and the pre-computation attack. Nonetheless, this might cause rejection of valid client requests due to the frequent change in randomness. Therefore, the randomness update time and VDF-puzzle solving time should be correlated and carefully chosen. The correlation and careful selection might reduce the number of client requests getting rejected. However, the problem of at least a few clients getting caught during the randomness update still persists. While a simple rejection solves these types of problems in practice, it would be interesting to analyze the implication of randomness update frequency in general. Another interesting direction would be to find a correlation among randomness update time, puzzle solving time, and network delay.
- *Comparison* The described three methods to generate verifiable random input differ significantly. Each method has its own advantage and disadvantage. 1) Randomness generation through a random beacon server can be easily applicable in organizations within a LAN. Therefore, all organizations can benefit from a single beacon server. However, as it is a centralized server, different attacks can be mounted against the beacon server, which further hinders the beacon server's availability, followed by the generation of fresh randomness. 2) Randomness generation from VRF shows that many public sources involving periodic change can be used to construct verifiable input for a client puzzle. The problem with this approach is the trust in the public source and its information update frequency. For example, in the case of the block hash of a blockchain as a public information source, problems might arise when a fork happens in the blockchain, and choosing the actual current block hash might lead to inconsistency for the clients. 3) From distributed randomness beacon protocols, the randomness generated for the client puzzle has all the properties of randomness beacon. Therefore, it can be easily verified by the clients and the server. The disadvantage with this approach is not being able to tweak the randomness update.

5 EVALUATION OF NON-INTERACTIVE VDF CLIENT PUZZLE

5.1 Security

In client puzzle schemes, an adversary $\mathcal{A}$ can perform two malicious actions, either by forging a valid client puzzle or by solving a client puzzle quickly without considering the difficulty parameter. However, our puzzle constructions already achieve puzzle unforgeability; therefore, this Section provides security against the adversary $\mathcal{A}$ trying to solve the puzzle quicker than expected time. To define security, we formalize the security notions in terms of games between an adversary $\mathcal{A}$ and a server $\mathcal{S}$. We first define necessary helper oracles that will be used to define a security game.

- $\mathsf{GetP}(i)$: Set a puzzle $p \longleftarrow \mathsf{GenPuz}(T, i, pp)$, record (i, p) in a list.
- $\mathsf{GetS}(i, p)$: If (i, p) was not recorded in the list by GetP, return $\perp$. Else, find a solution s such that $\mathsf{VerSol}(i, p, s, pp) =$ true. Record (i, p, s) and return s.
- $\mathsf{VerS}(i, p, s)$: Return true if all these conditions satisfy a) $\mathsf{VerSol}(i, p, s, pp) =$ true; b) (i, p) has been recorded by GetP; c) (i, p, s) has not been recorded by GetS. Else, return false.

Let λ be a security parameter, T be a difficulty parameter, and q be the number of queries made by an adversary $\mathcal{A}$ to each oracle. The security game between the adversary $\mathcal{A}$ and a server $\mathcal{S}$ for a VDF client puzzle scheme VP can be defined as follows:

```
Exec^A_{T,VP}(1^λ)
1 :  $(param, pp) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$
2 :  $\{(i_k, p_k, s_k) : k = 1, \ldots, q\} \leftarrow \mathcal{A}^{\mathsf{GetP},\mathsf{GetS}}(param)$
3 :  Return $\mathcal{A}^{\mathsf{VerS}}(param)$
```

In the above security game $\mathsf{Exec}^{\mathcal{A}}_{T,\mathsf{VP}}(1^\lambda)$, the adversary $\mathcal{A}$ make queries to the oracles, $\mathcal{A}$ wins only if $\mathcal{A}$ submits a correct solution s_j to a valid puzzle p_j (where $j \notin [1, \ldots, q]$) where s_j has not been recorded by GetS.

Theorem 5.1. *Let* VP *be a VDF client puzzle scheme with security parameter λ, and $T \in \mathcal{D}$. Let all the probabilistic polynomial-time adversaries $\mathcal{A}$ are making q number of queries to* GetP, GetS *in total. For the client puzzle* VP*, for all $i \in \mathcal{I}, p \in \mathcal{P}, s \in \mathcal{S}$, and for all adversaries $\mathcal{A}$ running in time $t \ll T$ the following condition holds:*

$$\Pr[\mathsf{Exec}^{\mathcal{A}}_{T,\mathsf{VP}}(1^\lambda) = \mathsf{true}] \leq \epsilon_{\lambda,q}(t) + \mathsf{negl}(\lambda). \quad (1)$$

where $\epsilon_{\lambda,q}(t)$ is a monotonically increasing function in t, such that for all $t, q : \epsilon_{\lambda,q}(t) \leq 1$, and it asymptotically reach 1 as the square of the number of queries q^2 gets closer to 2^λ.

The only way for an adversary $\mathcal{A}$ to solve VP in time $t \ll T$, is if $\mathcal{A}$ knows the factorization of modulus N. Thus, the sketch of the proof of the theorem follows the well-known *hard problem of factorization* and the reduction techniques introduced in the paper of Kurosawa and Takagi [24].

5.2 Performance Estimate

To estimate the performance, a few crucial points should be taken into account. The RSA modulus N as a setup parameter should be generated efficiently (in a decentralized way [8]), and the factorization should be unknown to the clients. We follow the implementation study of VDF [4] for performance estimation. The RSA group size has a significant impact on the evaluation time of the VDF puzzle. The evaluation time grows linearly with fine-tuning of the difficulty T of the puzzle. Let us consider the 128-bit security and the difficulty T between 2^{16} to 2^{20} for the DoS-resistant protocol. In this setting, the client can evaluate the puzzle in the $O(M)$ (order of minutes). The computation time for the proof of the puzzle output is approximately similar to the evaluation time. Therefore, the total running time for the FindSol algorithm in the protocol is in the order of minutes. Moreover, on the server-side, the verification of the client puzzle is in $O(ms)$ (order of milliseconds), which can be optimized further using Dimitrov's multiexponentiation [10]. The optimization in verification time will enhance the server's capability to verify more client puzzles during an event of DoS. Hence, with the above given estimate, our DoS-resistant protocol can be efficiently applied in real-world cases for DoS mitigation.

5.3 Applications

Besides the DoS mitigation, our non-interactive VDF client puzzle posses additional applicability. It can be applied in a variety of applications, e.g., metering client accesses (to solve the problem of forged client website visits) [15], achieving pseudonymous secure computation [23], constructing an offline submission protocol [19], achieving digital forgetting [3], and constructing a non-malleable commitment scheme [25]. Besides these applications, our VDF client puzzle can also be used to generate publicly verifiable proof of sequential work [26] with fast verification. Further, these proofs can be used to timestamp documents in a non-interactive manner.

Our DoS-resistant protocol also shows applicability in various domains. In what follows, we briefly present the pertinency of our DoS-resistant protocol in the blockchain ecosystem. Due to the lack of a trusted third party in a decentralized P2P system, and given the fact that our puzzle scheme is non-interactive along with other properties, our DoS-resistant protocol can be efficiently applied in the blockchain ecosystem. The setup parameter and difficulty (periodically update similar to bitcoin difficulty) is embedded in the blockchain. To generate different and publicly verifiable puzzle input i for each client C (blockchain user), its previously confirmed transaction tx_C on the blockchain is used as the input i for a puzzle p. In a case where the client announces its first transaction, it can use the timestamp as an input; this timestamp will also be present in the transaction to make it verifiable. Henceforth, a client C trying to publish a transaction during DoS attack, computes $i \longleftarrow tx_C$, constructs puzzle p, and corresponding solution s, and sends $(\mathsf{Request} : s, p, i)$ as defined in Section 3.1.

In most of the DoS events, the transaction flooding situation arises when the cost of creating a transaction is low. In most settings, these transactions are monetary payment transactions of a very tiny value, but for some cases, these can be data transactions (e.g., IoT blockchain transactions). By applying the non-interactive DoS-resistant protocol, we add a cost for creating a transaction, thereby reducing the attacker's throughput as in the following scenarios.

In a mempool, unconfirmed transactions are stored and picked by a miner for their addition to the blockchain. A DoS attack can be launched by flooding the mempool with several small fee transactions, making the other legitimate low fee transactions rejected by the miners [29]. Henceforth, it leads users to pay higher mining fees to prevent the rejection of their transactions. Our DoS-resistant protocol can force the clients to solve the VDF puzzle on its previously confirmed transaction before broadcasting a new transaction in the network and storing it in mempool.

A DoS attack can be mount on a mining pool by: 1) A hacker whose aim is to make money by asking the ransom from the attacked mining pool with the promise of stopping the DoS attack for the time being [18], 2) A competing mining pool whose goal is to increase his winning probability. Our non-interactive DoS-resistant protocol can be applied where the pool operator can act as the server and handles the client request.

Due to the small fee or no fee transactions in IoT blockchain networks [32], a DoS attack can be performed by creating several transactions. Our protocol can be applied where each node solves the puzzle before propagating its transaction in the network.

6 CONCLUSION

In this paper, we proposed a new client puzzle scheme that leverages the functionality of VDF. We explored the deficiencies in the existing client puzzle schemes and compared those schemes with our scheme. Our scheme achieves all the desirable properties, such as non-parallelizability and non-interactivity, that are preferable in DoS mitigation. Then, we constructed the DoS-resistant protocols from our new puzzle schemes. We further propounded the performance estimate of VDF in our protocols. To demonstrate the utility of the DoS-resistant protocol, we scrutinized applications of our protocol in the blockchain ecosystem.

Future Directions: There are various ways to adapt this work. The non-interactive puzzle construction can be further studied and optimized to generate verifiable random inputs for the puzzles that the server can efficiently verify. One possible way for efficient verification is to optimize the VDF verification time using parallelization and multi-exponentiation. Finding the applicability of our client puzzle scheme to combat some existing problems in existing systems or schemes can be an interesting avenue to explore. Another future direction is to have a formal analysis of our DoS-resistant protocol. Furthermore, studying the implication of randomness update in our DoS-resistant protocol will be an exciting area to research.

REFERENCES

[1] Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. 2005. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology (TOIT)* 5, 2 (2005), 299–327.

[2] Mehmud Abliz and Taieb Znati. 2009. A guided tour puzzle for denial of service prevention. In *2009 Annual Computer Security Applications Conference*. IEEE, 279–288.

[3] Ghous Amjad, Muhammad Shujaat Mirza, and Christina Pöpper. 2018. Forgetting with puzzles: using cryptographic puzzles to support digital forgetting. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 342–353.

[4] Vidal Attias, Luigi Vigneri, and Vassil Dimitrov. 2020. Implementation Study of Two Verifiable DelayFunctions. *IACR Cryptol. ePrint Arch.* 2020 (2020), 332.

[5] Adam Back et al. 2002. Hashcash-a denial of service counter-measure. *ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf* (2002).

[6] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. 2020. RandPiper-Reconfiguration-Friendly Random Beacons with Quadratic Communication. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1590.

[7] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer International Publishing, Cham, 757–788.

[8] Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and Abhi Shelat. 2020. Multiparty Generation of an RSA Modulus. In *Annual International Cryptology Conference*. Springer, 64–93.

[9] Alisa Cherniaeva, Ilia Shirobokov, and Omer Shlomovits. 2019. Homomorphic Encryption Random Beacon. *IACR Cryptol. ePrint Arch.* 2019 (2019), 1320.

[10] Vassil S Dimitrov, Graham A Jullien, and William C Miller. 2000. Complexity and fast algorithms for multiexponentiations. *IEEE Trans. Comput.* 49, 2 (2000), 141–147.

[11] Christos Douligeris and Aikaterini Mitrokotsa. 2004. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks* 44, 5 (2004), 643 – 666. https://doi.org/10.1016/j.comnet.2003.10.003

[12] Cynthia Dwork and Moni Naor. 1992. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*. Springer, 139–147.

[13] Cynthia Dwork, Moni Naor, and Hoeteck Wee. 2005. Pebbling and proofs of work. In *Annual International Cryptology Conference*. Springer, 37–54.

[14] Wu-chi Feng, Edward Kaiser, and Antoine Luu. 2005. Design and implementation of network puzzles. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Vol. 4. IEEE, 2372–2382.

[15] Matthew K Franklin and Dahlia Malkhi. 1997. Auditable metering with lightweight security. In *International Conference on Financial Cryptography*. Springer, 151–160.

[16] Yi Gao, Willy Susilo, Yi Mu, and Jennifer Seberry. 2010. Efficient trapdoor-based client puzzle against DoS attacks. In *Network Security*. Springer, 229–249.

[17] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. 2013. DoS and DDoS in Named Data Networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. 1–7. https://doi.org/10.1109/ICCCN.2013.6614127

[18] Stan Higgins. 2015. Bitcoin Mining Pools Targeted in Wave of DDOS Attacks. https://www.coindesk.com/bitcoin-mining-pools-ddos-attacks. [Online; accessed 07-June-2021].

[19] Yves Igor Jerschow and Martin Mauve. 2010. Offline submission with rsa time-lock puzzles. In *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 1058–1064.

[20] Yves Igor Jerschow and Martin Mauve. 2012. Secure client puzzles based on random beacons. In *International Conference on Research in Networking*. Springer, 184–197.

[21] Yves Igor Jerschow and Martin Mauve. 2013. Modular square root puzzles: Design of non-parallelizable and non-interactive client puzzles. *Computers & Security* 35 (2013), 25–36. https://doi.org/10.1016/j.cose.2012.11.008 Special Issue of the International Conference on Availability, Reliability and Security (ARES).

[22] Ari Juels. 1999. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. Networks and Distributed System Security Symposium (NDSS), 1999.*

[23] Jonathan Katz, Andrew Miller, and Elaine Shi. 2014. *Pseudonymous broadcast and secure computation from cryptographic puzzles.* Technical Report. Cryptology ePrint Archive, Report 2014/857, 2014. http://eprint. iacr. org ....

[24] Kaoru Kurosawa and Tsuyoshi Takagi. 2003. Some RSA-based encryption schemes with tight security reduction. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 19–36.

[25] Huijia Lin, Rafael Pass, and Pratik Soni. 2020. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. *SIAM J. Comput.* 49, 4 (2020), FOCS17–196.

[26] Mohammad Mahmoody, Tal Moran, and Salil Vadhan. 2013. Publicly verifiable proofs of sequential work. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 373–388.

[27] Krzysztof Pietrzak. 2018. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[28] Ronald L Rivest, Adi Shamir, and David A Wagner. 1996. Time-lock puzzles and timed-release crypto. *Massachusetts Institute of Technology, Laboratory for Computer Science* (1996).

[29] Muhammad Saad, Laurent Njilla, Charles Kamhoua, Joongheon Kim, DaeHun Nyang, and Aziz Mohaisen. 2019. Mempool Optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 285–292.

[30] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar Weippl. 2021. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. (2021).

[31] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. 2020. Hydrand: Efficient continuous distributed randomness. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 73–89.

[32] Jayasree Sengupta, Sushmita Ruj, and Sipra Das Bit. 2020. A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT. *Journal of Network and Computer Applications* 149 (2020), 102481. https://doi.org/10.1016/j.jnca.2019.102481

[33] Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Gonzalez Nieto. 2011. Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols. In *Topics in Cryptology – CT-RSA 2011*, Aggelos Kiayias (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 284–301.

[34] Suratose Tritilanunt, Colin Boyd, Ernest Foo, and Juan Manuel González Nieto. 2007. Toward non-parallelizable client puzzles. In *International Conference on Cryptology and Network Security*. Springer, 247–264.

[35] Brent Waters, Ari Juels, J Alex Halderman, and Edward W Felten. 2004. New client puzzle outsourcing techniques for DoS resistance. In *Proceedings of the 11th ACM conference on Computer and communications security*. 246–256.

[36] Benjamin Wesolowski. 2019. Efficient Verifiable Delay Functions. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 379–407.

[37] Nicky Woolf. 2016. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian* 26 (2016).

Paper E

DoS Attacks on Blockchain Ecosystem

M. Raikwar, D. Gligoroski

Published in Euro-Par 2021: Parallel Processing Workshop,
4th International Workshop on Future Perspective of Decentralized
Applications (FPDAPP), 2021

DoS Attacks on Blockchain Ecosystem

Mayank Raikwar and Danilo Gligoroski

Norwegian University of Science and Technology (NTNU) Trondheim, Norway
{mayank.raikwar,danilog}@ntnu.no

Abstract. Denial of Service (DoS) attacks are a growing threat in network services. The frequency and intensity of DoS attacks are rapidly increasing day by day. The immense financial potential of the Cryptocurrency market is a prevalent target of the DoS attack. The DoS attack events are kept on happening in cryptocurrencies and the blockchain ecosystem. To the best of our knowledge, there has not been any study on the DoS attack on the blockchain ecosystem. In this paper, we identify ten entities in the blockchain ecosystem and we scrutinize the DoS attacks on them. We also present the DoS mitigation techniques applicable to the blockchain services. Additionally, we propose a DoS mitigation technique by the use of verifiable delay function (VDF).

Keywords: Denial-of-service · Verifiable Delay Function· Non-Interactive · Blockchain

1 Introduction

Blockchain had brought a paradigm shift in digital innovation and the financial world since the advent of Bitcoin [26]. Today, the cryptocurrency market consists of 5424 cryptocurrencies that all together built a financial market worth around \$1.71 trillion (as of 26 May 2021) [9]. The immense financial potential of the cryptocurrency market has become a growing concern for the targeted attacks. Some of the well-known attacks in current blockchain systems are selfish mining, blockchain forks, 51% attack, double spending, Sybil attack, and Denial-of-service attacks [33]. A Denial-of-service (DoS) attack prevents legitimate user requests and depletes the server's resources. Due to the various configurations and decentralized features of blockchain, many of the attacks are preventable. Nevertheless, DoS attacks, especially its distributed variant (DDoS), are still prominent attacks on cryptocurrencies and blockchain-based applications.

Due to the increasing intensity and frequency of DoS attacks, it is contemplated as one of the biggest and severe threats for the Internet industries. One of the major DoS attacks was mounted on a DNS server in October 2016, which manifested in a cut of access to major websites, including PayPal, Netflix, and Twitter, for several hours [46]. The spectrum of DoS attacks can range from DNS services, cloud providers, IoT devices to the cryptocurrency and blockchain market. Nowadays, the cryptocurrency market is a popular target of DoS attacks, with the motivation of ransom, stealing funds, or business competition. In the past, many works[21,19,12] regarding the detection and prevention of DoS attacks have been carried out. Moreover, DoS/DDoS solutions based on blockchain are an emerging area of research. Applying the most recent advances of cryptographic research for the DoS/DDoS[1] problem can open new directions and avenues for addressing this ever-present problem.

In the general context of a DoS attack in blockchain, an adversary usually mounts a DoS attack when the cost of mounting an attack is very low. Therefore, various countermeasures, such as increased block size, increased transaction fees, or limiting transaction size have been proposed for mitigating the attacks. However, most of these countermeasures also force legitimate system users to invest their economic or computational power. This behavior shows a dire need to construct new methods for DoS prevention that do not require extra-economic or computational power of blockchain users. In this paper, we study and review DoS attacks on ten different entities in the blockchain ecosystem and possible mitigation techniques. In addition, we propose DoS mitigation by applying the astonishing functionality of verifiable random function (VDF) [8].

1.1 Related Work

Many DoS attacks have been mounted in the blockchain ecosystem and its services in the past few years. Some of these DoS attacks or threats on cryptocurrencies were disclosed a couple of years after they had been discovered. It requires new techniques to detect and counter the attack. Some of those new blockchain-based DoS mitigation techniques are devised from the decentralized nature and the deployed smart contracts of blockchain [30,36]. Even different machine learning techniques have been proposed to fight the DoS attack in cryptocurrency [14].

Specifically for the Bitcoin blockchain (as the blockchain of the most popular and valuable cryptocurrency), several DoS attacks have been mounted [40], which include mining pools, currency exchanges, eWallets, and financial services. Like most high visibility businesses, mining pools and currency exchanges are the primary DoS targets, which drives them to buy DDoS protection services such as Incapsula, CloudFlare, or Amazon Cloud. A report from September 2020 [18] revealed that the Bitcoin software implementation had a vulnerability for an uncontrolled memory consumption that was repeatedly used as a DoS vulnerability until it was patched in June 2018. This DoS vulnerability existed in many other branched Bitcoin implementations, including Litecoin and Namecoin.

Another major cryptocurrency, Ethereum [45] has also suffered from DoS attack [4]. In September 2016, a DoS attack against the Ethereum network was begun by exploiting a flaw in its client node. Furthermore, the same week, another DoS attack was mounted on the processing nodes of Ethereum [44]. A recent disclosure on Ethereum shows that a very cheap DoS attack could have brought down the Ethereum main net due to a bug in Geth Ethereum client [16].

[1] Throughout the paper, we use DoS to refer to both DoS and DDoS attacks, unless explicitly mentioned.

Recent work shows an Incentive-based blockchain denial of service attack (BDoS) [25] on Proof-of-work-based blockchains that exploits the reward mechanism to discourage the miner participation. This BDoS could theoretically be able to grind the (Bitcoin's) blockchain to a halt with significantly fewer resources (21% of the network's mining power). This attack raises a concern about the liveness of the Proof-of-work-based cryptocurrencies. This big concern and recent ongoing DoS attack disclosures compel researchers to find new ways to construct efficient DoS mitigation techniques.

1.2 Denial of Service Attack

A denial of service (DoS) attack targets to disrupt the availability of the network, server or application, and prevents legitimate requests from taking place. For a DoS attack to be successful, the attacker has to send more requests than the victim server can handle. These requests can be legitimate or bogus. The DoS attack depletes the server's resources such as CPU, memory, or network.

Definition 1. *(DoS): Let a server $\mathcal{S}$ be given, with the available resources $R_1, R_2, \ldots, R_n$ (R_i can be bandwidth, memory, CPU etc.). Let $\mathcal{A}$ or a set of $\{\mathcal{A}_j\}$ are an attacker or a set of attackers and let the legitimate users are represented by the set $\{\mathcal{U}_k\}$. A DoS attack on server $\mathcal{S}$ is expressed by a set of probabilities for successful resource-depletion $\{P_{R_1}, P_{R_2}, \ldots, P_{R_n}\}$. The total probability for a success of a DoS attack is then a probability the server $\mathcal{S}$ to refuse legitimate transactions from a user u, where $u \in \{\mathcal{U}_k\}$ and is modeled as the probability of blocking the legitimate traffic in at least one of the resources:*

$$P_{DoS} = 1 - (1 - P_{R_1})(1 - P_{R_2}).\ldots.(1 - P_{R_n}) \tag{1}$$

Note that the situation when attacker(s) exhausts at least one resource R_i implies the attack probability is $P_{R_i} = 1$, which from equation (1) further leads to $P_{DoS} = 1$.

DoS attacks can be categorized into several categories based on network and application layers or volume and protocol attacks. Network-level DoS attacks aim to overload the server's bandwidth or cause CPU usage issues. However, application-level DoS attacks focus on applications, websites, or online services.

1.3 Our Contribution

The contributions of our work are as follows:

1. We thoroughly investigate the DoS attacks in the blockchain ecosystem.
2. We present different mitigation techniques of DoS attacks in the blockchain ecosystem.
3. We propose a VDF-based DoS resistant protocol by using the functionality of VDF.

The rest of the paper is as follows: Section 2 shows a detailed analysis of DoS attacks in the blockchain ecosystem. Further, Section 3 presents DoS mitigation techniques, including our VDF-based proposal. Finally, in Section 4, we conclude the paper and discuss the possible future directions.

2 DoS Attacks on Blockchain Ecosystem

The blockchain ecosystem has suffered from many DoS attacks in the past, and that situation is a continuing trend. The DoS attack can be launched against a specific entity or a network in the blockchain. We present a nonexclusive list of ten entities in the Blockchain ecosystem with their corresponding DoS attacks.

1. *On cryptocurrency wallets* A crypto wallet is a software program in which a user stores cryptocurrency. The wallet contains a set of signing keys for the user to sign new transactions. Wallets are also integrated with decentralized applications (DApps) to hold and manage users' signing keys and transactions securely. In a wallet service, a user is the sole owner of his account keys. However, if someone steals the signing keys, then the cryptocurrency held in that account can be spent. Therefore, hardware wallets (e.g., TREZOR) are ways to store cryptocurrency and the signing key in an offline manner. Nevertheless, online wallets are still a preferable choice for blockchain users. These online crypto wallets also suffer from DoS attacks [28] due to inconsistency in its smart contracts that further hinders the services of integrated DApps. Recently, a DDoS attack was mounted on the Wasabi bitcoin wallet [15].
2. *On cryptocurrency exchange services* A cryptocurrency exchange allows clients to buy, sell and store crypto-currencies at some exchange rate and leverages the clients to trade their currencies and earn some profit due to the fluctuations in the price of currencies. Besides, the exchange charges some fee for every trade made by its client and also converts the cryptocurrency into fiat currencies. Many exchange services also provide a wallet, but the wallet signing keys are controlled by the exchange service apart from the wallet user. Furthermore, these exchange services are online platforms, hence susceptible to DoS attacks that can cause the temporary unavailability of the platform. In the past, many of the crypto-currency exchange services were jeopardized by the DoS attacks (especially DDoS). One such example is the Bitcoin exchange platform, *Bitfinex* which has been a victim of DDoS attacks several times [2]. Another well-known bitcoin exchange service, Mt. Gox, was completely disrupted by DDoS attacks over time [17]. Over the years, many cryptocurrency exchange platforms have suffered DoS attacks. Recently, a UK-based exchange *EXMO* was hit by DDoS attack [10].
3. *On memory (transaction) pools* A memory pool (mempool) is a repository of unconfirmed transactions in a cryptocurrency blockchain, e.g., Bitcoin. Once a user creates a new transaction, it is broadcast to the network and stored in the mempool. In the mempool, the transaction waits to be picked by a miner to be added in a block and subsequently to the blockchain, therefore acquiring the transaction's confirmation. If a transaction remains unconfirmed for a long time in the mempool, it gets rejected eventually. As the transactions with high fees are most likely to be selected by a miner, it poses a threat to flood the mempool by small fee transactions, consequently affecting the mempool size. In that direction, it creates uncertainty among the users for their transactions and leads them to pay higher mining fees to prevent the

rejection of their transactions. The work [34] studies such an attack on Bitcoin mempool and proposes a few countermeasures. However, the proposed solutions have limitations regarding the minimum payable fee and rejection of fast transactions. A follow-up work [32] provides similar prevention measures for Proof-of-work-based blockchain but suffers from the similar problems.

4. *On mining pools* Mining pools are the major players in Proof-of-work-based cryptocurrencies, e.g., Bitcoin. The mining pool's goal is to accumulate miners' power and solve the Proof-of-work puzzles. As the difficulty of Proof-of-work puzzles gives a very low probability of solving the puzzle to a single miner, the miner usually prefers to join a mining pool where the miner gets a fair share of the reward proportional to his/her effort, if the mining pool finds the solution. Two kinds of entities can mount a DDoS attack on a mining pool: 1) A hacker whose aim is to make money by asking the ransom from the attacked mining pool with the promise of stopping the DDoS attack [22], 2) A competing mining pool whose goal is to increase his winning probability by undermining the power of competing mining pools. Few game-theoretic studies [48,47] are also conducted to analyze DoS attacks in mining pools.
5. *On layer-two blockchain protocols* Layer-two blockchain protocols are built on the top of the main blockchain that moves a sufficient amount of transaction load from the main blockchain to the off-chain in sub-seconds (instead of minutes or hours) with a reduced fee and similar security. Hence, these protocols are referred to as an orthogonal solution for the scalability problem in the blockchain. In recent years, there has been tremendous growth in constructing new layer-two protocols [20] for blockchain scalability such as channel networks. In a channel network, channels are established between the parties of the network and governed by the smart contracts of the main chain. It provides a fast and scalable approach for off-chain interactions. These protocols also suffered from DoS attacks in the past [39,42].
6. *On sharding protocols* Similar to layer-2 blockchain solutions, sharding protocols [41] also tackle the scalability issue of blockchain. The idea of sharding is to partition the blockchain state into multiple shards. Each shard processes a set of transactions; therefore, all shards can process the transactions parallelly and hence increases the blockchain throughput. The majority of the sharding protocols are built on the top of the Bitcoin blockchain, and some are built for the Ethereum blockchain. A sharding protocol deals with challenges involving the shard assignment to validators, transaction assignment to shard, and intra-shard consensus. A DoS attack can be mounted on sharding protocol by flooding a single shard which becomes the bottleneck for the whole system. A recent work [27] studies the DoS-attack on sharding protocols and proposes a Trusted Execution Environment (TEE) based countermeasure.
7. *On commit-chain operator* A commit-chain [23] is an off-chain scaling solution where the transactions are performed off-chain by a non-custodial and untrusted operator. The operator commits the balances of users periodically to the blockchain by computing a checkpoint and feeding it to an on-chain smart contract. The scheme involves users publishing challenges to the smart contract in case of a dispute with the operator, which imposes a drawback

where a malicious user can flood the smart contract with unwarranted challenges. Another significant issue is the operator being a central entity can become a victim of a DoS attack, resulting in collapsing the whole system.

8. *On smart contract* A smart contract is a transaction protocol in blockchain that takes actions according to the terms of the contract. In the Ethereum blockchain, each block has a maximum gas limit that is spent by executing a smart contract, and exceeding the gas limit causes a DoS attack. An attacker can mount a DoS attack on smart contract [4] in several possible ways such as: 1) By sending a computationally intensive transaction to a contract thus preventing other transactions from being included in a block; 2) By adding a couple of refund addresses at once that can end up smart contract exceeding the gas limit while refunding to those addresses; 3) By unexpected revert of refund to a legitimate user by using fallback function. A recent work [35] shows a method to detect DoS attacks caused due to unexpected revert in Ethereum smart contract. An example of a DoS attack on a smart contract is an auction contract where an attacker can constantly call the bidding function (e.g., *bid()*), preventing other legitimate users from making their bids. In the NEO blockchain, a vulnerability allowed attackers to invoke a malicious contract that created a DoS attack by crashing each node that tried to execute the contract [37]. Moreover, a DoS attack on a smart contract triggers stopping a node from executing the functions for all the DApps it hosts.
9. *On mixing services* A mixing service is a protocol that allows a cryptocurrency user to utilize its currency anonymously. It provides unlinkability of the user's input to its output and prevents the user's identification from being revealed. There are centralized [6] and decentralized [31] mixing services. Centralized mixing services being a single-point-of-failure are more vulnerable to DoS attacks (e.g. by competing services). However, both types of mixing services suffer from DoS due to different actions of its users, such as 1) By providing inconsistent input for the shuffle, leading the whole verification step of shuffle to fail; 2) By denying to perform some required task e.g., to sign a group transaction; 3) By several participation requests in the mixing transaction pool leading to the depletion of a precomputed pool by participants [49].
10. *On consensus participants* In the blockchain, consensus participants are the major players who decide on the blockchain's new block. Therefore, consensus participants are the usual DoS target for an attacker. In deterministic leader election protocols of consensus, the leader of the consensus round can be a primary target for DoS attacks which can make the whole system halt if the leader suffers a DoS attack. Other main targets can be stakeholders in Proof of Stake consensus mechanisms that hold some stake in the system, therefore attracting an attacker to mount DoS. A DoS attack can be mounted on PBFT-based permissioned blockchains and its participants, where a DDoS attack can be launched if an adversary controls over 33 % of the replicas. As in the BFT-based blockchains, network size is known to the participants, an attacker creates the required number of Sybil replicas needed for a DoS attack. Hence, for each transaction sent by the primary, the Sybil replicas will not reply to their approvals, leading the whole system to halt.

3 DoS Mitigation Techniques for Blockchain Systems

In most of the DoS events, an attacker floods the network by creating multiple transactions in a short time period, hence maximizing his throughput. This kind of situation arises when the cost of creating a transaction is low. In most settings, these transactions are monetary payment transactions of a tiny value, but for some cases, these can be data transactions (e.g., IoT blockchain transactions). To mitigate the DoS attack, some cost should be imposed on the attacker to slow down or stop unnecessary requests in the blockchain system. Hence, following, we present the DoS mitigation techniques in the blockchain ecosystem.

Blockchain Ecosystem	**Applicable Solutions**
Cryptocurrency Wallets	Client Puzzle (Inside Smart Contract)
Cryptocurrency Exchange Services	Client Puzzle (On Exchange Clients)
Memory Pools	Fee-based Approach/NI-Client Puzzle
Mining Pools	Fee-based Approach/NI-Client Puzzle
Layer-2 Blockchain Protocols	Fee-based Approach
Sharding Protocols	Fee-based Approach
Commit-chain Operator	Client Puzzle (On Commit-chain Users)
Smart Contract	Client Puzzle (Inside Smart Contract)
Mixing Services	Fee-based Approach/NI-Client Puzzle
Consensus Participants	Client Puzzle (On Participant Registration)

Table 1. DoS Mitigation Techniques in Blockchain Ecosystem

- *Client Puzzles* Client puzzles are one of the most effective prominent techniques to defend against DoS attacks. In a client puzzle, a client has to solve a puzzle before being granted access to a service or a resource by a server. The initial introduction of the client puzzle was given by Dwork and Naor [13] to combat the spam attacks. Client puzzles can be categorized into different types based on the resource used by the client for solving the puzzle such as number of CPU cycles or a number of memory access, quantifying CPU-bound puzzles [5] and memory-bound puzzles [1] respectively. Several client puzzles such as Time-lock puzzles [29], Hash-chain [24] and Equihash [7] are employed in the blockchain ecosystem. A client puzzle scheme can be *Interactive* where server creates the puzzle for the client or *Non-Interactive* (NI) where the client creates a puzzle, solves the puzzle and sends it to the server.
- *Fee-based Approach* In many events of DoS attack, to disincentivize an attacker an extra or minimum fee can be introduced in the blockchain ecosystem. This fee can be of different types based on the underlying blockchain system. The fee can be a mining fee in mining pools, a mixing fee in mixing services, a transaction fee in transaction pools, a relay fee in a blockchain network, a registration fee for user registration (e.g. a user of a permissioned blockchain), etc. Therefore, with the introduction of a minimum fee, launching a DoS attack becomes costlier for an attacker. However, the fee-based approach adversely affects legitimate users who do not want to pay this minimum amount of fee.

Table 1 presents the possible DoS mitigation solutions for corresponding blockchain ecosystem. Fee-based approach can be applied in almost every case but will not be favorable for all blockchain users. In the table, for layer-2 and sharding protocols, the use of client puzzle will defeat the purpose of scalability due to its time consumption, therefore fee-based approach is a more viable option. For memory pools, mining pools, and mixing services, non-interactive client puzzle schemes can be applied where the miner/user presents a verifiable puzzle and its solution for the inclusion of its new transaction (Rewarding puzzle solution in case of mining pool). Apart from the above described techniques, other mechanisms such as packet filtering techniques or DoS protection services e.g. Incapsula can be used for DoS mitigation in some blockchain contexts.

3.1 VDF-based DoS-resistant Protocol

Most of the existing client puzzles lack public verifiability, non-parallelizability, non-interactivity, and easy verification. Therefore, the initial introduction of VDF [8] as a moderately hard function can be configured as a client puzzle for DoS mitigation achieving all these properties. A VDF can be described as a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ which takes a predefined number of steps T to compute the output $y \in \mathcal{Y}$, given an input $x \in \mathcal{X}$ and a polynomial number of processors. Furthermore, the verification of the output is exponentially easy. VDF produces a unique output that is efficiently and publicly verifiable. There have been a few constructions of VDF. We employ the Wesolowski VDF scheme [43] to construct our client puzzle due to its fast verification and short proof size properties.

We define an Interactive VDF client puzzle, where a server $\mathcal{S}$ creates a puzzle p and asks for solution s of the puzzle from the client $\mathcal{C}$ before giving access to its resource. In the following construction, $\mathcal{K}$ is a key space, $\mathcal{P}$ is a puzzle space, $\mathcal{O}$ is a solution space, $\mathcal{D}$ is a puzzle difficulty space, and $\mathcal{I}$ is a puzzle input space.

- $\mathsf{Setup}(1^\lambda)$: Select $\mathcal{K} = \varnothing, \mathcal{D} \subseteq \mathbb{N}, \mathcal{P} \subseteq \{0,1\}^*, \mathcal{O} \subseteq \{0,1\}^*, \mathcal{I} \subseteq \{0,1\}^*$. Generate a group $\mathbb{G}$ of unknown order, an RSA modulus N, a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathcal{D} \leftarrow T$. Set $param \leftarrow (\mathcal{P}, \mathcal{O}, \mathcal{D}, \mathcal{I})$ and $pp \leftarrow (\mathbb{G}, N, H, T)$, return $(param, pp)$.
- $\mathsf{GenPuz}(T, i, pp)$: Server runs this algorithm to create a puzzle for the client. It generates an input $i \in \mathcal{I}$ for VDF-evaluation, samples $l \overset{\$}{\leftarrow} Primes(\lambda)$. Return a puzzle $p = l$ to the client.
- $\mathsf{FindSol}(i, p, pp)$: Client runs this algorithm to solve the puzzle p. Client computes $g = H(i)$, further computes $y \leftarrow g^{(2^T)} \mathsf{mod}\, N$. It computes q, r such that $2^T = ql + r$ where $0 \leq r < l$, and computes a proof $\pi = g^q$. Send a solution $s = (y, \pi)$ to the server.
- $\mathsf{VerSol}(i, p, s, pp)$: Server computes $r \leftarrow 2^T \mathsf{mod}\, l$ and accepts if $g, y, s \in \mathbb{G}$ and $y = \pi^l g^r \mathsf{mod}\, N$.

An Interactive VDF-based DoS-resistant protocol can be designed using client puzzle as depicted in Figure 1. The protocol construction follows from the Stebila et al. [38]. To define this interactive protocol, we assume server and client have public identities $ID_{\mathcal{S}}$ and $ID_{\mathcal{C}}$. Our VDF-based client puzzle can also be made Non-Interactive where the client constructs a puzzle and its solution. The client and server share a common source of randomness (e.g. random beacon). The client creates publicly verifiable puzzles using randomness. Further, the non-interactive VDF client puzzle can be transformed into a DoS-resistant protocol that can be efficiently applied in the blockchain ecosystem during DoS events.

Interactive DoS-resistant Protocol

Client $\mathcal{C}$		**Server $\mathcal{S}$**
$SK_{\mathcal{C}}, ID_{\mathcal{C}}, N_{\mathcal{C}} \overset{\$}{\leftarrow} \{0,1\}^k$		$SK_{\mathcal{S}}, ID_{\mathcal{S}}$
	$\xrightarrow{(\mathsf{Request} : ID_{\mathcal{C}}, N_{\mathcal{C}})}$	
		$N_{\mathcal{S}} \overset{\$}{\leftarrow} \{0,1\}^k$ $i \leftarrow (ID_{\mathcal{C}}, ID_{\mathcal{S}}, N_{\mathcal{C}}, N_{\mathcal{S}})$ $p \leftarrow \mathsf{GenPuz}(T, i, pp)$, $\sigma \leftarrow \mathrm{MAC}_{SK_{\mathcal{S}}}(i, p)$
	$\xleftarrow{(\mathsf{Challenge} : N_{\mathcal{S}}, p, \sigma)}$	
$i \leftarrow (ID_{\mathcal{C}}, ID_{\mathcal{S}}, N_{\mathcal{C}}, N_{\mathcal{S}})$ $s \leftarrow \mathsf{FindSol}(i, p, pp)$		
	$\xrightarrow{(\mathsf{Response} : s, p, i, \sigma)}$	
		If $ID_{\mathcal{C}} \in \{$List of Recorded IDs$\}$, Reject If $\sigma \neq \mathrm{MAC}_{SK_{\mathcal{S}}}(i, p)$, Reject If $\mathsf{VerSol}(i, p, s, pp) = 0$, Reject Store $ID_{\mathcal{C}}$

Fig. 1. Interactive DoS-resistant Protocol

Following the implementation study of VDF [3], for 128-bit security and the difficulty between 2^{16} to 2^{20}, our DoS-resistant protocol can be efficiently employed for DoS mitigation in the blockchain. With the aforementioned setting, the running time for FindSol, VerSol algorithms are in order of minutes and order of milliseconds respectively. The verification time on the server side can be further optimized using Dimitrov's multiexponentiation method [11]. As a future work, we will put a demonstration of a proof-of-concept and initial experiments with Wesolowski VDF for DoS mitigation.

4 Conclusion

In this work, we offered a thorough study of DoS attacks in the blockchain ecosystem. To the best of our knowledge, this is the first investigation in the context of blockchain. As the frequency and intensity of DoS attacks are increasing rapidly, it raises a concern about efficient detection and mitigation techniques. Therefore, we listed out main mitigation approaches which can be used for DoS mitigation in the blockchain ecosystem. We also identify verifiable delay function as an effective primitive to mitigate DoS attacks. A proper construction of non-interactive VDF puzzle and experimental results will be provided in the continuation of this work. This paper will help academic and industrial researchers to study the possible venues and impact of the DoS attack in the blockchain context and to improve upon the existing solutions.

References

1. Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. ACM Transactions on Internet Technology **5**(2), 299–327 (2005)
2. Abhishta, A., Joosten, R., Dragomiretskiy, S., Nieuwenhuis, L.J.M.: Impact of Successful DDoS Attacks on a Major Crypto-Currency Exchange. In: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). pp. 379–384 (2019)
3. Attias, V., Vigneri, L., Dimitrov, V.: Implementation Study of Two Verifiable Delay Functions. IACR Cryptol. ePrint Arch. **2020**, 332 (2020)
4. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on ethereum smart contracts (sok). In: Maffei, M., Ryan, M. (eds.) Principles of Security and Trust. pp. 164–186. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)
5. Back, A., et al.: Hashcash - a denial of service counter-measure. ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf (2002)
6. Bao, Z., Shi, W., Kumari, S., Kong, Z.y., Chen, C.M.: Lockmix: a secure and privacy-preserving mix service for Bitcoin anonymity. International Journal of Information Security pp. 1–11 (2019)
7. Biryukov, A., Khovratovich, D.: Equihash: Asymmetric proof-of-work based on the generalized birthday problem. Ledger **2**, 1–30 (2017)
8. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 757–788. Springer International Publishing, Cham (2018)
9. CoinMarketCap: Total market capitalization. `https://coinmarketcap.com` (May 2021), [Online; accessed 26-May-2021]
10. Crawley, J.: UK Crypto Exchange EXMO Offline Amid DDoS Attack. `https://tinyurl.com/u8kk94ry` (Feb 2021), [Online; accessed 08-June-2021]
11. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Complexity and fast algorithms for multiexponentiations. IEEE Transactions on Computers **49**(2), 141–147 (2000)
12. Douligeris, C., Mitrokotsa, A.: Ddos attacks and defense mechanisms: classification and state-of-the-art. Computer Networks **44**(5), 643 – 666 (2004)
13. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference. pp. 139–147. Springer (1992)

14. Eduardo A. Sousa, J., Oliveira, V.C., Almeida Valadares, J., Borges Vieira, A., Bernardino, H.S., Moraes Villela, S., Dias Goncalves, G.: Fighting Under-price DoS Attack in Ethereum with Machine Learning Techniques. ACM SIGMETRICS Performance Evaluation Review **48**(4), 24–27 (2021)
15. Explica.co: Cryptocurrency : Wasabi bitcoin wallet servers suffered DDoS attack. `https://tinyurl.com/s6sbunam` (June 2021), [Online; accessed 10-June-2021]
16. Fadilpasic, S.: Disclosed: Ethereum 'Lived' With a Major Threat for 18 Months. `https://tinyurl.com/h7478aej` (May 2021), [Online; accessed 07-July-2021]
17. Feder, A., Gandal, N., Hamrick, J., Moore, T.: The impact of DDoS and other security shocks on Bitcoin currency exchanges: Evidence from Mt. Gox. Journal of Cybersecurity **3**(2), 137–144 (2017)
18. Fuller, B., Khan, J.: CVE-2018-17145: Bitcoin Inventory Out-of-Memory Denial-of-Service Attack. https://invdos.net/paper/CVE-2018-17145.pdf (2020)
19. Gasti, P., Tsudik, G., Uzun, E., Zhang, L.: DoS and DDoS in Named Data Networking. In: 2013 22nd International Conference on Computer Communication and Networks (ICCCN). pp. 1–7 (2013)
20. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: Sok: Layer-two blockchain protocols. In: Financial Cryptography and Data Security. pp. 201–226. Springer International Publishing, Cham (2020)
21. Gupta, B., Badve, O.P.: Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. Neural Computing and Applications **28**(12), 3655–3682 (2017)
22. Higgins, S.: Bitcoin Mining Pools Targeted in Wave of DDoS Attacks. `https://tinyurl.com/5jew979z` (March 2015), [Online; accessed 07-July-2021]
23. Khalil, R., Zamyatin, A., Felley, G., Moreno-Sanchez, P., Gervais, A.: Commit-Chains: Secure, Scalable Off-Chain Payments. Tech. rep., Cryptology ePrint Archive, Report 2018/642 (2018)
24. Mahmoody, M., Moran, T., Vadhan, S.: Publicly verifiable proofs of sequential work. In: Proceedings of the 4th conference on Innovations in Theoretical Computer Science. pp. 373–388 (2013)
25. Mirkin, M., Ji, Y., Pang, J., Klages-Mundt, A., Eyal, I., Juels, A.: BDoS: Blockchain Denial-of-Service. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 601–619. CCS '20, ACM, NY, USA (2020)
26. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf (2009)
27. Nguyen, T., Thai, M.T.: Denial-of-service vulnerability of hash-based transaction sharding: Attacks and countermeasures. arXiv preprint arXiv:2007.08600 (2020)
28. Praitheeshan, P., Pan, L., Doss, R.: Security Evaluation of Smart Contract-Based On-chain Ethereum Wallets. In: International Conference on Network and System Security. pp. 22–41. Springer (2020)
29. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Massachusetts Institute of Technology, Laboratory for Computer Science (1996)
30. Rodrigues, B., Bocek, T., Stiller, B.: Multi-domain ddos mitigation based on blockchains. In: Tuncer, D., Koch, R., Badonnel, R., Stiller, B. (eds.) Security of Networks and Services in an All-Connected World. pp. 185–190. Springer International Publishing, Cham (2017)
31. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: Practical decentralized coin mixing for Bitcoin. In: European Symposium on Research in Computer Security. pp. 345–364. Springer (2014)

32. Saad, M., Njilla, L., Kamhoua, C., Kim, J., Nyang, D., Mohaisen, A.: Mempool Optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 285–292. IEEE (2019)
33. Saad, M., Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D., Mohaisen, A.: Exploring the attack surface of blockchain: A systematic overview. arXiv preprint arXiv:1904.03487 (2019)
34. Saad, M., Thai, M.T., Mohaisen, A.: POSTER: deterring ddos attacks on blockchain-based cryptocurrencies through mempool optimization. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp. 809–811 (2018)
35. Samreen, N.F., Alalfi, M.H.: SmartScan: An approach to detect Denial of Service Vulnerability in Ethereum Smart Contracts. preprint arXiv:2105.02852 (2021)
36. Singh, R., Tanwar, S., Sharma, T.P.: Utilization of blockchain for mitigating the distributed denial of service attacks. Security and Privacy **3**(3), e96 (2020). https://doi.org/https://doi.org/10.1002/spy2.96
37. Sotnichek, M.: NEO Smart Contract Vulnerabilities: DoS Vulnerability. `https://tinyurl.com/faxjbby5` (October 2018), [Online; accessed 07-July-2021]
38. Stebila, D., Kuppusamy, L., Rangasamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. pp. 284–301. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
39. Tochner, S., Zohar, A., Schmid, S.: Route Hijacking and DoS in Off-Chain Networks, p. 228–240. ACM, New York, NY, USA (2020)
40. Vasek, M., Thornton, M., Moore, T.: Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) Financial Cryptography and Data Security. pp. 57–71. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
41. Wang, G., Shi, Z.J., Nixon, M., Han, S.: SoK: Sharding on blockchain. In: 1st ACM Conference on Advances in Financial Technologies. pp. 41–61 (2019)
42. Weintraub, B., Nita-Rotaru, C., Roos, S.: Exploiting Centrality: Attacks in Payment Channel Networks with Local Routing (2020)
43. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 379–407. Springer International Publishing, Cham (2019)
44. Wilcke, J.: The Ethereum network is currently undergoing a DoS attack. `https://tinyurl.com/ww6kp2nu` (2016), [Online; accessed 07-July-2021]
45. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)
46. Woolf, N.: DDoS attack that disrupted internet was largest of its kind in history, experts say. The Guardian **26** (2016)
47. Wu, S., Chen, Y., Li, M., Luo, X., Liu, Z., Liu, L.: Survive and Thrive: A Stochastic Game for DDoS Attacks in Bitcoin Mining Pools. IEEE/ACM Transactions on Networking **28**(2), 874–887 (2020)
48. Zheng, R., Ying, C., Shao, J., Wei, G., Yan, H., Kong, J., Ren, Y., Zhang, H., Hou, W.: New Game-Theoretic Analysis of DDoS Attacks Against Bitcoin Mining Pools with Defence Cost. In: International Conference on Network and System Security. pp. 567–580. Springer (2019)
49. Ziegeldorf, J.H., Matzutt, R., Henze, M., Grossmann, F., Wehrle, K.: Secure and anonymous decentralized Bitcoin mixing. Future Generation Computer Systems **80**, 448–466 (2018)

Paper F

PriBank: Confidential Blockchain Scaling Using Short Commit-and-Proof NIZK Argument

K. Gjøsteen, M. Raikwar, S. Wu

Published in Cryptographers' Track at the RSA Conference (CT-RSA), 2022

PriBank: Confidential Blockchain Scaling Using Short Commit-and-Proof NIZK Argument

Kristian Gjøsteen, Mayank Raikwar, and Shuang Wu

Norwegian University of Science and Technology, NTNU, Trondheim, Norway
{kristian.gjosteen,mayank.raikwar,shuang.wu}@ntnu.no

Abstract. Decentralized financial applications demand fast, cheap, and privacy-preserving cryptocurrency systems to facilitate high transaction volumes and provide privacy for users. Off-chain Layer-2 scaling solutions such as Plasma, ZK-Rollup, NOCUST are appealing innovations devised to enable the scalability and extensibility account-based blockchains that support smart contracts. The essential idea is simple yet powerful: move expensive computations off-chain and commit the abbreviated transaction data on-chain. Nevertheless, these solutions do not provide privacy for the users' balances and off-chain transaction data. In this paper, we propose *PriBank*, a novel privacy-preserving cryptocurrency system that enables private balances and transaction values on top of these Layer-2 scaling solutions. To construct PriBank system, we propose a Commit-and-Prove short NIZK argument for quadratic arithmetic programs. The Commit-and-Prove short NIZK argument is built on top of the existing zero-knowledge proof scheme: Bulletproof. It allows a prover to commit to an arbitrary set of witnesses by Pedersen commitments before proving, which may be of independent interest. We construct security models and definitions for Layer-2 privacy-preserving scaling solutions and analyse the security of our scheme under the security model. We also implement and evaluate the system, and present a comparative analysis with the existing solutions.

Keywords: Blockchain · Privacy · Scalability · Commitments · Zero-knowledge proofs · Smart contract

1 Introduction

Blockchain-based cryptocurrencies enable a peer-to-peer digital transfer of values by keeping an immutable, distributed but globally synchronised public ledger, the *blockchain*. However, the transactions in many of these blockchain-based systems such as Bitcoin [26], Ethereum [33] are public. Current blockchains are not suitable for daily financial transactions due to their privacy and scalability issues. For instance, the average throughput of Bitcoin is 4.6 transactions per second while Visa does around 1,700 transactions per second on average. Privacy and scalability, however, are hard to achieve at the same time. Since adding privacy and confidentiality for a blockchain inevitably adds more computation and data to the blockchain that results in reducing the transaction throughput and increasing the transaction fees and hence downgrading the scalability of the system.

There are many anonymous cryptocurrencies ranging from Zcash [30], Monero [32] based on Bitcoin-like blockchains, to works [8,31,20] built on top of the smart contracts. The systems built on these models can have private user balances, private and anonymous transactions. While most of the systems focus on improving privacy, a few (none) of them discuss the scalability of their designs. Monero employs anonymity sets in a ring structure to achieve privacy, however, these privacy sets are fairly small. Zcash and Monero rely on recording every transaction in the history to perform further transactions, this model is called unspent transaction output (UTXO) model. Zcash and its follow-up designs [23,7,20] inherit the same limitation of the UTXO model and also employ a zk-SNARK proof algorithm that has a sophisticated structured common reference string (CRS) requisite more trust for the parties to set up the system. Account-based systems [8,15] where transaction take place between accounts also incorporate zero-knowledge proof for privacy. The transactions in these systems are directly sent to the blockchain resulting in computation overload. All these systems do not achieve scalability due to the expensive blockchain transactions.

Many constructions such as Plasma [29], NOCUST [21], ZK-Rollup [19] aim to improve the scalability of blockchain by moving a large amount of data and computation off-chain through an untrusted operator. The operator puts a short summary of the transactions on-chain. The blockchain scaling designs following this architecture are called Layer-2 scaling solutions. Plasma/NOCUST claim to decrease the transaction cost nearly to zero. However, these systems send less data to the blockchain but they suffer from the problem of mass exit and long waiting time in case of withdraw (Detailed explanation in Section 1.2). Moreover, users need to keep online and monitor the behaviour of the operator and other users in case of a dispute. Dziembowski et al. [16] proved that mass exit is inevitable in these systems. ZK-Rollup is designed to avoid these issues by submitting a zero-knowledge proof with extra data to the blockchain, but it has less throughput and needs a trusted setup. We follow the similar scalability approach as ZK-Rollup but without trusted setup, and above all, provide privacy.

Based on the above observations, one possible approach to balance privacy and scalability is to build privacy on the top of these scaling architectures. However, it is not simply adapting the cryptographic techniques used in anonymous cryptocurrencies to the blockchain scaling solutions. First, the fundamental architecture difference of having an operator or not gives different security and threat models. Second, the goal of introducing an off-chain operator is to outsource the computation, while if the transactions are private, the operator needs to compute on private data. Third, the operator is supposed to send as minimal data as possible to the blockchain as long as the data can represent a unique and correct transaction history. While if the transactions are encrypted, inevitably it is more complex to “compress” the data and still be able to prove its correctness. Meanwhile, the overall cost ‘to build privacy on scaling solutions’ method is unknown, and is still a worthwhile question to be asked and to be investigated.

Motivated by the above, the contributions of this paper are as follows:

1. We construct an efficient Commit-and-Prove NIZK protocol for quadratic arithmetic program by modifying the Bulletproof protocol [9]. Our Commit-and-Prove NIZK protocol can be of independent interest (Section 3).
2. We formally define a privacy-preserving cryptocurrency system built on the top of layer-2 scaling and further, we present a security model of the system (Section 4).
3. We construct a privacy-preserving cryptocurrency system **PriBank** where the large computation of the user transactions is delegated to an off-chain operator while keeping the users' balances and transaction values private. Users trust the operator for confidentiality, but the system is trust-less for its integrity (Section 5).
4. We provide a security analysis of the system (Section 6) and implement the Commit-and-Prove NIZK protocol of PriBank in *Go* and further evaluate the performance of the protocol (Section 7).

Furthermore, in our system, the computation load and data sent to the blockchain is quadratic to the number of users in the worst case. The zero-knowledge argument for inner-product in our system allows a prover to commit to a subset of witness by Pedersen commitment, compared with Bulletproof where all the witness are committed by vector Pedersen commitment.

1.1 Overview of PriBank

The PriBank system aims to enhance the privacy of blockchain-based cryptocurrency scaling approaches, and concurrently manages the computation and data overload at a practical level. There are three types of entities in PriBank: *Users*, *Bank (Operator)*, and a *Smart Contract* running on the blockchain. The users make the transactions with other users of the system through the bank operator. From a privacy perspective, the operator serves as a bank on top of a blockchain where users' transactions and balances are hidden from other entities apart from the bank. The bank operator maintains users' balances and transactions, keeps them private, and periodically submits commitments to the users' data and zero-knowledge proofs to the smart contract. The smart contract validates the operator's commitments and proofs and records them on the blockchain only if they are valid. The difference between the traditional bank and PriBank is that PriBank is not trusted to execute users' requests honestly, yet the bank can do no harm apart from leaking user's information. As an additional benefit in the current regulatory climate, the information that the operator holds enables auditing of transactions, which is important to prevent financial crimes.

PriBank system operates in terms of *epochs*. An epoch is divided into three phases: *Transaction* phase, *Commit* phase and *Exit* phase. The beginning of an epoch is the *Transaction* phase which consists of three processes: *Register*, *Deposit* and *Transfer*. These process can run parallel. The operator collects all the transaction data in the *Transaction* phase, and sends the commitments and proofs to the smart contract in the *Commit* phase. At the end of epoch is the *Exit* phase where users can withdraw or exit the system with the balances that smart contract has confirmed during the *Commit* phase. Moreover, during the *Transaction* phase, a user can make multiple transactions to another user, but only the final balance is uploaded to the blockchain. Figure 1 depicts a general overview of PriBank.

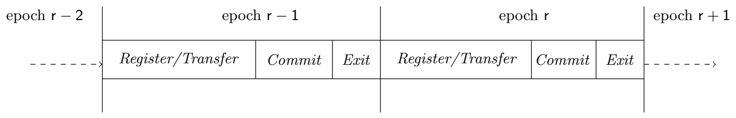

Fig. 1: An Overview of PriBank System

PriBank Workflow: A general working flow of PriBank is as follows:

– Firstly, a smart contract is set up on the blockchain. It includes all the public parameters in the system. Then user register and deposit funds in the system through this smart contract.
– After registration, a user sends a plain text transaction to the operator using a secure channel. The operator then commits to the transaction value and generates its proof. Then, it returns the commitment and proof to the user along with a collection of all the transactions commitments made for this user within a specific period and asks for a signature on the collection. The user checks that the collection is valid, and if so signs the transaction collection.
– Through above operation, the operator collects a list of transaction commitments of a user and also gets the user's signature on them. At the end of a period, the operator updates each user's balance, submits the balance commitments and the lists of transaction commitments from the users along with a zero-knowledge proof to the smart contract on blockchain. The balances of users in the system are represented in the form of their commitments.
– Upon receiving the data from the operator, the smart contract verifies the signatures and the zero-knowledge proof, and updates the new balance commitment for each user if the data passes all the verification checks.
– The withdrawal or exit process are on the blockchain between the smart contract and users. The users have the necessary proof that they received from the operator beforehand for the withdrawal. They submit the requests and the proof to the smart contract. The smart contract checks the proof and processes the withdrawal or exit.

Functionality: PriBank processes the user transactions in an off-chain manner through an operator and smart contract, hence amortizes the cost incurred in the parent blockchain. As PriBank is deployed alongside the parent blockchain, the parent blockchain has a global view of user accounts. The parent blockchain should be account-based, smart-contract enabled (e.g. Ethereum [33]), and operated by an honest majority. Furthermore, PriBank operations should take place under the partially synchronous network where messages between two entities should reach within an upper bound delay. Users of PriBank should also be online at least once in each epoch to send or receive transactions in the system.

Verifiable Operation through Commit-and-Prove Zero-knowledge Proof For the correct execution of the PriBank workflow, we build a commit-and-prove zero-knowledge proof system. The commit-and-prove approach in zero-knowledge proofs dates back to the works of Kilian [22] and Canetti et al. [11], following by the works [3,10,5]. The algorithms are zero-knowledge proof in which the prover proves statements about values that are committed. In *PriBank*, this proof system is built over an arithmetic circuit that represents the computation of the bank operator. We commit to the circuit wires and then prove that they satisfy the circuit. We use Pedersen commitment scheme to represent users' balances and transactions privately, these values are part of the wires of the circuit. For the other secret wires, we use the vector Pedersen commitment to shrink the size of the overall commitment to being logarithmic in the circuit size. We also use the signature and zero-knowledge proof verification in the smart contract that guarantees the correctness and verifiability of the updates from the bank operator. Details about the arithmetic circuit and mentioned commitment schemes are given in Section 2.

1.2 Related Work

The problem of scalability in blockchain has become more urgent recently. A large amount of work has been done to address this issue and many solutions have been proposed [25,14,21,4,29]. The majority of these solutions support off-chain interactions and computations. In these off-chain solutions, a large number of transactions take place off-chain through an operator who puts a short summary of these transactions on-chain. However, some of these systems are vulnerable to mass exit. In a mass exit scenario, the operator goes rogue and users of the systems need to send ownership proofs of their assets to the main chain, in order to exit the system with their respective assets. This event causes congestion on the main chain and the users might not be able to exit the system in time.

Apart from the problem of mass-exit, some of these solutions do not provide integrity of the transactions. Moreover, some solutions do incorporate zero-knowledge proofs to achieve the integrity of blockchain and off-chain systems. Nevertheless, transaction data in all these systems are public and therefore fail to address the privacy implications of blockchain.

In the blockchain context there are different privacy notions, ranging from privacy of transaction amounts [30,23,8] and transacting parties [15,17] to the privacy of embedded functional calls in a smart contract [7]. Different solutions have been proposed to achieve meaningful privacy notions. Several of these solutions employ advanced cryptographic techniques such as zero-knowledge proofs [30], ring signature [32], homomorphic encryption [34] and mixing techniques [6,24] to achieve various forms of privacy. Financial systems zkLedger [27], Solidus [12], RSCoin [13] also achieve privacy of their transactions, but banks regulate the supply of funds and a blockchain is used to make transactions.

Hawk [23] is the first framework to construct privacy-preserving smart contracts. Hawk combines the idea of Zcash and multi-party computation that achieves on-chain privacy through the use of zero-knowledge proofs. The on-chain privacy is guaranteed by a party that performs off-chain computation, generates a cryptographic zk-SNARKs proof, and puts these results on-chain.

A zk-SNARK provides a succinct cryptographic proof attesting to the correctness of a computation. However, the use of zk-SNARK puts a bound on the number of participants (in Hawk) due to its trusted setup and circuit-dependent CRS. Ledger-based construction, Zexe [7], follows the similar idea of performing off-chain computation using zk-SNARKs and subsequently produces privacy-preserving transactions. In addition to Hawk, Zexe also hides which computations were performed off-chain, but both suffer from limitation of zk-SNARKs.

To build a privacy-preserving ledger system along with stateful computations in the smart contract, several new constructions have been proposed such as zKay [31], Zether [8], and Kachina [20]. All these constructions extend Ethereum with privacy-preserving currency or data. zKay achieves privacy of data using encryption and correctness using NIZK proof of encryption. zKay presents a new language that extends Ethereum smart contracts with private data types. Zether, on the other hand, proposes a privacy-preserving payment mechanism using ElGamal encryption and zero-knowledge proofs (Bulletproof-based range proofs). Kachina realizes a large class of privacy-preserving smart contracts. Kachina uses NIZK proofs and state oracles to establish the desired privacy-preserving smart contract. The security model of Kachina is based on Universal Composition (UC) model, encompassing the other mentioned models. However, many find UC hard to work with.

Quisquis [17] is a hybrid construction of UTXO and account-based model that provides a provably secure notion of anonymity. It achieves this notion by combining a DDH-based updatable public key system with NIZK proofs.

Although all the above-described privacy-preserving constructions achieve various privacy notions, none of the constructions explicitly provide scalability analysis. Furthermore, many of these constructions provide implementations without integrity and do not have a proper security model. Moreover, even some of these constructions involve off-chain transactions that might result in better scalability, but fail to mention and analyse it. To the best of our knowledge, PriBank is the only construction that enables a useful form of privacy along with scalability in account-based blockchain with a proper security model.

Tables 1 and 2 compare PriBank with the existing popular schemes for scalability and privacy, respectively. In these tables, '✓' denotes that the respective system has the corresponding feature, and '✗' denotes that the scheme lacks that feature. Table 1 compares PriBank with existing scalable off-chain systems. Table 2 compares PriBank with the existing privacy-preserving systems. However, schemes such as Monero and Zcash do not have a concept of off-chain third party, therefore they achieve stronger privacy notions. Irrespective of that, we

Table 1: Comparison Matrix for different off-chain systems.

Scheme		Plasma [29]	ZK-Rollup [19]	State Channel [25]	Our Scheme
Security	No Mass Exit Assumption	✗	✓	✓	✓
	No Watch Tower Assumption	✗	✓	✗	✓
	Cryptographic Primitives	Standard	New	Standard	Standard
Usability	Withdraw Time	1 week	10 minutes	1 confirmation	1 epoch
	Transaction Finalization Time	1 confirmation	1 confirmation	Instant	1 confirmation
	User Verification	✗	✗	✓	✓
Performance	Cost of Transaction	Very low	Very low	Low	Low
	No Collateral Required	✓	✓	✗	✓

Table 2: Comparison Matrix for different privacy-preserving systems.

Scheme		Monero [32]	Zcash [30]	Zexe [7]	Zether [8]	Quisquis [17]	Our Scheme
Privacy	Confidential Transaction	✓	✓	✓	✓	✓	✓
	Confidential User Address	✓	✓	✓	✗*	✓	✓
	Anonymity Set	Small	Large	Large	Small	Small	Large
Security	Security Model	✗	✗	✓*	✓*	✗	✓
	Cryptographic Primitives	Standard	Standard	New	New	Standard	Standard
Performance	Transaction Size	2–∞KB	2KB	0.945–∞KB	1.472KB	13.4KB	0.033–1.9KB
	No Trusted Setup Required	✗	✗	✗	✓	✓	✓

compare our PriBank scheme to all these systems. '✗*' denotes that Zether itself does not provide anonymity, however, another construction Anonymous Zether does provide anonymity, though with extra cost. '✓*' denotes that the scheme does not provide extensive security analysis.

2 Preliminaries

Notation : Throughout the paper, we use bold font to denote vectors, i.e. $\boldsymbol{a} \in \mathbb{Z}_p^n$ is a vector with elements $a_1, ..., a_n \in \mathbb{Z}_p$. The inner product between two vectors $\boldsymbol{a}, \boldsymbol{b}$ is denoted by $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$. The Hadamard product or entry wise multiplication of two vectors $\boldsymbol{a}, \boldsymbol{b}$ is denoted by $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n) \in \mathbb{Z}_p^n$. We denote slices of a vector as: $\boldsymbol{a}_{[:l]} = (a_1, ..., a_l) \in \mathbb{Z}_p^l, \boldsymbol{a}_{[l:]} = (a_{l+1}, ..., a_n) \in \mathbb{Z}_p^{n-l}$.

2.1 Commitment schemes

A commitment scheme allows one to commit to a chosen value (or chosen statement) while keeping it secret, with the ability to only reveal the commitment to the committed value later. A commitment schemes has Hiding and blinding properties. In PriBank, commitment schemes are used to commit to user's balance or transaction data. For the construction, we use Pedersen commitment [28] and Vector Pedersen commitment scheme.

- Given a group $\mathbb{G}$ of order q and two generators g, h of group $\mathbb{G}$, a *Pedersen commitment* for a value $a \in \mathbb{Z}_q$ is defined as $C_a = g^a h^r \in \mathbb{G}$, where $r \xleftarrow{\$} \mathbb{Z}_q$.
- Given a group $\mathbb{G}$ of order q, $\boldsymbol{g} := (g_1, \ldots, g_n), h \leftarrow \mathbb{G}$ and a vector $\boldsymbol{a} := (a_1, ..., a_n)$, a *vector Pedersen commitment* is $C_{\boldsymbol{a}} = \prod_{i=1}^n g_i^{a_i} h^r \in \mathbb{G}$, $r \xleftarrow{\$} \mathbb{Z}_q$.

In Section 3.2, we are using the commitment scheme in slightly different way. We are constructing a collection of Pedersen commitments that are using the same randomness over different generators of the group.

2.2 Quadratic Arithmetic Program

We represent the operations that the bank operator do to compute the new balances from the old balances and transaction history into an arithmetic circuit satisfaction problem. The circuit gives the necessary range checks as well. The work of Gennaro et al. [18] shows how to further translate an arithmetic circuit satisfaction problem to a *Quadratic Arithmetic Program (QAP)*, where the circuit is reduced to a polynomial equation.

Definition 1 (Quadratic Arithmetic Program [18]). *A quadratic arithmetic program (QAP) Q over a field $\mathbb{Z}_p$ consists of three sets of polynomials $V = \{v_k(x) : k \in \{0,...,n\}\}, U = \{u_k(x) : k \in \{0,...,n\}\}, W = \{w_k(x) : k \in \{0,...,n\}\}$ and a target polynomial $z(X)$, all are defined over $\mathbb{Z}_p$.*

Let a circuit C, where all the wires including inputs, outputs and inner circuit wires variables are labelled $a_0, a_1, ..., a_n$ (where $a_0 = 1$). A QAP Q is said to compute C if the following holds:
$a_1, ..., a_n \in \mathbb{Z}_p^n$ is a valid assignment to the wires variables of C iff there exist $h(X)$ such that

$$\sum_{i=1}^{n} a_i u_i(X) \cdot \sum_{i=1}^{n} a_i v_i(X) = \sum_{i=1}^{n} a_i w_i(X) + h(X)z(X)$$

The size of QAP is n, and degree is deg$(z(X))$, which is also the number of gates in the circuit C. The polynomials $u_k(X), v_k(X), w_k(X)$ have degree at most deg$(z(X)) - 1$.

2.3 Commit-and-Prove Zero-knowledge Argument

A zero-knowledge argument is a protocol in which a prover wants to convince a verifier that a statement is true without revealing any private information. A commit-and-prove zero-knowledge argument is a zero-knowledge argument in which a prover proves statements about values that are committed. It allows a prover to commit to the secrets it holds before the prover knows what it is going to prove. For instance, a prover makes a commitment to a user's balance, later it can convince the verifier that this balance is in or out of a certain range.

We follow the notation of [9]. A commit-and-prove argument consists of three PPT algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. These are the common reference generator $\mathcal{G}$, the interactive prover $\mathcal{P}$ and verifier $\mathcal{V}$. Take input as 1^λ, $\mathcal{G}$ outputs the common reference σ. The communication transcript between $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. We write the output of the protocol as $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$. If verifier accepts, $b = 0$, otherwise $b = 1$.

The language of commit-and-prove zero-knowledge argument proving is defined over a polynomial time decidable relation R and a commitment scheme $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$. R is defined over $\mathcal{D}_\sigma \times \mathcal{D}_x \times \mathcal{D}_u \times \mathcal{D}_w$: given a common reference σ, for a triple (x, u, w), we call x the statement, u the committed witness and w the non-committed witness. Define $\mathcal{R}^{\mathsf{Com}}$ as a family of relations that every relation $\mathbf{R} \in \mathcal{R}_\lambda^{\mathsf{Com}}$ can be represented by a tuple

(σ, c, r, x, u, w). Let $\mathcal{L}$ be the language associated with $\mathbf{R}$, i.e.,

$$\mathcal{L}_\sigma = \{\sigma, c, x \mid \exists w, u, r \text{ s.t. } \mathsf{VerCommit}(c, u, r) = 1 \wedge R(\sigma, x, u, w) = 1\}$$

The commit-and-prove argument algorithm that we define has completeness, special soundness and perfect zero-knowledge. The formal definitions are given in Appendix F.

3 Commit-and-Prove Short NIZK Argument for Quadratic Arithmetic Program

In this section, we introduce the construction of commit-prove short interactive zero-knowledge proof for the quadratic arithmetic program. The protocol is lying on three sub-protocols: a zero-knowledge argument for a product of Pedersen commitments; a zero-knowledge argument for the inner product of a collection of Pedersen commitments and a public vector; a zero-knowledge argument for the inner product of a vector Pedersen Commitment and a public vector. We describe the sub-protocols at the beginning and then describe how we combine them and build the final protocol.

3.1 Zero-Knowledge Argument of Knowledge for Product of Pedersen Commitments

Consider two Pedersen commitments $c_a = g^a h^{r_a}$ and $c_b = g^b h^{r_b}$, the following Protocol 3.1 is to prove a Pedersen commitment c is committed to the product of a and b, i.e. $c = g^{ab} h^t$.

Protocol Input: $(g, h, c, c_a, c_b \in \mathbb{G}; a, b, r_a, r_b, t \in \mathbb{Z}_p)$
Protocol Output: ($\mathcal{V}$ accepts or $\mathcal{V}$ rejects)

$\mathcal{P}$'s input:$(g, h, a, b, r_a, r_b, t)$
$\mathcal{V}$'s input:(g, h, c_a, c_b, c)

1. $\mathcal{P}$ chooses randoms $\alpha, \beta, r_1, r_2, s_0, s_1$ and computes
   $d_1 = g^\alpha h^{r_1}$, $d_2 = g^\beta h^{r_2}$, $c_0 = g^{\alpha b + \beta a} h^{s_0}$, $c_1 = g^{\alpha\beta} h^{s_1}$.
   $\mathcal{P}$ sends (d_1, d_2, c_0, c_1) to $\mathcal{V}$.
2. $\mathcal{V}$: $x \xleftarrow{\$} \mathbb{Z}_p$, sends x to $\mathcal{P}$.
3. $\mathcal{P}$ computes $\theta_a = \alpha - ax$, $\theta_b = \beta - bx$, $\theta_1 = r_1 - r_a x$, $\theta_2 = r_2 - r_b x$ $\theta_{ab} = x^2 t - x s_0 + s_1$ then sends $\theta_a, \theta_b, \theta_1, \theta_2$ to $\mathcal{V}$.
4. $\mathcal{V}$ checks $c_a^x g^{\theta_a} h^{\theta_1} = d_1$, $c_b^x g^{\theta_b} h^{\theta_2} = d_2$, $g^{\theta_a \theta_b} h^{\theta_{ab}} c_0^x = c^{x^2} c_1$, output 1 if all the equations hold otherwise output 0.

Fig. 2: Protocol 3.1

We prove the protocol has perfect completeness,computational soundness and perfect zero-knowledge in Appendix B

3.2 Zero-Knowledge Argument for Inner Product of Pedersen Commitments and a Public Vector

Consider a vector $\boldsymbol{a} = (a_1, ..., a_n)$ and a collection of Pedersen commitments $\{c_i\}_{i=1}^n$ where $c_i = g^{a_i} h_i^{\gamma}$. These Pedersen commitments are commitments to the elements of $\boldsymbol{a}$ using the same randomness γ but over different generators h_i. We give a zero-knowledge argument Protocol 3.2 that claims c is a Pedersen commitment that commits to the inner product between $\boldsymbol{a}$ and a public vector $\boldsymbol{b}$, i.e. $c = g^{\sum_{i=1}^{n} a_i b_i} h^t$.

Statement: Vector $\boldsymbol{b}$, Pedersen commitments $\{c_i\}_{i=1}^n$ and generators $g, \{h_i\}_{i=1}^n$
Prover's Witness: Openings $\boldsymbol{a}, \gamma$ and t
Protocol Input: $(\{h_i\}_{i=1}^n, \{c_i\}_{i=1}^n, g, c \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n; \gamma, t \in \mathbb{Z}_p)$
Protocol Output: ($\mathcal{V}$ accepts or $\mathcal{V}$ rejects)

$\mathcal{P}$'s input:$(\{h_i\}_{i=1}^n, g, \boldsymbol{a}, \boldsymbol{b}, \gamma, t)$
$\mathcal{V}$'s input:$(\{h_i\}_{i=1}^n, g, c, \{c_i\}_{i=1}^n)$

1. $\mathcal{P}$ chooses randoms $\alpha, \beta, t \xleftarrow{\$} \mathbb{Z}_p$, computes $c_0 = h^{\gamma}, \quad \tau = \prod_{i=1}^{n} h_i^{b_i}, \quad \Omega = \tau^{\gamma} h^{-t}$, $d_1 = h^{\alpha}$, $d_2 = \tau^{\alpha} h^{\beta}$ and sends (c_0, Ω, d_1, d_2) to $\mathcal{V}$.
2. $\mathcal{V}$ chooses a challenge $x \xleftarrow{\$} \mathbb{Z}_p$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes $\theta_1 = \alpha - x\gamma, \theta_2 = \beta + xt$.
4. $\mathcal{V}$ computes $\tau = \prod_{i=1}^{n} h_i^{b_i}$, output accept if and only if $c = \frac{\prod_{i=1}^{n} c_i^{b_i}}{\Omega} \wedge d_1 = c_0^x h^{\theta_1} \wedge d_2 = \Omega^x \tau^{\theta_1} h^{\theta_2}$

Fig. 3: Protocol 3.2

We prove the argument of knowledge presented in Protocol 3.2 has perfect completeness, computational soundness and perfect special honest-verifier zero-knowledge in Appendix C.

3.3 Zero-knowledge Argument of Knowledge for Inner Product of Vector Pedersen Commitment and Public Vector

Consider a vector Pedersen commitment $c_{\boldsymbol{a}} = \prod_{i=1}^{n} g_i^{a_i} h^{r_a}$ that commits to $\boldsymbol{a} := (a_1, ..., a_n)$. We give a zero-knowledge argument protocol that claims $c_{\boldsymbol{ab}}$ is a Pedersen commitment that commits to the inner product between $\boldsymbol{a}$ and a public vector $\boldsymbol{b}$, i.e. $c_{\boldsymbol{ab}} = g^{\sum_{i=1}^{n} a_i b_i} h^{r_{ab}}$.

This algorithm is a variant of the inner product argument construction in Bulletproof [9]. We modify it to have the zero-knowledge property which we will use to build PriBank system. We prove the argument of knowledge presented in Protocol 3.3 has perfect completeness, computational knowledge soundness and perfect special honest-verifier zero-knowledge. The proofs for completeness and honest-verifier zero-knowledge are in Appendix D. The soundness proof follows the proof of Bulletproof and can be found in the full version of the paper.

Statement: Generators vector $\boldsymbol{g}{:} = \{g_1, ..., g_n\}$, generator h, vector $\boldsymbol{b}{:} = \{b_1, ..., b_n\}$, vector Pedersen commitment $c_{\boldsymbol{a}} = \prod_{i=1}^{n} g_i^{a_i} h^{r_a}$ and $c_{\boldsymbol{ab}} = g^{\sum_{i=1}^{n} a_i b_i} h^{r_{ab}}$

Prover's Witness: Openings for the commitments $\boldsymbol{a}, r_a, r_{ab}$

$\mathcal{V}$ randomly chooses a challenge x' and sends it to $\mathcal{P}$. Let $c = c_{\boldsymbol{a}} c_{\boldsymbol{ab}}^{x'}$, $r = r_a + r_{ab}x'$, $u = g^{x'}$ and runs the following protocol Prove on input $(\boldsymbol{g}, u, h, \boldsymbol{a}, \boldsymbol{b}, c, r)$.

Protocol Prove:

Input: $(\boldsymbol{g} \in \mathbb{G}^n, u, h, c \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n, r \in \mathbb{Z}_p)$ Output: ($\mathcal{V}$ accepts or $\mathcal{V}$ rejects)

$\mathcal{P}$'s input:$(\boldsymbol{g}, u, h, c, \boldsymbol{a}, \boldsymbol{b})$
$\mathcal{V}$'s input:$(\boldsymbol{g}, u, h, c, \boldsymbol{b})$
if $n = 1$ $(\boldsymbol{a} := \{a_1\}, \boldsymbol{g} := \{g_1\})$:

1. $\mathcal{P}$ chooses randomness $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p$, computes and sends $d = g_1^{\alpha_1} u^{\alpha_1 b_1} h^{\alpha_2}$ to $\mathcal{V}$.
2. $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p$ challenge x, sends it to the $\mathcal{P}$.
3. $\mathcal{P}$ computes $\theta_1 = \alpha_1 - xa_1, \theta_2 = \alpha_2 - xr$, sends θ_1 and θ_2 to $\mathcal{V}$
4. $\mathcal{V}$ accepts if $c^x g_1^{\theta_1} u^{b_1 \theta_1} h^{\theta_2} = d$, otherwise it rejects.

if $n > 1$:

1. Let $n' = \frac{n}{2}$, $\mathcal{P}$ chooses random $r_1 \xleftarrow{\$} \mathbb{Z}_p$ and $r_2 \xleftarrow{\$} \mathbb{Z}_p$, computes L, R as follows and sends L, R to $\mathcal{V}$.

$$L = \boldsymbol{g}_{[n':]}^{\boldsymbol{a}_{[:n']}} \cdot u^{\langle \boldsymbol{a}_{[:n']}, \boldsymbol{b}_{[n':]} \rangle} \cdot h^{r_1} \in \mathbb{G}$$

$$R = \boldsymbol{g}_{[:n']}^{\boldsymbol{a}_{[n':]}} \cdot u^{\langle \boldsymbol{a}_{[n':]}, \boldsymbol{b}_{[:n']} \rangle} \cdot h^{r_2} \in \mathbb{G}$$

2. $\mathcal{V}$ chooses challenge x and sends it to the prover, i.e. $\mathcal{V} \rightarrow \mathcal{P} : x \xleftarrow{\$} \mathbb{Z}_p$
3. Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\boldsymbol{g}' = \boldsymbol{g}_{[:n']}^{x^{-1}} \circ \boldsymbol{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$$
$$\boldsymbol{b}' = x^{-1}\boldsymbol{b}_{[:n']} + x\boldsymbol{b}_{[n':]} \in \mathbb{Z}_p^{n'}$$
$$c' = c \cdot L^{x^2} \cdot R^{x^{-2}} \in \mathbb{G}$$

4. $\mathcal{P}$ computes:

$$\boldsymbol{a}' = x\boldsymbol{a}_{[:n']} + x^{-1}\boldsymbol{a}_{[n':]}$$
$$r = r + x^2 r_1 + x^{-2} r_2$$

5. Recursively run Prove on input $(\boldsymbol{g}', u, h, c', \boldsymbol{a}', \boldsymbol{b}', r)$

Fig. 4: Protocol 3.3

3.4 Commit-and-Prove Zero-Knowledge Argument For QAP

We give a commit-and-prove zero-knowledge argument Protocol 3.4 for the satisfiability of a QAP for an arithmetic circuit C. For wires in the circuit $\{a_i\}_{i=0}^{n}$, we denote the input witnesses are $\{a_i\}_{i=0}^{k}$, the inner circuit witnesses are $\{a_i\}_{i=k+1}^{l}$ and the statements wires are $\{a_i\}_{i=l+1}^{n}$. The quadratic arithmetic program, Pedersen commitment and vector Pedersen commitment give a relation of the form $\boldsymbol{R} = (\mathbb{G}, \mathbb{Z}_p, k, l, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^{n}, z(X), \{a_i\}_{i=0}^{n}, c_l, \{c_i\}_{i=1}^{k}, \{a_i\}_{i=1}^{l}, \gamma, r)$ *such that with* $a_0 = 1$

$$\sum_{i=1}^{n} a_i u_i(X) \cdot \sum_{i=1}^{n} a_i v_i(X) = \sum_{i=1}^{n} a_i w_i(X) + h(X)z(X) \ \wedge \ \{c_i = g^{a_i} h_i^{\gamma}\}_{i=1}^{k}$$
$$\wedge \ c_l = h^r \prod_{i=k+1}^{l} g_i^{a_i} \wedge c_h = h^t \prod_{i=0}^{n-2} g_i^{e_i}$$

where $e_0, ..., e_{n-2}$ *are the coefficients of* $h(X)$.

Statements: A collection of Pedersen commitments $c_1, ..., c_k$ that commit to $a_1, ..., a_k$, two vector Pedersen commitments c_l and c_h that commit to $a_{k+1}, ..., a_l$ and the coefficients of polynomial $h(x) = e_0 + e_1x + ... + e_{n-2}x^{n-2}$, the public values $a_{l+1}, ..., a_n$
Witnesses: $a_1, ..., a_l, \gamma, r, t, e_0, ..., e_{n-2}$
Input: $(g, h, \{h_i\}_{i=1}^k, \{c_i\}_{i=1}^k, c_l, c_h \in \mathbb{G}, \boldsymbol{g} \in \mathbb{G}^{n-2}; \{a_i\}_{i=1}^n, \gamma, r, t, \{e_i\}_{i=1}^{n-2} \in \mathbb{Z}_p)$
Output: ($\mathcal{V}$ accepts or $\mathcal{V}$ rejects)

$\mathcal{P}$'s input:$(g, h, \{h_i\}_{i=1}^k, \{c_i\}_{i=1}^k, c_l, c_h, \boldsymbol{g}, \{a_i\}_{i=1}^l, \gamma, r, t, \{e_i\}_{i=1}^{n-2})$
$\mathcal{V}$'s input:$(g, h, \{h_i\}_{i=1}^k, \{c_i\}_{i=1}^k, c_l, c_h, \boldsymbol{g})$

1. $\mathcal{V}$ sends challenge $x_1 \xleftarrow{\$} \mathbb{Z}_p$ to the $\mathcal{P}$, computes $\{u_i(x_1), v_i(x_1), w_i(x_1)\}_{i=0}^m$.
2. $\mathcal{P}$ chooses $t_u, t_v, t_w \xleftarrow{\$} \mathbb{Z}_p$ and computes $c_u = g^{\sum_{i=1}^{k} a_i u_i(x_1)} h^{t_u}$, $c_v = g^{\sum_{i=1}^{k} a_i v_i(x_1)} h^{t_v}$, $c_w = g^{\sum_{i=1}^{k} a_i w_i(x_1)} h^{t_w}$ as the inner product between Pedersen commitments $\{c_i\}_{i=1}^k$ and $\{u_i(x_1)\}_{i=1}^k$,$\{v_i(x_1)\}_{i=1}^k$,$\{w_i(x_1)\}_{i=1}^k$ respectively. Run protocol 3.2 to give the proof of the correct constructions.
3. $\mathcal{P}$ chooses $s_u, s_v, s_w \xleftarrow{\$} \mathbb{Z}_p$ and computes
$$c_{\boldsymbol{u}} = g^{\sum_{i=k+1}^{l} a_i u_i(x_1)} h^{s_u},$$
$$c_{\boldsymbol{v}} = g^{\sum_{i=k+1}^{l} a_i v_i(x_1)} h^{s_v}$$
$$c_{\boldsymbol{w}} = g^{\sum_{i=k+1}^{l} a_i w_i(x_1)} h^{s_w}$$
as the inner product of vector Pedersen commitment c_l between $\{u_i(x_1)\}_{i=k+1}^l$, $\{v_i(x_1)\}_{i=k+1}^l$ and $\{w_i(x_1)\}_{i=k+1}^l$ respectively. $\mathcal{P}$ also chooses $s_h \xleftarrow{\$} \mathbb{Z}_p$ and computes $c_{hz} = g^{h(x_1)z(x_1)} h^{s_h}$, which is an inner product of vector Pedersen commitment and public vector $z(x_1), x_1z(x_1), ..., x_1^{n-2}z(x_1)$.
Run Protocol 3.3 to give a proof for the above constructions.
4. $\mathcal{P}$ computes
$$c_a = c_u \cdot c_{\boldsymbol{u}} \cdot g^{\sum_{i=l+1}^{n} a_i u_i(x_1)},$$
$$c_b = c_v \cdot c_{\boldsymbol{v}} \cdot g^{\sum_{i=l+1}^{n} a_i v_i(x_1)},$$
$$c_c = c_w \cdot c_{\boldsymbol{w}} \cdot g^{\sum_{i=l+1}^{n} a_i w_i(x_1)} \cdot c_{hz}$$
Run Protocol 3.1 to prove c_c commits to the product of the committed values of c_a and c_b.

Fig. 5: Protocol 3.4

We prove the protocol has perfect completeness, computational soundness and perfect special honest verifier zero-knowledge in Appendix E.

4 Blockchain-based Bank Protocol

In this algorithm we isolate the blockchain functionality. The transactions/data that are sent to blockchains are denoted by trans, when a trans is accepted by the blockchain, the public state, bank state and users' states are all updated.

Definition 2 (BBank). *A blockchain-based bank protocol* ***BBank*** *is a tuple of algorithms* ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EstablishBank}, \mathsf{NewUser}, \mathsf{Deposit}, \mathsf{Withdraw}, \mathsf{Pay}, \mathsf{Commit}$) *with the following syntax and semantics*

- ***Setup*** *The algorithm* **Setup** *generates the public parameters* pp*, to be distributed off-chain.*
- ***KeyGen*** *The algorithm* **KeyGen** *takes the public parameters* pp *and generates the key pair for users or for a bank.*
- ***EstablishBank*** *The algorithm establishes a bank, it takes the public parameters and a key pair as inputs, generates the initial state of the bank* $\mathsf{TempSt_b}$ *and a transaction* trans*. Once the transaction* trans *is accepted by the blokchain, it launches the smart contract.*
  $(\mathsf{trans}, \mathsf{TempSt_b}) \leftarrow \mathsf{EstablishBank}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pp})$
- ***NewUser*** *This algorithm is performed by a user to register on the smart contract, but without any deposit for her account yet. It takes the public parameters and the key pair of a user as inputs, outputs* trans *and the initial state of this user.*
  $(\mathsf{trans}, \mathsf{Tempst_u}) \leftarrow \mathsf{NewUser}(\mathsf{pk_b}, \mathsf{sk_u}, \mathsf{pk_u}, \mathsf{pp})$
- ***Deposit*** *The protocol is run by the operator and a user to deposit money on smart contract. It takes* pp*, the key pairs and states of a user and a bank, epoch counter, deposit value as inputs, outputs a* trans *and temporary states of user and bank. Once the transaction* trans *get accepted by the smart contract, the user gets a commitment for her initial balance.*
  $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Deposit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
- ***Withdraw*** *The algorithm is performed by a registered user who wants to exit the PriBank. It takes* pp*, the key pairs of a user, generates a* trans *and updates the temporary states of this user and the bank.*
  $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Withdraw}(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
- ***Pay*** *The protocol is run by the bank and a user (payer) to send transactions to other users. It takes the public key of the receiver, the key pair of the payer and the bank, the temporary states of the payer and the bank, the epoch counter and the transferred value as inputs, and then it updates the temporary states of both user and bank.*
  $(\mathsf{TempSt'_p}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Pay}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_p}, \mathsf{sk_p}, \mathsf{pk_r}, \mathsf{TempSt_p}, \mathsf{TempSt_{bank}}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
- ***Commit*** *The algorithm is performed by the bank. It takes the public state, the key pair of the bank, the state and temporary state of the bank as inputs and generates a* trans *and updates the temporary state of the bank.*
  $(\mathsf{trans}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Commit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{epoch})$
- ***Contract*** *The algorithm takes the public parameters, a* trans*, the public state, all users' states and bank public states as inputs and then updates all of them.*
  $(\mathsf{St'_{pb}}, \mathsf{St'_b}, \{\mathsf{St'_u}\}, \mathsf{TempSt'_b}, \{\mathsf{TempSt'_u}\}) \leftarrow \mathsf{Contract}(\mathsf{St_{pb}}, \{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{trans}, \mathsf{pp})$

4.1 Security Definition

We define two security definitions for the blockchain-based bank protocol; transaction indistinguishability and overdraft safety. Informally speaking, transaction indistinguishability is from typical left-or-right setting used for indistinguishability-based definitions, it specifies an adversary cannot distinguish two confidential transactions. Overdraft safety says the honest users are guaranteed to be able to withdraw all their funds from the system.

We firstly describe the experiment that defines the security of the above two security definitions, and the formal definitions for security follow behind. Given a (candidate) blockchain-based bank scheme Π, an adversary $\mathcal{A}$, and the security parameter λ, the (probabilistic) experiment consists of interactions between $\mathcal{A}$ and the experiment. We assume the adversary $\mathcal{A}$ has full control of the network, we also assume that the adversary forwards the transactions to the blockchain on time (i.e. send query $\mathbf{Q} = \mathbf{Contract}$ on time, we explain the query below). The experiment accepts different types of queries from the adversary. Figure 6 describes each type of query $\mathbf{Q}$.

4.2 Transaction Indistinguishability

Informally, transaction indistinguishability specifies an experiment where an adversary sends two different transactions to the ledger. Only one will be recorded, while the adversary is not able to distinguish which one of these two is recorded. This security property could indicate the anonymity of the users as well as the privacy of the transaction values, depending on leakage.

Transaction indistinguishability is defined by an experiment $\mathsf{Tx\text{-}IND}$, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given (candidate) BBANK scheme. We now describe the $\mathsf{Tx\text{-}IND}$ experiment mentioned above. Given a (candidate) BBANK scheme Π, an adversary $\mathcal{A}$, and security parameter λ, the (probabilistic) experiment $\mathsf{Tx\text{-}IND}(\Pi, \mathcal{A}, \lambda)$ proceeds as the adversary is capable of sending the listed queries in the experiment described in the previous section, while the adversary is not allowed to send reveal query for the secret key of the bank. In addition, the adversary sends the challenge queries we describe next. In the challenge epoch, the experiment randomly chooses $b \leftarrow \{0, 1\}$, the adversary sends many challenge queries as $\mathbf{Q} = \mathbf{Challenge}(\mathbf{Q_0}, \mathbf{Q_1})$; for each challenge query,these two queries leaks some information and the experiment only performs Q_b. After finishing the queries, the adversary sends query $\mathbf{Q} = \mathbf{Commit}$ and gets the output trans_b. At the end, the adversary outputs $b' \in \{0, 1\}$. The adversary wins the game if $b' = b$. During the challenge epoch, we require the queries sent by the adversary to be *Public Consistent* as defined.

Definition 3 (Leakage Function). *A leakage function* $\mathsf{Leakage}$ *takes the output from the experiment as input, and the function outputs the leaked information about the related queries.*

$\eta \leftarrow \mathsf{Leakage}(\mathbf{Q})$

Definition 4 (Public Consistent). *To avoid an adversary winning the experiment trivially, we require the query pairs for* **Commit** *and* **Pay.User** *must be jointly consistent with respect to public information and* $\mathcal{A}$*'s view, namely,*

– *For all the users that the adversary controls (adversary has asked* **Reveal** *query for them), their states in the two banks should be consistent.*

$\mathbf{Q} = (\mathbf{KeyGen})$

1. Compute $(pk, sk) := \mathsf{KeyGen}(\mathsf{pp})$
2. Add (sk, pk) to the key list $\mathsf{KeyList}$
3. Output the public key pk

$\mathbf{Q} = (\mathbf{EstablishBank}, pk)$

1. If (pk, sk) is not in $\mathsf{KeyList}$, output $\perp$.
2. Start a bank instance $(\mathsf{trans}, \mathsf{TempSt_b}) \leftarrow \mathsf{EstablishBank}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pp})$
3. Store key pair and the temporary state of the bank, initiate the bank epoch counter as $n = 1$
4. Output trans

$\mathbf{Q} = (\mathbf{NewUser}, \mathsf{pk}, \mathsf{sk})$

1. If (pk, sk) is not in $\mathsf{KeyList}$, outpt $\perp$.
2. Compute $(\mathsf{trans}, \mathsf{Tempst_u}) \leftarrow \mathsf{NewUser}$ $(\mathsf{pk_b}, \mathsf{sk_u}, \mathsf{pk_u}, \mathsf{pp})$
3. Store the temporary state of the user $(\mathsf{pk}, \mathsf{sk}, \mathsf{TempSt_u})$
4. Output trans

$\mathbf{Q} = (\mathbf{Deposit}, \mathsf{pk_{user}}, \mathsf{v}, \mathsf{epoch})$

1. If $(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{TempSt_u})$ is not recorded, output $\perp$
2. Execute i-th instance of deposit protocol $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Deposit}$ $(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$, when the bank/user sends m, sends (i, m) to $\mathcal{A}$.
3. Output trans

$\mathbf{Q} = (\mathbf{Pay}, \mathsf{pk_p}, \mathsf{pk_r}, \mathsf{v})$

1. If $\mathsf{pk_p}$ or $\mathsf{pk_r}$ is not in the $\mathsf{KeyList}$, output $\perp$
2.Execute the ith pay protocol instance $(\mathsf{TempSt'_p}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Pay}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_p}, \mathsf{sk_p}, \mathsf{pk_r}, \mathsf{TempSt_p}, \mathsf{TempSt_{bank}}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$, when the bank/ user send m, send (i, m) to $\mathcal{A}$.

$\mathbf{Q} = (\mathbf{Send}, i, m)$

1. If $(i, \mathsf{TempSt_b}, \mathsf{TempSt_u}, \mathsf{trans})$ is recorded, send $\perp$ to adversary.
2. Send m to the ith instance. If the ith instance outputs $\perp$, record $(i, \perp)$ and sends $(i, \perp)$ to the adversary. If the ith instance outputs $(\mathsf{TempSt_b}, \mathsf{TempSt_u}, \mathsf{trans})$, then record $(i, \mathsf{TempSt_b}, \mathsf{TempSt_u}, \mathsf{trans})$ and output trans. If the instance sends a message m', send (i, m') to the adversary.

$\mathbf{Q} = (\mathbf{Commit}, \mathsf{epoch})$

1. Compute $(\mathsf{trans}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Commit}$ $(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{epoch})$
2. Output trans

$\mathbf{Q} = (\mathbf{Withdraw}, \mathsf{pk_u}, \mathsf{v})$

1. Compute $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow$ $\mathsf{Withdraw}(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
2. Output trans

$\mathbf{Q} = (\mathbf{Contract}, \mathsf{trans})$

1. Compute $(\mathsf{St'_b}, \{\mathsf{St'_u}\}, \mathsf{TempSt'_b}, \{\mathsf{TempSt'_u}\})$ $\leftarrow \mathsf{Contract}(\{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{trans}, \mathsf{pp})$
2. Output $\{\mathsf{St'_u}\}, \mathsf{St'_b}$

$\mathbf{Q} = (\mathbf{Reveal}, \mathsf{pk})$

Output the secret key and the state of the user/bank who owns pk, i.e., output $\mathsf{Sk_u}, \mathsf{St_u}$

Fig. 6: Query Description in Blockchain-Bank Experiment

– *If one of the queries* $\mathbf{Q}_0$ *and* $\mathbf{Q}_1$ *is not legitimate, the other query will not proceed by the experiment as well.*
– *The leaked information of* $\mathbf{Q}_0$ *and* $\mathbf{Q}_1$ *should be the same, i.e.,* $\mathsf{Leakage}(\mathbf{Q_0}) = \mathsf{Leakage}(\mathbf{Q_1})$

Definition 5. *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EstablishBank}, \mathsf{NewUser}, \mathsf{Deposit}, \mathsf{Withdraw}, \mathsf{Pay}, \mathsf{Commit}, \mathsf{Contract})$ *be a candidate* BBANK *scheme and* λ *is the security parameter. We define the advantage of an adversary* $\mathcal{A}$ *in the* $\mathsf{Tx} - \mathsf{IND}$ *experiment as follows,*

$$\mathsf{Adv}^{\mathsf{Tx\text{-}IND}}_{\Pi,\mathcal{A}(\lambda)} = |2\Pr[b = b'] - 1|$$

4.3 Overdraft Safety

Informally, overdraft safety specifies that an honest user can withdraw all the balance that she owns according to her state in the withdraw phase of any epoch. This security requirement prohibits an adversary to withdraw more than what it has, since otherwise there must be an honest user who cannot withdraw all of his balance because of the lack of the funds in the smart contract.

Overdraft safety is defined by an experiment Overdraft, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given (candidate) BBANK scheme. We now describe the Overdraft experiment mentioned above. Given a BBANK scheme Π, an adversary $\mathcal{A}$, and security parameter λ, the (probabilistic) experiment $\mathsf{Overdraft}(\Pi, \mathcal{A}, \lambda)$ proceeds as the adversary is capable of sending all the queries in the experiment that we define in the beginning of this section. In addition, adversary can send $\mathbf{Q} = \mathbf{Reveal}$ for the secret key and state of the bank. In the challenge epoch, the adversary wins if in a certain epoch, there is an honest user who tries and fails to withdraw all his balance within one epoch.

Definition 6. *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EstablishBank}, \mathsf{NewUser}, \mathsf{Deposit}, \mathsf{Withdraw}, \mathsf{Pay}, \mathsf{Commit}, \mathsf{Contract})$ *be a candidate* BBANK *scheme and* λ *is the security parameter. We define the advantage of an adversary* $\mathcal{A}$ *in the* Overdraft *game as*

$$\mathsf{Adv}^{\mathsf{Overdraft}}_{\Pi,\mathcal{A}(\lambda)} = Pr[\exists \mathsf{u} \in \mathcal{U} \textit{ such that } \perp \leftarrow \mathsf{Withdraw}(\mathsf{b}_{\mathsf{max}}, \mathsf{pk}_{\mathsf{u}}, \mathsf{v})]$$

5 PriBank: Privacy-Preserving Scaling Construction

We present the construction of algorithms of PriBank in Figure 7 with a brief descriptions as follows. A taxonomy of symbols is provided in Appendix G.

– **Proof of commitment** Operator commits to a user's balance, transaction values, a randomness using Pedersen commitment, ComProve algorithm. A user verifies these commitments using ComVerify algorithm.
– **NIZK for updated balance** Operator collects all transactions of the users in one epoch, computes the new commitments for users' updated balances and gives NIZK proof using ProBal. The verification algorithms is VeriBal.
– **Signature** To provide the authenticity of data exchanged between users and operator, standard digital signatures are used in PriBank.

```
ProBal($\{c\}, \{c'\}, \{c_v\}, \{c_{t_i}\}, \{d_i\}, \{v_{ij}\}, g, h, \gamma, c_0, \{h_i\}, \{h'_i\}, \{t_i\}, \{b_i\}, \{b'_i\}, C$)
Take $(\{d_i\}, \{t_i\}, \{b_i\}, \{b'_i\}, \{v_{ij}\})$ as inputs of the circuit $C$, compute all the inner wires of $C$ as $a_{k+1}, ..., a_l$
set $\{c_k\} = \langle\{c_i\}, \{c'_i\}, \{c_v\}, \{c_{t_i}\}\rangle$
Compute $c_l$ as described in protocol 3.4
Run protocol 3.4 as a prover, generate proof $\pi_{zk}$
return : $\pi_{zk}$

VeriBal($\{c\}, \{c'\}, \{c_v\}, \{c_{t_i}\}, g, h, \gamma, c_0, \{h_i\}, \{h'_i\}, , C, \pi_{zk}) \to \{0,1\}$:
Run protocol 3.4 as a verifier, generate $\mathfrak{b} \in \{0,1\}$
return : $\mathfrak{b}$
```

```
ComProve($h^\gamma, h_i^\gamma, \gamma, h_i$):
$\alpha \leftarrow \mathbb{Z}_p$
Compute $d = h^\alpha, d' = h_i^\alpha$
$x \leftarrow$ Hash($h, h_i, h^\gamma, h_i^\gamma, d, d'$)
Compute $\theta \leftarrow \theta - x\gamma$.
return : $\pi \leftarrow (d, d', \theta)$.

ComVerify($g, h, h_i, c_0, c_i, b_i, \pi$):
Compute $x \leftarrow$ H($h, h_i, h^\gamma, c_0, d, d'$)
Let $c' = c_i / g^{b_i}$.
if $c_0^x h^\theta = d \wedge c'^x h^\theta = d'$ then
    return 1
else
    return 0

Sign($m, sk_E) \to \sigma_E$
VerifySig($m, \sigma_E, pk_E) \to \{0,1\}$
```

Fig. 7: PriBank Construction Algorithms, including syntax for digital signatures.

Following, we provide a brief description of the main processes of PriBank:

Register To register an account, a user u_i sends a request Reg consisting of signature σ_{b_i} on its balance b_i to the operator during an epoch r. The operator returns a randomness t_i with its commitment proofs and signature on these values. Further, user verifies all the details and accordingly sends a registration request to the smart contract along with a deposit transaction. The smart contract verifies the request and registers the user accordingly (Figure 8). After a send/receive transaction, the user's balance becomes private in later epochs.

```
Operator $O$
$\{$Reg : $pk_i, b_i, \sigma_{b_i}\} \leftarrow$
If VerifySig($b_i, \sigma_{b_i}, pk_i$)

1. $t_i \overset{\$}{\leftarrow} \mathbb{Z}_p, c_{t_i} \leftarrow g^{t_i} h_{t_i}^\gamma$
2. $\pi_1 \leftarrow$ ComProve($c_0, h_{t_i}^\gamma, \gamma, h_{t_i}$)
3. $m := \{c_{t_i}, r, pk_i\}$
4. $\sigma_o \leftarrow$ Sign($m, sk_o$)
5. Send $\{m, \sigma_o, t_i, \pi_1\}$ to $U \rightarrow$
```

```
User $U$

$d_1 \leftarrow$ VerifySig($m, \sigma_o, pk_o$)
$d_2 \leftarrow$ ComVerify($g, h, h_{t_i}, c_0, c_{t_i}, t_i, \pi_1$)
If $d_1 \wedge d_2$ :

1. $\sigma_i \leftarrow$ Sign($\langle m, \sigma_o\rangle, sk_i$)
2. Send $\{$Reg : $m, \sigma_o, \sigma_i, b_i\}$ to $SC \rightarrow$
```

```
Smart Contract $SC$

$d_1 \leftarrow$ VerifySig($m, \sigma_o, pk_o$)
$d_2 \leftarrow$ VerifySig($\langle m, \sigma_o\rangle, \sigma_i, pk_i$)
If $d_1 \wedge d_2$ :

1. $c_{b_i} \leftarrow g^{b_i} c_0$
2. Record($c_{t_i}, c_{b_i}, pk_i$)
3. $B \leftarrow B + b_i$
```

Fig. 8: User Registration

Transfer In each epoch r, each user u_i needs to get a fresh randomness t_i from the operator. The randomness is computed in similar way as in registration. To agree on the randomness computed by the operator, a user sends $\sigma_i \leftarrow \mathsf{Sign}(\langle m, \sigma_o \rangle, sk_i)$ to the operator and the operator verifies the signature by $\mathsf{VerifySig}(\langle m, \sigma_o \rangle, \sigma_i, pk_i)$ and records c_{t_i}, t_i. This randomness allows users to compute their balances $d_i = t_i - b_i$ during the end of the epoch, where d_i is published by the operator. The user u_i can compute the correctness of his balance by the published d_i before the end of the epoch. The randomness t_i is used as a 'mask' for the user's balance. The necessity of this randomness is for the balance to be sealed by the operator. Therefore, if a user receives some transactions that it doesn't know the transaction value, then, the user cannot compute its balance directly from its own transaction history.

Operator O	**User** U
Transaction lists: $\mathcal{T}_r,\ \mathcal{CT}_r$	Transaction lists: $\mathcal{T}_r,\ \mathcal{CT}_r$
	$\mathsf{Tx}'_{ij}{:}(pk_i, pk_j, v_{ij}, \mathsf{Null}, r, n, \sigma_{ij})$
	$\leftarrow$ Send Tx'_{ij} to O
$d \leftarrow \mathsf{ValidateTx}(\mathsf{Tx}'_{ij}, b_i)$	
If $\neg(\mathsf{Online}(u_i) \wedge \mathsf{Online}(u_j) \wedge d)$: abort	
${b_i}' \leftarrow b_i - v_{ij}, \quad c'_i \leftarrow g^{{b_i}'} h_i^{\gamma}, \quad c_{ij} \leftarrow g^{v_{ij}} h_{ij}^{\gamma}$	
Let $\mathsf{Tx}_{ij} : (pk_i, pk_j, v_{ij}, c_{ij}, r, n)$, $\mathsf{CTx}'_{ij} : (pk_j, c_{ij})$	
$\mathcal{CT}'_r = \{\mathsf{CTx}_{ik} \vert \mathsf{CTx}_{ik} \in \mathcal{CT}_r \wedge k \neq j\} \cup \{\mathsf{CTx}'_{ij}\}$	
$\pi \leftarrow \mathsf{ComProve}(c_0, h_{ij}^{\gamma}, \gamma), m := \{\pi, \mathcal{CT}'_r\}$	
$\sigma_o \leftarrow \mathsf{Sign}(m, sk_o)$	
Send $\{m, \sigma_o\}$ to $U \rightarrow$	Check if $\mathcal{CT}'_r$ is the correct history
	$d_1 \leftarrow \mathsf{VerifySig}(m, pk_o, \sigma_o)$
	${b_i}' \leftarrow b_i - v_{ij}$
	$d_2 \leftarrow \mathsf{ComVerify}(g, h, h_{ij}, c_0, c_{ij}, v_{ij}, \pi)$
	If $d_1 \wedge d_2 : \sigma_i \leftarrow \mathsf{Sign}(\mathcal{CT}', sk_i)$
$d_1 \leftarrow \mathsf{VerifySig}(\mathcal{CT}', \sigma_i, pk_i)$	$\leftarrow$ Send σ_i to O
If d_1; Update transaction lists	
$\mathcal{T}_r \leftarrow \{\mathsf{Tx}_{ik} \vert \mathsf{Tx}_{ik} \in \mathcal{T}_r \wedge k \neq j\} \cup \{\mathsf{Tx}_{ij}\}$	
$\mathcal{CT}_r \leftarrow \mathcal{CT}'_r$	

Fig. 9: Transfer

If a user wants to send transactions in an epoch, he/she needs to keep records of two transaction lists $\mathcal{T}_r$ and $\mathcal{CT}_r$, the operator keeps records of these two lists for each user. The list $\mathcal{T}_r$ contains the plain transactions that user sent in epoch r, while $\mathcal{CT}$ only contains the confidential abbreviated transactions, i.e. $\mathcal{CT} = \{\mathsf{CTx}_{ij} \vert \mathsf{CTx}_{ij} : (pk_j, C_{ij})\}$ To send a transaction, a user sends the plain transaction to the operator. The operator checks its validation, commits to the transaction value, signs the transaction, and gives a proof of commitment. Meanwhile, the operator aggregates all the transactions sent by the user in this epoch, signs the confidential transaction list, and sends it to the user. If the user agrees to the confidential transaction list, he/she returns its signature on the list. Henceforth, a user confirms all its sent transactions in this epoch. Before the end of one epoch, the operator collects all the confidential lists of each user and sends them to the smart contract. The smart contract checks these lists and records the transactions accordingly. Figure 9 depicts the transfer process.

Commit Before the end of an epoch, the operator collects all the confidential transaction lists of the users, compute the new balance and its commitments for each user i.e. $c_i = g^{b_i} h_i^{\gamma}$. Further, it computes $d_i = b_i - t_i, \forall i \in \{1, ..., N\}$ where t_i is the randomness that the operator agrees on with each user during the start of epoch. In case where a user has some receiving transactions during an epoch but the user did not agree on a randomness with the operator, the operator sets $d_i = \perp$. Subsequently, it generates a zero-knowledge proof for the correct computation by $\pi \leftarrow \mathsf{ProvBal}(\{\mathcal{CT}_r\}, \{c_i\}, \{d_i\})$, where the inputs are the confidential transaction lists, the balance commitments in the previous epoch, the updated balance commitments, the randomness commitments and d_i of all users. Finally, it submits the following data to the smart contract:

User Index	$\mathcal{CT}$ and Sig	t_i and Sig	Balance	d
u_1	$\mathcal{CT}_1^r$, σ_1	c_{t_1}, σ_{t_1}	c_1	d_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_i	$\perp$, $\perp$	c_{t_i}, $\perp$	c_i	d_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_N	$\mathcal{CT}_N^r$, σ_N	c_{t_N}, σ_{t_N}	c_N	d_N
Operator signature on the above data: σ_o				
zero-knowledge proof: π				

Upon receiving the data submitted by the operator, the smart contract checks the validation of each signature by $\mathsf{VerifySig}(m, \sigma_i, pk_i)$ and the validation of the zero-knowledge proof $\mathsf{VerifyBal}(\{\mathcal{CT}_r\}, \{c_i\}, \{d_i\})$. For every user u_i who did not agree on a randomness with the operator in this epoch, smart contract updates this user's balance commitment as his/her balance commitment in the previous epoch and for the other users who have send transaction with value C_{ji} to this user, it updates their balance commitments as $c_j = c_i \cdot C_{ji}$.

Exit At the end of every epoch, a user who wants to exit the system can send a request to smart contract during the exit phase. The smart contract transfers the funds back to the user if the request is valid. The user who wants to exit gets the randomness and its proof of the latest randomness commitment that it agreed on with the operator. Generate the request as $\{\mathsf{exit}, pk_i, t_i, \pi, \sigma_i\}$ where σ_i is the user's signature on the request. Upon receiving the request, the smart contract verifies the signatures and verifies the proof of the most recently committed randomness by $\mathsf{ComVerify}(g, h, h_{t_i}, c_0, c_{t_i}, t_i, \pi)$. Transfer user's balance $b_i = t_i - d_i$ if all the verification get accepted.

6 Security Proof

6.1 Proof for Transaction Indistinguishability

We describe a simulation experiment $\mathbf{Exp}_{sim}$ in which the adversary $\mathcal{A}$ interacts with an experiment as in the $\mathsf{Tx\text{-}IND}$ experiment. While the answer of the experiment to the challenge queries is independent of b, therefor the $\mathcal{A}$'s advantage in the $\mathbf{Exp}_{sim}$ is 0. We further prove that the simulation experiment is indistinguishable from the real experiment $\mathbf{Exp}_{real}$.

$\mathbf{Exp}_{sim}$ In the challenge phase, how the simulated experiment answers the queries from the adversary is different from the $\mathbf{Exp}_{real}$ as follows,

– $Q = (\mathbf{Commit}, \mathsf{epoch})$
  1. The operator collects all the transaction lists in this epoch, calculates the new balance for each user. Then the operator computes the balance masks as $d = t - b$ for users that the adversary has asked reveal key query before and select random values as balance masks for honest users.
  2. Simulate a zero-knowledge proof for the statement.
  3. Output trans.

Experiment $\mathbf{Exp}_1$. The experiment $\mathbf{Exp}_1$ modifies the real experiment by simulating the zero-knowledge proof. Since the zk-SNARK system is perfect honest-verifier zero knowledge, the distribution of the simulated proof is identical to the proof computed in a real experiment. Hence, the advantage of distinguishability of the adversary for experiments $\mathbf{Exp}_{real}$ and $\mathbf{Exp}_1$ is 0.
Experiment $\mathbf{Exp}_2$. The experiment $\mathbf{Exp}_2$ modifies $\mathbf{Exp}_1$ by replacing all the commitments for honest users' balances, randomness t_i and transactions values by commitments to random values. Precisely, in the real experiment, every time a user sends a transaction to the operator, the operator hides the transaction value by making a commitment to it (Figure 9) and publishes the commitment on the blockchain later. In $\mathbf{Exp}_2$, the operator commits to a random value instead of the transaction value, by Lemma 1 (see below), $|\mathsf{Adv}^{\mathbf{Exp}_1} - \mathsf{Adv}^{\mathbf{Exp}_2}| \leq q \cdot \mathsf{Adv}^{\mathbf{DDH}}$.
Experiment $\mathbf{Exp}_{sim}$. $\mathbf{Exp}_{sim}$ modifies $\mathbf{Exp}_2$ by replacing all the balance masks d_i for honest users to random values. Since the adversary does not know the honest users' t_i and these values are also random, hence the distributions for d_i in Exp_{sim} is indistinguishable.

As argued above, the responses of the adversary $\mathcal{A}$ to $\mathbf{Exp}_{sim}$ is independent of b, so that $\mathsf{Adv}^{\mathbf{Exp}_{sim}} = 0$. Then, by summing over $\mathcal{A}$'s advantages in the hybrid experiments, we can bound $\mathcal{A}$'s advantage in $\mathbf{Exp}_{real}$ by

$$\mathsf{Adv}^{\mathbf{Exp}_{real}} \leq q \cdot \mathsf{Adv}^{\mathbf{DDH}}$$

Lemma 1. *Let Adv^{DDH} be the advantage of an adversary in DDH problem. Then after q **Pay.User** queries,* $|\mathsf{Adv}^{\mathbf{Exp}_2} - \mathsf{Adv}^{\mathbf{Exp}_1}| \leq q \cdot \mathsf{Adv}^{\mathbf{DDH}}$

Proof. In the challenge phase of the experiment, when the adversary sends the query Commit, the experiment replies with a trans that includes all the commitments for the transaction values and all the commitments for users' balances in the current epoch. While in $\mathbf{Exp}_2$, the commitments for the transactions and balances of honest users are actually commitments to the random values. We argue the two experiments are indistinguishable.

In $\mathbf{Exp}_1$, a commitment for a transaction from u_i to u_j is $c_{ij} = g^v h_{ij}^{\gamma}$ where v is the transaction value that known by the adversary, h_{ij} is a random generator such that the commitment key is unknown, γ is a secret held by the operator, while h^{γ} is publicly known. In $\mathbf{Exp}_2$, a commitment for such a transaction is computed as $c = g^v h_{ij}^s$ where s is uniformly selected from $\mathbb{Z}_p$. If an adversary can distinguish the tuple $(h^{\gamma}, h_{ij}, g^v h_{ij}^{\gamma})$ from $(h^{\gamma}, h_{ij}, g^v h_{ij}^s)$, it can also solve the DDH tuple $(h^{\gamma}, h_{ij}, h_{ij}^z)$. The similar argument applies to the commitments for users' balances and the commitments for agreed randomness t_i. Hence, we have $q \cdot \mathsf{Adv}^{\mathbf{DDH}} \geq |\mathsf{Adv}^{\mathbf{Exp}_1} - \mathsf{Adv}^{\mathbf{Exp}_2}|$ where q is the total number of commitments. $\square$

6.2 Proof for Overdraft Safety

Assuming the challenge epoch is n, the output of the experiment for query $\mathbf{Q} = \mathbf{Commit}$ is trans_n. At the end of this epoch, an honest user u_i fails to withdraw her total balance b_i. b_i is the balance recorded in the state of u_i. It is computed by the user's balance in the previous epoch being deduced by the user's spending and being added by the user's receiving in the current epoch. The experiment is performed under a setting that the signature forgery is impossible.

In the exit phase, an honest user u_i submits an exit request $\{\mathsf{exit}, pk_i', t_i, \pi, \sigma_i\}$, the smart contract accepts the request only if $1 \leftarrow \mathsf{ComVerify}(g, h, h_{t_i}, c_0, c_{t_i}, t_i, \pi)$. Suppose $\mathsf{ComVerify}$ fails, it indicates the proof π is incorrect. While we observe that the same algorithm with the same parameter is run by the user before the operator commits to blockchain (due to randomness agreement), since the user is honest, it should abort the execution in the transfer phase and refuse to sign the randomness commitment at that point.

Apart from the failure of $\mathsf{ComVerify}$, a user can also fail to withdraw in the case that $b_i \neq b_i'$, i.e., the user withdraws less than what she believes to have. This can be the result of $b_i \neq t_i - d_i$ or the lack of funds in the smart contract. In the earlier case, it indicates the soundness of zero-knowledge proof is broken. In the later case, it implies that there must be at least one adversary, say u_j, who withdraws more than what it has, namely, $b_j \geq t_j - d_j$, which also indicates the soundness of zero-knowledge is broken.

7 Implementation and Evaluation

We evaluate the usability of the commit-and-prove zero-knowledge proof of *PriBank* and give its implementation in *Go*. We use elliptic curve secp256k1 with 128-bit security, the balances and transactions are 8-bits length, we test the proof size per user, transaction size, the proof generation time, verification time, the time for pre-processing. We do not deploy optimization on the implementation, therefore the performance in terms of time is not desirable that results in "low" cost of transaction instead of "Very low" cost similar to ZK-Rollup as depicted in Table 1. Particularly, the pre-processing to generate a QAP takes too long to perform further experiments on more users cases. We refer to the work [9,2] for optimization and faster implementation. The experiments were performed on 2×24 Xeon 2.4 GHz cores. The code of the implementation can be found in [1].

The commitment sent by the operator to the smart contract includes balance and transaction commitments, balance mask d, two signatures for each user, and a proof. The computation circuit of the operator allows each user in the system to send a transaction to every other user, Therefore, for an n user circuit the maximum transaction number is $n(n-1)$. Assuming a user can send a transaction with a zero value, the total commitment size of the operator is computed based on the maximum transaction number. While the number of transactions that have positive value might be much less, we measure the transaction size in a case where we assume a user send non-zero transactions to half of the other users in one epoch. Following, we depict the proof size, transaction size, and circuit gate numbers dependency of users in Figure 10.

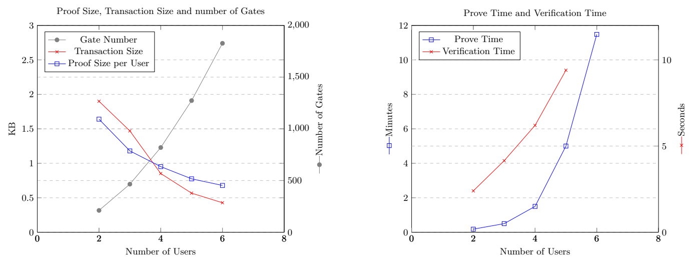

Fig. 10: Experimental Results

8 Conclusion

Our goal in this work was to implement privacy over an account-based blockchain system. We formally defined a blockchain-based bank model along with security definitions. Following that, we presented a novel privacy-preserving cryptocurrency system PriBank, for that, we constructed an efficient Commit-and-Prove NIZK protocol. Our construction achieves privacy together with scalability in the blockchain which has not been achieved by the previous schemes. As most of the schemes are concerned with achieving privacy or scalability alone, hence, we compared our scheme separately with the popular privacy and scalability solutions. We provided a detailed security analysis and performance evaluation.

Future Directions There are various ways to adapt and extend this work. One possibility is to reduce the complexity of the data which is sent to the smart contract in each epoch, henceforth increasing the efficiency of the overall system. Another direction of work is to build more efficient proof algorithm to reduce the verification complexity.

References

1. Pribank in Go
2. libsnark: a C++ library for zkSNARK proofs
3. Agrawal, S., Ganesh, C., Mohassel, P.: Non-interactive zero-knowledge proofs for composite statements. In: Annual International Cryptology Conference. pp. 643–673. Springer (2018)
4. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., Wuille, P.: Enabling blockchain innovations with pegged sidechains. http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains **72** (2014)
5. Benarroch, D., Campanelli, M., Fiore, D., Kolonelos, D.: Zero-knowledge proofs for set membership: Efficient, succinct, modular. IACR Cryptol. ePrint Arch. **2019**, 1255 (2019)
6. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: Anonymity for bitcoin with accountable mixes. In: International Conference on Financial Cryptography and Data Security. pp. 486–504. Springer (2014)
7. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: Zexe: Enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 947–964 (2020). https://doi.org/10.1109/SP40000.2020.00050
8. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: International Conference on Financial Cryptography and Data Security. pp. 423–443. Springer (2020)
9. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018)
10. Campanelli, M., Fiore, D., Querol, A.: Legosnark: Modular design and composition of succinct zero-knowledge proofs. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2075–2092 (2019)
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. pp. 494–503 (2002)
12. Cecchetti, E., Zhang, F., Ji, Y., Kosba, A., Juels, A., Shi, E.: Solidus: Confidential distributed ledger transactions via pvorm. In: Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 701–717 (2017)
13. Danezis, G., Meiklejohn, S.: Centrally banked cryptocurrencies. arXiv preprint arXiv:1505.06895 (2015)
14. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Symposium on Self-Stabilizing Systems. pp. 3–18. Springer (2015)
15. Diamond, B.E.: Many-out-of-many proofs and applications to anonymous zether. In: 2021 2021 IEEE Symposium on Security and Privacy (SP). pp. 1800–1817. IEEE Computer Society, Los Alamitos, CA, USA (may 2021). https://doi.org/10.1109/SP40001.2021.00026, `https://doi.ieeecomputersociety.org/10.1109/SP40001.2021.00026`
16. Dziembowski, S., Fabiański, G., Faust, S., Riahi, S.: Lower bounds for off-chain protocols: Exploring the limits of plasma. In: 12th Innovations in Theoretical Computer Science Conference (ITCS 2021) (2021)
17. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 649–678. Springer (2019)

18. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 626–645. Springer Berlin Heidelberg (2013)
19. Gluchowski, A.: Zk rollup: scaling with zero-knowledge proofs. Matter Labs (2019)
20. Kerber, T., Kiayias, A., Kohlweiss, M.: Kachina–foundations of private smart contracts. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). pp. 1–16. IEEE (2021)
21. Khalil, R., Zamyatin, A., Felley, G., Moreno-Sanchez, P., Gervais, A.: Commit-chains: Secure, scalable off-chain payments. Cryptology ePrint Archive, Report 2018/642 (2018)
22. Kilian, J.: Uses of randomness in algorithms and protocols. In: Massachusetts Institute of Technology (1990)
23. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE symposium on security and privacy (SP). pp. 839–858. IEEE (2016)
24. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world. In: Post on Bitcoin forum
25. Miller, A., Bentov, I., Kumaresan, R., McCorry, P.: Sprites: Payment channels that go faster than lightning. CoRR abs/1702.05812 **306** (2017)
26. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf (2009)
27. Narula, N., Vasquez, W., Virza, M.: zkledger: Privacy-preserving auditing for distributed ledgers. In: 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18). pp. 65–80 (2018)
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)
29. Poon, J., Buterin, V.: Plasma: Scalable autonomous smart contracts. White paper (2017)
30. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
31. Steffen, S., Bichsel, B., Gersbach, M., Melchior, N., Tsankov, P., Vechev, M.: zkay: Specifying and enforcing data privacy in smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1759–1776 (2019)
32. The Monero Project: Monero (2014), `https://web.getmonero.org`
33. Wood, G.: Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow Paper (2014)
34. Zyskind, G., Nathan, O., et al.: Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops. pp. 180–184. IEEE (2015)

A Commit-Prove Zero-Knowledge Proof Construction

Circuit: The arithmetic circuit C of the zero-knowledge proof is in Figure 11

B Proof of Protocol 3.1

Proof. **Soundness**. By the rewinding, the prover, the extractor $\mathcal{X}$ gets two valid transcripts that have the same commitments:

Inputs:

- Transaction values v_{ij} where $i, j \in 1, .., N, i \neq j$
- Users' randomness $\{t_i\}_{i=1}^N$
- Users' balances of the previous epoch b_i^{r-1} and the updated balance b_i^r
- User's balance mask $d_i, i \in \{1, ..., N\}$
- Total balance of the contract B

The circuit check the following constraints for the inputs:

$$b_i^r = b_i^{r-1} - \sum_{j=1}^N v_{ij} + \sum_{j=1}^N v_{ji} \wedge b_i^r = t_i - d_i \wedge B = \sum_{i=1}^N b_i^r \wedge v_{ij} \geq 0 \wedge b_i^{r-1} - \sum_{j=0}^N v_{ij} \geq 0$$

Fig. 11: Circuit

$(d_1, d_2, c_0, c_1, x, \theta_a, \theta_b, \theta_1, \theta_2, \theta_{ab})$, $(d_1, d_2, c_0, c_1, x', \theta'_a, \theta'_b, \theta'_1, \theta'_2, \theta'_{ab})$ from the verification, we get equations

$$c_a^x g^{\theta_a} h^{\theta_1} = d_1 \qquad c_a^{x'} g^{\theta'_a} h^{\theta'_1} = d_1$$

By the binding property of Pedersen commitment, This implies $a = \frac{\theta'_a - \theta_a}{x - x'}$, by the same technique, $\mathcal{X}$ can compute $b = \frac{\theta'_b - \theta_b}{x - x'}$ and α, β.

Next, assume c is a commitment that committed to z, we will prove $z = ab$. Assume $c_0 = g^u h^{r_{c_0}}, c_1 = g^v h^{r_{c_1}}$, observe that $g^{\theta_a \theta_b} h^{\theta_{ab}} c_0^x = c^{x^2} c_1$, it implies

$$g^{abx^2 - (a\beta + b\alpha)x + \alpha\beta + ux} h^{\theta_{ab} + x r_{c_0}} = g^{zx^2 + v} h^{r_c x^2 + r_{c_1}}$$

Since $a, b, \alpha, \beta, u, v$ are all predefine value, either $\mathcal{X}$ can extract non-trivial relation between g, h or $u = \alpha b + \beta a$ and the extractor can extract $z = ab = \frac{\theta_a \theta_b - \theta'_a \theta'_b + (\alpha b + \beta a)(x - x')}{x^2 - x'^2}$

Perfect special honest-verifier zero-knowledge. The simulator randomly chooses $\theta_1, \theta_2, \theta_a, \theta_b, \theta_{ab}, u, r \leftarrow \mathbb{Z}_p$ and randomly chooses a challenge $x \leftarrow \mathbb{Z}_p$, it computes $d_1 = c_a^x g^{\theta_a} h^{\theta_1}$, $d_2 = c_b^x g^{\theta_b} h^{\theta_2}, c_0 = g^u h^r, c_1 = g^{\theta_a \theta_b} h^{\theta_{ab}} c_0^x / c^{x^2}$. Thus the simulator produces a valid transcript $(d_1, d_2, c_0, c_1, x, \theta_a, \theta_b, \theta_1, \theta_2, \theta_{ab})$ that has the identical probability distributions with the real proof.

□

C Proof of Protocol 3.2

Proof. **Soundness**. For an accepting transcripts $(c_0, \Omega, d_1, d_2, x, \theta_1, \theta_2)$, assume that

$$c_0 = h^\gamma, \Omega = \tau^u h^{-v}, d_1 = h^\alpha, d_2 = \tau^\delta h^\beta$$

since $d_1 = c_0^x h^{\theta_1}$, $d_2 = \Omega^x \tau^{\theta_1} h^{\theta_2}$ we have

$$h^{x\gamma + \theta_1} = h^\alpha, \quad \tau^{ux + \theta_1} h^{\theta_2 - vx} = \tau^\delta h^\beta$$

If $u \neq \gamma$ then it means $\theta_1 = \frac{\delta\gamma - u\alpha}{\gamma - u}$ and $x = \frac{\alpha - \delta}{\gamma - u}$ or the cheating prover is able to compute the Pedersen commitment key $\log_g h$. Since α, δ, γ are pre-defined values, $\Pr[x = \frac{\alpha - \delta}{u - \gamma}] = \frac{1}{p}$.

Since in the verification, $c = \prod_{i=1}^{n} c_i^{b_i} / \Omega$, assume $c = g^w h^t$, this implies

$$g^w h^t = g^{\sum_{i=1}^{n} a_i b_i} h^v$$

Hence, either $w = \sum_{i=1}^{n} a_i b_i$ or the prover is able to compute the discrete logarithm.

□

Perfect special honest-verifier zero-knowledge The simulator randomly chooses $\gamma, \theta_1, \theta_2 \leftarrow \mathbb{Z}_p$ and computes $c_0 = h^\gamma$, $\Omega = \prod_{i=1}^{n} c_i^{b_i}/c$. Then the simulator chooses a challenge randomly $x \leftarrow \mathbb{Z}_p$ and computes $d_1 = c_0^x h^{\theta_1}, d_2 = \Omega^x \tau^{\theta_1} h^{\theta_2}$. The transcript $trs = (c_0, d_1, d_2, x, \theta_1, \theta_2)$ is a valid transcript that has the identical probability distributions with the real proof.

D Proof of Protocol 3.3

Proof. We follow the proof of [9] for the soundness, and give our proof for the zero-knowledge property.

Soundness. We firstly construct an extractor $\mathcal{X}_1$ of protocol Prove, then construct an extractor $\mathcal{X}_2$ for protocol 3.3. For $\mathcal{X}_1$, we use an inductive argument showing that in each step, we either extract a witness or a discrete log relation. If $n = |g| = 1$, rewinding $\mathcal{P}$ to get 2 transcripts with the same randomness used by $\mathcal{P}$ but different challenges from $\mathcal{V}$, assume the witness of $\mathcal{P}$ are (a_1, c, r), $d = g_1^{t_1} u^{t_2} h^{t_3}$, the transcripts are

$$tr := (d, x, \theta_1, \theta_2)$$
$$tr' := (d, x', \theta_1', \theta_2')$$

then we get $g_1^{a_1 x + \theta_1} u^{cx + b_1 \theta_1} h^{\theta_2 + xr} = g_1^{a_1 x' + \theta_1'} u^{cx' + b_1 \theta_1'} h^{\theta_2' + x' r} = d$.

Since a_1, c, d are predefined value, either extractor can compute

$$\log_{g_1} u + \log_{g_1} h = \frac{\theta_1 - \theta_1'}{b_1 \theta_1' + \theta_2' - b_1 \theta_1 - \theta_2}$$

or $a_1 = \frac{\theta_1 - \theta_1'}{x' - x}$ and $c = a_1 b_1$

Next, on the k-th recursive step that on input($\boldsymbol{g}, u, h, c, \boldsymbol{b}$), assume that the $(k+1)$th recursive step has input($\boldsymbol{g}', u, h, c', \boldsymbol{b}'$) and the witness can be extracted from this recursive are $r', \boldsymbol{a}', \langle \boldsymbol{a}', \boldsymbol{b}' \rangle$. We show that with the witness of the $(k+1)$th recursive step, an extractor can effectively compute a witness of the k-th recursive step or a non-trivial discrete logarithm relation between the generators.

On k-th recursive step, the extractor runs the prover to get L and R. Then, by rewinding the prover four times and giving it four different challenges x_1, x_2, x_3, x_4, the extractor obtains four $\boldsymbol{a}_i' \in \mathbb{Z}_p^{n'}$ such that

$$c \cdot L^{x_i^2} \cdot R^{x_i^{-2}} = \left(\boldsymbol{g}_{[:n']}^{x_i^{-1}} \circ \boldsymbol{g}_{[n':]}^{x_i^2}\right)^{\boldsymbol{a}'_i} h^{r'_i} u^{\langle \boldsymbol{a}'_i, \boldsymbol{b}'_i \rangle} \quad \text{for } i = 1, ..., 4 \tag{1}$$

compute $v_1, v_2, v_3 \in \mathbb{Z}_p$ such that

$$\sum_{i=1}^{3} v_i x_i^2 = 1, \qquad \sum_{i=1}^{3} v_i = 0, \qquad \sum_{i=1}^{3} v_i x_i^{-2} = 0 \tag{2}$$

Then taking a linear combination of the first three equations with v_1, v_2, v_3 as the coefficients,

$$c^{v_i} \cdot L^{x_i^2 v_i} \cdot R^{x_i^{-2} v_i} = (\boldsymbol{g}_{[:n']}^{x_i^{-1}} \boldsymbol{g}_{[n':]}^{x_i})^{\boldsymbol{a}'_i v_i} h^{v_i r'_i} u^{\langle \boldsymbol{a}'_i, \boldsymbol{b}'_i \rangle v_i} \text{ for } i = 1, 2, 3$$

we can compute

$$L = \boldsymbol{g}^{\boldsymbol{a}_L} h^{r_L} u^{s_L} \text{ , where}$$

$$r_L = \sum_{i=1}^{3} v_i r'_i, \quad \boldsymbol{a}_{L[:n']} = \sum_{i=1}^{3} x_i^{-1} \boldsymbol{a}'_i v_i, \quad \boldsymbol{a}_{L[n':]} = \sum_{i=1}^{3} x_i \boldsymbol{a}'_i v_i, \quad s_L = \sum_{i=1}^{3} \langle \boldsymbol{a}'_i, \boldsymbol{b}'_i \rangle v_i$$

Repeating this process with different combinations (compute v_1, v_2, v_3 of equation 2 with different summations), we can also compute R, c such that

$$R = \boldsymbol{g}^{\boldsymbol{a}_R} h^{r_R} u^{s_R}$$
$$c = \boldsymbol{g}^{\boldsymbol{a}_c} h^{r_c} u^{s_c}$$

Now, we can rewrite equation 1, for each $x \in \{x_1, x_2, x_3, x_4\}$ as

$$\boldsymbol{g}^{\boldsymbol{a}_L x^2 + \boldsymbol{a}_c + \boldsymbol{a}_R x^{-2}} h^{x^2 r_L + r_c + x^{-2} r_R} u^{x^2 s_L + s_c + x^{-2} s_R} = \boldsymbol{g}_{[:n']}^{\boldsymbol{a}' \cdot x^{-1}} \boldsymbol{g}_{[n':]}^{\boldsymbol{a}' \cdot x} h^{r'} u^{\langle \boldsymbol{a}', \boldsymbol{b}' \rangle}$$

This implies that

$$\boldsymbol{a}' \cdot x^{-1} = x^2 \boldsymbol{a}_{L[:n']} + \boldsymbol{a}_{c[:n']} + x^{-2} \boldsymbol{a}_{R[:n']}$$
$$\boldsymbol{a}' \cdot x = x^2 \boldsymbol{a}_{L[n':]} + \boldsymbol{a}_{c[n':]} + x^{-2} \boldsymbol{a}_{R[n':]}$$
$$\langle \boldsymbol{a}', \boldsymbol{b}' \rangle = x^2 s_L + s_c + x^{-2} s_R$$

Either the extractor can obtain a non-trivial discrete logarithm relation between the generators $(\boldsymbol{g}, h, u)$ if these equations do not hold, or we can deduce that for each challenge $x \in \{x_1, x_2, x_3, x_4\}$

$$x^3 \boldsymbol{a}_{L[:n']} + x(\boldsymbol{a}_{c[:n']} - \boldsymbol{a}_{L[n':]}) + x^{-1}(\boldsymbol{a}_{R[:n']} - \boldsymbol{a}_{c[n':]}) - x^{-3} \boldsymbol{a}_{R[n':]} = 0$$

The only way the above equation hold for all challenges is if

$$\boldsymbol{a}_{L[:n']} = \boldsymbol{a}_{R[n':]} = 0, \quad \boldsymbol{a}_{c[:n']} = \boldsymbol{a}_{L[n':]}, \quad \boldsymbol{a}_{R[:n']} = \boldsymbol{a}_{c[n':]}$$

Thus $\boldsymbol{a'} = x\boldsymbol{a}_{c[:n']} + x^{-1}\boldsymbol{a}_{c[n':]}$ Using these values we can see that:

$$\begin{aligned} x^2 s_L + s_c + x^{-2} s_R &= \langle \boldsymbol{a'}, \boldsymbol{b'} \rangle \\ &= \langle \boldsymbol{a}_{c[:n']}, \boldsymbol{b}_{[n':]} \rangle \cdot x^2 + \langle \boldsymbol{a}_c, \boldsymbol{b} \rangle + \langle \boldsymbol{a}_{c[n':]}, \boldsymbol{b}_{[:n']} \rangle \cdot x^{-2} \end{aligned}$$

Since the relation holds for all $x \in \{x_1, x_2, x_3, x_4\}$, it must be that

$$\langle \boldsymbol{a}_c, \boldsymbol{b} \rangle = s_c$$

The extractor, thus, either extracts a discrete logarithm relation between the generators, or the witness $\boldsymbol{a}_c$.

We now show that at the beginning of the protocol 3.3, on input $(c_{\boldsymbol{a}}, c_{\boldsymbol{ab}}, \boldsymbol{g}, \boldsymbol{b})$, the extractor $\mathcal{X}_2$ runs $\mathcal{P}$ with challenge x and uses $\mathcal{X}_1$ to obtain a witness $\boldsymbol{a}, r$ such that $c_{\boldsymbol{a}} c_{\boldsymbol{ab}}^x = \boldsymbol{g}^{\boldsymbol{a}} g^{x\langle \boldsymbol{a}, \boldsymbol{b} \rangle} h^r$. Rewinding $\mathcal{P}$ with a different challenge x' and $\mathcal{X}_1$ extracts new witness $\boldsymbol{a'}, r'$ such that $c_{\boldsymbol{a}} c_{\boldsymbol{ab}}^{x'} = \boldsymbol{g}^{\boldsymbol{a'}} g^{x\langle \boldsymbol{a'}, \boldsymbol{b} \rangle} h^{r'}$. Then we get

$$g^{s(x-x')} h^{r_{ab}(x-x')} = \boldsymbol{g}^{\boldsymbol{a}-\boldsymbol{a'}} g^{x\langle \boldsymbol{a}, \boldsymbol{b} \rangle - x' \langle \boldsymbol{a'}, \boldsymbol{b} \rangle} h^{r-r'}$$

Unless $\boldsymbol{a} = \boldsymbol{a'}$ we get a not trivial discrete log relation between $\boldsymbol{g}, h$and g. Otherwise we get $s = \langle \boldsymbol{a}, \boldsymbol{b} \rangle, r_{ab} = \frac{r-r'}{x-x'}, r_a = r - \frac{x(r-r')}{x-x'}$. □

Perfect Zero-Knowledge. The simulator chooses randomly a vector $\boldsymbol{a} \in \mathbb{Z}_p^n$ as witness and we show it can generate a valid transcripts for this vector.

For each recursive step when a prover asks for L, R, the simulator chooses randomly $r_1, r_2 \in \mathbb{Z}_p^*$, and computes

$$\begin{aligned} L &= \boldsymbol{g}_{[n':]}^{\boldsymbol{a}_{[:n']}} \cdot u^{\langle \boldsymbol{a}_{[:n']}, \boldsymbol{b}_{[n':]} \rangle} \cdot h^{r_1} \in \mathbb{G} \\ R &= \boldsymbol{g}_{[:n']}^{\boldsymbol{a}_{[n':]}} \cdot u^{\langle \boldsymbol{a}_{[n':]}, \boldsymbol{b}_{[:n']} \rangle} \cdot h^{r_2} \in \mathbb{G} \end{aligned}$$

Assume that at the last recursive step the input commitment is c', the challenge is x. The simulator randomly choose $\theta_1, \theta_2 \in \mathbb{Z}_p^*$, compute $d = c'^x g_1^{\theta_1} u^{b_1 \theta_1} h^{\theta_2}$.

The transcript $trs = (c, L_1, R_1, x_1, L_2, R_2, x_2, ..., d, x, \theta_1, \theta_2)$ is a valid transcript that has the identical probability distributions with the real proof.

E Proof of Protocol 3.4

Proof. **Soundness** A valid transcript of protocol 3.4 consists of 8 sub-transcripts: three transcripts of Protocol 3.1 on statements

$$\begin{aligned} &(\{c_1\}_{i=1}^k, \{h_i\}_{i=1}^k, g, h, c_u, \{u_i(x_1)\}_{i=1}^k); \\ &(\{c_1\}_{i=1}^k, \{h_i\}_{i=1}^k, g, h, c_v, \{v_i(x_1)\}_{i=1}^k); \\ &(\{c_1\}_{i=1}^k, \{h_i\}_{i=1}^k, g, h, c_w, \{w_i(x_1)\}_{i=1}^k) \end{aligned}$$

respectively; four transcripts of Protocol 3.2 on statements $(\boldsymbol{g}, \boldsymbol{b} := \{u_{k+1}(x_1), ..., u_l(x_1)\}, c_l, c_{\boldsymbol{u}})$, $(\boldsymbol{g}, \boldsymbol{b} := \{v_{k+1}(x_1), ..., v_l(x_1)\}, c_l, c_{\boldsymbol{v}})$, $(\boldsymbol{g}, \boldsymbol{b} := \{w_{k+1}(x_1), ..., u_l(x_1)\}, c_l, c_{\boldsymbol{w}})$ and $(\boldsymbol{g}, \boldsymbol{b} := \{xz(x_1), ..., x^{n-2}z(x_1)\}, c_h, c_{hz})$ respectively; one transcript of Protocol 3.3 on statement (c_a, c_b, c_c).
The soundness of protocol 3.1 implies

$$c_u = g^{\sum_{i=1}^{k} a_i u_i(x_1)} h^{t_u}, \quad c_v = g^{\sum_{i=1}^{k} a_i v_i(x_1)} h^{t_v}, \quad c_w = g^{\sum_{i=1}^{k} a_i u_i(x_1)} h^{t_w}$$

The soundness of protocol 3.3 implies

$$\begin{aligned} c_{\boldsymbol{u}} &= g^{\sum_{i=k+1}^{l} a_i u_i(x_1)} h^{s_u}, \quad c_{\boldsymbol{v}} = g^{\sum_{i=k+1}^{l} a_i v_i(x_1)} h^{s_v}, \\ c_{\boldsymbol{w}} &= g^{\sum_{i=k+1}^{l} a_i w_i(x_1)} h^{s_w}, \quad c_{hz} = g^{h(x_1)z(x_1)} h^{s_h} \end{aligned}$$

The knowledge extractor described in the proof of Protocol 3.1 can extract a, r_a and b, r_b such that

$$\begin{aligned} c_a &= c_u \cdot c_{\boldsymbol{u}} \cdot g^{\sum_{i=l+1}^{n} a_i u_i(x_1)} = g^a h^{r_a} \\ c_b &= c_v \cdot c_{\boldsymbol{v}} \cdot g^{\sum_{i=l+1}^{n} a_i v_i(x_1)} = g^b h^{r_b} \\ c_c &= c_w \cdot c_{\boldsymbol{w}} \cdot g^{\sum_{i=l+1}^{n} a_i w_i(x_1)} \cdot c_{hz} = g^{ab} h^{r_c} \end{aligned}$$

which means

$$\sum_{i=1}^{n} a_i u_i(x_1) \cdot \sum_{i=1}^{n} a_i v_i(x_1) = \sum_{i=1}^{n} a_i w_i(x_1) + h(x_1)z(x_1)$$

Apart from the challenge x_1, all the variables in the above equation are predefined, therefore either the prover can compute the non-trivial discrete logarithm relation between the generators or $\sum_{i=1}^{n} a_i u_i(X) \cdot \sum_{i=1}^{n} a_i v_i(X) = \sum_{i=1}^{n} a_i w_i(X) + h(X)z(X)$.
Perfect special honest-verifier zero-knowledge The zero-knowledge property follows by the zero-knowledge properties of the sub-protocols. The simulator can utilize the sub-protocols' simulator to produce a valid transcript without knowing the witnesses. □

F Definitions for Commit-and-Prove Zero-Knowledge Proof

Definition 7 (Perfect Completeness). *The triple $(\mathcal{G}, \mathcal{V}, \mathcal{P})$ has perfect completeness if for all non-uniform PPT adversary $\mathcal{A}$ such that*

$$\Pr\left[\begin{array}{l}(\sigma,c,r,x,u,w)\notin\mathcal{R}_\lambda^{\mathsf{Com}}\\ \textit{or}\ \langle\mathcal{P}(\sigma,c,r,x,u,w),\mathcal{V}(\sigma,c,x)\rangle=1\end{array}\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda)\\(c,r,x,u,w)\leftarrow\mathcal{A}(\sigma)\end{array}\right]=1$$

Definition 8 (Computational Soundness). *$(\mathcal{G},\mathcal{V},\mathcal{P})$ has computational soundness if it is not possible to prove a false statement where no witness exist, i.e. for all non-uniform polynomial time interactive adversary $\mathcal{A}_1,\mathcal{A}_2$, the function* $\mathsf{negl}(\lambda)$ *is negligible.*

$$\Pr\left[\begin{array}{l}\mathcal{A}_1(tr)=1\ (\textit{i.e. tr is accepting})\ \wedge\\(\sigma,c,r,x,u,w)\notin\mathcal{R}_\lambda^{\mathsf{Com}})\end{array}\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda)\\(c,x,s)\leftarrow\mathcal{A}_2(\sigma)\end{array}\right]\leq\mathsf{negl}(\lambda)$$

Definition 9 (Computational Knowledge Soundness). *$(\mathcal{G},\mathcal{V},\mathcal{P})$ has computational knowledge soundness if for all deterministic polynomial time $\mathcal{P}^*$, there exists an polynomial time knowledge extractor $\mathcal{E}$ such that for all non-uniform polynomial time interactive adversary $\mathcal{A}_1,\mathcal{A}_2$, the function* $\mathsf{negl}(\lambda)$ *is negligible.*

$$\left|\begin{array}{l}\Pr\left[\mathcal{A}_1(tr)=1\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda),(c,x,s)\leftarrow\mathcal{A}_2(\sigma)\\ tr\leftarrow\langle\mathcal{P}^*(\sigma,c,x,s),\mathcal{V}(\sigma,c,x)\rangle\end{array}\right]-\\ \\ \Pr\left[\begin{array}{l}\mathcal{A}_1(tr)=1\wedge\\(tr\ \textit{is accepting i.e.}\ (\sigma,c,r,x,u,w)\in\mathcal{R}_\lambda^{\mathsf{Com}})\end{array}\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda)\\(c,x,s)\leftarrow\mathcal{A}_2(\sigma)\\(tr,w)\leftarrow\mathcal{E}^{\mathcal{O}}(\sigma,c,x)\end{array}\right]\end{array}\right|\leq\mathsf{negl}(\lambda)$$

where the oracle is given by $\mathcal{O}=\langle\mathcal{P}^*(\sigma,c,x,s),\mathcal{V}(\sigma,c,x)\rangle$.

The oracle $\mathcal{O}$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. Informally, if there is an adversary that can produce an argument that satisfies the verifier with some probability, then there exists an emulator that can extract the witness. The value s is the internal state of $\mathcal{P}^*$, including randomness. The emulator is permitted to rewind the interaction between the prover and verifier to any move, then resuming with fresh randomness for the verifier.

Definition 10 (Perfect Special Honest-Verifier Zero-Knowledge). *A triple $(\mathcal{G},\mathcal{P},\mathcal{V})$ is a perfect special honest verifier zero knowledge argument of knowledge for $\mathcal{R}_\lambda^{\mathsf{Com}}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all pairs of interactive adversaries $\mathcal{A}_1,\mathcal{A}_2$*

$$\Pr\left[(\sigma,c,r,x,u,w)\in\mathcal{R}_\lambda^{\mathsf{Com}}\ \textit{and}\ \mathcal{A}_1(tr)=1\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda),(c,x,r,u,w,\rho)\leftarrow\mathcal{A}_2(\sigma)\\ tr\leftarrow\langle\mathcal{P}^*(\sigma,c,x,r,u,w),\mathcal{V}(\sigma,c,x;\rho)\rangle\end{array}\right]$$

$$=\Pr\left[(\sigma,c,r,x,u,w)\in\mathcal{R}_\lambda^{\mathsf{Com}}\ \textit{and}\ \mathcal{A}_1(tr)=1\middle|\begin{array}{l}\sigma\leftarrow\mathcal{G}(1^\lambda),(c,x,r,u,w,\rho)\leftarrow\mathcal{A}_2(\sigma)\\ tr\leftarrow\mathcal{S}(\sigma,c,x,\rho)\end{array}\right]$$

where ρ is the randomness used by the verifier.

Definition 11 (Commit-and-Prove Zero-knowledge Argument of Knowledge).

The triple $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ is a commit-and-prove zero-knowledge argument of knowledge for a family of relations $\mathcal{R}^{\mathsf{Com}}$ if it satisfies the perfect completeness, perfect special honest-verifier zero-knowledge and computational soundness or computational knowledge soundness.

G Notations

- Op : the operator
- SC : the smart contract
- r : the epoch number
- pk_i : the public key of user u_i
- b_i : the balance of user u_i
- c_i : the commitment for the balance of user u_i
- t_i : the randomness of user u_i
- c_{t_i} : the commitment for the randomness t_i
- v_{ij} : the value of transaction that is sent from u_i to u_j
- σ : a signature
- B : the total balance in the smart contract.
- Tx'_{ij}:$(pk_i, pk_j, v_{ij}, \mathsf{Null}, r, n, \sigma_{ij})$: The plain transaction sent by user to the operator.
- Tx_{ij}:$(pk_i, pk_j, v_{ij}, \mathsf{c_{ij}}, r, n, \sigma_{ij})$: The plain transaction after that the operator commits to the value and replace the Null with the commitment.
- CTx_{ij} : (pk_j, c_{ij}) The abbreviated confidential transaction that sends value from u_i to u_j
- $\mathcal{T}r$: the plain transaction list
- $\mathcal{CT}_r$: the abbreviated confidential transaction list
- H : a collision resistant hash function
- $\{x_i\}_{i=1}^{N}$: a set of values $\{x_1, ..., x_N\}$, we use curly brackets to indicates a set of values.

Paper G

Security Model for Privacy-preserving Blockchain-based Cryptocurrency Systems

K. Gjøsteen, M. Raikwar, S. Wu

Submitted to 13th Conference on Security and Cryptography for Networks (SCN), 2022

This paper is awaiting publication and is therefore not included.

Paper H

SoK: Decentralized Randomness Beacon Protocols

M. Raikwar, D. Gligoroski

Accepted in 27th Australasian Conference on Information Security and Privacy (ACISP), 2022
(To be presented during 28-30 Nov 2022)

Paper I

Competitive Decentralized Randomness Beacon Protocols

M. Raikwar

Published in ACM ASIACCS Workshop,
4th ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI), 2022

Competitive Decentralized Randomness Beacon Protocols

Mayank Raikwar
Norwegian University of Science and Technology
Trondheim, Norway
mayank.raikwar@ntnu.no

ABSTRACT

A distributed and reliable source of randomness is always a critical element in cryptography, both in the construction and application of cryptographic primitives. Modern cryptography such as blockchain, cryptocurrencies, decentralized finance is heavily dependent on a trusted randomness source. In particular, Decentralized Randomness Beacon (DRB) protocols can be a reliable source of randomness. A DRB protocol generates a continuous stream of publicly verifiable random values. Almost all the available DRB protocols are collaborative in nature where participants of a DRB protocol collaborate their local entropy to generate global randomness.

A DRB protocol can also be competitive where the participants compete to generate the global randomness. RANDCHAIN (ePrint 2020/1033) is the first Competitive Decentralized Randomness Beacon Protocol. Although RANDCHAIN claims to provide fairness and scalability, it still has a few problems concerning blockchain-oriented attacks and fairness to the participants. Therefore, to solve the existing problems of RANDCHAIN, we present a general model to construct competitive DRB protocols. Our competitive DRB model is a composition of committee selection strategy followed by a moderately hard cryptographic puzzle. To provide different levels of fairness, we present a variety of committee selection strategies. Our competitive DRB protocols provide better fairness, linear communication complexity, and better scalability compared to the existing DRB protocols.

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Theory of computation → Cryptographic protocols**.

KEYWORDS

Random Beacon; Bias-resistance; Unpredictability; Secret Sharing; Verifiable Delay Function

ACM Reference Format:
Mayank Raikwar. 2022. Competitive Decentralized Randomness Beacon Protocols. In *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI '22), May 30, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3494106.3528679

1 INTRODUCTION

Provably secure cryptographic primitives and protocols require a reliable source of randomness. Generating trustworthy randomness in a network of mutually distrusting participants was first introduced in a coin-tossing protocol presented by Blum [5]. Further, Rabin formalized the notion of random beacon protocol [38]. Since then, there have been many works addressing the construction of public randomness by using different well-established approaches. However, to make the public randomness source distributed and unbiasable (by a third party), the notion of a decentralized randomness beacon (DRB) was proposed. To date, most of the available constructions of beacon protocol are decentralized where trust is distributed among participants.

Furthermore, due to the prolific development in blockchain technology since the advent of bitcoin [34], there have been a plethora of blockchain consensus protocols and blockchain applications that heavily rely on publicly verifiable randomness. Therefore, many DRB protocols have been proposed in recent years.

Existing DRB protocols [4, 10–13, 15, 17, 18, 20–24, 26, 29, 30, 41–43] are *Collaborative* where participants work together to generate a global random output (beacon output) using their local entropy. These DRB protocols differ in their designs, underlying cryptographic primitives, and security properties. Current DRB protocols are based on Publicly Verifiable Secret Sharing (PVSS) schemes [4, 11, 12, 17, 29, 42, 43], Threshold Crypto-Systems [10, 15, 21, 26], Verifiable Random Functions (VRF) [18, 23, 24], or Verifiable Delay Functions (VDF) [20, 22, 25, 30, 41].

These DRB protocols have inherent limitations due to their design, initial setup, and applied cryptographic primitive. The main limitations are complexity, scalability and fairness; scalability refers to the number of participants a DRB can handle, and fairness refers to each participant having comparable power on deciding DRB output. A few recent protocols [17, 26, 41] try to address some of these limitations, but most of these protocols are based on time-sensitive cryptographic primitives e.g., Sloth [30], VDF [6].

DRB Protocols based on time-sensitive cryptography: Lenstra and Wesolowski [30] constructed a random beacon protocol, Unicorn, using a delay function. The main idea of the Unicorn protocol is to collect a pool of inputs from a set of distrusting participants and feed those inputs to a slow-time hash function named *sloth*. The output of the hash function is the random beacon value. As long as at least one participant provides a random input value, the beacon's output remains bias-resistant and unpredictable. However, the verification of the output of the delay function was not efficient. Keeping Unicorn protocol as a successor to VDF, a few VDF-based DRB protocols [20, 22, 41] are constructed.

The participants of a VDF-based DRB evaluate an Iteratively Sequential Function (ISF) to generate their local random values. The verification of these values can be efficiently done using the verification algorithm of VDF. Further, these local random values are used to generate the DRB output. The main idea behind using VDF is that an adversary cannot bias the output of the DRB due to the non-parallelizable property of VDF. However, hardware solutions may give some advantage in accelerating the VDF computation.

A smart contract-based DRB protocol [20] was constructed by Justin Drake that also leverages VDF to produce unbiasable random values. However, the presented construction lacks a formal security analysis. In accordance, Ephraim et al. [22] constructed a notion of Continuous Verifiable Delay Function (cVDF) that can be utilized to construct a random beacon protocol. Nevertheless, in the continuous VDF based DRB, a node equipped with a fast processor is always able to learn the beacon output before other participants hence breaking the unpredictability property of a beacon protocol.

A recent DRB construction based on VDF is RandRunner [41]. RandRunner employs trapdoor VDF with the property of strong uniqueness in its construction. The idea of RandRunner is fairly simple and the protocol works in rounds. During the bootstrapping (Initialization phase), each node P_i of RandRunner initializes its public parameters pp_i along with a secret key sk_i. All the nodes participating in RandRunner during initialization exchange their public parameter with each other and verify the received parameters. In each round, a node is selected as a leader and it tries to solve a VDF using its trapdoor information (secret key). Nonetheless, in the same round, the other nodes attempt to solve the VDF using the public parameter of the leader. The main drawback of RandRunner is its unpredictability guarantee of random beacon output. In RandRunner, in each round, a leader is elected using a leader election mechanism. Once an adversarial node with powerful hardware becomes a leader, it can withhold the beacon output and further hinder the unpredictability by corrupting the leaders in each round while working on the next round output.

Competitive DRB Protocol: The above described DRB protocols provide good scalability and fairness guarantee but still face problems regarding the beacon properties e.g., unpredictability. Therefore, to solve these limitations, Han et al. [25] introduced a new class of DRB, called *Competitive* DRB. In a Competitive DRB, participants compete to solve cryptographic puzzles, and the participant who solves the puzzle first becomes the leader and broadcasts its puzzle solution. The participants in the beacon network apply Nakamoto consensus [34] to agree upon the puzzle solution that is subsequently used to generate the beacon output. The competitive DRB protocols not only solve the main limitations of DRB protocols but are also easy to integrate with blockchain platforms. It enables the randomness generation as a side effect of the normal system operation in the blockchain. Han et al. [25] presented the first construction of competitive DRB called RANDCHAIN.

RANDCHAIN is built upon a cryptographic puzzle that takes an unpredictable and random number of steps to solve. RANDCHAIN is scalable and provides fairness to the participants. However, the fairness of RANDCHAIN can be weakened by selfish mining attacks e.g., a front-running attack. For which the authors cited some of the existing defense mechanisms against selfish mining to be deployed in RANDCHAIN. However, most of these mechanisms are hard to exploit in RANDCHAIN. We present a brief discussion about existing front-running attacks' defense mechanisms and why these mechanisms cannot be deployed in RANDCHAIN.

1.1 Problems in RANDCHAIN

RANDCHAIN is the only competitive DRB protocol as per our knowledge. It introduces Sequential Proof of Work (SeqPoW) which is a non-parallelizable, memory-hard cryptographic puzzle and takes an unpredictable and random number of sequential steps. RANDCHAIN presents a formal definition, followed by constructions of SeqPoW based on VDF and Sloth which uses public key pk for its algorithms, namely $\mathsf{Init}(), \mathsf{Solve}(), \mathsf{Prove}()$.

The idea of SeqPoW is fairly straightforward: In the $\mathsf{Init}()$ algorithm, an initial solution is computed as $S_0 \leftarrow H_G(pk||x)$ by a prover, where pk is the public key of the prover and x is an input. Given the initial solution S_0, the prover further keeps incrementing an ISF and produces a new output S_i in each increment (step i) using $\mathsf{Solve}()$ algorithm. In each step, the prover checks whether the output satisfies a certain threshold (difficulty) condition. If S_i satisfies the checks, then S_i is considered as a valid solution and the prover runs $\mathsf{Prove}()$ algorithm to generate a proof π_i for the correctness of the output S_i. A verifier verifies the solution S_i using the proof π_i and executing the algorithm $\mathsf{Verify}()$. To note, S_0 is computed in $\mathsf{Prove}(), \mathsf{Verify}()$ algorithms.

Furthermore, the authors also mentioned that SeqPoW can also be instantiated using secret key sk instead of public key pk in the above-mentioned algorithms except for the $\mathsf{Verify}()$ algorithm. Because a verifier should not know the secret key of a prover. The RANDCHAIN DRB construction proposed by the author uses the secret key for the $\mathsf{Init}(), \mathsf{Solve}(), \mathsf{Prove}()$ algorithms. With this regard, there are two shortcomings with the RANDCHAIN construction:

- $\mathsf{Init}(), \mathsf{Solve}(), \mathsf{Prove}()$ algorithms use sk and hence the initial solution is computed as $S_0 \leftarrow H_G(sk||x)$. This initial solution can easily be computed for $\mathsf{Prove}()$ algorithm as a part of the construction to compute the proof. However, for the verification algorithm $\mathsf{Verify}()$, the authors did not justify how a verifier will construct the initial solution S_0 without having knowledge about the secret key sk of the prover.
- Another point about using secret key sk for the computation of S_0 is how the prover will prove that the initial solution has been computed correctly without revealing his secret key sk. A zero-knowledge proof can be a useful tool to deal with this issue but the authors have not mentioned this issue and the corresponding solution. This is crucial for the correctness of S_0.
- Consider a case where the prover's public key pk is used to construct the initial solution and also for all the other algorithms. In this scenario, an adversary can easily compute the initial solutions for every node in the DRB. After computing the initial solutions, a powerful adversary can run the $Solve()$ algorithm in parallel for each honest node in the DRB and can easily get to know which node is going to solve the puzzle first. Here, the adversary breaks the unpredictability guarantee of the DRB protocol. In this case, an adversary can either make the targeted attack towards the winning node or can mount a bribery attack (ask for a bribe) on the winning node. The adversary can keep corrupting the expected winning nodes in DRB protocol and can further lead to the unavailability of the system.

In conclusion, RANDCHAIN suffers from mainly three issues, front-running attack, breaking the unpredictability guarantee, and not providing enough fairness to all the DRB participants.

Basic Idea of our competitive DRB protocol

To solve the challenges of RANDCHAIN and to provide better fairness and mitigation against blockchain-related attacks, we present a general model to construct competitive DRB protocols. The general model is a composition of committee selection strategy followed by a moderately hard cryptographic puzzle. Depending upon the different committee-selection strategies and different cryptographic puzzles, a class of competitive DRB protocols can be constructed. The committee selection is an integral part of the construction because it confers fairness in the DRB protocol and also makes blockchain attacks less probable.

The basic idea of a competitive DRB protocol is as follows: the protocol operates in a permissioned setting similar to the existing DRB protocols and works in rounds. Each participant[1] of DRB maintains its local chain and syncs its chain with the main chain in each round. In each round of the DRB protocol, a committee is selected from the participants of DRB. Further, the members of the committee compete to solve a moderately hard cryptographic puzzle. The one who solves the puzzle first becomes the leader of that round. The leader creates a new block, appends the solution of the puzzle with corresponding proofs in the block, and further broadcasts the block in the DRB network. The other participants verify the received block by checking the solution using the proof. If the verification is successful, the participants add that block to their local chains. The current random value is computed using the solution of the puzzle appended in the block.

1.2 Our Contribution

Motivated by the challenges in RANDCHAIN DRB, the contributions of the paper are as follows.

- We propose a model to construct a class of competitive DRB protocols using a committee selection strategy followed by a moderately hard cryptographic puzzle.
- We present the advantages of competitive DRB over the current DRB protocols.
- We discuss the security properties of our competitive DRB and present the proofs for the properties.

1.3 Outline

The remainder of the paper is as follows: In Section 2, we describe the preliminaries needed to construct competitive DRB protocols. In Section 3, we define the model to construct competitive DRB protocols. In Section 4, we give a brief description of the main challenges faced in present DRB protocols and how a competitive DRB overcomes some of these challenges. Further, the properties of our competitive DRB are presented in Section 5. Finally, in Section 6, we conclude the paper and provide a few future research directions.

2 PRELIMINARIES

2.1 Verifiable Delay Function

Verifiable delay function (VDF) is a cryptographic primitive proposed in 2018 by Boneh et al. [6]. A function $f : \mathcal{X} \to \mathcal{Y}$ is a VDF if given an input $x \in \mathcal{X}$, the computation of output $y \in \mathcal{Y}$ takes predefined number of steps T; additionally the verification of y is exponentially easy and efficient. Furthermore, the computation of y cannot be parallelized even if a polynomial number of processors are available.

Definition 2.1. (VDF): *A VDF is defined as a tuple of following algorithms:*

- $\mathsf{Setup}(\lambda, T)$: It is a randomized algorithm that takes security parameter λ, time parameter T and outputs public parameter pp.
- $\mathsf{Eval}(pp, x, T)$: The evaluation algorithm takes public parameter pp, input value $x \in \mathcal{X}$ and time parameter T, returns an output value $y \in \mathcal{Y}$ together with a proof π. The algorithm may use random coins to generate the proof π but not for the computation of output y.
- $\mathsf{Verify}(pp, x, y, \pi, T)$: The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

There have been two main constructions of VDF based on the repeated squaring (modular exponentiation). These two main proposals are Wesolowski Scheme [44] and Pietrzak Scheme [37]. In both schemes, the number of VDF evaluation cycle is already known since the initialization of VDF. Both schemes evaluate an output value $y \leftarrow H(x)^{(2^T)} \bmod N$, along with a proof π, given an input value x. Here $H : \mathcal{X} \to \mathbb{G}$ is an efficiently computable hash function, T is the number of squarings needed to compute the output, and N is an RSA modulus. More information about these VDF construction can be found in Appedix A.

2.2 Verifiable Random Function

Definition 2.2. (VRF): *A VRF is defined as a tuple of following algorithms:*

- $\mathsf{KeyGen}(r)$: On input value r, the algorithm generates a secret key sk and a verification key vk.
- $\mathsf{Eval}(sk, M)$: Evaluation algorithm produces pseudorandom output o and the corresponding proof π on input sk and a message M.
- $\mathsf{Verify}(vk, M, o, \pi)$: Verify algorithm outputs 1 if and only if the output produced by evaluation algorithm is o and it is verified by the proof π given the verification key vk and the message M.

More information about VRF is presented in Appendix B.

2.3 Decentralized Randomness Beacon

A decentralized randomness beacon (DRB) allows a group of participants to continuously produce random values without having any central party. A DRB requires following properties [15, 42]:

- *Unpredictability:* An adversary's ability to predict (precompute) beacon outcome is negligible.
- *Bias-resistance:* Any single participant or colluding participants should not be able to influence the beacon outcomes to their advantage.
- *Availability (or liveness):* Any single participant or colluding participants should not be able to prevent the progress of the DRB protocol.
- *Public Verifiability:* Third parties should always be able to verify the correctness of the beacon outcome using publicly available information.

[1]*Note*: Throughout the paper, we use node and participant interchangeably.

3 COMPETITIVE DRB

In a competitive DRB, instead of all the participants competing in each round, we can select a committee in each round. The selection of the committee should provide better fairness to all the participants so that all the participants can get an equal chance to be elected as a leader. The motive of selecting a committee is that a powerful adversary should not be able to take part in the committee of each round and so the adversary will not be able to continuously build his advantage to be the leader of future rounds. Therefore, following we define a new definition of a fair competitive DRB.

A fair competitive DRB can be defined as a composition of committee selection strategy followed by a process of solving a cryptographic puzzle. Different committee selection strategies provide different levels of fairness for competitive DRB protocols. Therefore, in the following Section 3.2 we present a few methods of committee selection.

3.1 DRB Model

- *System Model*: A competitive DRB consists of N participants $\mathcal{P} = (P_1, P_2, \ldots, P_N)$. Each participant P_i has a key-pair (pk_i, sk_i) and the node can be identified by its public key pk_i. These participants are connected in a distributed manner and for every round $r \in \{1, 2, \ldots\}$, the protocol $\text{DRB}(I_r)$ receives a generically denoted input I_r and further collectively produces a random output O_r.
- *Network Model*: Our DRB protocol works in a synchronous network where messages between participants are delivered within a bounded network delay δ. The DRB protocol works in rounds, and the synchronous network assumption provides guaranteed message delivery.

3.2 Committee Selection Strategies

In a competitive DRB protocol, in each round, a set of participants are selected as committee members using a predetermined committee selection technique. These committee members further advance the DRB protocol to the next step where these committee members compete to solve a cryptographic puzzle and henceforth, propose the next beacon output.

Definition 3.1. (Committee Selection) Given a set of participants $\mathcal{P} = (P_1, P_2, \ldots, P_N)$, for a round r a committee selection strategy CS generates a committee set $C = (C_1, C_2, \ldots, C_k)$, where $C \subseteq \mathcal{P}$, provided *aux* as an auxiliary information. *aux* can be a previous round output O_{r-1} or a seed s_r computed from the previous round.

There are many constructions of committee selection in the literature due to mainly its applicability in consensus protocols. In most of the committee-based consensus protocols, the committee is selected to execute Byzantine fault-tolerant (BFT) distributed consensus to decide on the next block that will be added to the blockchain. Most of the committee selection strategies show strong statistical similarities [16]. Moreover, as committee members are responsible to generate the next block (or next random output in our DRB), in order to enforce the honest behavior, the committee members can be incentivized using a fair rewarding mechanism [2].

A committee can also be selected as proposed by benhamouda et al. [3]. In their construction, a selected committee has an honest majority and committee members remain anonymous until the start of the next round where a new committee is selected. Similarly, a committee can be selected for our DRB protocol where an adversary can not be able to guess the current round committee members and due to the honest majority of committee members, the next random output can be generated fairly. A committee with a predefined size should be selected in a distributed verifiable manner (Using verifiable lotteries). The committee size depends on the total number of participants in the DRB and it can be updated based on the total number of existing and new participants from time to time.

For this work, we describe a few committee selection strategies that can be used for the construction of competitive DRB protocols. Each of these strategies brings a different level of fairness for the participants of the DRB protocol. Nevertheless, perfect and universal committee selection strategies are hard to find, for which more research is required.

- *Robust Round Robin* : In this method, each DRB participant maintains a round-robin queue of participants' identities. Identity of a participant P_i corresponds to his public key pk_i. This queue is maintained and sorted in decreasing order based on the age of the participants' identities. Once a committee of size k is selected from the queue for a round r, those k committee members are all placed at the end of the queue for round $r+1$. In this manner, each participant gets a fair and equal chance of being a committee member. In case, a new participant joins the DRB protocol, the identity of the new participant is placed at the end of the queue.
- *Randomized Round Robin* : A randomized round robin method can be employed to select a committee. In this method, an initial seed s_0 is used to deterministically derive a random sequence $\mathcal{P}^1$ of DRB participants $\mathcal{P}$ for round 1. The seed s_r is used to shuffle the DRB participant set $\mathcal{P}$ and hence to obtain a randomized set $\mathcal{P}^r$. The initial seed is computed using distributed random number generation [9].
  The seed is refreshed in every round and a new random sequence is generated for the round. A new seed for round r is computed as $s_r = H_s(s_{r-1}||r||O_{r-1})$, where $H_s : \{0,1\}^* \rightarrow \{0,1\}^{256}$. A committee member C_i from a committee of k members is computed as follows from the random sequence $\mathcal{P}^r$ for a round r:

$$C_i = \mathcal{P}^r[r \ mod \ N] + i \tag{1}$$

- *Uniformly Random Sampling* : In this method, a committee can be uniformly sampled using the round output of the last round. By interpreting the output as a 256-bit number, a committee of size k can be sampled using a simple approach. A committee member C_i from a committee of k members is computed as follows, for round r:

$$C_i = (O_{r-1} \ mod \ N) + i \tag{2}$$

- *Cryptographic Sortition* : A committee can be selected using cryptographic sortition as defined in Algorand [24]. In this method, all the DRB participants individually apply VRF on a common input using their secret keys and if the VRF output computed by a participant is less than a certain threshold then that participant is selected as a committee member. The threshold can be set or modified to control the number of committee members. Further, these committee members execute the cryptographic puzzle to be the leader of the round and to propose beacon output.

In Algorand, if a committee is of size C, then an adversary only needs to corrupt $\frac{C}{3N}$ out of N nodes in the Algorand blockchain. Due to the local predictability of VRF output, only the participants can know becoming a committee member or leader in Algorand. An adversary can perform a bribery attack by advertising it in a black market and hence adversary can abrupt the Algorand consensus. Nevertheless, in our DRB, the output of DRB in a round does not depend on every member of the committee, as the output is generated by a participant who solves the cryptographic puzzle first. Therefore, an adversary cannot decide which or how many committee participants need to be corrupted.

3.3 Cryptographic Puzzle

For a competitive DRB protocol, a cryptographic puzzle should be provided to each selected committee participant to compete in each round. This cryptographic puzzle should be moderately hard and non-parallelizable. The reason for making the puzzle non-parallelizable is that a powerful adversary with more processors should not have an advantage in solving a puzzle. Therefore, we use VDF to make it non-parallelizable. Sloth can also be used for the puzzle construction instead of VDF (similar to RANDCHAIN), but the verification cost in sloth is linear which makes it less favorable for the puzzle construction.

As mentioned above, Pietrzak [37] and Wesolowski [44] schemes for VDF are based on repeated squaring. Nevertheless, both schemes require the number of VDF evaluations to be known during the initialization of VDF. Moreover, even though both schemes have fast verification, these scheme needs to store all the intermediate proofs to verify the final output value. Wesolowski scheme solves this issue by aggregating the intermediate proofs to a single proof but the verification time still grows with the number of iterations. Therefore, we adopt Continuous Verifiable Delay Function (cVDF) [22] where verification time does not depend on the number of iterations. cVDF was introduced by Ephraim et al. which is based on the construction of Pietrzak scheme. A VDF f is a continuous VDF (cVDF) if it provides the computation of function f on intermediate steps (i.e. $f(t)$ for $t < T$) along with an efficient proof π^t, in addition, the intermediate outputs of cVDF are public and continuous verifiable.

Following, we provide a formal definition of non-parallelizable, publicly verifiable, moderately hard puzzle for a participant P.

Definition 3.2. (Non-parallelizable Cryptographic Puzzle) It consists of following polynomial time algorithms :

- $\mathsf{Init}(R_{r-1}, pk_P, r)$: Given input as previous puzzle output R_{r-1}, the participant's public key pk_P, and current round r, it outputs a verifiable input x for the puzzle. An extra information e_1 might also be output needed for public verifiability of x. The initialization of input x can also be done using secret key sk_P of the participant instead of pk_P.
- $\mathsf{Solve}(x, \tau, r)$: Given input as initial puzzle input x, threshold τ, and the current round r, it outputs a solution to the puzzle R_r, a proof π^r for the correct computation and an extra information e_2 might be needed for solution verification satisfying the threshold.
- $\mathsf{Verify}(x, r, R_r, \pi^r, pk_P, \tau, e_1, e_2)$: Given inputs as $x, r, R_r, \pi^r, pk_P, \tau, e_1, e_2$, it outputs 1 if R_r is a valid solution for round r computed by participant P; otherwise it outputs 0.

Cryptographic Puzzle based on VDF:

A VDF-based cryptographic puzzle CP is constructed using the repeated squaring functionality of VDF as defined in [19, 25, 31, 39]. Following, we present the definition of VDF-based puzzle

Definition 3.3. (VDF-based Puzzle): For a participant P in round r, given an input x, a function f, a threshold τ, a VDF-based puzzle CP has a solution $\mathrm{S}_r = (t, R_r, \pi^r)$ such that

$$R_r = H(x)^{2^t} \tag{3}$$

$$f(R_r) \leq \tau \tag{4}$$

In the above puzzle, solution $\mathrm{S}_r = (t, R_r, \pi^r)$ is an intermediate output (on step t) of cVDF on input x using random oracle H. In the puzzle, the main output R_r is computed using equation 3, which satisfies the equation 4. The proof of correct computation π^r for output R_r is provided using cVDF. In equation 4, function f can be a one-way hash function or it can even be a VRF. The threshold τ depends on the function and it can be fixed for each round or it can also vary from participant to participant. Function f is defined considering the way of input computation.

Input Computation:

In the direction of constructing cryptographic puzzle CP based on VDF, we want the puzzle CP to be different for each participant and also have the same difficulty level. Therefore, in each round, input for each participant's puzzle should be different and should be publicly verifiable. On the contrary, initial input in RANDCHAIN is computed using the participant's secret key which can not be verified by other participants. We want to avoid this situation, therefore, following we present two ways of input computation.

(1) An input x for participant P in a round r can be computed by applying VRF on previous round puzzle output R_{r-1} (last winner solution). Furthermore, VRF output $x \in \mathcal{X}$.

$$(x, \pi^x) \leftarrow VRF_{sk_P}(R_{r-1}) \tag{5}$$

From the above method, a verifiable input x is computed. Furthermore, an adversary can not compute the input of the participant P without having knowledge of participant's secret key sk_P. With this context, function f can be replaced by a hash function $H_I : \{0,1\}^* \rightarrow \mathcal{X}$ in equation 4, hence the new condition would be:

$$H_I(pk_P || R_r) \leq \omega.M \tag{6}$$

In equation 6, ω is a parameter controlling the difficulty of the puzzle and M is the maximum value of hash function H_I. In this case of input computation, the extra information needed in the puzzle definition 3.2 e_1, e_2 are π^x and ϕ respectively.

(2) An input x for participant P in a round r can be computed by applying hash function as follows:

$$x \leftarrow H_I(pk_P || R_{r-1}) \tag{7}$$

Here, input x can be verified by again performing the hash on public key pk_P and previous round puzzle output R_{r-1}. Nevertheless, a powerful adversary can easily compute the puzzle input for every participant as everything needed to compute x is public. We don't want a powerful adversary to compute the puzzle solution S for the honest participants beforehand and then corrupt/targeted attack the honest participants. Furthermore, as computation of output R_r is performed using continuous VDF (as in equation 3), we don't want delegation of any participant's puzzle to the powerful adversary. In order to prevent corruption and delegation, we want function f to be solely computed by every participant. Therefore, we consider function f as a VRF because it can only be computed using the secret key of the participant. Henceforth, equation 4 is replaced by the following equation.

$$VRF_{sk_P}(R_r) \leq \tau \tag{8}$$

The above equation checks whether the output of the VRF is $\leq$ threshold τ. This τ can be a difficulty parameter and it can be updated from time to time (similar to bitcoin difficulty). VRF also produces the proof which is used to prove the correct generation of VRF output. In this case of input computation, the extra information needed in puzzle definition 3.2 e_1, e_2 are ϕ and (o, π^o) respectively; where o is VRF output on R_r and π^o is the corresponding proof about correct computation of VRF.

In our DRB construction, the above puzzle can be replaced by any cryptographic puzzle that requires an unknown number of sequential steps and is moderately hard in nature. Therefore, an interesting research direction would be to construct new non-parallelizable, moderately hard cryptographic puzzles.

3.4 Structure of Competitive DRB

Following the Nakamoto consensus, each participant P_i maintains his local chain in a form of a directed acyclic graph of blocks. With each round, a new block is added to the main chain $\mathcal{B}$ following the Nakamoto consensus and longest chain rule in the case of forks. In each round, a participant becomes the leader of the round by being a committee member first, followed by solving a puzzle first. Further, the leader P_l creates a block B_r for round r. In case, we follow the first input computation method (Following equation 5, 6) for the DRB protocol, the header h_r of the block B_r contains $(pk_l, S_r, x, \pi^x, O_r, \sigma_{sk_l}(h_r))$, where pk_l is the public key of the leader node P_l; $S_r = (t, R_r, \pi^r)$ is the solution for round r; x, π^x corresponds to the input and proof for input computation in round r for the leader P_l; O_r is the output of the random beacon which can be computed as $O_r = H_{out}(R_r)$ where $H_{out} : \mathcal{Y} \rightarrow \{0,1\}^{256}$; and $\sigma_{sk_l}(h_r)$ is a signature on the block header.

A signature $\sigma_{sk_l}(h_r)$ is required to avoid the fork by the leader node P_l in the blockchain. The cryptographic puzzle used in our DRB protocol can have multiple valid solutions due to not having uniqueness in solutions (as in eq. 4). Therefore, an adversarial leader $\mathcal{A}$ can create forks by solving the same puzzle and providing different valid solutions to the puzzle. If an adversarial leader tries to fork the chain, other nodes can easily confirm it by checking that the new block in the fork chain and in the main chain have the same signature and share the same parent block. To avoid the fork, a *slashing rule* can be applied in the protocol where an adversarial leader trying to fork the chain will be punished. The punishment can be a direct exit from the protocol or slashing of stakes deposited in the case of stake-based blockchain.

Each participant P_j receiving the block B_r performs several checks before accepting and adding the block to his local chain. First P_j verifies the input x using the proof π^x, if it verifies, P_j verifies the solution S_r. To verify the solution S_r, P_j checks whether the equation 6 satisfies or not. Further, P_j verifies the proof π^r for t VDF-evaluations on input x.

Beacon Output:
The output of our DRB beacon O_r can be computed by applying a deterministic function to the puzzle solution R_r for round r. In our case, we apply hash function H_{out} to the puzzle solution R_r to compute the output. However, due to the non-uniqueness property of the cryptographic puzzle, an adversarial leader A can fork the chain by biasing the output. Nevertheless, a slashing rule can be applied in the event of a fork as mentioned previously but an adversary can still try to gain financial advantage by manipulating the output. If the puzzle outputs a unique random output in each round, then our DRB protocol is considered as strong bias-resistance.

To provide strong-bias resistance property, we can make an adversary not able to manipulate the beacon output instantly during the end of the round. In simple words, an adversary (or any node) has to wait for a few rounds to compute the beacon output. In that sense, a VDF can be applied to the puzzle output R_r such that the output O_r can not be extracted instantly. Therefore, before learning the output O_r, an adversary has to decide on whether to publish the block (makes the block irreversible) or withhold the block (makes it as an invalid block from round $r + 1$).

4 MAIN CHALLENGES IN CURRENT DRB PROTOCOLS

There has been a growing interest in the construction of new DRB protocols. However, many of these constructions still face different challenges due to their design and underlying cryptographic primitive. In this section, we give a brief description of some of these challenges. This section provides a complete overview of the limitations in existing DRB protocols and it also describes the advantage of competitive DRB protocol to solve some of these challenges.

4.1 Attacks

- *Front-Running Attack* In this attack, an adversary learns the beacon output earlier than the honest participants. This attack is more probable in non-interactive DRB protocols where participants do not interact with each other to generate the beacon output in each round. In some of these DRBs, each participant can generate the global beacon output from its local entropy, and further, it can withhold the output and let the other honest participants invest their energy to find another output. Therefore, these attacks are hard to capture as it is hard to differentiate whether a message (e.g., beacon output) is withheld by the adversary or delayed by the network.
  In the case of cryptocurrencies (blockchain), this attack has been an everlasting problem and a few defense mechanisms [27, 35, 45] against front-running attacks (selfish mining) have been proposed in the literature. The motive of an attacker in the case of cryptocurrencies is to gain more reward by mounting an attack which also puts a risk on the attacker's withholding reward. However, in the case of DRB, the motive of an attacker is to disrupt the unpredictability and availability property of the DRB protocol.

As in the DRB case, the attacker has no risk of losing anything, the attacker can always try to mount this attack.
Heilman [27] presented a strategy to combat this attack by using a freshness preferred approach. In this approach, the block with a recent valid timestamp is chosen by a miner (participant) when the miner receives two blocks from the branches of equal length. However, the approach fails to defend against a powerful attacker whose selfish chain is longer than the public chain. Zhang et al. [45] improves Hellman approach and proposes backward compatible defense against selfish mining by revising the fork-resolving technique.
These mechanisms are hard to deploy in DRB protocols directly as there is no award affiliated with the participants, which goes for RANDCHAIN as well. Nevertheless, the freshness preferred approach can be introduced by introducing a timestamp parameter in competitive DRB protocols. A slot parameter defined in PoSAT [19] used in a necessary check for leader election. This slot parameter is analogous to time-ordering. In the same way, it can also be used in RANDCHAIN or in our cryptographic puzzle inside DRB protocol. Nevertheless, the committee selection and the cryptographic puzzle in our DRB construction already make the front-running attack less probable.

- *Nothing at Stake Attack* This is a general problem in Proof-of-Stake (PoS) consensus. In this attack, a rational stakeholder publishes many blocks in different forks in the blockchain. The reason for extending all forks is that there is no opportunity cost so the rational stakeholder can extend all the branches in order to maximize his reward.
  A similar situation might arise in the case of DRB protocols that use blockchain as a public bulletin board for publishing the beacon output where an attacker can always try to create many forks. However, DRB protocols based on slow-time functions, especially the protocols not knowing the number of sequential steps for computation in advance, are less probable to have nothing at stake attack. Hence, our competitive DRB protocol has fewer chances of having nothing at stake attack.
- *Bribery Attack* Some DRB protocols are leader-based where a leader is elected using a leader-election mechanism and the leader is solely responsible to compute the beacon output. In these DRB protocols, a bribery attack is possible where an adversary can advertise a reward in a black market or bribe a leader to get to know the beacon output beforehand; hence tampering with the unpredictability guarantee of beacon output. The motive of an attacker can not only be just to tamper with the beacon properties but also to use the beacon output to his advantage by using it in an application such as a lottery protocol.
  For example, in RandRunner [41] DRB, all the participants know the identities of all the upcoming leaders if a randomized round-robin strategy is used for leader election. Hence, an adversary can bribe one of the upcoming leaders and can get to know the beacon output before anyone else knows it. This opens the possibility of different strategic attacks on the DRB. As knowing the output first gives an advantage to the adversary, the adversary can try to compute the next beacon output through mounting a front-running attack to the DRB protocol. In our competitive DRB protocol, committee selection instead of a leader-election mechanism makes this attack harder to be mounted.
- *Network-related Attacks* During network-related attacks, the properties of a DRB protocol can be obstructed. For example, during the network outage (a rare event), an adversary can hamper the availability and bias-resistance of the beacon output. Moreover, an external entity can also eclipse some participants from the beacon (e.g., by a DoS attack) or even the whole beacon from the participants. This attack is challenging to realize.
  In the case of competitive DRB, each participant maintains his local chain and tries to sync his local chain with the main chain through all his connected peers (neighboring participants). Furthermore, in our competitive DRB, as a committee is responsible to generate the beacon output in each round, the network-related attacks might not disrupt the beacon properties. Some countermeasures [28, 33] can also be used to protect against these attacks.

4.2 Complexity

Complexity in a DRB protocol is inferred from the computation of beacon output, further dissemination of the output, and then verification of the output. Therefore, protocols can be computation-intensive or communication expensive. The major challenge while designing a DRB protocol is to reduce its computation and communication effort. Therefore, many recent constructions focus on reducing the complexity of their protocol. The computation complexity is measured as the total amount of computation required to generate a beacon output. However, the communication complexity is the total amount of communication required to complete the DRB protocol. These complexities are mainly dependent on the underlying network structure, underlying cryptographic primitive, involved interaction among participants, and output dissemination method.

Based on the interaction, DRB protocols can also be categorized into interactive and non-interactive protocols as defined in [23]. Interactive DRB protocols involve multiple rounds of communication to generate a beacon output. These protocols are based on Publicly Verifiable Secret Sharing schemes (PVSS) [4, 11, 13, 17, 29, 42, 43] and hence require high communication (at least $O(n^2)$). A recent DRB protocol, ALBATROSS [12] reduces the communication cost to $O(n)$ by using packed Shamir secret sharing scheme.

A DRB protocol incurs three major costs (complexity): computation cost for beacon output, the communication cost of output generation, and dissemination and verification cost of the beacon output. Following, we describe all these costs and existing approaches used in DRB protocols to reduce these costs. In addition, we also discuss the complexity of our competitive DRB protocol.

- *Communication Complexity* Communication cost highly depends on the network model (Synchronous V/s Asynchronous) and message dissemination approach. It can also incur during the initial setup of the DRB protocols. Many DRB protocols [10, 15, 21, 26] require a Distributed Key Generation (DKG) setup that incurs high communication cost.
  The initial interactive DRB protocols e.g., Ouroboros [29], RandShare [43], and SCRAPE [11] require a broadcast channel to broadcast DRB messages (e.g., PVSS share, beacon output); henceforth, suffer from a high communication complexity of $O(n^3)$. Further, to improve the communication cost, committee-based or leader-based strategies were employed where respectively a

committee is responsible for output computation [26, 43] or a leader is responsible to performs the share distribution [12, 23]. The non-interactive DRB protocols [8, 18, 24, 41] incur less communication cost, usually $O(n)$, as in these protocols a single participant has to perform only one broadcast to disseminate the beacon output among the protocol participants. Furthermore, a few DRB constructions use blockchain as a public bulletin board to publish the beacon output which can result in very less communication complexity but threatens the finality of the random beacon output. As our competitive DRB protocol is also a non-interactive DRB protocol, hence it incurs linear communication complexity.

- *Computation Complexity* Similar to communication complexity, the initial interactive DRB protocols e.g., Ouroboros [29], RandShare [43] suffer from high computational cost of $O(n^3)$. This cost in interactive DRB was reduced in SCRAPE [11] to $O(n^2)$ and further reduced to $O(\log n)$ in ALBATROSS [12]. Nevertheless, DRB protocols based on VRF have very less computational complexity.
  Some of the DRB protocols (non-interactive DRB) are puzzle-based where participants are required to solve puzzles in each round. These DRB protocols incur high computation complexity for their underlying puzzle design. These puzzles can be based on Proof-of-Work [7] or Proof-of-Delay [8]. DRB protocols based on slow-time functions also face the same problem of high computational cost. Therefore, the puzzle should be designed in a way that is moderately hard and requires less computation.
  The computation complexity remains arguable for puzzle-based DRB protocols compared to protocols based on PVSS or threshold cryptographic primitives. The question arises as to whether the computation required to generate a DRB output using a threshold cryptographic primitive is less computation-intensive than computation needed in a puzzle-based DRB. In our competitive DRB protocol, computational complexity depends not only on the hardness of the VDF-based cryptographic puzzle but also on the size of the committee in each round.
- *Verification Complexity* Verification cost is the total number of operations performed by an external participant to verify the beacon output. Verification cost in almost all of the interactive DRB protocols is $O(n)$ due to the share verification. Nonetheless, VRF-based DRB protocols incur less verification cost similar to less computation cost. Furthermore, puzzle-based DRB protocols involving VDF as their underlying cryptographic primitive also outputs less verification cost.
  Ideally, the verification cost in a DRB protocol should be constant so that any external participant can verify the DRB output in constant time which is more preferable in real-time applications of DRB protocols such as in e-voting protocols [1], or in online gaming and lottery services [7].
  Our competitive DRB protocol is based on VDF, therefore, its verification complexity is equal to the cost needed to verify the output of the VDF puzzle. The output of the VDF puzzle includes proof of correctness for the puzzle output generation which is required to verify the puzzle output. Generally, VDF has fast verification, hence, our competitive DRB protocol has constant verification cost.

4.3 Scalability

Scalability implies that even having a large number of participants in DRB protocol, the protocol should advance as usual and produce beacon output regularly. To scale a DRB protocol, the communication complexity (should be linear) for output generation and the network latency should be minimized.

Scalability in DRB protocols is still an everlasting problem despite a decade of research that has been conducted to design new efficient DRB protocols. A few recent DRB protocols [17, 25, 41] have taken scalability into account while constructing their DRB. DRB protocols having DKG in their setup offer poor scalability. As DKG requires a fixed number of participants (key holders) and inclusion of new participants will trigger the setup modification which will incur huge communication costs and does not seem realistic to scale.

Scalability is directly related to communication complexity as mentioned above. Therefore, the same approaches described above to improve communication complexity can also be applied to achieve better scalability. Therefore, committee-based approach [24, 26, 43] (sharding) or leader-based approach can be leveraged to improve the scalability of DRB protocols. In general, sharding is a very efficient approach to scale a blockchain where participating nodes are grouped into smaller size groups.

Another important factor directly impacting the scalability of DRB protocol is *Reconfiguration Friendliness*. A DRB protocol is said to be reconfiguration-friendly if the public parameters and the list of participating nodes are allowed to be modified dynamically without causing any disturbance in the current protocol execution. Therefore, non-interactive DRB protocols and competitive DRB protocols show better scalability as new participants are allowed to join at any time.

4.4 Fairness

Fairness in a DRB protocol refers to that each participant gets a fair and equal chance of computing the beacon output. In the case of PVSS-based collaborative DRB protocols, all the participants contribute their local entropy to compute the global entropy. However, in puzzle-based DRB protocols, an adversary having extensive computational resources have an advantage in solving the puzzle, hence generating the beacon output.

The definition of fairness in RANDCHAIN is about comparable voting power for output computation regardless of participants' hardware resources. In other words, it is the maximum voting power difference between an honest participant and a powerful adversary. Achieving fairness in a real-world scenario where we can not bound the participants to the actual available hardware resources is hard. In RANDCHAIN, fairness is $> \frac{1}{5}$ due to powerful VDF hardware which computes 5 times faster than normal hardware.

Fairness in a competitive DRB protocols can be improved by giving equal chance for randomness computation for a set of participants (committee) in each round so that a powerful adversary can not participate in each round and hence can not have a withholding advantage (by becoming a leader in a round and withholding the output) for many consecutive rounds. In that way, better fairness can be provided in a competitive DRB protocol which is embraced in our DRB construction.

4.5 Problems with Competitive DRB

There can be following problems or possible attacks due to the chosen committee selection strategy. We described these problems in detail and propose some solutions to deal with the problems.

(1) *Block Withholding* An adversary being a committee member in a round can solve the puzzle first and can try to withhold the block (puzzle solution) in order to increase his advantage for the subsequent rounds. To mount the withholding (front-running) attack, the adversary has to be in the committee for each subsequent round and has to be able to solve the cryptographic puzzle first among all the committee members. Due to the selection of committee through a fair committee selection strategy and having a different cryptographic puzzle than other committee members in each round (requiring a different number of unknown steps to solve the puzzle), the chances of an adversary to mount the withholding attack are less likely.
(2) *Local Predictability* Depending on the committee selection strategy, an adversary might be able to know the committee members for the upcoming rounds of DRB protocol. The adversary can advertise for the winning leader in some round on a black market. In that scenario, the adversary would be able to know the beacon output through the bribed leader participant in some round before the other honest participants get to know about the output. The adversary can use this as an advantage to either plan for a front-running attack or break the unpredictability guarantee of the DRB protocol.
(3) *Finality* Finality refers to the participants of DRB protocol should have a consistent view on random beacon output. Due to following Nakamoto consensus, finality is a probabilistic event as the fork might arise in some cases. As finality is essential in some randomness-based applications, there should be some ways to achieve finality in DRB protocols.
To achieve finality in competitive DRB protocols, a few mechanisms can be used. Mainly, herding [14] or quorum mechanism [32] can be particularly helpful in achieving finality for our random beacon output. These mechanisms might increase the time complexity of the beacon output. Therefore, it would be interesting to integrate these mechanisms with the protocol or to look for other mechanisms that can help to achieve finality.
(4) *Sybil Attack* An adversary can create multiple identities (public keys) and form a majority in the DRB network. In Proof-of-Work (PoW) and Proof-of-Stake (PoS) consensus forming a majority requires having huge computational power (mining power) or a majority of stake (voting power). In our DRB protocol, an adversary should not be able to create multiple identities, otherwise, an adversary can be a part of every committee selection and might increase his advantage in each round of DRB protocol.
To prevent Sybil attack, either a staking mechanism can be involved which will require a fixed minimum amount of stake from each participant to be put on the blockchain, or binding of the public key of a participant with the participant's unique identifier such as a passport is to be incorporated. Moreover, each participant should provide a Proof of Possession (PoP) of a secret key corresponding to his public key, during the bootstrapping of DRB protocol or during the joining in the DRB protocol.

5 PROPERTIES

Following are the few main properties of our DRB protocol.

- *Fairness* In our DRB, each participant gets a fair and equal chance for being a committee member due to the respective committee selection strategy. Even the participants having a difference in their available resources, the chances of being a committee member are independent of the available resources to the participants. The resources (computational power) can help to solve the puzzle faster than honest participants having commodity available resources. The collusion of participants will not give any more advantage in solving the puzzle compared to the fastest processor available to a powerful adversary. Since the puzzle is non-parallelizable and takes an unknown number of steps to solve brings more fairness to the protocol. Nevertheless, the composition of committee selection and cryptographic puzzle provides a fair chance for each participant to be a leader and propose the random beacon output.
- *Scalability* Scalability is hard to achieve in DRB protocols, especially those involving DKG setup or secret sharing among participants. As mentioned in Section 4.3, in order to scale the protocol with a large number of participants, the communication complexity needs to be minimized. In our competitive DRB protocol, in each round a leader participant has to send his block to other participants of DRB protocol, it requires only $O(n)$ communication. Moreover, a new participant can easily join our DRB protocol by following the Nakamoto consensus and adapting the longest chain from his neighboring participants. Low communication complexity and easy joining of participants guarantee our DRB protocol scalable.
- *Unpredictability* Unpredictability ensures that the adversary's availability to predict the random output for the future rounds is bounded. In case of this property breaks, an adversary can take advantage of beacon output in randomness-based applications.

Theorem 5.1. *An adversary $\mathcal{A}$ can not predict the random beacon output O_{r+m} of DRB protocol at the beginning of round r.*

Proof. We present a sketch for the proof. Due to the fair selection of committee selection strategy, the committees of these rounds $C_r, C_{r+1}, \ldots, C_{r+m}$ will not always contain many adversarial participants. Moreover, having a different cryptographic puzzle with an unpredictable number of solution steps poses one claim such that in consecutive rounds $\{r, r+1, \ldots, r+m\}$, there is at least one correct leader participant P_l^e for round e. For the round e, the best an adversary can do is guess with negligible probability. Therefore, the adversary can not compute the output O_e, and output of the consecutive rounds are unpredictable. □

- *Unbiasability* Unbiasability refers to the adversary's ability to bias the produced random beacon output for its advantage. An adversary cannot manipulate an output R_r of round r, as it will not pass the required checks in order to be accepted by other participants. Even if an adversary tries to withhold the beacon output, it will not give any advantage for the next round because, in the next round, the adversary might not be in the committee and does not know the exact number of steps to solve the puzzle for the next round. Therefore, the adversary will not be able to bias the beacon output.

Table 1: Comparison of DRB Protocols based on Time-sensitive Cryptography

Protocol	Network Model	Adaptive Adversary	Liveness	Unpredictability	Bias-Resistance	Fault-tolerance	Communication Complexity	Computation Complexity	Front-running Attack	Trusted Dealer or DKG required
Unicorn [30]	asyn.	✗	✓†	✓	✓	1/2	$O(1)$	Sloth	Hard	✓
Continuous VDF [22]	asyn.	✗	✓†	✓	✓	1/2	$O(1)$	VDF	Easy	✗
RANDAO [40]	asyn.	✗	✓	✗	✗	1/2	$O(n)$	VDF	Easy	✓
RANDCHAIN [25]	syn.	✓	✓	✓	✓	1/3	$O(n)$	VDF	Moderate	✓
RandRunner [41]	syn.	✓	✓	✓‡	✓	1/2	$O(n^2)$	VDF	Moderate	✓
Our Protocol	syn.	✓	✓	✓	✓	1/3	$O(n)$	VDF	Hard	✓

• Network-model refers to whether messages in the DRB protocol are delivered within a known time-bound (synchronous (syn.)) or without a known time-bound (asynchronous (asyn.)).
• Fault-tolerance refers to the number of byzantine faults a DRB can tolerate.
• Adaptive Adversary corrupts the participants during the protocol execution.
• Front-running Attack vector shows whether it is easy or moderate or hard to mount a front-running (block withholding) attack in the DRB protocol.
‡ refers to probabilistic guarantees for unpredictability and has a bound on the number of future rounds an adaptive rushing adversary can predict the beacon output.
† The node with more computational power learns the beacon output earlier than others.

THEOREM 5.2. *An adversary $\mathcal{A}$ cannot influence the random beacon output O_r for any round $r \geq 1$.*

PROOF. The output O_r is derived from the puzzle solution R_r using a deterministic hash function H_{out} that can not be influenced by an adversary. In case, an adversarial leader $\mathcal{A}$ tries to send two (or multiple) valid solutions $S_r^1, \ldots, S_r^j$ (forking) will be caught due to his signature on all the block headers corresponding to these solutions. Further, due to the slashing rule, the adversary $\mathcal{A}$ will be punished. Therefore, the adversary will not try to bias the beacon output. On the contrary, this shows a weak bias-resistance. □

To achieve strong bias-resistance, the output should be computed using VDF as described in Section 3.4. In this case, due to the sequential property of VDF, the adversary can get to know (compute) the random beacon output after a certain number of rounds and till then the main chain has already grown by the honest participants.
In this case of random beacon output computation using VDF, the header h_r parameter O_r (as O_r will be available after $r+j$ rounds, where $j \geq 1$) will no longer be considered as actual random beacon output, however, it will be used for committee selection. The actual random beacon output is computed by applying VDF for a certain time parameter T_{out}.

- *Availability* Availability ensures that the protocol will always proceed and generate continuous random beacon output. Given the maximum network delay, and an approximate solution time for the VDF puzzle, the protocol should produce a random beacon output within the total of these times, let say t_{max}. Therefore, in each round, at least one committee member would be able to solve the puzzle and produce the random beacon output within the known time bound t_{max}.

THEOREM 5.3. *Each correct participant knowing the random beacon output O_{r-1}, can always access the random beacon output O_r by the end of round r.*

PROOF. Due to the fact that our competitive DRB applies Nakamoto consensus, it also follows the chain-growth definition [36] where the blocks are added to the main chain by the correct participants at a certain rate. Depending upon the hardness of the cryptographic puzzle, the puzzle output is produced at a certain rate δ. Since the time required for both the committee selection and for applying the hash function H_{out} is negligible, a correct participant can always get the output O_r within δ seconds. Therefore, at the end of a round r, the chain will always make progress and an output O_r will be available to the participants. □

- *Public Verifiability* Given a block, it is always possible to verify the correctness of random beacon output. In each block, a transcript for the corresponding round execution is provided in the header which involves a proof that corresponds to the random output. Hence, any third party first verify whether the given transcript is correct and further verifies the output using the provided information in the header. Therefore, our DRB protocol achieves public verifiability and can be used efficiently in randomness-based applications.

Theorem 5.4. *An external participant $\mathcal{E}$ can always verify the correctness of random beacon output O_r at the end of round r.*

Proof. Given a valid transcript needed to prove the correctness of beacon output, an external participant can easily verify the output. A valid transcript $\mathcal{T}$ for a round r can consist of the following necessary information and is provided to $\mathcal{E}$ by any correct participant.
– Public key of leader participant pk_l
– Proof of correct input generation x, π^x
– Solution to the puzzle (t, R_r, π^r, τ)
The external participant $\mathcal{E}$ can perform verify algorithms of VRF and VDF to check the correctness of x and R_r by using π^x, π^r respectively. It can further check the threshold condition and beacon output correctness using equation 4 and applying H_{out} on R_r respectively. □

6 CONCLUSION

In this paper, we presented a general way to construct competitive DRB protocols. Our competitive DRB protocol is scalable and provides better fairness to the participants compared to the existing competitive DRB protocol. We pointed out some of the main challenges in existing DRB protocols. Our competitive DRB protocols solve most of these challenges and achieve all the necessary properties of a random beacon. We also mentioned some of the major problems in competitive DRB protocols and provided a few possible solutions. We also compare time-sensitive cryptography-based DRB protocols in Table 1. At the last, we discuss all the properties achieved by our competitive DRB protocol.

There are many ways to extend and adapt our work. As this work proposes a general way for competitive DRB construction, it would be fulfilling to provide a brief construction of a competitive DRB protocol with detailed security proofs. It would be interesting to conduct a thorough experiment to check the robustness and efficiency of our DRB protocols, especially provided the number of participants and adversary-controlled participants, what should be an optimal committee size is an intriguing question to work with. Another exciting direction would be to follow the solutions mentioned to deal with the problems in competitive DRB and also to research for finding better solutions. Instead of proof sketch, it would be interesting to prove all the properties of the DRB protocol by providing concrete detailed proofs.

REFERENCES

[1] Ben Adida. 2008. Helios: Web-based Open-Audit Voting.. In *USENIX security symposium*, Vol. 17. 335–348.
[2] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. 2019. On fairness in committee-based blockchains. *arXiv preprint arXiv:1910.09786* (2019).
[3] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. 2020. Can a Public Blockchain Keep a Secret?. In *Theory of Cryptography Conference*. Springer, 260–290.
[4] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. 2020. RandPiper-Reconfiguration-Friendly Random Beacons with Quadratic Communication. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1590.
[5] Manuel Blum. 1983. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News* 15, 1 (1983), 23–27.
[6] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer International Publishing, Cham, 757–788.
[7] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. 2015. On Bitcoin as a public randomness source. *IACR Cryptol. ePrint Arch.* 2015 (2015), 1015.
[8] Benedikt Bünz, Steven Goldfeder, and Joseph Bonneau. 2017. Proofs-of-delay and randomness beacons in ethereum. *IEEE Security and Privacy on the blockchain (IEEE S&B)* (2017).
[9] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*. Springer, 524–541.
[10] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. *Journal of Cryptology* 18, 3 (2005), 219–246.
[11] Ignacio Cascudo and Bernardo David. 2017. SCRAPE: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*. Springer, 537–556.
[12] Ignacio Cascudo and Bernardo David. 2020. Albatross: publicly attestable batched randomness based on secret sharing. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 311–341.
[13] Ignacio Cascudo, Bernardo David, Omer Shlomovits, and Denis Varlakov. 2021. Mt. Random: Multi-Tiered Randomness Beacons. Cryptology ePrint Archive, Report 2021/1096.
[14] T-H Hubert Chan, Rafael Pass, and Elaine Shi. 2019. Consensus through herding. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 720–749.
[15] Alisa Cherniaeva, Ilia Shirobokov, and Omer Shlomovits. 2019. Homomorphic Encryption Random Beacon. *IACR Cryptol. ePrint Arch.* 2019 (2019), 1320.
[16] Tarun Chitra and Uthsav Chitra. 2019. Committee selection is more similar than you think: Evidence from avalanche and stellar. *arXiv preprint arXiv:1904.09839* (2019).
[17] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. 2021. SPURT: Scalable Distributed Randomness Beacon with Transparent Setup. *IACR Cryptol. ePrint Arch.* 2021 (2021), 100.
[18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 66–98.
[19] Soubhik Deb, Sreeram Kannan, and David Tse. 2021. PoSAT: Proof-of-work availability and unpredictability, without the work. In *International Conference on Financial Cryptography and Data Security*. Springer, 104–128.
[20] Justin Drake. 2018. Minimal VDF randomness beacon. *Ethereum Research* (2018).
[21] drand. 2020. Drand - a distributed randomness beacon daemon. https://github.com/drand/drand
[22] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. 2020. Continuous verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 125–154.
[23] David Galindo, Jia Liu, Mihai Ordean, and Jin-Mann Wong. 2020. Fully Distributed Verifiable Random Functions and their Application to Decentralised Random Beacons. *IACR Cryptol. ePrint Arch.* 2020 (2020), 96.
[24] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*. 51–68.
[25] Runchao Han, Haoyu Lin, and Jiangshan Yu. 2020. RandChain: A Scalable and Fair Decentralised Randomness Beacon. *Cryptology ePrint Archive* (2020).
[26] Timo Hanke, Mahnush Movahedi, and Dominic Williams. 2018. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548* (2018).
[27] Ethan Heilman. 2014. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*. Springer, 161–162.
[28] Sebastian Henningsen, Daniel Teunis, Martin Florian, and Björn Scheuermann. 2019. Eclipsing ethereum peers with false friends. *arXiv preprint arXiv:1908.10141* (2019).
[29] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.
[30] Arjen K Lenstra and Benjamin Wesolowski. 2015. A random zoo: sloth, unicorn, and trx. *IACR Cryptol. ePrint Arch.* 2015 (2015), 366.
[31] Jieyi Long and Ribao Wei. 2019. Nakamoto consensus with verifiable delay puzzle. *arXiv preprint arXiv:1908.06394* (2019).
[32] Dahlia Malkhi and Michael Reiter. 1998. Byzantine quorum systems. *Distributed computing* 11, 4 (1998), 203–213.
[33] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. 2018. Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network. *IACR Cryptol. ePrint Arch.* 2018 (2018), 236.
[34] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf.
[35] Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. 2020. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*. Springer, 61–78.

[36] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.
[37] Krzysztof Pietrzak. 2018. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*.
[38] Michael O Rabin. 1983. Transaction protection by beacons. *J. Comput. System Sci.* 27, 2 (1983), 256–267.
[39] Mayank Raikwar and Danilo Gligoroski. 2021. R3V: Robust Round Robin VDF-based Consensus. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 81–88.
[40] Randao. [n.d.]. RANDAO: A DAO working as RNG of Ethereum. https://github.com/randao/randao. [Online; accessed 1-Nov-2021].
[41] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar Weippl. 2020. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. *IACR Cryptol. ePrint Arch.* 2020 (2020), 942.
[42] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. 2020. Hydrand: Efficient continuous distributed randomness. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 73–89.
[43] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. 2017. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*. Ieee, 444–460.
[44] Benjamin Wesolowski. 2019. Efficient Verifiable Delay Functions. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 379–407.
[45] Ren Zhang and Bart Preneel. 2017. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*. Springer, 277–292.

A VERIFIABLE DELAY FUNCTION

Following, we define a basic construction of VDF based on modular exponentiation as it is discussed in the two schemes.

Definition A.1. (VDF based on modular exponentiation): *A VDF based on modular exponentiation is defined as a tuple of following algorithms:*

- $\mathsf{Setup}(\lambda, T)$: It is a randomized algorithm that takes security parameter λ, time parameter T and outputs public parameter $pp := (\mathbb{G}, N, H, T)$, where $\mathbb{G}$ is a finite abelian group of unknown order, N is an RSA modulus, and $H : \mathcal{X} \rightarrow \mathbb{G}$ is a hash function.
- $\mathsf{Eval}(pp, x, T)$: The evaluation algorithm applies T squarings in $\mathbb{G}$ starting with $H(x)$ and outputs the value $y \leftarrow H(x)^{(2^T)} \bmod N$, along with a proof π.
- $\mathsf{Verify}(pp, x, y, \pi, T)$: The verification algorithm outputs a bit $\in \{0, 1\}$, given the input as public parameter pp, input value x, output value y, proof π, and time parameter T.

A VDF ensures the following three security properties:

(1) *ϵ-Evaluation time*: The evaluation algorithm Eval of VDF runs in at most $(1 + \epsilon)T$ time.
(2) *Sequentiality*: An algorithm using at most $poly(\lambda)$ parallel processors can not compute the function output in less than time T.
(3) *Uniqueness*: Evaluation algorithm should produce exactly one output value y for an input value x which can be verified using the Verification algorithm.

B VERIFIABLE RANDOM FUNCTION

A VRF has the following security properties:

(1) *Uniqueness*: It ensures that for any input M and for a fixed public key of VRF vk, there is a unique VRF output O that can be proved valid using vk.
(2) *Collison resistance*: VRF should be collision resistant which means that a single VRF output O should not be generated two from different inputs M_1, M_2.
(3) *Pseudorandomness*: VRF output O should be indistinguishable from a random value in a view of adversarial verifier without having proof π in his view.
(4) *Unpredictability*: Pseudorandomness does not hold if the VRF keys are generated in adversarial manner. Therefore, a different type of unpredictability property is desirable in certain VRF applications. This property states that if the VRF input has enough entropy, then the VRF output is indistinguishable from uniform, similar to an ordinary hash function.

Part III

Secondary Papers

Paper J

Trends in Development of Databases and Blockchain

M. Raikwar, D. Gligoroski and G. Velinov

Seventh International Conference on Software Defined Systems (SDS), IEEE, 2020, pp. 177-182

This work is about the mutual influence between two technologies: Databases and Blockchain. It addresses two questions: 1. How the database technology has influenced the development of blockchain technology?, and 2. How blockchain technology has influenced the introduction of new functionalities in some modern databases? For the first question, we explain how database technology contributes to blockchain technology by unlocking different features such as ACID (Atomicity, Consistency, Isolation, and Durability) transactional consistency, rich queries, real-time analytics, and low latency. We explain how the CAP (Consistency, Availability, Partition tolerance) theorem known for databases influenced the DCS (Decentralization, Consistency, Scalability) theorem for the blockchain systems. By using an analogous relaxation approach as it was used for the proof of the CAP theorem, we postulate a "DCS-satisfiability conjecture." For the second question, we review different databases that are designed specifically for blockchain and provide most of the blockchain functionality like immutability, privacy, censorship resistance, along with database features.

Paper K

Aggregation in Blockchain Ecosystem

M. Raikwar, D. Gligoroski

Eighth International Conference on Software Defined Systems (SDS), IEEE, 2021, pp. 1-6

Blockchain has evolved rapidly in the past decade, but it still faces challenges such as scalability, large block size, slow block verification, high communication overhead. Though multiple solutions have been proposed to overcome these challenges, a few solutions perform aggregation of blockchain data. Moreover, blockchain data can be a blockchain state, transaction, or consensus message. Therefore, the solutions involving aggregation have immense potential to solve several existing challenges in the blockchain ecosystem. In this work, we investigate and scrutinize possible aggregations in the blockchain. For that purpose, we first briefly describe cryptographic primitives with aggregation scheme and their applicability in the blockchain. These schemes can empower the blockchain with improved scalability, reduced block size, low communication overhead, and fast block verification. Then, we identify nine research problems related to these cryptographic primitives.

Paper L

Efficient Novel Privacy Preserving PoS Protocol Proof-of-concept with Algorand

K. Stevenson, O. Skoglund, M. Raikwar, and D. Gligoroski

3rd Blockchain and Internet of Things Conference (BIOTC), ACM, 2021, pp. 44-52

Proof of Stake (PoS) emerged to replace and tackle the problem of vast energy consumption in Proof of Work (PoW) consensus. PoS is based on the assumption that the majority of the stake is owned by honest participants. Consequently, instead of solving a computationally hard puzzle to propose the next block in the blockchain, PoS selects a participant with probability proportional to its stake in the network. In contrast to the solution to the puzzle, the proof of selection in PoS has inherent privacy issues. The identity of the selected participant is revealed to other participants to verify the proof, and the stake of the selected can be deducted by frequency analysis. Therefore, Private Proof of Stake (PPoS) emerged to provide a valid alternative to PoW, aiming to tackle the energy consumption in PoW while preserving the privacy of the selected participant in a consensus round. Recent PPoS protocols by Baldimtsi et al. and Ganesh et al., rely on an anonymous broadcast channel and have a large proof size that hinders the practical implementation of the protocols.

In this paper, we identify issues and areas of improvement within the current PPoS protocols. We built our privacy-preserving PoS scheme upon the anonymous lottery by Baldimtsi et al. with an instantiation of Algorand as the underlying PoS protocol. We apply fully homomorphic encryption along with zero-knowledge proof techniques to reduce the proof size and to achieve privacy of selected participant's stake and identity. In comparison with the original anonymous lottery scheme, our scheme achieves better efficiency and complexity.

Paper M

Databases fit for blockchain technology: A complete overview

J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, D. Gligoroski

Submitted to Elsevier Journal on Blockchain: Research and Applications, 2022

Efficient data storage and query processing systems play a vital role in many different research areas. Blockchain technology and distributed ledgers attract massive attention and trigger multiple projects in various industries. Nevertheless, Blockchain still lacks features from databases, such as high throughput, low latency, and high capacity. There have been many proposed approaches for handling the data storage and query processing solutions in Blockchain for that purpose. This paper presents a complete overview of many different types of databases and how these databases can be used to implement, enhance, and further improve Blockchain technology. More concretely, we give an overview of 13 transactional databases, an extensive overview of 16 analytical database engines, and 15 hybrids, i.e., translytical databases. We explain how the database technology has influenced the development of Blockchain technology by unlocking different features such as Atomicity, Consistency, Isolation, and Durability (ACID), transaction consistency, rich queries, real-time analysis, and low latency. Using a relaxation approach analogous to the one used to prove the CAP theorem, we postulate a "Decentralization, Consistency, and Scalability (DCS)-satisfiability conjecture" and give concrete strategies for achieving the relaxed DCS conditions. We also provide an overview of the different databases, emphasizing their architecture, storage manager, query processing, and implementation.

Paper N

Cryptographic Primitives in Blockchain

M. Raikwar, S. Wu

Book chapter in S. Kanhere, M. Conti, and S. Ruj, (editors) “Blockchains - A Handbook on Fundamentals, Platforms, and Applications”, Springer, 2022

Blockchain is promising, powerful, and still a growing technology. However, it still encounters many research challenges. Some of the important challenges are scalability, key management, protection against different attacks, smart contract management, and further improvements on security and privacy in blockchain designs. These challenges emerge due to the underlying consensus mechanism, network infrastructure, and participants’ behavior. Therefore, to overcome these challenges and to achieve the proper functioning of blockchain, many cryptographic primitives can be investigated, scrutinized, and applied in the blockchain.

In this chapter, we present a brief description of cryptographic primitives employed in blockchain. For each cryptographic primitive, we provide the definition, properties and their use in blockchain domain. Furthermore, we also postulate research problems which can be of independent interest.

ISBN 978-82-326-5180-1 (printed ver.)
ISBN 978-82-326-6420-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)