Integrating Blockchain and Fog Computing Technologies for Efficient Privacy-preserving Systems

PhD Theses

Hamza Baniata
Supervisor: Attila Kertesz, PhD

Doctoral School of Computer Science
Department of Software Engineering
Faculty of Science and Informatics
University of Szeged

Szeged
2022

Contents

Acknowledgments

I would like to thank, first, my supervisor Assoc. Prof. Attila Kertesz for his guidance, who is the leader of the IoTCloud Research Group at the Department of Software Engineering, University of Szeged. The door to Prof. Kertesz office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently supported my work and steered me into the right direction, whenever he thought I needed it.

I would also like to thank the experts who were involved in reviewing this thesis, and the papers included within. Their thoughtful recommendations were a major reason behind the successful defence of the work of a 3-year-long journey. Without their passionate participation and input, the validation of my proposals and experiments could not have been as reliable as they are.

I would also like to acknowledge the participation opportunities in several national and international research projects. My research was mostly supported by those projects providing instrumentation, extensions, publication fees, and travel costs. Specifically, I would like to acknowledge:

- The TruBlo project of the European Union's Horizon 2020 research and innovation program under grant agreement No. 957228;
- The European COST program under action identifier CA19135 (CERCIRAS);
- The European Regional Development Fund under grant number GINOP-2.3.2-15-2016-00037;
- The European Social Funds (EFOP-3.6.3-VEKOP-16-2017-00002, and EFOP-3.6.1-16-2016-00008);
- The Hungarian Scientific Research Fund under grant number OTKA FK 131793;
- National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme;
- The Ministry of Innovation and Technology of Hungary with the grants number TUDFO/47138-1/2019-ITM, TKP2021-NVA-09, NKFIH-1279-2/2020 and UNKP-21-4.

Finally, I must express my very profound gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. I am deeply thankful as this accomplishment would not have been possible without you.

Hamza Baniata, 2022.

List of Figures

List of Tables

Abbreviations

3DDL Three-Dimensional DAG-based model of the Distributed Ledger 109, 114, 117, 122, 123, 132

ACO Ant Colony Optimization 92, 93, 95–99, 108

AnoLE Anonymous Leader Election 76, 79, 80, 85, 86, 88, 89

BaaS Blockchain-as-a-Service 11

BC Blockchain ii, 3, 7, 11–21, 25–47, 49–51, 53–61, 63–69, 71–73, 76, 81, 82, 85–89, 91–95, 97, 98, 101–103, 105, 107–115, 117, 118, 120, 121, 123–127, 129, 131, 132, 169

CA Consensus Algorithm 14, 17, 28, 29, 34–37, 39, 42, 44, 46, 47, 49, 55, 56, 60, 61, 93, 95, 98, 109, 111, 112, 116, 118, 121, 123, 124, 132

CC Cloud Computing 11, 15, 16, 28, 30, 54

DID Decentralized Identifier 112, 113, 115–118, 121–125, 129–131

DL Distributed Ledger 11, 14, 16, 63–67, 69, 70, 81, 87, 88, 112–116, 118, 121, 122, 125

DMSTP Distributed Minimum Spanning Tree Problem 68, 76, 85

DONS Dynamic Optimized Neighbor Selection 65, 76, 78, 80, 82, 85, 86, 88, 89

DS Digital Signature 109, 111, 115, 125, 132

DTTP Distributed Trusted Third Party 111, 115–118, 121, 122, 125, 126, 132

FC Fog Computing 13, 15–17, 19–21, 25–30, 32–46, 60, 61, 68, 86, 91, 94, 97, 108–111, 131, 132, 169

FCFS First-Come-First-Served 92, 93, 95, 98, 99, 127

GDPR General Data Protection Regulations 109, 111, 112, 125

GUI Graphical User Interface 44

IoT Internet of Things 13, 15, 16, 25, 26, 28–30, 32, 33, 35, 36, 43, 91, 94

MST Minimum Spanning Tree 65–68, 75, 76, 78–80, 84, 85, 88

MSTP Minimum Spanning Tree Problem 68

Nonce Number Used Once 12

NS Neighbor Selection 64, 65, 67, 68, 76, 80, 81, 85, 102

NSP Neighbor Selection Problem 67, 68, 88

ONS Optimum Neighbor Selection 75, 79, 80, 84, 88

P2P Peer-to-Peer 11, 13

PBFT Practical Byzantine Fault Tolerance 14, 34, 36

PoA Proof-of-Authority 13, 15, 20, 35, 42, 46, 47, 51, 52, 55–57, 60, 109, 111, 114, 116, 118, 123, 124, 132

PoCot Proof-of-Contribution 13, 27

PoET Proof-of-Elapsed-Time 13, 14, 34, 64

PoL Proof-of-Luck 13

PoS Proof-of-Stake 13–15, 20, 27, 33, 42, 46, 51, 52, 55–57, 60, 64

PoSch Proof-of-Schedule 13

PoW Proof-of-Work 3, 12–15, 20, 30, 33–38, 42, 45, 46, 50, 51, 53, 55–58, 60, 64, 71, 87, 89, 95, 98, 112–114

QoS Quality of Service 16, 36, 38, 109

RNS Randomized Neighbor Selection 64, 65, 67, 80, 82, 85

RR Round-Robin 93, 95, 98, 99

RTT Round Trip Time 36, 67, 70, 76, 83

RTT-NS Round Trip Time based Neighbor Selection 65, 80, 82

SC Smart Contracts 12, 13, 16, 20, 32, 34, 35, 42, 43, 94, 95, 97–101, 112

SJF Shortest-Job-First 93, 95, 98, 99

SoW Signatures of Work 109, 111, 114, 116, 123, 124, 131, 132

SS Smart Systems 13, 110, 131, 132

TTP Trusted Third Party 13, 14, 16, 30, 34–36, 39, 67, 76, 111, 113

TX Transaction 4, 12, 13, 16, 17, 28, 34, 35, 44, 45, 47, 49–53, 55–57, 59, 60, 63, 67–69, 80, 92–94, 101–105, 107, 108, 111, 112, 114, 122–126, 129–132

VC Verifiable Credential 109, 112, 113, 115–118, 120–125

VM Virtual Machine 13, 15

VR Virtual Resource 15, 91–97, 99–101, 108

ZKP Zero-Knowledge-Proof 109, 111, 112, 115, 118, 120, 125, 132

Chapter 1

Introduction

1.1 Blockchain Technology

Blockchain (BC) is the base technology of many available cryptocurrency systems. BC was firstly introduced by Nakamoto [1], as an underlying technology that guarantees a trusted, fully-distributed, digital money system. The ability of providing decentralized decisions by a BC network has excited many researchers to deploy it in the decentralized, distributed cloud servers at the edge of the network. Subsequently, Blockchain-as-a-Service (BaaS) was proposed and analyzed in the literature, as a service model similar to the Process-as-a-Service schemes [2]. Thus, different BaaS models are strongly expected to support Cloud Computing (CC) during the life cycle of information processing and management in complex scenarios [3]. Providing computational power and immutable storage abilities, BC should lighten the burden on clouds.

A BC system is a distributed computing system that is able to store and process a Distributed Ledger (DL) in a Peer-to-Peer (P2P) network model [4]. A BC can be permissioned, where participants added by authorized entities within the system are capable of appending (or mining) new blocks onto the DL. Permissionless BCs, on the other hand, allow anyone to participate as a miner in the system without the need for an access grant. Data saved on BC DLs can be either public or private, depending on the domain of users able to request or perform tasks within the system. That is, anyone can access a public BC (e.g. read data on the chain and/or request to be a miner), while only users within a predefined domain within the organization, region, etc. can access data on a private BC. Different BC functionalities appeared in the literature, e.g. data management [5, 6], payment and trading management [7], reputation management [8, 9], e-voting [10], identity management [11, 12], etc.

A BC miner is, in most cases, a rather strong computer that is able to relatively conduct huge number of computational tasks per second [13]. Miners are typically challenged to prove the correctness of their claims by solving a mathematical problem

that is hard to find and easy to verify. Reaching a consensus of 51% or more on the validity of the solution, as well as on other information related to the puzzle and the performed TXs by end users, results in confirming the block of TXs and adding it to a publicly available ledger [14].

Generally, a BC puzzle is a computational challenge $f(.)$, whose solution S must fulfill the condition Ω. In order for S to be sufficiently hard to find, yet easy to verify, Ω should be set moderately according to the network conditions (e.g. the avg. computational capacity C of miners or the avg. transmission delay T between neighbors). S shall be coupled with every newly mined block so that other miners can verify it referring to Ω. A probabilistic finality based system requires several miners to search for S at the same time, while an absolute finality-based system requests one selected miner to find S.

The miner node who solves the puzzle is then rewarded by the network and by the end users whose TXs were confirmed. The difficulty of the puzzle is conditionally determined [15]. For example, Bitcoin requires one block to be confirmed every 10 minutes, while Ethereum average block time is 15 seconds [16]. Hence, if a BC miner is able to solve the puzzle in less time, the algorithm increases the difficulty so that the average time of puzzle solution, and hence the block generation, remains [17]. Utilizing this approach for reaching a consensus is what the Proof-of-Work (PoW) actually is.

Bitcoin is a cryptocurrency secured by a PoW-based BC, where money TXs, merchandise and exchange can be performed. Ethereum, on the other hand, is a PoW-based BC[1] that can handle wider range of applications, including money transfer. Specifically, Ethereum provides a platform where on-line Smart Contracts (SC) can be generated and agreed-on by different users. SCs are small chunks of code, run by BC nodes upon fulfillment of algorithmically verifiable conditions [18]. Those contracts are conditionally automated providing higher trust measures for sensitive applications. Furthermore, the smartness of these contracts is highly flexible as users write and agree on the terms of each contract, hence each contract is different than other contracts and users are not forced to play by any body's rules. The need of SCs does not appear in Bitcoin, or similar cryptocurrency systems, as the rules and terms are unified for all TXs. Miner nodes select and accumulate TXs according to some criteria, such as the preferred TX fees waived by the first user. Once a block reaches a completeness status (configured by the system) such as specific number of TXs per block, the block is mined (i.e. hashed and a Number Used Once (Nonce) is found in a Brute-Force manner). Consequently, the first miner node to find the Nonce broadcasts the new block with its unique Nonce and all network peers validate the correctness of the solution and add the block to the locally saved chain. Once the

[1]Until September 2022 when Ethereum main chain adopted Proof-of-Stake. Throughout this thesis, reference to Ethereum indicates its PoW-based version.

block is added to the chain, all TXs within are confirmed and hence the amounts of senders' digital coins are decreased while the amounts of the receivers' digital coins are increased[2].

In Ethereum, a SC does not have to include transferring digital coins to another user, although it can. All the above mentioned steps apply in Ethereum, except that users actually write a code, which is the SC, that is run on one Ethereum node. The SC can include any type of implementation, and the user who generates this SC provides TX fees that is given for the node who runs the SC code. Depending on how power consuming the SC code is, and fees decided by the system per each computational cycle, the TX fee (typically termed Gas) is decided. However, gas coupled with each SC is also flexible as it is decided by the user, not the system. If the gas was sufficient to completely run the SC, its conditions are successfully performed and the unused gas is returned to the generator. Thus, the SC is run as long as there is available gas.

Generally, BC-based systems are developed with reference to different layers depicted in Figure 1.1. The Hardware/Infrastructure Layer represents devices and equipment used to perform the BC computations, communications and storage services. Examples of such devices include Computers, Virtual Machines (VMs), containers, messaging models, etc. The Data Layer represents the structure of shared messages among entities of the first layer, definitions of encryption and signature methods utilized within upper layers, storage schemes, etc. The Network Layer represents the P2P connections definition between entities of the first layer, communication protocols with technical consideration referred to by the OSI network model, etc. The Consensus Layer represents the rules which network members must follow to maintain the security of the system. Additionally, this layer defines different conceptual types of system elements such as miners, TTPs, users, servers, full nodes, light nodes, etc. Finally, the Application Layer defines the interfaces of the developed system, business models, system monitoring, etc. Examples of applications where the BC technology was found beneficial include the Internet of Things (IoT), Smart Systems (SS), Fog Computing (FC), Cryptocurrency applications, decentralized credential management, etc.

The PoW algorithm, used in most famous BC systems, exposed high consumption of energy for BC systems to maintain its consistency [19]. This motivated the proposal of other proof-based variants for BC applications [20], to mention some: Proof-of-Stake (PoS) [21], Proof-of-Contribution (PoCot) [22], Proof-of-Luck (PoL) [23], Proof-of-Elapsed-Time (PoET) [24], Proof-of-Authority (PoA) [25] and Proof-of-Schedule (PoSch) [26]. All of those, and others, require miner nodes to solve/contribute a challenge, yet the challenge is way less energy consuming com-

[2]Some Blockchain-based systems use the UTXO model instead of the (simply mentioned) ABOT model.

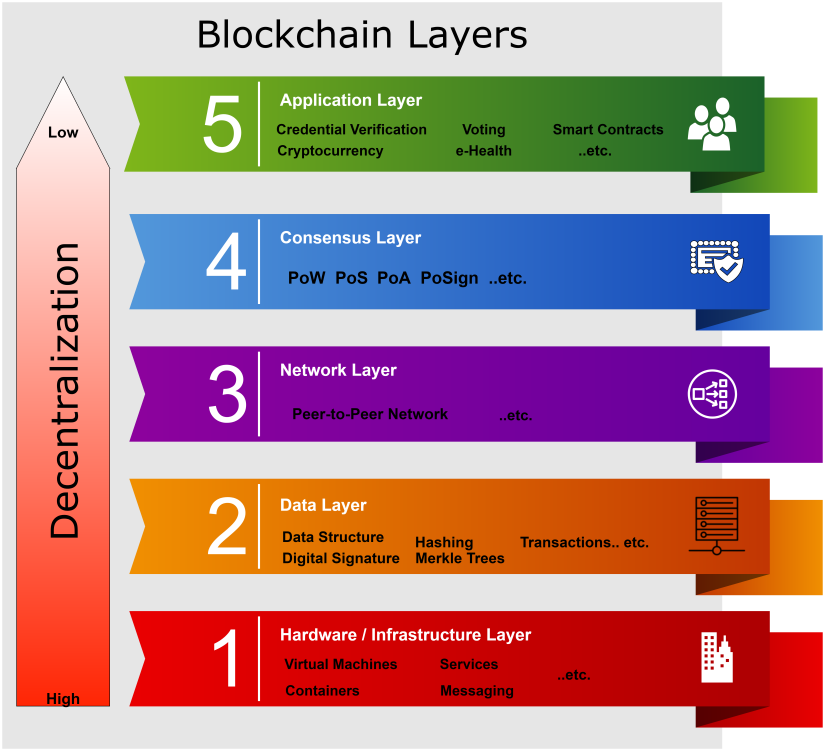


Figure 1.1: *Layers of Blockchain Systems*

pared to PoW. Other non-proof based CAs can be found in the literature, such as the Practical Byzantine Fault Tolerance (PBFT) [27].

The PoS algorithm [28] is currently being optimized to provide similar advantages to PoW, so that it can be deployed in the industry. However, some drawbacks of PoS need to be solved before its official deployment, such as The Monopoly Problem [29], The Bribe Attack [30, 31], and relatively low reliability in TTP-less applications [32].

In PoS-based BCs, miners are chosen randomly by the algorithm. Staking more digital coins in deposit shall increase the probability of being selected as the miner to generate the next block. This provides high trust measures as faulty generated blocks are not tolerated by the system, and the staked coins of malicious/faulty miners would be burned as a penalty.

The probability that different nodes solving the puzzle at the same time is quite low yet existing. Such event is termed *Forking*. Once a fork appears, two different versions of the DL will be considered valid by two different groups of nodes. After a while, the distribution of the two versions through the network will result in accrediting the longer chain, and withdrawing the shorter. However, each CA has a different fork handling protocol.

PoET-based BCs generate randomly selected times for BC nodes. The node whose randomly picked time elapses first, is the one who is granted the opportunity to

generate the next block. PoA, on the other hand, implies that only blocks signed by authorized members are validated and confirmed by the network. Those authorized miners must be known, trusted participants that can be tracked and penalized in case of faulty behaviour. Both of these algorithms share the property of being suitable for private and/or permissioned BCs, while PoW and PoS are known for being suitable for public and permissionless BCs.

Generally, a hash of a/the previous block is added to each newly generated block, so that any alteration attack of this block in the future will be impractical, and hence almost impossible.

1.2 Fog Computing technology

CC is the paradigm that has been used for years now, performing tasks for end-users in a reliable, and efficient manner. CC paradigm offers different types of tasks that can be performed in order to meet storage, computation and communication needs of end-users. Cloud services mainly rely on the use of VMs concept (interchangeably termed as Virtual Resources (VRs)), which logically divides the resources in the cloud, into separate machines, in a way to make it easier for users to get services in a Pay-as-you-Go manner. This means that the less VMs/VRs used, the less payment for the services is to be committed. Consequently, in big applications that require a large number of VRs, optimizing VR usage in terms of utilization time may save the application users a great deal of money. Furthermore, the decreased usage of VRs leads to enhanced levels of energy utilization and system efficiency [33].

Constant streams of data and information are expected to rise as an issue in such complex scenarios. Additionally, those streams may carry highly sensitive and private data that need to be secured during, and after processing and storage. As an extension of the cloud at the edge of the network, Fog Computing (FC) is envisioned to provide cloud services closer to end-users, and to solve the aforementioned issues.

FC as defined in [34] is a geographically distributed computing architecture, in which various heterogeneous devices at the edge of the network are ubiquitously connected to collaboratively provide elastic computation, communication and storage services. FC can also be defined as a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud datacenters [35]. FC, also known as Fog Networking and Fogging, was introduced by Cisco in 2013, as the future of the current CC-based systems.

The basic idea of FC is to create a layer of distributed fog entities between the centralized cloud data centers/processors and end-user devices (or Things in the case of IoT systems [36]) at the edge of the network. The FC layer should, conceptually, control the handling of users' data in a private and secure manner, while providing cloud services. Different reference architectures were proposed for the FC paradigm,

e.g. by Habibi et al. [37], Dastjerdi et al. [38], the OpenFog consortium [39], and Cisco [40]. Utilizing FC was proven to be more efficient, than classical, centralized cloud-based services, in terms of overall system throughput [41], response latency [42], storage efficiency [43], and privacy [44].

The FC layer can be studied in three levels, namely the node level, the system level, and the service level [45]. The fog consists of several nodes connected to each other and to the cloud. In such FC-enhanced cloud settings, the service is requested by end-users to the fog layer, which provides this service if possible. Otherwise, the request is forwarded to the cloud where complex and time consuming actions are performed. However, information of the complexity of the system, and the decision making process in the fog layer, should not be within the concern of end-users. That is, end-users require their tasks to be performed within a privacy-aware context and the QoS measures implications that were agreed on.

1.3 Integration of Fog Computing and Blockchain technologies

The integration of FC with BC had been recently discussed by many researchers [46]. On one hand, more efficient services can be provided by FC over CC, mostly required by IoT systems. On the other hand, the BC technology can be deployed for reliable, TTP-free, and secure TXs ledger in such distributed environments.

Within a BC context, data storage service model implies that pieces of data are saved on the immutable DL. Such data may be of any type including data records, IDs, digital payment registration, reputation measures of end-users or fog components, real estate ownership tokens, etc. Other applications do not require transferring assets rather than saving data on the chain only, such as voting applications and eHealth applications. However, the mentioned second type of applications may also need, on some level, a digital payment method be embedded. In such cases, SCs on other payment platforms can be implemented and generated.

Performing computations for end-users is the second service model that the BC can provide. That is, computational tasks can be sent by end-users/fog entities to the BC in the form of SC. After running the SCs, the results can be saved in a centralized or decentralized form according to the pre-run configuration. Figure 1.2 presents how the services, classically provided by a Cloud/Fog system, can be interpreted into the form of services that can be provided by a BC system. It can be noticed from the figure that SCs can be considered relevant to cloud computational services, while different types of data saved on the DL can be considered a relevant option to the centralized storage model provided by a cloud system.

As a BC-assisted FC system can provide computational and storage services, the

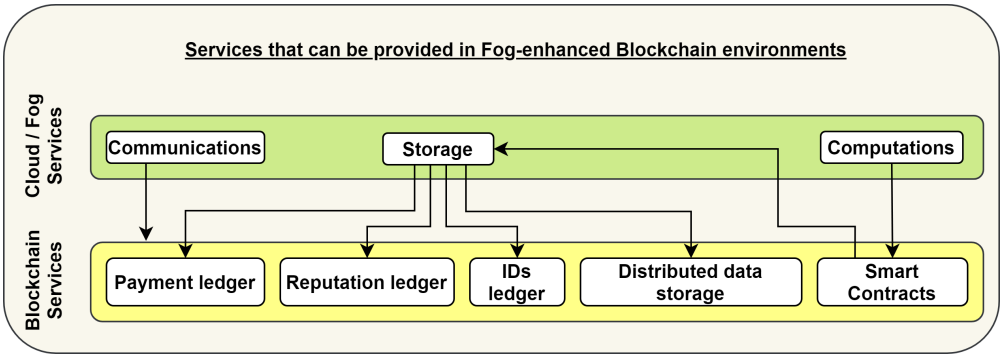


Figure 1.2: *Service models provided by Cloud/Fog systems, and their relevant service models provided by BC systems*

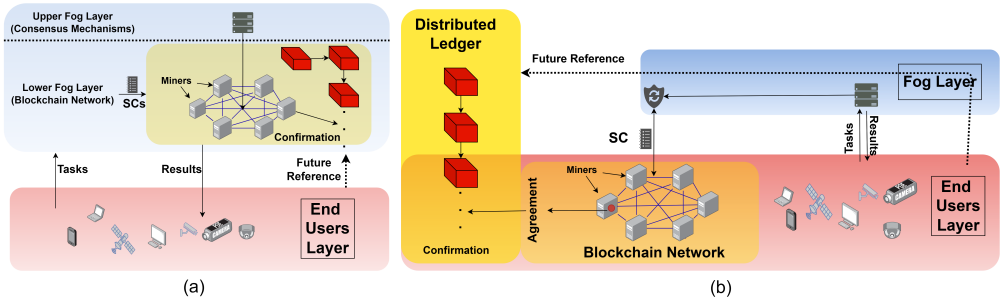


Figure 1.3: *FC-BC integration system model, where (a) the BC is deployed in the fog layer, and (b) the BC is deployed in the end-user layer*

BC placement within the the FC architecture may differ. That is, BC can be placed in the fog layer, the end-user layer, or the cloud layer. If the fog layer is controlling and automating the communications between the end-user layer and the BC network, as in [47], the TXs are sent from end-users to the fog layer. After that, some communications take place between the fog layer and the BC network in order to successfully perform the requested tasks. In such system model, it is assumed that the BC network lays in a different layer than the fog layer. Feedback with the appropriate result of each TX should be easily achievable then by end-users as it is saved on-chain.

When the BC is deployed in the fog layer, storage and computational services are performed by the fog nodes themselves. In other words, fog nodes wear a second hat, which is a BC network hat. Thus, when storage to be provided by the fog, while fog nodes are also BC nodes, data is stored in *all* fog nodes in the fog layer. A simple system model is demonstrated in Figure 1.3.a, where only one chain is constructed in the lower fog layer and one fog control point in the upper layer monitors the BC functionality. However, such model is not practical and more complexities appear in a real-life scenarios, including heterogeneous fog nodes, multiple BCs deployment, different CAs, and different service models.

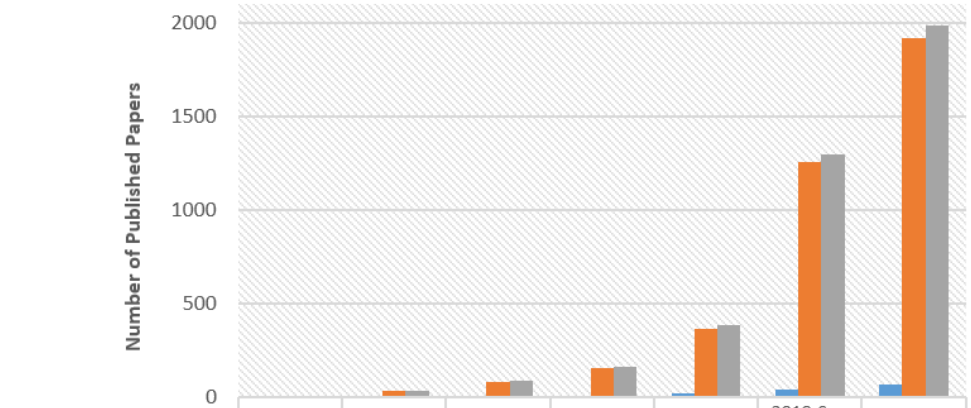
In a system model where the BC is deployed in the end-user layer, two types of end-users are distinguishable; namely task requester and BC node. In a Fog-enhanced BC system, the fog controls the communications between the two types of end-users. Specifically, BC nodes perform the tasks that were sent by the fog. Those tasks were originally requested by task requester end-users. The fog can preserve the privacy of data and incentivize BC nodes in the form of digital currency, as in [48]. BC nodes can be further sub-categorized in such settings according to the scenario to be simulated. Adding other types of BC nodes is up to the developers and the system model. For example, the Bitcoin system is modeled in a simpler way, were BC is directly connected to task requester end-users, and it only provides a payment ledger service. Ethereum, on the other hand, surpass Bitcoin because it can provide more services to end-users. The system model when the BC is deployed in the end-user layer is demonstrated in Figure 1.3.b.

I have identified several challenges that can be inherited from both technologies into their integrated solutions:

- **Challenge 1:** I have not found any reliable FC-BC simulation tools that can directly address FC-BC integration.
- **Challenge 2:** As FC provides enhanced QoS, BC tends to secure these services with methods that lower the QoS measurements.
- **Challenge 3:** BC solutions generally consume more power compared to centralized ones.
- **Challenge 4:** Privacy awareness of BC and FC technologies is a major challenge for both of these technologies. Thus, integrated FC-BC solutions must provably maintain, or better enhance, the level of different types of user privacy.
- **Challenge 5:** The consistency of distributed ledgers is the main challenge that BCs attempt to solve, while the distributed infrastructure of the fog subjectively implies that such solutions are needed. Other complex trade-offs are considered major factors, contributing to successful integration of FC and BC, resulting in secure, privacy-preserving, efficient and, most importantly, trusted solutions.

1.4 Privacy

As declared in the global BC survey [49], 62% of Chinese surveyed business owners believed that the biggest concern for them adopting BC-based technologies is privacy. The percentage was close in Malaysia and USA with 51%. On the other hand, 50% of surveyed companies in twelve different countries think it would be better, if they



	2013	2014	2015	2016	2017	2018-Sep 2019	Total
concerned in Privacy	1	0	4	2	22	40	69
not concerned in Privacy	11	39	84	160	365	1260	1919
Total	12	39	88	162	387	1300	1988

Figure 1.4: *Privacy-Concerned vs. None Privacy-concerned published papers from 2013 to 2019*

could only use BC technologies privately/internally. Following these insights, Data-Protection-By-Design, a concept that was recently introduced for enhancing privacy [50], needs to be adopted in BC systems. The NIST computer security handbook defines Privacy as "*a confidentiality term which assures that individuals control of influence what information related to them may be collected and stored, and by whom and to whom that information may be disclosed*" [51].

Privacy in, specifically, FC-enabled solutions is not well defined yet, despite the property of being mostly people-centric [52]. Figure 1.4 shows the ratio of published papers from the year of 2013 until September 2019 citing the phrase "Fog Computing" in their title[3]. Searching for papers whose titles also include the word "Privacy", I found only 69 papers with a percentage of less than 4% relative to the total number of published papers as shown in Figure 1.4. Out of those 69 papers, 14 were survey articles, which are concluded in Table 1.1. The privacy factor that is assigned " $\surd$ " for some surveys indicates that those surveys discussed this factor and recommended specific applicable solutions that were proposed in the literature.

It can be seen in Figure 1.4 that the research work citing *Fog Computing* and *Privacy* is relatively low. Privacy provisioning is considered a big challenge for fog developers [66, 67] because of some major barriers such as heterogeneity, mobility issues, low control upon open air transmission medium, and different business relationships in a multi-owned scenario.

Through my research, I found that privacy is usually discussed and categorized

[3]scholar.google.com; last accessed: September 24, 2019

Table 1.1: *Privacy-Concerned surveys from 2013 to 2019*

Reference	Data	Usage	Location
[53]	$\surd$	$\surd$	$\surd$
[54, 55]	$\surd$	$\surd$	$\surd$
[56]	$\surd$	X	$\surd$
[57]	$\surd$	$\surd$	X
[52]	X	X	$\surd$
[58, 59]	$\surd$	X	X
[60, 61, 62, 63, 64, 65]	X	X	X

into three pivotal types [54] namely Data Privacy, Usage Privacy, and Location Privacy. Some works also categorize the Identity Privacy as a fourth type, while others consider it as part of the Data Privacy. Referring to several privacy and FC concerned references [68, 69, 70, 71, 72], a definition of a Private Fog System can be thus provided as "*a system that is capable of maintaining and preserving fog properties along with data privacy, usage privacy, and location privacy*".

1.5 Contributions

I have attempted to address all of the aforementioned challenges by optimizing different integration models of FC and BC. First, I performed detailed and extended literature review of related simulation tools and integration approaches [J1, J2, C1]. Accordingly, I implemented FoBSim [J2][C2]; a novel FC-BC simulation tool that allows for reliable and realistic integration simulation. FoBSim facilitates the simulation of different consensus algorithms (PoW, PoA and PoS) and different applications (e.g. cryptocurrency, SCs, etc.). It also allows to deploy the BC at different layers of an FC-enabled cloud system, with the advantage of easy parameterization of simulation scenarios (**Challenge 1**). Using this tool, I experimentally proved how integrating FC and BC meets my expected enhancement in terms of latency and cost (**Challenges 2 & 3**). Additionally, I analyzed different factors affecting distributed ledger consistency and trust in [C3], which motivated the development of novel methods for quantifying the consistency and reliability of BCs. Using these methods, I could introduce a decision-making model resulting in better integration potential of FC and BC technologies (**Challenges 2, 3 & 5**).

Based on the state-of-the-art, and following my experiments and their evaluation results, I designed two novel protocols aiming to enhance FC-BC integrated applications. The DONS protocol [J3] allows for dynamic and optimized neighbour selection in BC networks, which provably enhanced block finality and optimized message fidelity (**Challenges 2 & 3**). The AnoLE protocol [J3] allows for privacy-preserving

leader election in public permissionless BCs (**Challenges 4 & 5**).

I have also designed two privacy-preserving solutions where BC is exploited to enhance FC efficiency in terms of latency, namely PF-BTS [J4], and where FC is exploited to enhance BC efficiency in terms of block validation time and energy consumption, namely PF-BVM [C4] (**Challenges 2, 3, 4 & 5**). Finally, in order to exploit the advantages of integrated FC-BC solutions, I designed and developed PriFoB [J5]; a novel Blockchain-based Fog-enhanced global accreditation and credential verification system. Comparing PriFoB to widely adopted BC platforms and solutions, such as Ethereum, Hyperledgr Fabric, Indy and Besu, PriFoB outperformed all of these in terms of latency and throughput. Meanwhile, PriFoB showed high levels of security and privacy, along with guaranteed compliance with the GDPR (**Challenge 5**). Following the results I obtained throughout my research, I discussed future works, regarding the utilization of FC-BC integrated systems, for trusted and robust smart system applications [C5] (**Challenge 5**).

The ideas, figures, tables and results included in this thesis were published in scientific papers (listed at the end of the thesis). To summarize, I made the following contributions:

- Thesis 1:

  Chapter 2: I performed a comprehensive literature review related to integrated Fog Computing and Blockchain solutions, tools and applications.

  Chapter 3: I designed a novel and extensible simulation tool called FoBSim that can comprehensively and realistically mimic such integrated systems. I experimentally validated and evaluated the simulation environment with different use cases and simulation parameters.

- Thesis 2:

  Chapter 4: I formalized and quantified the concepts of Blockchain network consistency and reliability. I developed two novel protocols (DONS and AnoLE) for the neighbor selection problem, targeting optimal block finality and privacy-preserving data propagation in public and permissionless Blockchains. I experimentally proved that Blockchain and Fog Computing integration is advantageous.

- Thesis 3:

  Chapter 5: I designed and developed two privacy-aware protocols for FC latency enhancement using BC, namely PF-BTS, and for BC energy efficiency us-

ing FC, namely PF-BVM. PF-BTS allows the fog layer to exploit BC in a privacy-aware manner to provide optimal task schedules for cloud infrastructures. PF-BVM allows BC networks to exploit fog nodes for faster privacy-aware block validation.

- Thesis 4:

  Chapter 6: I developed a Fog-enhanced Blockchain-based solution (called PriFoB) to realize privacy-aware institution accreditation and credential validation. I designed a novel hybrid Proof-of-Authority and Signatures-of-Work consensus algorithm to enhance security and efficiency. I proposed a partially immutable, three dimensional, DAG-based Distributed Ledger to support revoking on-chain data.

Publications, Theses and Citations

Table 1.2 summarizes the relation between the theses and the corresponding publications.

Table 1.2: *Relation between the theses and publications.*

no.	Publication	Year	Thesis				Citations
			1	2	3	4	
1.	[J1] IEEE-Access	2020	•				35
2.	[C1] FMEC	2020	•				5
3.	[C2] CSCS	2020	•	•			1
4.	[J2] PeerJ-CS	2021	•	•			5
5.	[C3] Euro-Par	2021		•			5
6.	[J3] FGCS	2022		•			5
7.	[C4] CLOSER	2020		•	•		6
8.	[J4] IPM	2021			•		58
9.	[C5] CERCIRAS	2021				•	–
10.	[J5] JNCA	2022				•	–

Journal publications

[J1] Hamza Baniata and Attila Kertesz. A survey on blockchain-fog integration approaches. *IEEE Access*, vol. 8, 102657-102668, 2020.

[J2] Hamza Baniata and Attila Kertesz. FoBSim: an extensible open-source simulation tool for integrated fog-blockchain systems. *PeerJ Computer Science*, vol. 7, e431, 2021.

[J3] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. Dons: Dynamic optimized neighbor selection for smart blockchain networks. *Future Generation Computer Systems*, vol. 130, 75–90, 2022.

[J4] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. *Information Processing & Management*, vol. 58, i. 1, 102393, 2021.

[J5] Hamza Baniata and Attila Kertesz. PriFoB: a Privacy-aware Fog-enhanced Blockchain-based system for Global Accreditation and Credential Verification. *Journal of Network and Computer Applications*, vol. 205, 103440, 2022.

Papers in conference proceedings

[C1] Hamza Baniata, Wesam Almobaideen, and Attila Kertesz. A privacy preserving model for fog-enabled mcc systems using 5g connection. In *Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 223-230, 2020.

[C2] Hamza Baniata. Fog-enhanced blockchain simulation. In *The 12th Conference of PhD Students in Computer Science (CS2)*, University of Szeged. 2020.

[C3] Attila Kertesz and Hamza Baniata. Consistency Analysis of Distributed Ledgers in Fog-enhanced Blockchains.. In *27th International European Conference on Parallel and Distributed Computing (Euro-Par)*, 2021.

[C4] Hamza Baniata and Attila Kertesz. PF-BVM: A Privacy-aware Fog-enhanced Blockchain Validation Mechanism.. In *The 11th International Conference on Cloud Computing and Services Science (CLOSER)* SCITEPRESS, 430-439, 2020.

[C5] Hamza Baniata, Dragi Kimovski, Radu Prodan, and Attila Kertesz. Towards Blockchain-based Smart Systems. In *1st Workshop on Connecting Education and Research Communities for an Innovative Resource Aware Society*, CERCIRAS Cost Action CA19135. 2021.

Further publications

[J6] Hamza Baniata, Sami Mahmood, and Attila Kertesz. Assessing anthropogenic heat flux of public cloud data centers: current and future trends. *PeerJ Computer Science*, vol. 7, e478, 2021.

[J7] Hamza Baniata, Ahmad Sharieh, Sami Mahmood, and Attila Kertesz. GRAFT: A Model for Evaluating Actuator Systems in terms of Force Production. *Sensors*, vol. 20, i. 7, 1894, 2020.

[C6] Hamza Baniata, Pflanzner, T., Feher, Z., & Kertész, A. Latency Assessment of Blockchain-based SSI Applications Utilizing Hyperledger Indy In *CLOSER, 2022, (pp. 264-271)*

[J8] Pflanzner, Tamas, Hamza Baniata, and Attila Kertesz. Latency Analysis of Blockchain-Based SSI Applications. *Future Internet*, VOL(14(10)), 282, 2022.

Chapter 2

Literature Review

In this chapter, I present the state-of-the-art of FC-BC integration with a detailed literature review and classification. I discuss and categorize the related papers according to the year of publication, domain, used algorithms, BC roles, and the placement of the BC in the FC-enhanced architecture. This chapter presents detailed observations, analysis, and open challenges for the BC-FC integration, which justifies the general topic of my dissertation, clarify the general future vision of BC-FC integration applications, and calibrate the research compass towards open issues and highly required directions.

2.1 Introduction

IoT applications researched and applied in the past decade by thousands of researchers and industry specialists, mainly depend on high rates of network response time, and reliability. Meanwhile, such applications require extended storage and computation abilities. This encouraged many to deploy FC for achieving the goals of their proposed IoT systems. In fact, FC is believed to have the major purpose of serving IoT applications at the edge of the network [73]. However, the integration of FC and IoT includes various challenges, such as the security and efficiency of communications. The development of a successful IoT system is usually challenged by Security and Privacy issues, the need of efficient data management schemes, the limitations of device resources (i.e. Memory, Processing power, etc.), Energy consumption, and connectivity into long distances and periods of time [74]. IoT-Cloud integration solved some of these challenges like providing processing power and unlimited storage, leading to have more than five billion devices connected nowadays to the internet on account of IoT [75]. FC, as the extension of CC is expected to solve more issues such as some privacy issues, energy consumption, real-time applications, and connectivity.

Nevertheless, major challenges remained open even when FC is integrated with

IoT, such as the need of efficient data management schemes and Security issues. Also, IoT paradigm itself had branched new similar paradigms serving different purposes. A famous example of that is the Internet of Vehicles (IoV) paradigm, which is only similar to IoT in the general concept, yet different because it serves for different components, goals, standards, and technical solutions. All of the mentioned challenges excited the integration of FC systems with BC technology.

In this chapter, my aim is to provide an assessment of BC deployment in FC environments. In contrast, I aim at highlighting the roles the BC played in such systems, and presenting how the research community visualizes the future BC-FC integration. To get close to my goals, I searched for published papers and surveys whose main topic is BC and FC[1]. I found 8 surveys and 43 articles. Hence, the a total number of papers concerned with my research topic was 51 papers.

I present and discuss these surveys and articles, and classify each according to their year of publication. As demonstrated in Figure 2.1 the first research work discussing the integration of BC and FC was published in 2016. Finally, I conclude my observations and analysis.

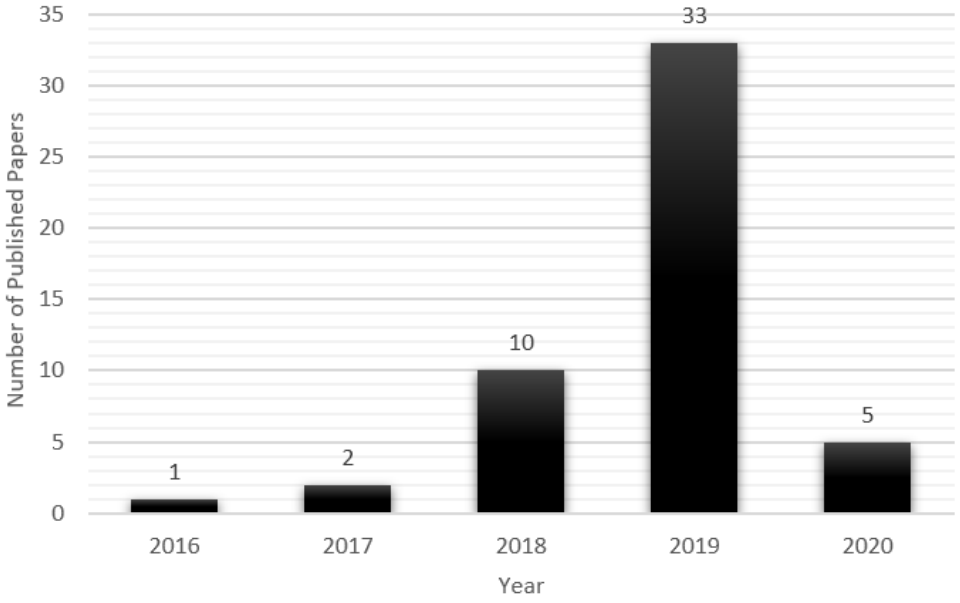


Figure 2.1: *Distribution of published papers according to the year of publication*

[1]papers whose titles include the words ”Blockchain” AND ”Fog”, at: scholar.google.com; last accessed: February-15-2020, and ScienceDirect.com; last accessed: February-16-2020.

2.2 Surveys on Blockchain and Fog Computing Technologies

In this section, I briefly present the surveys I found in the literature discussing BC and FC. Table 2.1 summarizes these surveys. However, the discussion of all papers that survey/propose FC-BC integration was comprehensively presented in this chapter, which provided wider vision of the domain, and hence, deeper understanding, and more generalized observations.

Table 2.1: *Surveys Discussing FC-BC Integration, Including our work*

Ref.	Domain	Year	Aim
[76]	Economy & Politics	2017	Compare Cryptocurrency to national currencies
[77]	CC-BC	2017	Compare Ethereum BC vs. Amazon SWF
[78]	IoT-BC	2018	State-of-the-art Assessment
[79]	FC-BC	2018	Compare Golem, iExec and SONM
[74]	IoT-BC	2018	State-of-the-art Assessment
[80]	Smart environments	2019	Assessment of Smart IoT-BC
[81]	IoT-FC	2019	Security and Privacy
[82]	IoT-FC-BC	2019	Cryptography assisment
[83]	SIoV	2019	Trust factors, challenges, models, and vision
[84]	eHealth-BC	2020	State-of-the-art Identity management systems
[85]	IoT-FC-BC	2020	General concepts (Book Chapter)
This chapter	FC-BC	2020	State-of-the-art Integration Assessment

A brief conceptual research surveying the criteria needed to develop a cryptocurrency system that integrates neuron technologies, artificial intelligence, BC, and FC was presented in [76]. The research mainly focused on the economical aspects rather than technical details, in a step towards understanding the threats, challenges, benefits, and expectations of replacing national currencies with cryptocurrencies.

A comparison of the cost of computation and storage when using Ethereum BC vs. when using Amazon SWF was conducted in [77]. Accordingly, the average cost of executing the same process instance on Ethereum was reported to be two orders of magnitude higher than on Amazon SWF.

Uriarte and De Nicola [79] technically surveyed three ongoing Fog-Based BC projects, namely Golem, iExec and SONM. The survey concluded that even for those three most mature FC-Based BC solutions, they still lack standardization since they are mainly based on ad-hoc communications. The three solutions use Ethereum[2] platform with different properties. The PoCot [22] algorithm was utilized in IExec, while a security deposit is made by providers, just like in the PoS [86].

[2]https://ethereum.org/

Fernández-Caramés and Fraga-Lamas [80] provided a comprehensive study on approaches of smart campuses and universities. The study highlighted main features, communications architectures, BC potential applications, examples, and challenges of smart IoT-FC-CC campus deployment. It was indicated that using traditional database systems provides more efficient latency and energy consumption than using BC, hence, BC deployment is not always the best choice. However, only one out of 13 studied smart campus deployments supported FC, while none of them supported BC.

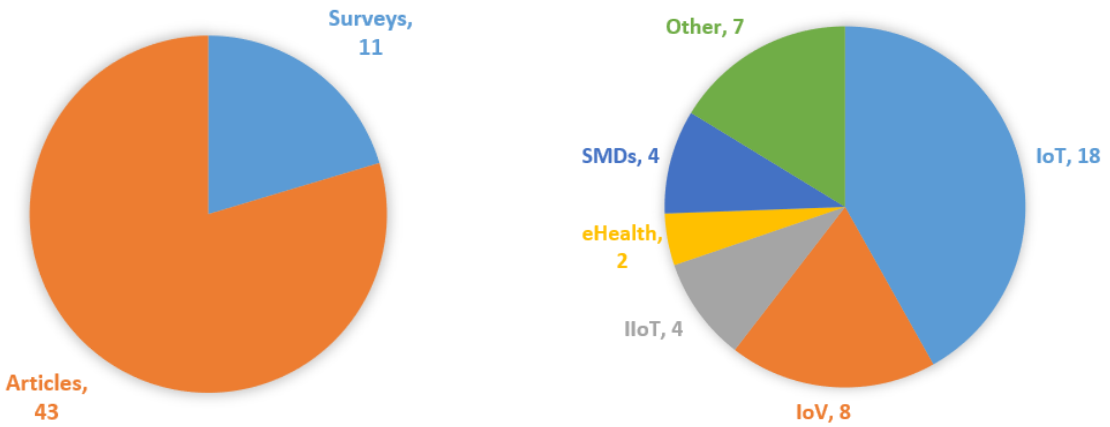
Tariq et al. [81] surveyed potential security and privacy challenges in fog-enabled IoT systems, while they shallowly discussed how BC may enhance such systems.

George and Sankaranarayanan [82] investigated light weight cryptographic solutions that might be suitable for IoT-FC-BC systems. As TXs must be signed in order to be validated, the faster the signing, the faster the system. Accordingly, an experimental comparison evinced that hashing and encoding using ChaCha with EdDSA, instead of SHA-256 with elliptic curve cryptography (ECC), respectively, enhances a fog-based BC system in terms of CPU utilization and number of network transmitted packets.

In [83], trust management models for the Social Internet of Vehicles (SIoV), were surveyed and discussed. In contrast, the authors analyzed the trust factors in such systems, such as the reputation, the environment, system expectations and goals, etc. The challenges faced by a trust management system in SIoV systems, such as the privacy, the heterogeneity, mobility, and Quality of Service (QoS), were also analyzed. The survey also reviewed existing trust models, and trending solutions to solve the challenges faced by such models, e.g. how BC and FC can boost the development of trusted SIoV systems. Accordingly, they presented and discussed an envisioned future SIoV network when enhanced by BC and FC.

Bouras et al. [84] surveyed decentralized BC-based identity management systems, and the possible scenarios of adopting such systems to enhance health-care applications. Alli and Fahadi [85] presented the basic concepts of IoT, FC, BC, the FC-BC general deployment frameworks, opportunities, and challenges. They clarified how the decentralization property of BC can be applied at the device level, the fog level, or the cloud level, and briefly discussed some deployable CAs.

All presented surveys came to an agreement on the advantages of the BC-FC integration, which include enhanced security, integrity, reliability, fault-tolerance, and credibility, thanks to the distribution of processing units of IoT and FC, and the decentralization and trust management mechanisms deployed within the BC mechanisms. On the other hand, such combination of different technologies suggested agreed on challenges, such as privacy, latency, legalization, and standardization issues.



(a) *Publications classification* **(b)** *Article classification*

Figure 2.2: *Classification of found publications into Surveys and Articles (a), and of Articles according to their domain (b)*

2.3 Blockchain and Fog Computing Integration Solutions

In this section, I present and discuss the remaining 43 articles that proposed systems to benefit from the advantages of BC-FC integration, or proposing solutions for different challenges faced by the FC-BC integration. Having analyzed these papers, I found that most of the papers discuss solutions for IoT-FC-BC integration. This may be caused by the fact that FC was initially introduced to specifically rise and enhance IoT applications. However, I found that other papers discuss FC-BC integration when deployed in different environments, such as Smart Mobile Devices (SMDs), Internet of Vehicles (IoV), Industrial Internet of Things (IIoT), and eHealth. Figure 2.2 depicts my a) classification of found publications into Surveys and Articles and b) classification of Articles according to their domain. Table 2.2 concludes those articles regarding their domain, the role that the deployed BC played, the used CA, the layer of FC where BC was deployed (as FC architecture's default definition clarifies it has three main layers [87]), and the properties/challenges that the corresponding article had enhanced. In Table 2.2: D, I, T, R, E, F, C, and O notations stand for: Data Storage Management, Identity/Authentication Management, Trading/Payment Management, Rating/Reputation Management, End-User layer, Fog Layer, Cloud Layer, and Other purpose/layer, respectively. In the following sub-sections, I discuss the articles according to the domain categorization presented in Figure 2.2b.

2.3.1 Internet of Things Applications (IoT)

Similarly to the SaaS and PaaS paradigms provided by the cloud, Samaniego and Deters [2] proposed Blockchain-as-a-Service in FC-IoT systems, for which finding the hosting environment was declared as the biggest challenge. That is, the Things are, by definition, resource and energy limited. On the other hand, hosting the BC in the cloud increases the latency, which was an original drawback of CC that FC was proposed to solve. Hence, the authors were only left with the option of hosting the BC in the fog layer, which was experimentally proven to be the best choice.

Sharma et al. [88] proposed a distributed cloud architecture based on BC technology with Proof-of-Service (PoSER) consensus, in order to speed up the processing of large amounts of IoT data. In this architecture, fog nodes provides the computation capabilities while the cloud layer plays the controlling and monitoring role. Computational and storage tasks are handled by fog nodes as long as they are able to, otherwise they are offloaded to the cloud. This, obviously, may increase the latency and resource consumption. The BC is deployed in the cloud layer to allow users to select/award the service provider, and to enhance the transparency of the cloud reputation regarding provided services. Contributions, such as the performance of a computation, the transfer/storage of a file, are registered into the BC, hence providing a proof of provided service. Jung et al. [89] proposed mapping the identity of Things to the IP address of the gateway they are connected to, and save these data in a BC. Accordingly, no sybil or spoofing attacks occurs, nor does a single point of failure, which are issues that may occur when using a classical database.

Xiong et al. [90] investigated deploying cloud/fog resources as computing power of the PoW based BC miners. That is, exhaustive puzzle solving computations being offloaded to the cloud instead of being locally solved, if the profit was maximized. Almadhoun et al. [91] proposed a TTP-free BC-FC authentication scheme using Ethereum platform, aiming to control the remote access to Things in IoT systems. The fog here is responsible for the storage, computing, and access management tasks on behalf of a group of IoT resource-poor devices. The access authority, granted for users, is configured by system administrators at the initialization phase of the system, and can be updated with time. Davcev et al. [92] presented their BC-based IoT-Cloud supply chain system, whose main purpose is evaluating quality metrics for agriculture and food production. BC was deployed as a trusted database to collect and save sensed data.

Table 2.2: *Blockchain Deployment in FC-BC Integration*

Domain	Ref.	Blockchain Role	Algorithm	FC Layer	Enhanced Property/Solved Challenge
IoT:	[2]	D, I, T, & R	N/A	F	Latency & Standardization
	[88]	R	PoSer	C	Latency & Optimization
	[89]	I	N/A	F	Privacy & Security
	[90]	T	PoW	C	Energy Consumption
	[91]	I	PoW, keccak256	C	Privacy
	[92]	D	N/A	C	Security
	[93]	T & I	PoW	F & C	Privacy & Security
	[94]	D	PoC	C	Security & Standardization
	[95]	D, I, T, & R	PoC	F & C	Privacy & Security
	[96]	D	PoW	F & C	Standardization
	[75]	D	PoW	F	Security & Standardization for DTB apps
	[97]	D	N/A	C	N/A
	[98]	D	N/A	C	N/A
	[99]	O	N/A	F	Security
	[100]	D & R	N/A	C	Reliability & Credibility
	[101]	D	PoW	C	High efficiency using dockerized components
	[9]	R	PoW	F & C	Security, Privacy, & credibility
	[102]	R & I	PoA	F	Trust
SMDs:	[103]	D & I	PoC	F	Mobility
	[104]	D & T	PoW	E & C	Offloading Computations to the Fog
	[105]	T	PoW	C	Resource efficiency
	[106]	D	BFT	C	Standardization & access control
IoV:	[6]	D	PoS	F	Privacy
	[107]	D	PoS	C	Heterogeneity & Privacy
	[108]	D	PoET	F & C	Latency
	[109]	D, I & R	PoW & PBFT	F & C	Inter-operation & Mobility of VANETs
	[110]	D	PoW	C	Security
	[111]	D & T	zk-SNARKs, DAP	C	Decentralization and Anonymity
	[112]	D & I	PBFT	F & C	Privacy & Security
	[113]	D & I	PBFT	F & C	Security & Network Overhead
eHealth:	[114]	D	N/A	F	Security
	[115]	T	PoW & PoA	C	Latency & Security
IIoT:	[116]	D, T & I	PoW & PoA	C	Trust & Optimization
	[117]	D	HLF	F & C	Security
	[118]	D	PoW OR PoS, OR PoA	C	Heterogeneity & Resource efficiency
	[119]	T	PoC	C	Trust
Other:	[12]	I	PoC	F	Security
	[120]	I	PoS	O	Privacy & Security
	[121]	I	PoAh	C	Privacy
	[122]	O	PoW	O	Energy & Memory Consumption
	[123]	T	PoC	C	Flexible Heterogeneous deployment
	[124]	D & I	PBFT	C	Security
	[125]	T	PoW	F & C	QoS & Security

Gu et al. [93] proposed CrowdChain, a Fog-assisted BC-based crowd sensing framework, where BC is deployed for performing payments/rewards, and record the identities in a chain located in the fog. Proof-of-Concept (PoC) [20] was used by [94] in the MultiChain[3] platform.

[3]https://www.multichain.com/

Ziegler et al. [95] proposed the concept of Plasma BCs which suggests various BCs be dedicated for different purposes in the same system, each is a parent or a child of other chain. Clearly, such concept fits perfectly, if well implemented, in a FC architecture, where fog nodes manage edge devices, and are managed by fog/cloud servers. The proposed integration enhances the security and privacy as parents and children are only aware of the least information about each other.

Tuli et al. [96] proposed an integration framework for IoT-Fog-Cloud infrastructures, namely FogBus, wherein a Java implemented PoW-based BC is supported for applications requiring high data integrity. Introducing a real-world case study, different criteria of FogBus were measured resulting higher latency, network usage, and energy consumption when Blockchian is used instead of regular database. Such results agree with the results presented in [80], and explains the results in [77].

Memon et al. [75] proposed virtual segregation of the fog layer into two clusters. One behaves similarly as a middle layer between Things and the cloud, while the other is dedicated to BC-related tasks. Here, the latency effect, caused by the addition of a PoW-based BC, was deeply discussed. The authors concluded that out of the three main categories of IoT-FC applications (i.e. Real-Time, Non-Real-Time, and Delay-Tolerant BC applications), only Delay-Tolerant applications are advised to deploy a BC.

Muthanna et al. [97] and El Kafhali et al. [98] proposed frameworks for IoT-FC systems controlled and managed by SDN networks. BC was added to these frameworks only as a structural component and was not integrated nor simulated within the proposed frameworks.

Saputro and Sari [99] proposed a multi-fog BC model to increase the availability in the model proposed in [126]. The proposal was to mainly distribute the tasks to different FC domains in order to decrease the probability of security attacks. Wang et al. [100] proposed utilizing BC to hold information about the contributed resources by fog nodes. The BC presents a log of satisfaction of system components by fog nodes; i.e. the more completed tasks and contributed resources fog nodes provide, the higher the satisfaction degree, and hence the more profit for the fog nodes. Holste et al. [101] presented an intuitive bench framework aiming to enable easy design of software, namely VarOps. The proposed framework considers the variability property, which makes it possible to re-use docker components, and hence, increase the efficiency of new proposed solutions. The proposed framework deploys BC as a data management controller, SCs for validating requests, and exemplifies with realistic cases.

Debe et al. [9] proposed a reputation system for fog nodes delivering services to the IoT devices, using BC Ethereum SCs. The system suggests that IoT devices rate fog nodes according to specific criteria. Fog nodes obtain, accordingly, trustworthiness value that would indicate how reliable they are. Meanwhile, IoT devices'

credibility is also computed, according to specific contributions, for the more credible the IoT device, the more effective its evaluation is on the final score of evaluated fog nodes. On the same topic, yet on the contrary, Cinque et al. [102] proposed a BC-Based Trust Management model in which chains in the fog layer store the trustworthiness values of network entities according to given criteria, and store entities' real identifiers. However, this trust management model requires the IoT device to be connected to at least three fog nodes in order to forbid faulty responses from a probable malicious fog node, which might be considered a drawback.

2.3.2 Smart Mobile Devices Applications (SMDs)

Sharma et al. [103] proposed a BC-FC Distributed Mobility Management handover scheme. The solution focused on the resolution of hierarchical security issues without affecting the network layout. The proposed scheme deployed three different BCs for handover attempts, mobility anchors and system entities' information, respectively. Tang et al. [104] used a Spacial-Temporal Database, a PoW-based BC, and digital token for rewarding fog entities. Each mobile device in this solution maintains a local BC, which suggests high energy and storage consumption by the resource-limited end mobile devices.

Jiao et al. [105] proposed an auction mechanism for offloading computations, such as PoW puzzle solving tasks, from resource-poor BC miners, such as SMDs, to the fog or the cloud. The allocation of computing resources to miners was shown to be computationally efficient. Podsevalov et al. [106] presented their BC-FC based Corda distributed ledger platform[4]. Here, groups of vacuum cleaners were connected to Raspberry Pi nodes, where maps of cleaned areas were transmitted. The maps then were transmitted to the Corda platform, representing the cloud, in which data are processed and saved on the BC. Finally, the user monitors and controls the system through a web-server software.

2.3.3 Internet of Vehicles Applications (IoV)

Li et al. [6] proposed a privacy-preserving BC-assisted Fog-Cloud carpooling scheme, where BC is deployed for data management. The deployed BC in uses the PoS algorithm for clients selection, and only stores the hash values of encrypted carpooling data, while the actual data are saved on the cloud. Fog nodes on the other hand are deployed for collecting real-time carpooling queries, and for matching passengers with drivers.

Li et al. [107] proposed the integration of IoV with FC and BC in a system where drivers from different service providers can be paired with riders. Such proposal

[4]https://www.corda.net/

makes it possible to provide more consumption of the service, and hence more revenues. Meanwhile, the privacy of users is preserved by anonymous authentication scheme, and BC is deployed for recording rides and creating SCs to pair riders with drivers. Bonadio et al. [108] deployed an alternative of the PoW algorithm, similar to the recently proposed PoET CA [127]. To do so, authors investigated classical epidemic flooding based, network coding inspired, and chord protocols. The system has been tested using the OMNeT++ framework to achieve the needed results of reacting on traffic anomalous conditions. The BC was deployed here as a log of past TXs related to a specific important incident that needs to be kept unchanged, such as the occurrence of an accident.

Gao et al. [109] analyzed the BC-SDN integration for effective operation of Vehicular Ad-hoc Networks (VANETs) in 5G and FC paradigms. The BC was deployed here for authentication, access control, data management, reputation management (through a proposed trust model), and policy enforcement (using SCs). FC on the other hand, was deployed to enhance the handover problems in such high mobility environment. Nadeem et al. [110] suggested a distributed PoW-based BC architecture for securing VANETs, where the BC keeps a record of services, provided by different cloud providers. Meanwhile, FC is deployed for connecting the vehicles directly to the BC.

Ou et al. [111] proposed a BC-based IoV data transaction scheme, where BC is deployed for payment purposes. The proposed scheme allows data consumers to anonymously get data they need and pay for the service, using asymmetric encryption and SCs. The full anonymity in this scheme was guaranteed by using a Decentralized Anonymous Bitcoin Payment scheme, which is a part of the ZeroCash proposal [128].

Yao et al. [112] proposed a BC-assisted authentication for distributed Vehicular Fog Services. A consortium, permissioned, semi-decentralized BC model, in which selected group of nodes are responsible for block validation, and PBFT CA [129], were adopted. Pseudonyms were used in this mechanism to guarantee the anonymity, as with each authentication a new pseudonym is generated by the client vehicle itself. However, BC is not deployed for keeping authentication keys, but for storing authentication results, while the keys are generated in a corporation with TTP.

Kaur et al. [113] deployed ECC in a BC-based IoV authentication and key-exchange scheme, where PBFT-based BC was deployed for maintaining network information, and ECC was deployed for the actual authentication. The proposed scheme was compared with [112], and was found more efficient in terms of computational and communications overhead, and was validated in terms of security and safety using the AVISPA tool [130].

2.3.4 e-Health Applications

Islam et al. [114] proposed a BC-based human activity monitoring framework for eHealth applications without declaring the properties, or deployment methodologies of the used BC. Fernández-Caramés et al. [115] proposed a BC-IoT system that monitors Glucose levels for Diabetes patients. Their proposed system takes advantage of the low latency of computations offered by FC for mobile sensors, which is highly beneficial in emergency situations. Meanwhile, the BC was deployed to incentivize patients for sharing their private health information, and to allow them to securely and privately buy medical equipment. In this solution, BC is built using a metacoin called GlucoCoin, and the system was evaluated by having it run on two different Ethereum testnets, namely Rinkeby with PoA CA and Ropsten with PoW CA.

2.3.5 Industrial Internet of Things Applications (IIoT)

Seitz et al. [116] proposed a BC-based Industrial Internet of Things (IIoT) Bazaar, using Ethereum and PoA. The main goal of this Bazaar idea was to provide a marketplace for IIoT applications based on the technologies of FC, BC, and Augmented Reality. BC was deployed for performing trusted payment transactions, trusted authentication for consumers and providers, and application data storage.

Jang et al. [117] studied how to integrate BC and FC in a smart factory environment. Accordingly, they proposed an IoT-Fog-Cloud system architecture where the cloud and fog nodes act as BC nodes. The main usage of the BC was to record and register the TXs performed between the three layers of the system.

Lallas et al. [118] studied the deployment of BC in supply chain network. The BC in this study replaced regular databases to save data generated by Things and fog nodes, and the decisions made by the cloud. This replacement was theoretically shown to be beneficial for connecting highly-heterogeneous resources within the network. In [119] a Trust Management architecture for a CCTV system in FC platforms, was proposed. In this architecture, BC was basically deployed for collecting payments using a proposed smart AI protocol.

2.3.6 Other Applications

Zhu and Badr [12] proposed a BC-enhanced FC security architecture, namely FOCUS, where the BC was deployed as an identity management ledger by recording users and organizations identities.

Huang et al. [120] surveyed the SC protocols and proposed three-party TTP-Free BC-based SC signing protocol in FC environments. The BC role was to guarantee all signing parties would eventually reveal their signature to avoid a penalty. Deep et al. [121] proposed a BC-based authentication mechanism aiming to forbid cloud insider

attacks that may actively manipulate, or passively disclose, private clients' data. Using this system, data was saved regularly at the cloud, but can only be disclosed by authenticated users. credentials are saved on BC while any entry to the data shall be preceded by a proof of authentication (PoAh) [131]. The proposed mechanism was proven mathematically and experimentally optimal against insider data manipulation. Kumar et al. [122] proposed a statistical method to solve the puzzle in the PoW algorithm using the expectation maximization algorithm and polynomial matrix factorization. The proposed method achieves the puzzle solution with less iterations, leading to less required time, energy, and memory consumption.

Savi et al. [123] presented their initial work results on the DECENTER project[5]. The showcase deployed Ethereum BC for payment orders, while using a PoC CA. The project aims to help users extend their infrastructures, and easily get access to private computational resources using FC through simple GUI. Alshehri and Panda [124] suggested adding the BC to their previously-proposed approach in [132], for protecting fog-enabled systems from malicious nodes. PBFT-based BC with a TTP was deployed in this approach for data management and access control. The approach was not tested nor simulated as the authors considered it as their future research direction.

Debe et al. [125] proposed a BC cryptocurrency-based payment system for the provided public FC services. In this approach, fog nodes provide computation and storage services for end-users, while end-users pay for the provided services, depending on the QoS and satisfaction level using Ethereum platform. FC service providers, and end-users, are evaluated by the reputation system presented in [9].

2.4 Concluding Remarks

Observations

Summarizing all previously discussed works, I found that the following key observations can be made:

1. BC can highly enhance FC systems in terms of security, reliability, and decentralization. On the other hand, deploying BC in FC systems is costly (compared to non-BC-based solutions) in terms of Money, Energy, and Latency. Hence, systems that require lower costs, lower RTT or lower energy consumption, should not use the BC technology.

2. Most of BC-FC integration solutions were proposed for IoT and related applications, such as IoV, IIoT, and eHealth applications.

[5] https://www.decenter-project.eu/

3. As clarified in Figure 2.3, most BC-FC integration solutions deployed BC for Data Management purposes, as a more reliable alternative of a classical Database.

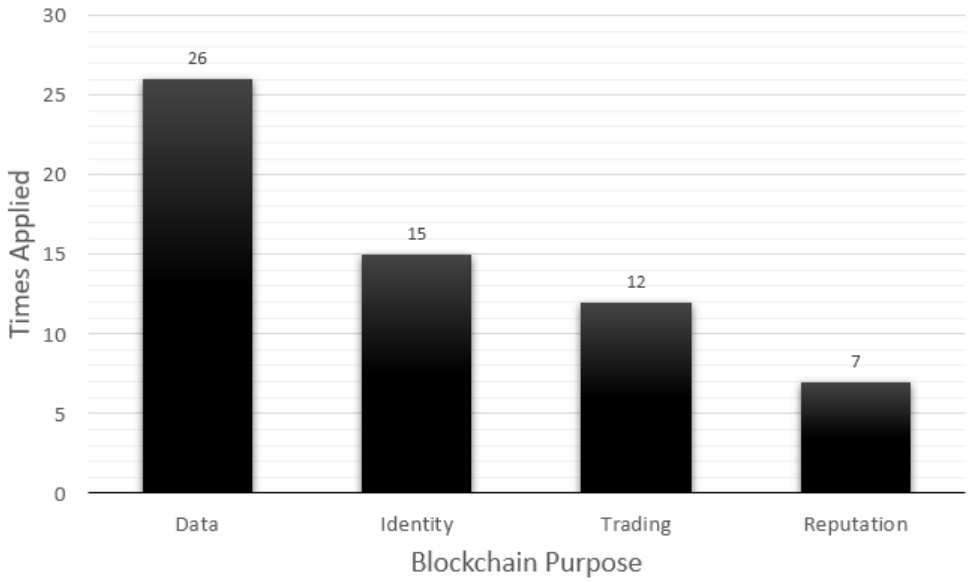


Figure 2.3: *Usage of BC in FC environments.*

4. The vast majority of BC-FC integration approaches used Proof-based algorithms. Most of these deployed a variation of PoW-based CA, Despite the fact that PoW-based BCs are the highest energy consuming compared to other algorithms. Exact statistics are provided in Figure 2.4.

5. Unless clearly defined, I assumed that BC is deployed in the cloud layer when it is used for Payment/Trading purposes. Following this assumption, most BC-FC integration solutions deployed the BC in the cloud layer. This is clarified in Figure 2.5.

6. At the time of writing, BC has not yet been implemented in SMDs, IIoT, or eHealth applications for Reputation management purposes. Also, BC has not yet been implemented in eHealth applications for Identity management purposes.

I tried to find a correlations between the role of the deployed BC and the used algorithm, or between the role of the deployed BC and the layer where the BC is deployed, but unfortunately I could not find any relation. Next, I conclude the enhanced properties of an FC-BC integrated solution, referring to the results presented within all the studied proposals.

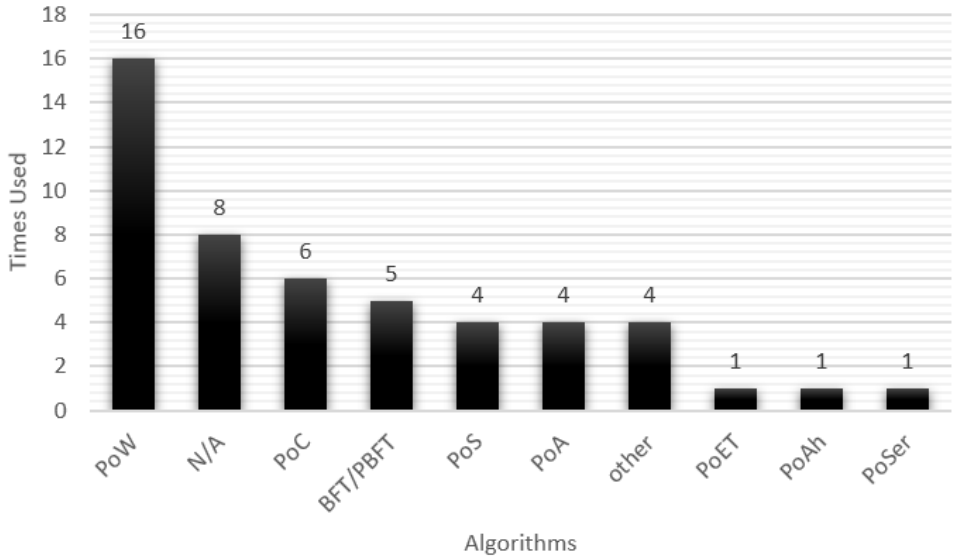


Figure 2.4: *Usage of different consensus algorithms in FC environments.*

Enhanced Properties:

1. Optimization: Several solutions proposed models or protocols that enhance the output of the system. The reached optimization solutions are mostly local, yet outperforms other solutions.

2. Security: Several articles enhanced the security using the BC instead of regular databases. The deployment of BC was highly recommended in systems that rely on the integrity and accountability.

3. Reliability & Credibility: Deploying the BC in the cloud made the proposed systems highly reliable, and almost impossible for its database to be altered by any party, especially when using the PoW algorithm. This criterion motivated many researchers to deploy the BC in the cloud.

4. Resource efficiency: The best placement of the BC in a FC architecture is the fog layer. This is because of decreasing the usage of the virtual resources in the cloud, hence decreasing the cost and latency, and increasing the QoS. However, deploying the BC increases the latency in most scenarios, hence, the balance of BC latency, database latency, cloud latency, shall be individually studied for each case.

5. Access control: Deploying the BC for controlling the authentication was proposed in several articles. This deployment made it nearly impossible to access information without the correct permission.

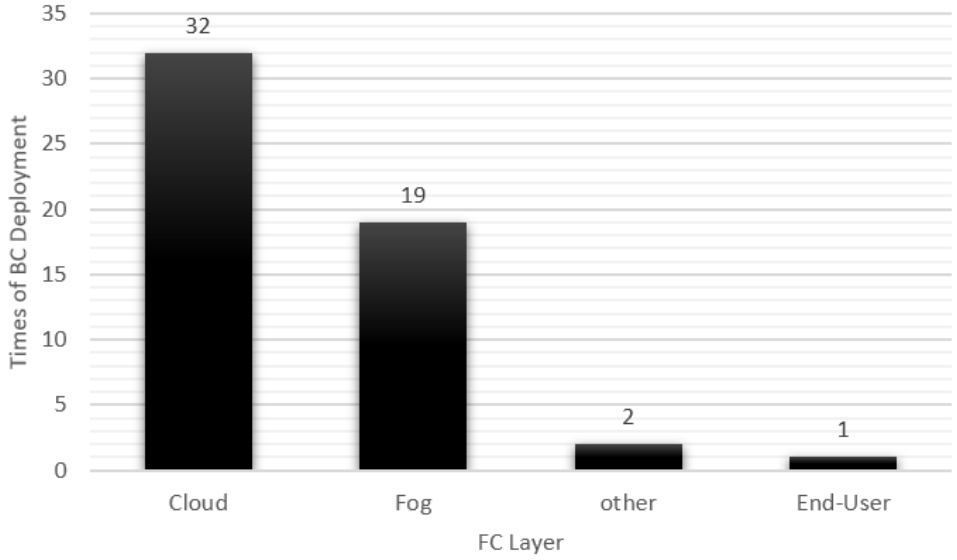


Figure 2.5: *Placement of BC in the Proposed BC-FC solutions.*

6. Decentralization: This property was shown to be highly beneficial in many applications. The BC fulfils the needed criteria to cope up with the decentralized fog/cloud, hence the successful deployment of BC in FC was shown to be beneficial, applicable, and practical.

7. Anonymity: As this is an important success factor for applications that require high levels of privacy, several articles deployed BC for obtaining anonymity of clients while using public systems. This is achieved in BC by the the deployment of asymmetric encryption, and CAs that do not functionally require TTPs.

Although the BC deployment has several advantages, there were several challenges faced by researchers and developers of different BC-FC integrated solutions. These challenges include the lack of standardized methods, informally guaranteed privacy, higher latency compared to centralized solutions, higher energy consumption, and social mistrust of BC-based solutions due to the technology being juvenile. Additionally, some applications in the IoV and the eHealth domains required highly adaptive mobility controls, due to the continuous movement of clients, which had been usually solved by deploying FC mechanisms. Specifically, some articles approached some enhancement of the mobility handling while deploying BC, yet this negatively affected other criteria, like latency and privacy. Finally, as cryptocurrency concepts are still not accepted nor legalized in many countries around the world, BC technology is ignorantly illegal as well.

In this chapter, I have discussed how BC can be deployed for different reasons than digital money, such knowledge needs to be globally provided that BC is not

the same as digital money, yet it is the backbone of it. Having such technology being illegal leads to falling behind the global technological trends, hence, makes it a challenge for the adoption of BC-based solutions.

I made the the following contributions presented in this chapter:

I/1. I performed a comprehensive literature review related to integrated FC and BC solutions, tools and applications.

I/2. I categorized the studied papers according to the year of publication, domain, used algorithms, BC roles, and the placement of BCs in FC-enhanced solutions.

I/3. I provided concluding observations, characteristics and challenges of BC-FC integrated solutions.

Chapter 3

FoBSim: a simulation tool for integrated Fog-Blockchain systems

As hundreds of integrated BC-FC projects, ideas, and systems were proposed, one can find a great R&D potential for integrating those technologies. Examples of organizations contributing to the R&D of these two technologies, and their integration, include Linux, IBM, Google, Microsoft, and others. To validate an integrated BC-FC protocol or method implementation, before the deployment phase, a suitable and accurate simulation environment is needed. Such validation should save a great deal of costs and efforts on researchers and companies adopting this integration. Current available simulation environments facilitate FC, or BC simulation, but not both. This chapter presents my proposed solution to solve this issue.

3.1 Introduction

In light of the general tendency towards skepticism around BC systems being reliable, huge research and industrial projects are being encouraged to address issues and vulnerabilities of those systems. This is because it is believed that a successful BC deployment would definitely advance the Internet-of-Everything (IoE) applications. Dubai, for example, has been planning for becoming the first smart city powered by BC [133]. China had launched, in late 2019, a BC-based smart city ID system [134], while it is planning to have its own official digital currency [135]. Before that, Liberstad, a private smart city in Norway, has officially adopted City Coin as its official currency [1].

The research and industry communities have been working hand-in-hand to solve the major challenges discussed earlier, along with other technical issues. Such efforts require reliable and flexible simulation environments that can mimic real-life

[1] https://www.liberstad.com/

scenarios with the lowest possible costs. Old, out-dated, or somewhat limited simulation tools that were initially implemented for classical Peer-to-Peer networks, such as PeerSim [136], may not be able to cover all the mechanisms of a modern BC system. Although some recently proposed systems use PeerSim, such as [137], it surly required vast amount of changes, modifications, and additions to redesign it into a BC simulation tool.

In this chapter, I propose FoBSim, a BC-FC simulation environment that is able to simulate different integration scenarios of FC and BC technologies. This chapter presents the modules, algorithms, and strategies implemented in FoBSim. It also describes in detail the validation, the incentivization, and the confirmation mechanisms deployed in the current version of FoBSim. To exemplify its utilization, a discussion regarding possible application scenarios of FC-BC integration is provided. Furthermore, a clarification on how such applications can be simulated and optimized using FoBSim is provided. The main properties of the current version of FoBSim includes:

1. FoBSim provides different CAs, namely PoW, PoS, and PoA that are ready to be deployed in any scenario.

2. FoBSim facilitates the deployment of BC miners in the fog or end-user layer.

3. FoBSim allows different services to be reliably provided by the BC network, namely Data Management, Identity Management, Computational Services (through SCs), and Payment/Currency transfer Services.

4. FoBSim provides both, parallel execution and non-parallel execution, of mining processing. While gossiping is optionally and efficiently available, leading to higher consistency in different possible network topologies.

5. FoBSim is the first simulation environment whose primary goal is to mimic integration scenarios of FC and BC technologies.

FoBSim allows to choose the suitable CA according to the simulated scenario. While there are many versions of each CA mentioned, I currently provide the simplest version of each so that modifications can be performed with no complexities. To obtain more information about them, however, more detailed information can be found at [127, 138, 139].

In FoBSim, the fog layer can be configured according to the scenario that needs to be simulated. For example, the number of fog nodes, the communications within the fog layer and with other entities of the simulated system, and the services provided by the fog, can all be modified.

In FoBSim, the BC network can provide two models of services; namely data storage, and computations. Meanwhile, the communications within the BC network and with the fog layer are configurable. Data storage service model implies that

pieces of data are saved on the immutable distributed ledger. Such data may be of any type including data records, IDs, digital payment registration, or reputation measures of end-users or fog components. It can also be noted that some applications require assets to be transferred between clients, such as cryptocurrency transfer applications or real estate ownership applications. Other applications do not require transferring assets rather than saving data on the chain only, such as voting applications and eHealth applications.

Performing computations for end-users is the second service model that the BC in FoBSim can be configured to provide. That is, computational tasks can be sent by end-users/fog entities to the BC in the form of SCs, which are small chunks of code, run by BC nodes upon fulfillment of algorithmically verifiable conditions [18]. After running the SCs, the results can be saved in a centralized or decentralized form according to the pre-run configuration.

As previously discussed, Figure 1.2 clarifies how the services, classically provided by a Cloud/fog system, can be interpreted into the form of services that can be provided by a BC system. In such complex systems, FoBSim can be easily extended by adding the needed classes and modules and, hence, cover necessary proposed scenario entities. Note it is important to differentiate between services provided by fog nodes who are BC nodes, and the services provided by fog nodes who are not BC nodes. The first type gets incentivized by end-users for providing both fog services and BC services, while the second type gets incentivized by end-users for providing only fog services. Such confusions need to be taken care of, when simulating FC-BC scenarios, to maximize the reliability of the obtained results.

The remainder of the chapter is organized as follows: Section 3.2 presents and discusses the state-of-the-art simulation environments that might be suitable to simulate FC-BC systems. Referring to previously discussed FC-BC solutions and paradigms, I present the proposed components, algorithms, and functions of FoBSim in Section 3.3. To validate FoBSim, I perform the simulation of several use cases and present the simulation results in Section 3.4, after which I conclude the chapter.

3.2 Related Works

To the best of my knowledge, there is no previous work that was specifically implemented for simulating FC-BC integration scenarios. Based on the extensive literature search I made for tools, I found several simulation tools that mimic FC-enhanced cloud systems, IoT-FC-Cloud scenarios, etc., and several tools that mimic BC scenarios, each with specific constraints on the used CAs. For example, the ABSOLUT tool, investigated in [140], models the deployment of BCs in IoT environments. Accordingly, some critical analysis were provided regarding network latency, effects of miners number on the overall efficiency of the IoT network, and simulation errors.

Liaskos et al. [141] proposed a general architecture that a BC simulation needs to follow in order to be considered comprehensive. Furthermore, some properties were declared as necessary for encouraging the adoption and re-usability of the tool. The proposed architecture includes extensible connection strategies, BC nodes, BC chains, TXs and Transaction pools, users, events, Blocks, and most importantly CAs. Events can include different triggers to other events - that may be performed by any entity of the network (such as TX or block arrival, TX or block validation, connection requests, etc.). Events need also to be handled by concise and well implemented strategies.

In a project that targets FC-BC integration applications, general-purpose FC simulators are typically used, where the BC is only implemented as an application case, such as in [142]. The results of such simulation approach can be trusted valid for limited cases, such as providing a proof of concept of the proposal. However, critical issues, such as scalability and heterogeneity in huge networks, need to be simulated in a more specialized simulation environments. To mention one critical case, the BC protocols, deployed in different CAs, require more precise and accurate deployment of the BC entities. Additionally, inter-operation in different layers of a FC-enhanced IoT-Cloud paradigm is rather critical. As some simulation scenarios need an event-driven implementation, while others need a data-driven implementation, a scenario outputs may differ when simulated using different simulation environments. Such possibility of fluctuated simulation outputs should normally lead to unreliable simulation results.

In light of the lack of simulation tools similar to my proposed tool, I found it more suitable to discuss related works in two separate subsections, i.e. FC simulation tools, and BC simulation tools.

3.2.1 FC simulation tools

The state-of-the-art in cloud, IoT and fog simulation tools was analyzed in [143]. Within this study, several simulation tools were classified, compared, and analyzed, such as the DockerSim tool [144], FogNetSim++ [145], and EdgeCloudSim [146]. Furthermore, technical details, advantages, vulnerabilities, and software quality issues were also addressed.

Rahman et al. [147] surveyed 15 simulation tools for cloud and data centers networks scenarios. Out of those 15 presented simulation tools, seven were defined as extensions of the CloudSim toolkit [148]. The tools were discussed and compared according to criteria such as GUI availability, implementation language, and communications model. Consequently, they proposed the Nutshell tool which addresses some drawbacks that were ignored by most of the surveyed simulators, such as the abstract network implementation, missing low-level details, and lack of addressing

schemes, congestion control mechanisms, or traffic pattern recognition mechanisms.

Yousefpour et al. [87] presented an extended survey, in which some FC simulation tools, such as iFogSim [149, 150], Emufog [151], Fogbed [152], and MyiFogSim [153] were discussed. As iFogSim was conceptually built using the CloudSim communications model, it inherited some of its properties, such as the ability to co-execute multiple tasks at the same time and the availability of plugable resource management policies.

Generally speaking, any cloud simulation tool can be extended to be a FC-enabled simulation tool. This is because of the fundamental property of the fog layer acting as a bridge between end-users and the cloud.

3.2.2 BC simulation tools

Anilkumar et al. [154] have compared different available simulation platforms specifically mimicking the Ethereum BC, namely Remix Ethereum [155], Truffle Suite [156], Mist [157], and Geth [158]. The comparison included some guidelines and properties such as the initialization and the ease of deployment. The authors concluded that truffle suite is ideal for testing and development, Remix is ideal for compilation and error detection and correction, while Mist and Geth are relatively easy to deploy. Alharby and van Moorsel [159] and Faria and Correia [160] proposed a somewhat limited simulation tool, namely BlockSim, implemented in Python, which specifically deploys the PoW algorithm to mimic the Bitcoin and Ethereum systems. Similarly, Wang et al. [161] proposed a simulation model to evaluate what is named Quality of Blockchain (QoB). The proposed model targeted only the PoW-based systems aiming to evaluate the effect on changing different parameters of the simulated scenarios on the QoB. For example, average block size, number of TXs per block/day, the size of the memPool, etc. affecting the latency measurements. Furthermore, the authors identified five main characteristics that must be available in any BC simulation tool, namely the ability to scale through time, broadcast and multi-cast messages through the network, be Event-Driven, so that miners can act on received messages while working on other BC-related tasks, process messages in parallel, and handle concurrency issues.

Gervais et al. [162] analyzed some of the probable attacks and vulnerabilities of PoW-based BCs through emulating the conditions in such systems. Sub-consequently, they categorized the parameters affecting the emulation into consensus-related, such as block distribution time, mining power, and the distribution of the miners, and network-related parameters, such as the block size distribution, the number of reachable network nodes, and the distribution of those nodes. However, they basically presented a quantitative framework to objectively compare PoW-based BCs rather than providing a general-purpose simulation tool.

Table 3.1: *Blockchain simulation tools and their properties*

Ref.	PL	PoW	PoS	PoA	SC	DM	PM	IDM	FC-enhanced
[159, 160]	Python	✓	χ	χ	✓	χ	✓	χ	χ
[161]	Python	✓	χ	χ	χ	χ	✓	χ	χ
[163]	Java	✓	χ	χ	✓	χ	χ	χ	χ
[164]	Python	✓	✓	χ	χ	✓	χ	χ	χ
[165]	Python	χ	χ	χ	χ	χ	✓	χ	χ
[166]	Java	✓	χ	χ	✓	χ	✓	χ	χ
FoBSim	Python	✓	✓	✓	✓	✓	✓	✓	✓

Memon et al. [163] simulated the mining process in PoW-based BC using the Queuing Theory, aiming to provide statistics on those, and similar systems. Zhao et al. [164] simulated a BC system for specifically validating their proposed Proof-of-Generation (PoG) algorithm. Hence, the implementation objective was comparing the PoG with other CAs such as PoW and PoS. Another limited BC implementation was proposed by Piriou et al. in [165], where only the blocks appending and broadcasting aspects are considered. The tool was implemented using Python, and it aimed at performing Monte Carlo simulations to obtain probabilistic results on consistency and ability to discard double-spending attacks of BC protocols. In [166], the eVIBES simulation was presented, which is a configurable simulation framework for gaining empirical insights into the dynamic properties of PoW-based Ethereum BCs. However, the PoW computations were excluded in eVIBES, and the last updates on the code were committed in 2018.

Table 3.1 highlights a comparison between the mentioned BC simulation tools and the proposed FoBSim tool. PL, DM, PM, and IDM are abbreviations for Programming Language, Data Management, Payment Management, and Identity Management, respectively. As shown in the table, none of the previously proposed BC simulation tools made the PoA algorithm available for simulation scenarios, provided a suitable simulation environment for identity management applications, or, most importantly, facilitated the integration simulation of FC and BC technologies.

Many other references can be found in the literature, in which a part of a BC system, or a specific mechanism is implemented. The simulated 'part' is only used to analyze a specific property in strict conditions, or to validate a proposed technique or mechanism under named (and probably biased) circumstances, such as in [167] and [168]. It is also worth mentioning here that some open-source BC projects are available and can be used to simulate BC scenarios. For example, the HyperLedger [169] projects administered by the Linux Foundation are highly sophisticated and well implemented BC systems. One can locally clone any project that suits the application needs and construct a local network. However, those projects are not targeting the simulation purposes as much as providing realized BC services for industrial projects.

Additionally, most of these projects, such as Indy, are hard to re-configure and, if re-configured, very sensitive to small changes in their code. Indy, for example, uses a modified version of PBFT CA, namely Plenum, while Fabric uses RAFT.

3.3 The proposed FoBSim tool

I have implemented FoBSim according to the conceptual workflow demonstrated in Figure 3.1. FoBSim was implemented so that it is inline with the general architecture of a reliable BC simulation tool presented in [141]. In fact, many more services and scenarios can be simulated using FoBSim, covering the fog layer inclusion besides the BC. As presented in Figure 3.1, different CAs can be used, different services of the BC network can be declared, and different placement scenarios of the BC network can be chosen. When the BC network is located in the fog layer, the number of BC nodes does not need to be input because each fog node is also a BC node. Nevertheless, number of task requester end-users connected to each fog node needs to be input, while some fog nodes in a PoA-based scenario might be not authorized to mint new blocks. Once the network is built, the system model can be run and tested.

The FoBSim environment is implemented using Python v3.8, with the inclusion of some common packages such as: *random*, *randrange*, *multiprocessing*, *time*, and *hashlib*. The current version of FoBSim can be cloned and directly run as all the variables, lists, dictionaries, and sets have been given initial values. However, these parameters can be modified before running the code in the *Sim_parameters.json* file. FoBSim tool is open-source and freely available at [170]. Due to the page number limitations of this thesis, only major descriptive discussions are provided here. More comprehensive details are provided in [171].

3.3.1 FoBSim Modules

Fog nodes can be extensible to provide privacy-preserving mechanisms (such as described in [172]), computational services (such as described in [173]), or reputation and trust management services (such as described in [9]). In the ”memPool.py” module, TXs are accumulated in a python multiprocessing queue that allows different processes to synchronously add() and get() TXs. There are other minor methods from other modules which are also called by FoBSim entities to mint new Blocks, or receive new TXs/Blocks, in order to synchronously and smoothly apply each different CA’s policies, as declared in its simple version. After each simulation run, some temporary files can be found in the ”temporary” folder of FoBSim. These files are originally initiated by the main module, the BC module, or the miner module. The temporary files are used synchronously by different FoBSim entities, mimicking the real-world interaction between BC entities. The current version of FoBSim generates

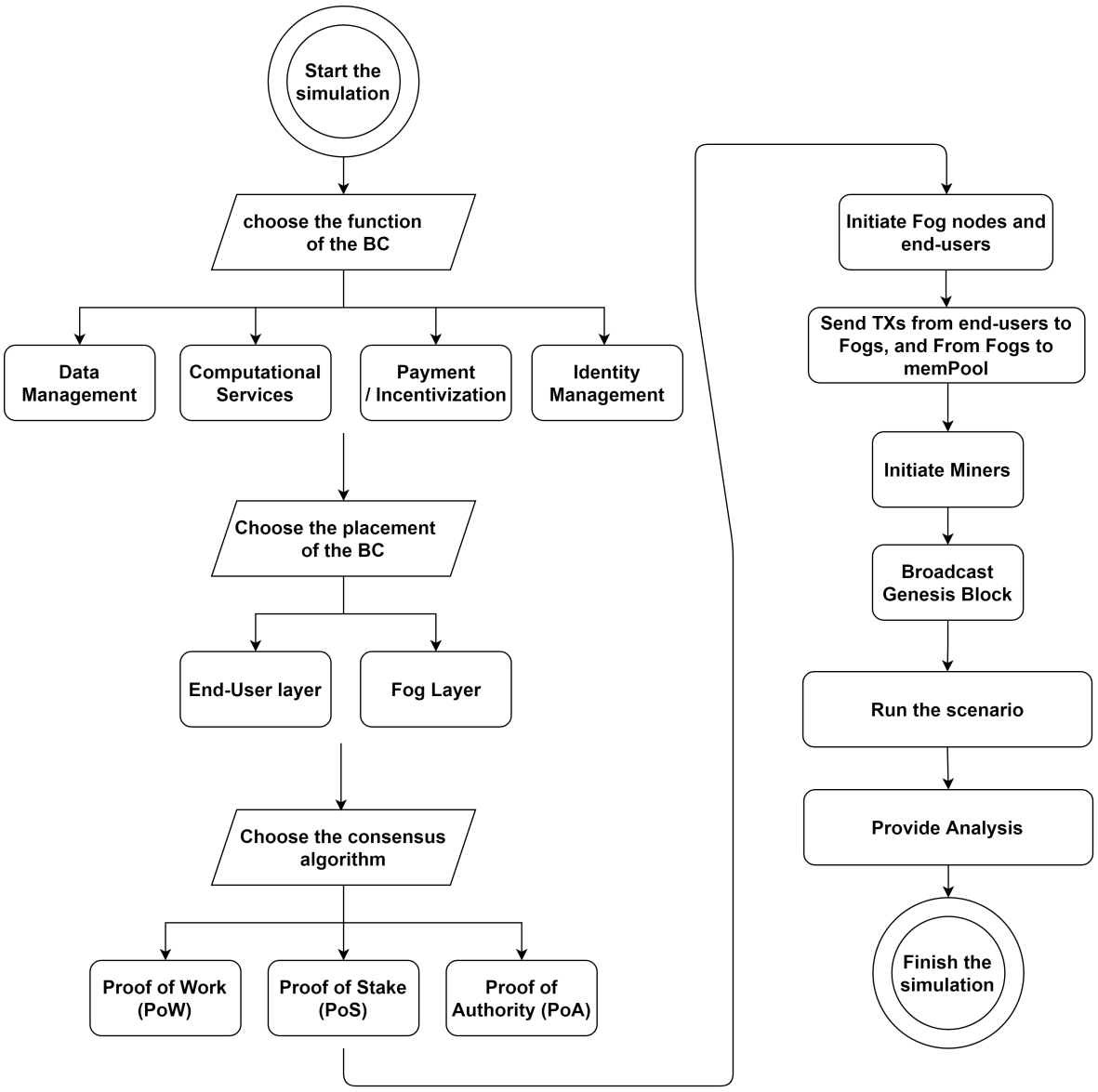


Figure 3.1: *Workflow of a simulation run using the proposed FoBSim tool*

some files depending on the simulated scenario (e.g. Miners' local chains, Miners' local records of users' wallets, Log of blocks confirmed by the majority of miners, Log of final amounts in miners' wallets (initial values - staked values + awards), Log of coin stakes, The longest confirmed chain, Forking log, etc.

3.3.2 Genesis Block Generation

The first block added to the chain in each simulation run is the most important block of the chain. Different scenarios imply different formats of this block, and different

methods to broadcast it among, and be accepted by, miners. In the current version of FoBSim, a genesis block is initiated with a list of miners available when this block was generated. The block number, previous hash, and nonce are all set to 0. The timestamp of the genesis block indicates when the chain was launched, hence all blocks shall have later timestamp values than this first timestamp.

3.3.3 FoBSim Consensus Algorithms

Currently, there are three available CAs ready to be used in different simulation scenarios. The following subsections describe each one individually as to facilitate any modifications by developers. However, the three included CAs are in their simplest versions and may require some individual modification in case of the need of more complicated ones. Before delving into the CAs, a discussion of the Gossip protocol in FoBSim is provided, as it can be (de)activated regardless of the selected CA.

Gossip Protocol

A Gossip Protocol [174] is usually deployed in peer-to-peer (P2P) systems for maintaining the consistency of distributed data saved in decentralized networks. Specifically in BC systems, miners regularly, yet randomly, gossip with their neighbours about current versions of the chain, aiming to reach consensus finality as soon as possible. According to specific characteristics of the BC, locally saved chains are updated so that all confirmed chains are soon enough equivalent at any given moment [175]. The equivalency that any BC system is seeking is defined by the contents similarity of the chains (i.e. TXs, hashes, etc.), and the order similarity of the confirmed blocks. That is, a chain [b_1, b_2, b_3] is not equivalent to [b_1, b_3, b_2] despite the fact that both have similar contents.

Gossip protocols are usually fault tolerant as many failing nodes do not affect the protocol. Furthermore, they can adapt to the dynamics of the network as several solutions have been proposed in the literature for nodes joining and leaving the network. However, gossiping is an iterative method that never quits as long as the network is up, and it may take time to converge. Additionally, high level of communication costs is expected for gossiping, while randomly chosen neighbors are informed about updates. Thus, one cannot provide precise analysis about the needed time for network agreement on a piece of data.

Although the implementation of such protocol is relatively simple, it is differently implemented in different systems. Some famous examples of efficient gossiping protocols include the Push-Sum protocol [176], the Push-Flow algorithm [177], and different versions of the Push-Pull averaging protocol [178]. Additionally, it can be easily noted that the number of simulated TXs/blocks and the initial TX per block configuration affects the speed of the system to reach consensus finality. That is, for

low number of TXs, blocks, and low ratios of TX per block, miners might not have sufficient time to converge locally saved chains. Final versions of local chains in some FoBSim simulations, under such circumstances, may not coincide, which is normal and expected as described in [179].

In FoBSim, a simple Push-Pull Gossip version is deployed. Modifications can be easily conducted if needed on the deployed approach, where a Time To Live (TTL) parameter was not added to the Pull requests when gossiping. This, as expected, floods the network with Pull and Push requests each time a node wants to gossip. Simulating up to 1500 miners, no memory or cash issues appeared. However, simulating more miners, where gossiping is activated, raised congested memory as Python does not allow more than 1500 parallel threads. Thus, it is recommended in such a case to either configure gossip requests to have a TTL (i.e. a number of hops the request walks before it is terminated), and/or decreasing the number of neighbors the gossiping node is sending the gossip request to. More details on such implementation approach can be found in [180], while detailed analysis regarding the success rate of gossiping, with a given TTL in a given P2P network, can be found in [181].

If the gossiping property was set to *true*, each miner will gossip once a new block is received as default. A miner who is gossiping requests information about the longest chain, and adopts it if its contents were agreed on by the majority of the network. Additionally, if a miner receives a new valid block, and the resulting local chain was longer than the global chain, the miner updates the global chain instantly, which represent the Push request.

The Proof of Work

In a simplified scenario of a PoW-based BC, miners collect TXs from the mempool (which is a shared queue in FoBSim) and accumulate them in blocks that they mint. Specifically, all available miners *compete* to produce the next block that will be added to the chain. The fastest miner producing the next block is the miner whose block is accepted by all other miners of the BC. Synchronously, all blocks that are being minted by other miners are withdrawn, and all TXs within are sent back to the mempool. To mimic this scenario in FoBSim, I deployed the multiprocessing package of Python and trigger all miners to work together on the next block.

Each miner then works within an isolated core of the device on which the simulation is conducted. A second approach is implemented where consequent calls for miners to mint new blocks (using a FOR loop) is run. Both approaches are available in the **miners_trigger()** function in the main.py module, and developers are free to use either. The Sim_parameters.json facilitates such parameterization. Detailed instructions for implementing different memory-sharing scenarios for advanced access control of the processes can be found in the Python official documentation [182].

When a Miner receives a new block, it checks whether the hash of the block (in

which the nonce or the puzzle solution is included) is in line with the acceptance condition enforced by the blockchain.py module. Further, the receiver miner checks whether sender end-users have sufficient amount of digital coins to perform the TX (in the case of payment functionality). Any miner is authorized to produce a block and there is no miner verification required.

The Proof of Stake

In a simplified version of PoS, miners stake different amounts of digital coins (which they temporarily are not allowed to claim) in the BC network. The network then randomly chooses a miner to mint the next block, with higher probability to be chosen for miners who stake more coins. Once a miner is chosen, it is the only one authorized to mint and broadcast the next block. In case of faulty/invalid block, the miner loses its staked coins as a penalty. Otherwise, it is awarded a predefined amount of digital coins.

To mimic this in FoBSim, each miner is initiated with specific amount of coins in its wallet. After that, randomly generated number of coins (up to the amount of coins in its wallet) is staked by each miner. Every miner then shall gain different probability to be chosen by the network. Next, the network randomly chooses, say 10% of the available miners, and picks the one with the highest stake. This chosen miner's address is immediately broadcast to all miners so that any block received from any other miner is rejected. Once the new block is received, it is validated and added to the local chain.

The Proof of Authority

In a simplified version of the PoA algorithm. only authorized network entities (by network administrators) are illegible to mine new blocks. Regardless of the BC functionality, there is no need to deploy the multiprocessing package for PoA-based scenarios as there is no competition as in PoW-based scenarios.

To mimic the PoA in FoBSim, the user is optionally allowed to declare authorized miners to mine new blocks. When the BC is deployed in the end-user layer, authorized miners are responsible for mining new blocks and maintaining the distributed ledger. When the BC is deployed in the fog layer, fog nodes perform both fog services and BC services according to the service definition. In both deployment options, unauthorized miners are only responsible for validating new blocks, received from their neighbors, and maintaining the distributed ledger.

This approach allows for comfortably emulating a scenario where the BC in the fog layer and part of the fogs are included in the BC functionality. Notice that a fog node that is also a BC node performs all the required tasks in logical isolation. This means that a fog node that is administering a group of end-users has a buffer to save

end-users TXs, but it does not use these TXs to mine a new block. Rather, it sends these TXs to the mempool as required, and then, only if it was authorized, it collects TXs from the mempool.

3.3.4 Transaction/Block Validation in FoBSim

Here, some differences between the terms Verification, Validation and Confirmation need to be underlined [47]. Accordingly, it can be understood how FoBSim differentiates between those terms in different scenarios.

Validation is the process when a miner (either a minter or receiver) checks the correctness of a claim. That is, in the case of a minter miner, the puzzle solution (or nonce) provided with the minted block needs to be correct before the block is broadcast. Otherwise, a new solution is searched for. In the case of a receiver miner, the nonce is checked once. If the solution was valid, the block is accepted, otherwise, the block is rejected.

In the case of payment functionality, the validity of TXs fetched from the mempool is tested. This means that the amount of coins in the wallet of the sender of each TX, in the payment functionality, is compared to the amount to be transferred. If the wallet contains less than the transferred amount, the TX is withdrawn from the block. Later when the new block is received by a miner, the same hash validation and TXs validation take place, except if one of the TXs were invalid, the whole block is rejected. In the case of a block rejection, the minter miner is usually reported in a reputation-aware context. If all the contents of a newly received block are valid (i.e. the hash, the TXs, the wallets, the block number, and the nonce) the block is added to the locally saved chain. Here, TXs are confirmed as the block containing them is added to the chain (i.e. the block is confirmed).

The verification, on the other hand, is the process of verifying the identity of an entity. For example, in the case of PoA, only authorized miners are allowed to mint new blocks. Similarly, in the case of PoS, a received block should be generated by a miner that all other miners expect to receive the new block from. Additionally, public information about end-users' wallets need to be accessible by miners to validate their TXs. Thus, a received block, with some TXs generated by end-users who do not have wallets, or whose wallets contents are not readable by miners, can not be validated and confirmed. Failing to confirm a TX is not necessarily caused by end-users not having sufficient coins to transfer, but may also happen for end-users who can not be *verified*.

All of these critical principles are, by default, taken care of in FoBSim. All miners are informed about the end-users public identities and wallets contents. After that, transferred coins are updated locally in each miner. Consequently, a new TX from the same end-user will be compared to the updated amount of coins in its wallet.

Invalid TXs are not included in the block being minted, while invalid TXs cause the rejection of the whole received block. Once a block contents are validated, and the TXs/block generators are verified, the TXs are confirmed. That is, the locally saved wallets amounts are updated, and the block is locally confirmed and added to the chain. The most interesting thing is that the very small probability of a double spend attack [183], which can appear in PoW-based scenarios, can be easily simulated in FoBSim. All processes are actually happening during each simulation run, rather than substituting them with a small delay as in most BC simulation tools previously discussed. Hence, validation, verification, and confirmation processes can be modified according to the scenario to be simulated. To facilitate the simulation of such critical scenarios, I deployed two broadcasting approaches for newly minted blocks. The first allows the broadcast process using a simple FOR loop, where miners sequentially validate and confirm new blocks. The second allows the broadcast process using the multiprocessing package, which allows all miners to receive and process new blocks at the same time. Relatively, developers need to be cautious when using the second approach, because of some critical challenges similar to those mentioned in Subsection 3.3.3.

3.3.5 Awarding winning miners

Generally speaking, BC miners get rewarded for providing BC services by two methods. The first is usually called TXgas, which is a small fee payed by the end-user who generated the TX. The second is the block reward granted by the BC network itself (i.e. all miner nodes), who update their locally saved winning miner's wallet once a new block is confirmed.

In the current version of FoBSim, miners are rewarded by the second approach only. This part is hard because the reward needs to be agreed on by the majority of BC miners (i.e. at least 51%). The first incentivization mechanism, is not applicable in many different scenarios, hence I left it for the developers to add it if needed.

3.3.6 Strategies in FoBSim

As had been discussed so far, there are some default strategies used by FoBSim entities throughout each simulation run. To mention some, TXs are picked by miners with no preference, e.g. the highest GAS or priority. Also, a default chain is a single linear chain and new blocks are added to the top of this chain. Some applications, however, have multiple chains or multi-dimentional chains, e.g. Directed Acyclic Graph (DAG) based chain. Additionally, if two blocks appear in the network, the block that was accepted by the majority of miners is confirmed rather than, in some BC systems, the older one is confirmed even if it was confirmed by the minority. Furthermore,

a valid block is immediately announced, once found, into the FoBSim network. In some other applications, there might be a conditional delay. For example, if a selfish mining attack scenario is to be simulated, miners would prefer to keep their newly found blocks secret, hoping they will find the next block as well [184].

The current version of FoBSim assumes data flows from end-users to the fog, and from the fog to the BC network. However, there are other possible data flow schemes that can be simulated. The BC in the current version provides DLT services to end-users, which are communicating with the BC through the fog layer. Services might be provided by the fog layer to the BC network or from the BC network to the fogs in some other applications. Furthermore, an application where end-users may need to request data directly from the BC might be possible, which implies different data flow as well. FoBSim facilitates the modification of data flow in the simulated application, and presents an extra Cloud Computing module that can add more possibilities to the simulated application.

Network connectivity characteristics are a major and critical concern in any BC system. To facilitate network architects job, FoBSim allows to define the number of nodes in each layer, the number of neighbors of each BC node, and the general topology of the network. Additionally, all BC nodes are connected into one giant component by default, whether they were deployed in the fog layer or end-user layer. Accordingly, the effect of manipulating the topology of simulated networks can be easily captured.

A full discussion of FoBSim constraints and their mitigation methods, including Merkle Trees, mining pools, and digital signatures, is available in [185].

3.4 Case Studies

Following the validation and verification methods of simulation models presented in [186], I have so far discussed the technologies and the paradigms lying within my proposed FoBSim environment. Furthermore, I highlighted my proposal novelty compared to other related works, discussed the event validity in FoBSim, and presented the algorithms and modules lying within to facilitate a structured walk-through validation.

Next, I follow an operational validity approach by presenting case studies that I simulated using FoBSim. The setup and behaviour of FoBSim is discussed, and the results of the simulation runs are presented afterwards. For all experiments presented in this chapter, I deployed the FoBSim environment on the Google Cloud Platform, using a C2-standard-16 VM (clocked at 3.8 GHz, 16 vCPUs, 64 GB memory), running Ubuntu 20.04 LTS OS.

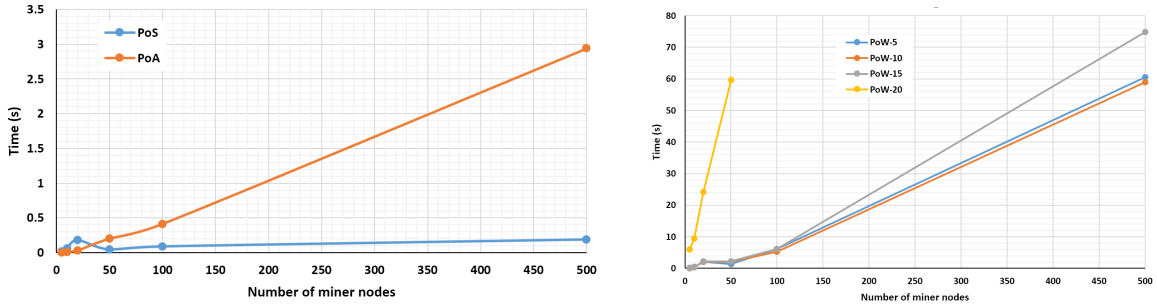


Figure 3.2: *Average block confirmation time (a) consumed by PoS-based BC vs. PoA-based BC, relatively to the number of miner nodes (b) consumed by PoW-based BC (the cases of difficulty = 5, 10, 15, and 20), relatively to the number of miner nodes*

Table 3.2: *Simulation parameters configuration for Case 1*

Simulation parameter\CA	PoW	PoS	PoA
no. of miners	5–500	5–500	5–500
neighbours per miner	4	4	4
puzzle difficulty	5–20	–	–
Authorized miners	All	Random choice	2–25
Initial wallet	–	1000	–
BC functionality	Data Management	Data Management	Data Management
BC deployment	end-user layer	end-user layer	end-user layer

3.4.1 Case-1: Comparing time consumption of PoW, PoS, and PoA

When PoW, PoS and PoA are compared in terms of average block confirmation time, PoW is expected to present the highest rates. This is because of the mathematical puzzle that each miner needs to solve in order to prove its illegibility to mine the next block. In PoS, the network algorithm randomly chooses the next miner, while it slightly prefers a miner with higher amount of staked coins. Once a miner is chosen, all miners are informed about the generator of the next block and, thus, the miner needs to perform no tasks other than accumulating TXs in a new standard block. Other miners then accept the new block if it was generated by the expected generator. The verification process takes nearly no time (assuming that the transmission delay between miners is negligible).

Using PoA, all authorized miners mine new blocks, verify newly mined blocks, and maintain the chain locally. Meanwhile, unautorized nodes verify new blocks and maintain the chain, but do not mine new blocks [187]. Every BC node has a regularly-updated list of authorized miners. This implies that the more authorized entities, the more complex the verification can be on the receiver side. Accordingly, it is advised to lessen the number of authorized miners for decreasing the complexity of verification [188]. Meanwhile, the more maintainers in a PoA-based BC, the higher

Table 3.3: *Average block confirmation time (results of Case-1), where the PoW puzzle difficulty ranged from 5 to 20, and the number of Miners (M) ranged from 5 to 500.*

Algorithm	M=5	M=10	M=20	M=50	M=100	M=500
PoS	0.018	0.06	0.18	0.046	0.09	0.19
PoA	0.002	0.008	0.03	0.2	0.41	2.94
PoW-5	0.08	0.36	2.1	1.31	6.15	60.6
PoW-10	0.07	0.44	2.1	2.03	5.21	58.9
PoW-15	0.25	0.42	2.23	2.26	6.18	74.76
PoW-20	6.02	9.5	24.2	59.62	–	–

the overall security level of the system.

In this case study, I run FoBSim several times deploying different CAs under similar conditions. The simulation runs targeted specifically the measurement of the average time consumed by each CA, from the moment where a miner is triggered to mine a new block, until this mined block is confirmed by, at least, 51% of other BC miners. To accurately measure this average, I added some variables representing the starting time and the elapsed time, exactly before calling the build_block() function and right after a block is confirmed by reaching the required number of confirmations.

As described in Table 3.2, I changed the difficulty of the puzzle during the PoW-based BC simulation runs from an easy level (i.e. 5), to a harder level (10), and finally to very hard levels (15) and (20). During the runs where PoA was used, I changed the number of authorized miners from 2/5 (2 authorized out of a total of 5 miners), 5/10, 10/20, and 25 authorized miners for the rest of runs.

Abstractly measuring the average confirmation time, I avoided the Computational Services and the Payment functionality, because both imply extra time consumption for performing the computational tasks, and validating the payments, respectively. I also avoided the Identity management functionality because the number of TXs per end-user is limited by the number of ID attributes required to be saved on-chain. Hence, the best choice was the Data Management functionality. I stabilized the total number of TXs delivered to the mempool unchanged, which gives equivalent input for all simulation runs. However, I changed the number of TXs generated by each end-user as to be equal to the number of miners in each run. Concerning the runs where a PoS is deployed, miner nodes were initiated with a wallet that has 1000 coins, allowing miners to stake random amounts of coins (with a block reward of 5 coins).

I have placed the BC in the end-user layer for all runs. Table 3.3 presents the exact results I obtained, which are depicted in Figures 3.2.a and 3.2.b.

It is noticeable from the results that PoW-based BCs consume much more time as

expected, to confirm a block, than PoA and PoS-based BCs. Additionally, the average block confirmation time, in PoW-based and PoA-based BCs, seems to be directly proportional to the BC network size, which complies with the results presented in [189]. Comparatively, an average block confirmation time in a PoS-based BC seems unaffected by the network size, which complies with the results presented in [190].

3.4.2 Case-2: Capturing the effect of using the Gossip protocol

Table 3.4: *Simulation parameters configuration for Case-2, where the Gossiping property is interchangeably activated and deactivated*

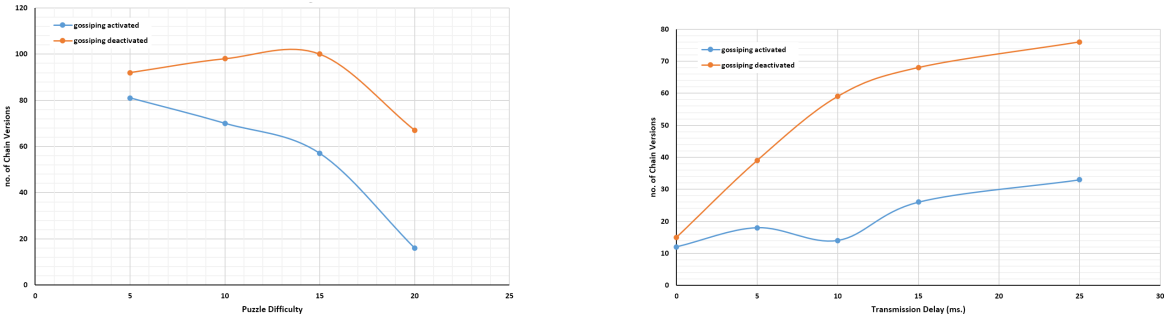
Simulation parameter	Puzzle difficulty effect	Transmission delay effect
no. of Fog Nodes	5	5
no. of users per fog node	5	5
no. of TX per user	5	5
no. of miners	100	100
no. of neighbours per miner	2	2
no. of TX per Block	5	5
puzzle difficulty	5, 10, 15, 20	20
Max enduser payment	100	100
miners initial wallet value	100	100
mining award	5	5
delay between neighbors	0	0, 5, 10, 15, 20

Table 3.5: *Results of Case-2, where the puzzle difficulty ranged from 5–20, and the Gossiping in FoBSim was interchangeably activated and deactivated*

Configuration	diff.=5	diff.=10	diff.=15	diff.=20
Gossip activated	81	70	57	16
Gossip deactivated	92	98	100	67

Table 3.6: *Results of Case-2, where the transmission delay between neighbors ranged from 0–25 ms., and the Gossiping in FoBSim was interchangeably activated and deactivated*

Configuration	T.D.=0	T.D.=5	T.D.=10	T.D.=15	T.D.=25
Gossip activated	12	18	14	26	33
Gossip deactivated	15	39	59	68	76



(a) *the puzzle difficulty fluctuates from 5 to 20*

(b) *the transmission delay between neighboring miners fluctuates from 0 to 25 ms*

Figure 3.3: *The effect of activating the gossiping protocol in FoBSim, on the number of chain versions at the end of PoW-based BC simulation runs*

In this case, I compare the number of chain forks at the end of several simulation runs, where I interchangeably activate and deactivate the gossiping property in a PoW-based BC. Accordingly, I could notice the effect of gossiping on ledger finality under different conditions, namely the puzzle difficulty and the transmission delay between miners. In this case, I detect the number of chain versions at the end of simulation runs, which can be decreased to one version under strictly designed parameters, such as medium network size, high puzzle difficulty, low transmission delay, low number of neighbors per miner, etc. Nevertheless, my goal in this case is to demonstrate how the (de)activation of the gossiping property during a simulation run affect the number of chain versions and, thus, the consistency of the distributed ledger.

Table 3.4 presents the initial configuration in each simulation scenario, while Tables 3.5 and 3.6 present the results I obtained by running the described scenarios, which are depicted in Figure 3.3. As can be noted from the results, the default gossip protocol in FoBSim could decrease the number of chain versions at the end of each simulation run. Although the number of chain versions did not reach the optimum value (i.e. one chain version), it is obvious that activating the gossiping property decreases the number of chain versions at each simulation run and, thus, enhances the distributed ledger consistency.

3.4.3 Case-3: Comparing deployment efficiency of BC in the fog layer vs. end-user layer

In this case, I compare BC deployment efficiency in the fog layer versus the end-user layer. The efficiency is determined by both the total time needed to perform all requested BC services and total storage cost. To fairly compare the BC efficiency when

Table 3.7: *Simulation parameters configuration for Case-3, where the efficiency of BC is assessed in the fog layer and end-user layer, in terms of total run time and total storage cost*

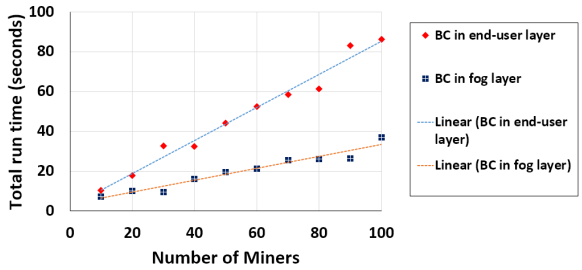
Simulation parameter	For total time efficiency	For total storage efficiency
no. of Fog Nodes	10–100	100
no. of users per fog node	2	5
no. of TX per user	2	5
no. of miners	10–100	100
no. of neighbours per miner	3	5
no. of TX per Block	5	5
puzzle difficulty	20	15
Max end-user payment	100	100
miners initial wallet value	1000	1000
mining award	5	5
delay between neighbors	fog layer: 12 ms., end-user layer: 1000 ms.	fog layer: 12 ms., end-user layer: 1000 ms.

deployed in those two layers, I stabilized all BC parameters that are configurable in FoBSim, except for the network size to deduce the trend of total time consumption when the network dynamically allows for new nodes to join the network. The detailed parameter configuration while running the described scenarios is presented in Table 3.7.

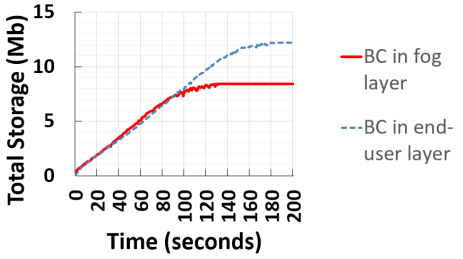
Recalling the results presented in [191] and [192], average transmission delay between miners in the fog layer can be estimated by 12 ms., while it can be estimated between miners in the end-user layer to 1000 ms. (higher transmission delays were reported in well known BC networks, such as Bitcoin [193]). The number of requested tasks was automatically modified due to the continuous change in the number of fog nodes (since the number of fog nodes was oscillated to deduce the trend of total time consumption). The total average time for performing requested BC services, in similar simulation sittings, while the BC is deployed in end-user and fog layers, is compared in Figure 3.4.a.

To accurately measure the storage cost during the simulation run, I implemented an independent storage cost analysis script, available in the FoBSim repository. This script analyzes the storage consumed by FoBSim modules represented by the size of the 'temporary' folder. The size of the folder is measured regularly (every one second), while the simulation is running. The measured sizes are then saved into an Excels sheet to facilitate performing the analysis. The total storage used by the BC network is compared in Figure 3.4.b, where similar simulation sittings were configured (detailed in Table 3.7), except for the layer where the BC is deployed.

It can be noted from the results presented in this case that deploying the BC



(a) *Total elapsed time for the BC network to perform requested services*



(b) *Total storage used by the BC network to perform requested services*

Figure 3.4: *BC efficiency comparison while deployed in end-user layer vs. fog layer*

network in the fog layer shall enhance its efficiency in terms of total time consumed to perform similar tasks with similar configuration, and in terms of total storage cost by the BC network to maintain the same distributed ledger (same number of confirmed blocks by the end of the simulation run).

3.5 Conclusions

In this chapter, I proposed a novel simulation tool called FobSim that mimics the interaction between the entities of an integrated FC-BC system. I briefly described the architectural elements of FC and BC technologies, and designed FoBSim in order to cover all the elements I described. I deployed three different CA algorithms, namely PoW, PoS and PoA, and different deployment options of the BC in an FC architecture, namely the end-user layer and the fog layer. Additionally, I fine tuned FoBSim modules so that various services, provided by FC and BC, can be adopted for any proposed integration scenario. The services that can be simulated are distributed Payment services, distributed Identity services, distributed Data storage and distributed Computational services (through Smart Contracts). I described the modules of FoBSim, the TX modelling, the Genesis block generation, the gossiping in FoBSim, the CAs, TX and block validation, incentive mechanisms, and other FoBSim strategies. I validated FoBSim with case studies comparing the average block time for different CAs, and analyzing the effect of gossiping on the consistency of the distributed ledger, in fluctuated puzzle difficulty and transmission delay configurations. Additionally, I compared the efficiency of the BC network, in terms of block confirmation time and total storage required to perform similar tasks, when deployed in the fog layer versus the end-user layer.

The results of the first case showed that PoS provides the least average block confirmation time, followed by PoA and PoW, respectively. The results of the second case

showed how FoBSim's gossip protocol performs as expected in realized BC applications, which contributes to enhance the consistency of the distributed ledger. The last case showed that deploying the BC network in the fog layer shall drastically enhance the BC performance, in terms of average block confirmation time and total storage cost.

In the future releases of FoBSim, I am willing to make more CAs available, as well as enhancing the identity management scheme in FoBSim. I will further investigate adding a Reputation management service in a generalized and simple manner so that analysis can be provided, while proposed reputation management ideas, conditions, or properties can be easily implemented/modified.

Furthermore, I plan to extend the simulator to support complex models for simulating the scheduling and provisioning process of the entire application. The simulation models shall adaptively respond to significant changes in the pool of available smart devices (Cloud or FC instances) during application execution and identify provisioned devices that do not provide good performance for a given smart application component. They will further enable the replacement of low-performing smart devices, e.g., provisioned as VMs or containers that no longer meet the application requirements, or reconfigure existing ones (increase number of CPUs to a VM running).

The application scheduling and provisioning models shall enable the simulation of decentralized data-aware resources scheduling over multiple control and network domains with increased trust. The model will utilize the transaction logs, stored in the simulated BC, to manage the smart devices in an efficient manner. The approach will use semantics to describe the simulated smart devices and check their compatibility through an Application Definition Machine (ADM). The ADM will describe the recommended resources for a given smart application.

I also plan to model different migration techniques to provide accurate simulation of the deployment and communication overhead for using smart devices from multiple providers. In case of over-provisioning, I will simulate the release or downgrading of resources to minimize the overall resource consumption without violating application requirements. Finally, the model will enable the simulation of resources provisioning through ADM for each individual application defined in the simulator.

I am responsible for the following contributions presented in this chapter:

I/4. I designed a novel and extensible simulation tool called FoBSim that can comprehensively and realistically mimic BC-FC integrated systems.

I/5. I experimentally validated and evaluated the simulation environment with different use cases and simulation parameters.

I/6. I experimentally proved that deploying BC in the fog layer outperforms a BC deployed in the end-user layer, in terms of block finality and storage cost.

Chapter 4

Enhancing Blockchain Efficiency

As discussed earlier, several challenges are faced when BC is integrated into fog-enabled systems. System efficiency can be defined differently for different integration models due to various elements of a given system. Response latency and resource utilization are examples of efficiency benchmarks of a given fog-enabled system. A BC-based solution can similarly be evaluated, with additional benchmarks relating to the application layer of the solution, such as overall system throughput and block finality time. In this chapter, I propose methods to evaluate BC systems, to decide where to deploy those systems, and how to parameterize them to get better efficiency. Additionally, I propose protocols, that are not specifically aware of the fog element, targeting enhanced block finality and message fidelity in BC-based systems in general.

4.1 Introduction

A major component of a BC-based system is the Distributed Ledger (DL), whose consistency is a problem that describes the unreliability of DLs in dense and highly dynamic networks [194]. This problem concerns maintaining exact copies of the DL, as the appearance of different DL versions is trivially expected in realized scenarios. Reasons for such issue include both, the transmission delay between network entities and the continuous and concurrent alteration of DL data [195]. The concept of Finality is usually related to the DL consistency, which is the state of the BC, under which TXs cannot be canceled, reversed or changed by any member of this network under any circumstances [195]. Although Nakamoto's model did not perfectly solve the consistency problem, it proposed a highly accurate, probabilistic solution. Specifically, a next block of data is introduced into the network when its previous block had, most likely, sufficient time to be confirmed on the DL (i.e. synchronized between the majority of network entities). The occurrence of two different data blocks at the same time would mostly lead to a temporary inconsistent state of the DL, which is

termed forking [196].

The enforced delay between blocks depends mainly on the ability of system entities to find a puzzle solution. The difficulty of the puzzle is indeed updated through time according to the design requirements (e.g. Bitcoin's predefined delay is 10 mins). Such model is classified as a 'Probabilistic Finality System'. That is, the system continuously lowers the probability of concurrent DL updates, yet the probability never reaches zero. Examples of algorithms belonging to this class include the PoW and the PoET. The other class is the 'Absolute Finality Systems', where the system allows its entities to produce a next block, only when the previous block is confirmed. Examples of algorithms belonging to this class include the PBFT and some versions of PoS [197].

The mechanisms deployed in BCs to decrease the probability of forking can be concluded by three main approaches. First, a gossiping protocol to synchronize confirmed blocks and longer DLs. Second, the continuous increment of the puzzle difficulty, which gives a window to network entities to gossip, leading to increased total energy consumption of the system, and/or decreased total throughput. Third, the utilization of full and light nodes in the network [198], so that gossiping is performed by fewer network entities. Accordingly, data propagation through the network shall consume less time.

A scalable system is one that maintains constant, or slowly degrading, overheads and performance as its size increases [199]. The dynamicity in P2P networks, which are the physical infrastructure of BCs, imposes even more complicated problems than the DL inconsistency as it directly affects its scalability. That is, the constantly changing topology of the network leads to non-consistent propagation delays between its entities. Peers in the vast majority of adopted BC solutions are connected to several neighboring peers with the adoption of a Randomized Neighbor Selection (RNS) method. Selected neighbors are the connected peers with which a named peer shares data [200]. Generally, shared data between peers include new blocks or information regarding the state of the sender's ledger (i.e. gossiping). Gossiping also includes sharing the best DL version between peers (best is defined according to certain criteria, e.g. longest chain). The RNS method implies randomized paths, walked by shared data [201], leading to an inefficient data propagation scheme. This is due to several redundant exchanged messages caused by the probability of cycle appearance on the randomly selected path of data, leading to higher average finality time and lower consistency levels. Other than that, few methods were proposed to locally optimize Neighbor Selection (NS), and indeed addressed the dynamicity issue (e.g. [191]). However, none of these solutions proposed *optimum* NS.

In this chapter, I analyze the forking phenomenon in BC-based systems that utilize a probabilistic finality mechanism. Specifically, I use the PoW algorithm to represent the class of probabilistic finality. As I perform my analysis under different conditions,

I contribute a method for evaluating the consistency of the DL, by finding the ratio of forks appearing within the DL to the maximum possible number of chain versions. Additionally, I study the effect of different BC deployment scenarios in the fog and end-user layer, and propose a decision making model that may help practitioners choose the optimal deployment approaches of BCs in Fog-enhanced environments.

Depending on the results I obtained, I propose a Dynamic Optimized Neighbor Selection (DONS) protocol. That is, more neighbors per miner and higher delivery time rates between neighbors, both lead to lower levels of DL consistency [202, 203]. These results served as a motivation to the proposal of DONS. This protocol shall require minimum number of neighbors per miner, directing the miners to communicate with globally-optimized selection of neighbors. I discuss how DONS decreases the number of cycles within a path, that shared data walks, from any peer to any other peer (i.e. no cycles, hence a Spanning Tree is an optimal solution [204]). I also show how DONS decreases the maximum time spent from generating data, by any peer, till it reaches all the peers of the network. Additionally, I discuss how DONS addresses the scalability issues of the network, leading to adaptive optimization of NS in spite of continuous change in network topology.

The proposed protocol includes a privacy-preserving leader election method, allowing one of the peers within the BC network to compute the MST without previous knowledge of network peers identities (e.g. IP address). Using one of the famous MST algorithms (e.g. Prim's or Kruskal's), the leader computes the MST and broadcasts it to all the network. Every recipient of the MST then can read only its identity and its neighbors' identities, leading to each peer of the network communicating with the optimized selection of neighbors. As a result, DONS shall address the NS Problem of public BCs.

I evaluate the DONS protocol against other approaches, utilizing two randomized network models, namely Erdős-Rényi (ER) model [205] and Barabási-Albert (BA) model [206]. The DONS protocol is analytically evaluated in terms of security and privacy, and is experimentally evaluated in terms of propagation time and message overhead against the currently used RNS and Round Trip Time based Neighbor Selection (RTT-NS) methods. The leader election method is theoretically and experimentally evaluated, in terms of time and message complexity [207], against a recent solution proposed in [208]. The evaluation results show high levels of security, privacy and efficiency.

4.2 Related Works

4.2.1 Consistency of Distributed Ledgers

It was proven by Brewer [209] that a DL system cannot guarantee ledger consistency, nodes availability and partition tolerance, altogether. Accordingly, any attempt for decreasing the inconsistency levels in a given BC system, would either decrease the availability level of the system, or decrease the overall tolerance against network partitioning.

Kiffer et al. [210] analyzed the consistency of Nakamoto's BC from a security point of view. In their study, the block delay, puzzle difficulty, and the adversarial fraction out of the total network size, were the considered parameters. The proposed method aimed at presenting the probability of some versions of the delay attacks, that may alter data saved on the DL. Under similar conditions, Zhao et al. [211] proposed a formula for ensuring the consistency property against the delay attacks.

Misic et al. [189] presented some ledger forking probability analysis depending on the absence hours of miners and the network size. They have discussed the optimization of mean delivery time of TXs, while the network size is fluctuating. Accordingly, the effects of varying properties of their proposed network model, such as block type and number of TCP connections per node, were analyzed.

Lan et al. [180] proposed consistency maintenance techniques for P2P networks (e.g. BC networks), based on push, pull, and hybrid gossiping algorithms. They found that a push-based approach achieves near-perfect consistency in stable P2P networks, although the flooding of messages through the network was found a burden, while a hybrid approach is very good for highly dynamic networks.

Wang et al. [212] compared their proposed CMV algorithm with the rumor spreading based scheme, and the Update Propagation Through Replica Chain (UPTReC) scheme, in terms of finality time and messages overhead. Similar benchmarks were used in [213], [214] and [180] to evaluate the consistency of the DL.

4.2.2 Neighbor Selection in Blockchain-based Solutions

Finding the MST of a network or a distributed system by different means have already been proposed in the literature [215, 216]. For example, building a binary tree in distributed fashion within the network and select the root node to search for shortest paths was proposed. Additionally, many leader election algorithms have been proposed within other contexts, e.g. general distributed systems, or even BC networks that do not consider the identity privacy as a constraint [217], or ad-hoc wireless sensor networks that have gateway controllers [218, 219]. In such methods, the leader is utilized to administrate the network, mine new blocks, select next miners, or perform specific computations for specific slave nodes [199, 220].

It has been shown in several previous works how optimizing the NS decreases the probability of DL forking [221]. The RNS [222] methods is the most used in BC networks while it is the least optimized. Examples of such networks include Bitcoin [1] and Hyberledger Fabric [223]. Few others, however, attempted to optimize NS. Bi et al. [191] proposed miners favor neighbors with lowest Round Trip Time (RTT) when they need to perform NS. Similarly, a bandwidth informed NS protocol was proposed by Wang and Kim [224], where BC miners favor communications with neighbors that offer higher bandwidth transmission leading to decreased congestion and enhanced overall throughput. Aoki and Shudo [225] proposed a score-based NS protocol that depends on the difference between block generation and block receiving times. Consequently, neighbors with relatively higher scores are preferred for NS.

Jin et al. [226] proposed network clustering and TX ID sharing instead of full TX. Exchanged messages are, then, shared with targeted destinations rather than in a randomized fashion. The proposal was found efficient in terms of network traffic, yet it deviates the network model towards a centralized one as each cluster has its own leader. Additionally, security and privacy analysis were not conducted, although the protocol required private information to be run (e.g. length of online time, miners' IDs, etc.).

Other local optimization approaches were proposed in the literature, such as signing immature blocks [227], utilizing concurrent communication schemes [175], or maintaining local logs of neighbors updated per each new block [228]. Although all of the presented approaches indeed perform better than the currently adopted RNS approach, they only address the Neighbor Selection Problem (NSP) depending on local views of the network, leading to local NS optimization. Additionally, a group of those approaches requires modifying the underlying network topology and/or violates the identity privacy constraints usually present in public BCs.

I argue that a protocol that solves the NSP can be assumed comprehensive if it fulfils three main criteria: 1) It optimizes the NS depending on a *global* view of the network topology in a timely manner 2) It requires no modification of the underlying network topology and 3) It preserves the *Identity privacy* of all peers within the network.

To fulfil the first two criteria, one needs to utilize the global view of the BC network using Graph Theory. That is, finding the MST of the graph that represents the BC network is, in fact, finding the global solution of the NSP. Meanwhile, no new edges are enforced into the graph.

It is trivial to find the MST of a given network in polynomial time, if its topology is known, using famous algorithms such as Prim's [229] or Kruskal's [230]. Accordingly, BC networks that consist of a TTP can calculate the MST. However, public and permissionless BCs don't usually consist of a TTP, which implies that no entity within

the network can build a graph that demonstrates the network. Thus, polynomial-time algorithms that solve the Minimum Spanning Tree Problem (MSTP) cannot be used in fully distributed BCs.

The NSP in fully-distributed BC networks can be formalized using the Distributed Minimum Spanning Tree Problem (DMSTP) [231]. This problem aims at computing the MST of a distributed system without prior knowledge of network topology. This problem has a long line of research dating back to 1926 [232], until 2018 when the problem could finally enjoy a singular optimality state with the protocol proposed by Pandurangan et al. [208]. That is, the proposed algorithm solved the DMSTP with, simultaneously, optimal time and optimal message complexity.

Although DMSTP has been solved in [208], it actually cannot be adopted by current public BCs. That is, the algorithm requires its participants to share their identities, along with other (perhaps considered private) data. Such requirement imposes a privacy issue that will mostly forbid public BCs from utilizing the solution of [208]. To be specific, such approach does not fulfil the third criterion of a comprehensive NS solution.

To address all of these issues, I propose a 3-phase protocol in this chapter. The first phase solves a privacy-preserving leader election. In the second phase, the elected leader solves the MSTP. Finally, the third phase anonymously allows all miners to retrieve anonymized optimal NS. As a summery for the detailed leader election protocols and leader election problem [233], discussed in [234], most of previously proposed leader election algorithms are implementable in public BCs, if sharing the IDs and domains (e.g. IP-addresses) of network peers have no privacy implications. That is, if peers of the network trust all other peers with their identity information. However, this is generally not the case in public BCs, where each peer is only aware of its neighbors' identities.

4.3 Trust in Blockchain

Security, Privacy, and Trust are different but related concepts [235]. Trust, specifically, can mainly be described using probability [119], which majorly increases by the reliability the system provides [236]. In FC environments, trust should rely on binary decisions: Trusted or not Trusted.

Privacy in BC is preserved by different means, e.g. ledgers and blocks are transmitted to all connected nodes, blocks can be easily detected but can hardly be related to specific identity. This anonymity is caused by keeping public keys available for all nodes, yet anonymous. BC relies on the exponential reduction of attack probability as the chain is growing, leading to the state where it is computationally impractical to attack the chain and change TXs. However, multi TX generations with the same public key indicates the ownership of all these TXs by the same entity, which was

proven to be a privacy threat in Bitcoin [237]. Also, authors of [238] have shown the ability to infer identity information from smart contracts' source codes.

Here, I propose a concept for measuring the reliability of a BC system called Un-Reliability (denoted by R), which depends on the probability of withdrawing a confirmed TX, after a while of adding it to the chain, because of a malicious behaviour of the miner who generated the block, or simply because of the forking "Longest-Chain-Remains" protocol. If the probability of forking is *p*, then the probability a block being withdrawn is *p*/2 since one block will be withdrawn and the other will remain. For example, if *p* equals to 0.001%, and the number of TXs per block *B* equals to 5, then the probability that these TXs will be withdrawn, equals to 5 **p*/2 relative to total number of generated TXs on that day *T*. To generalize the concept of R, I propose equation 4.1.

$$R = \frac{t * (p/2)}{T} \tag{4.1}$$

In the case of Bitcoin, the average number of TXs per block is 2,700 TXs as announced on Mar. 29. 2019 [239]. While the probability of forking evaluates to approximately $(5.54 * 10^{-6})$. This is about 0.000554% i.e. we'd expect two blocks to occur within two seconds once every 180k blocks [240]. The total average TXs per day announced on Nov. 03. 2019 is about 290 thousand TXs [241]. Here we can calculate R /day as: $2700 * (0.000554/2)/290000\% = 2.6 * 10^{-6}$.

Out of the box, the concept of Un-reliability can be extended using other factors to indicate how reliable and trust-worthy the evaluated system is. Factors may include the probability of attacks, such as 51% attack [242] or selfish mining attack [243], encryption deployment, or the privacy-awareness in the system all in all [237]. For instance, some famous BC systems, like Bitcoin and Ethereum, do not encrypt network messages [244], which shall negatively affect the level of trust in the system due to the lack of privacy, which is a foundation principal in some applications [96]. Private keys in those systems, however, are specifically encrypted using the Elliptic Curve Digital Signature (ECDS) which requires $((2^2)^5)^6$ trials in order to successfully fraud a signature. This is computationally impossible [20], which shall positively affect the level of trust in the system.

4.4 Quantifying Distributed Ledger Consistency

Here, I use the FoBSim tool for mimicking BC operations both in the end-user and fog layers. Deactivating the gossip functionality in FoBSim, I could detect the appearance of a fork during a simulation run. That is, more forks appearing in the ledger indicate higher levels of DL inconsistency under the simulated scenario conditions. Manipulating simulation conditions facilitates the analysis of direct effect of those

conditions on DL consistency.

In each simulation scenario, I oscillated the configuration of one condition and stabilized the others. I ran each simulation scenario five consequent times under the same conditions and computed the average number of chain versions. The five parameters that were oscillated are the number of miners M, the number of neighbors per miner N, the puzzle difficulty Ω, the number of simultaneously mined blocks β, and the average RTT between neighbors τ. Table 4.1, and the second column of Table 4.2 present the configuration of different parameters for each simulation scenario.

Table 4.1: *FoBSim configuration parameters*

Scen.	(M)	(N)	(Ω)	(β)	(τ) (in ms)
1	100,500,1000,1500	2	20	10	0
2	500	2,3,5,8,15	5	10	0
3	1500	2	5,10,15,20,25	10	0
4	500	2	15	2,3,5,8,12,18	0
5	1500	2	25	10	0,5,10,15

I defined a set $P(\beta)$ that contains all chain versions that can be possibly formed, independently, out of all confirmed blocks. By referring to the Probability Theory and the principles of Enumerative Combinatorics, the number of elements in $P(\beta)$ can be computed according to Equation 4.2. Note that the order of blocks in a given chain matters, which increases the number of elements in $P(\beta)$ even more than $\beta!$.

$$|P(\beta)| = \sum_{k=0}^{\beta} \frac{\beta!}{(\beta - k)!} \tag{4.2}$$

let δ be the number of chain versions obtained at the end of a simulation run and ξ be the maximum δ that can appear at the end of a simulation run. It is trivial that $\delta = \xi$ in a given scenario is the worst case in terms of DL consistency, while $\delta = 1$ is the best case. Naturally, ξ shall be equal to the number of miners. However, if M is large enough and β is relatively small, ξ shall be equal to the number of elements in the set $P(\beta)$. Consequently, as long as $M > 0$ and $\beta > 0$, ξ is determined according to Relation 4.3:

$$\xi = \min\{M, |P(\beta)|\} \tag{4.3}$$

Quantitative conclusions, then, can be drawn regarding the inconsistency Y of a DL by calculating the ratio δ to ξ. This can be formalized using Equation 4.4.

$$Y = \delta/\xi \tag{4.4}$$

As one cannot obtain an inconsistency level of 0% using Equation 4.4, Equation 4.4 can be modified into Equation 4.5.

$$Y' = \frac{\delta - 1}{\xi - 1} \tag{4.5}$$

I attempt to simulate a probabilistic finality system represented by the PoW algorithm because it is useless to assess the consistency of an absolute finality system as it offers a perfect consistency with the cost of lower security. Accordingly, the puzzle difficulty Ω is the number of Zero's at the beginning of $f(x)$, where $f(x) = H(x \oplus S)$ ($H(.)$ is a hash function, x is data being mined, and S is a random integer being searched for by the miner). I conducted the following experiments on the Google Cloud Platform using an E2-standard-32 VM instance (up to 3.8 GHz, 32 vCPUs, 128 GB memory) running a Ubuntu 20.10 OS. Table 4.2 concludes the results I obtained, where δ and Avg. δ are presented. The table also concludes the general observed effect of each factor oscillation on δ.

For Scenario-1, simulation shows that δ is proportional to the number of miner nodes participating in the BC network. For Scenario-2, simulation shows that δ is inversely proportional to the number of neighbors per miner, as long as the ratio $N : M \leq 1\%$. This result is similar to the results presented in [180], because the impact of N was studied for a maximum of four, and with a network size of 500.

However, I found that for the case where the ratio $N : M > 1\%$, δ is directly proportional to N. This observation is somewhat consistent with the observations presented in [202]. That is, it was argued that $N < \log M$ may increase the number of forks in a given BC, due to several weak links acting as bottlenecks. Accordingly, it was recommended that N shall be set to $\geq \frac{M-1}{M} \log M$. Taking the Bitcoin network as an example, the authors argued that it is safe, with regards to N, since it operates within a stable range of 22–99 connections (per Full miner nodes). However, I argue that although such range is safe to guarantee partition tolerance, it is not optimized in terms of consistency. As [202] sets the recommended lower bound of N for safety, my results recommend an upper bound of N, for optimization, to be $N \leq \lceil M/100 \rceil$.

For Scenario-3, simulation shows that δ is inversely proportional to the puzzle solution difficulty. Such result is naturally expected for a system with a probabilistic finality. Increasing Ω is the BC solution to decrease the probability of $\beta > 1$. Additionally, the increment of Ω provides sufficient window for miners to gossip, and compensates for the continuous enhancement of mining machines. Such continuous enhancement (predicted by Moore's law [245]) may lead to faded effect of a static Ω through time. From another point of view, compensating the advancement of computational capacities of mining machines only by increasing Ω implies higher energy consumption through time as discussed earlier.

For Scenario-4, the simulation shows that δ is directly proportional to the number of blocks simultaneously mined and broadcast in the BC network. In the case where $M \leq 500$, simultaneous blocks appearing in the network are tolerated in terms of Y

Table 4.2: *Number of chain versions at the end of each simulation run, the average number of chain versions for each scenario, and the observed effect of oscillating the corresponding factor on the average number of chain versions*

Scenario	oscillated factor	run-1	run-2	run-3	run-4	run-5	Average δ	Effect
1	M = 100	1	1	1	1	3	**1.4**	
	M = 500	5	1	4	2	2	**2.8**	↑
	M = 1000	37	11	4	2	9	**12.6**	↑
	M = 1500	26	57	54	28	24	**37.8**	↑
2	N = 2	91	105	78	69	91	**86.8**	
	N = 3	87	66	42	65	79	**67.8**	↓
	N = 5	45	50	53	65	64	**55.4**	↓
	N = 8	117	73	71	45	71	**75.4**	↑
	N = 15	417	418	409	374	413	**406.2**	↑
3	Ω = 5	138	125	142	134	144	**136.6**	
	Ω = 10	143	123	128	126	142	**132.4**	↓
	Ω = 15	129	135	125	140	136	**133**	↑
	Ω = 20	22	14	20	9	31	**19.2**	↓
	Ω = 25	1	1	1	8	1	**2.4**	↓
4	β=2	3	3	3	3	3	**3**	
	β=3	4	3	4	4	3	**3.6**	↑
	β=5	66	72	42	52	89	**64.2**	↑
	β=8	38	51	39	62	58	**49.6**	↓
	β=12	393	376	389	405	379	**388.4**	↑
	β=18	459	461	469	466	460	**463**	↑
5	τ = 0 ms.	2	5	2	1	17	**5.4**	
	τ = 5 ms.	26	47	7	24	6	**22**	↑
	τ = 10 ms.	21	46	2	6	40	**23**	↑
	τ = 15 ms.	31	72	37	11	103	**50.8**	↑

up to $\beta = 8$. This can be justified by the exponential growth rate of ξ, which compensates the linear growth of δ and M, and almost hides the effect of the increasing β. Once the value of ξ is switched to be equal to M (according to Equation 4.3), the actual effect of β can be clearly noted.

Lastly, for Scenario-5, the simulation shows that δ is directly proportional to the average transmission delay between neighboring miners. These results conform with the proportionality characteristic presented in [246], where higher transmission delays predicted higher forking probability. Furthermore, it agrees with [247], where it was shown that lower delay between BC miners implies higher efficiency in terms of consistency.

Accumulating my findings, I could propose Relation 4.6, where η is $1/N$ if $N/M \leq 1\%$ and is N otherwise.

$$\delta \propto \frac{M \times \eta \times \tau \times \beta}{\Omega} \tag{4.6}$$

Solving for Y using Equations 4.2, 4.3 and 4.4, I could compute Y as a percentage. For δ and ξ values, I use the average values in Table 4.2 and educe the relevant values of M and β.

Figure 4.1 presents the results I obtained from each scenario assessment, in terms of percentage Y. The figure presents the average number of chain versions according to the simulation of Scenarios 1–5, and maximum possible number of chain versions that could appear during the simulation, represented by blue bars and orange bars, respectively (correlated with the primary y-axis on the left). The figure also depicts the percentage value of Ledger inconsistency, represented by the grey curve (correlated with the secondary y-axis on the right). According to these results I could deduce that Relation 4.7 applies, where λ is M if $M \leq |P(\beta)|$, and is $1/|P(\beta)|$ otherwise.

$$Y \propto \frac{\lambda \times \eta \times \tau \times \beta}{\Omega} \tag{4.7}$$

One can also notice the individual relative effect of each analyzed factor. That is, increasing M can indeed decrease the ledger consistency level, yet it is not as effective as increasing the number of neighbors per miner (e.g. adding 500 miners to the network increases Y with about 1%, while changing the number of neighbors per miner from 8 to 15 increases Y with about 60%). Furthermore, increasing M leads to increasing both δ and ξ in case $\xi = M$, while it only leads to increasing δ in case $\xi = |P(\beta)|$. Such notion can particularly justify that increasing M does not strongly affect Y as increasing M participates in increasing and decreasing Y in parallel. Additionally, it justifies the results of Scenario-4, where β increases yet Y decreases. According to the results of Scenario-1, increasing M can guarantee higher ledger consistency (in the cases where $M \leq 500$), as the effect of M is higher on ξ than it is on δ.

4.5 Optimizing Neighbor Selection in BC Networks

4.5.1 Preliminaries and problem statement

Similarly to [248], let a public-permissionless BC network be a connected, undirected, and weighted graph $G = (V, E, w)$, where V is the set of nodes in G representing miner nodes, E is the set of edges in G, representing the communication lines between the miners, where each $e_{i,j} \in E$, connecting exactly two nodes $i, j \in V$, can be traveled in both directions. Nodes of G communicate by message passing via (strictly) the edges of G. Each $e \in E$ is associated with a distinct non-negative value,

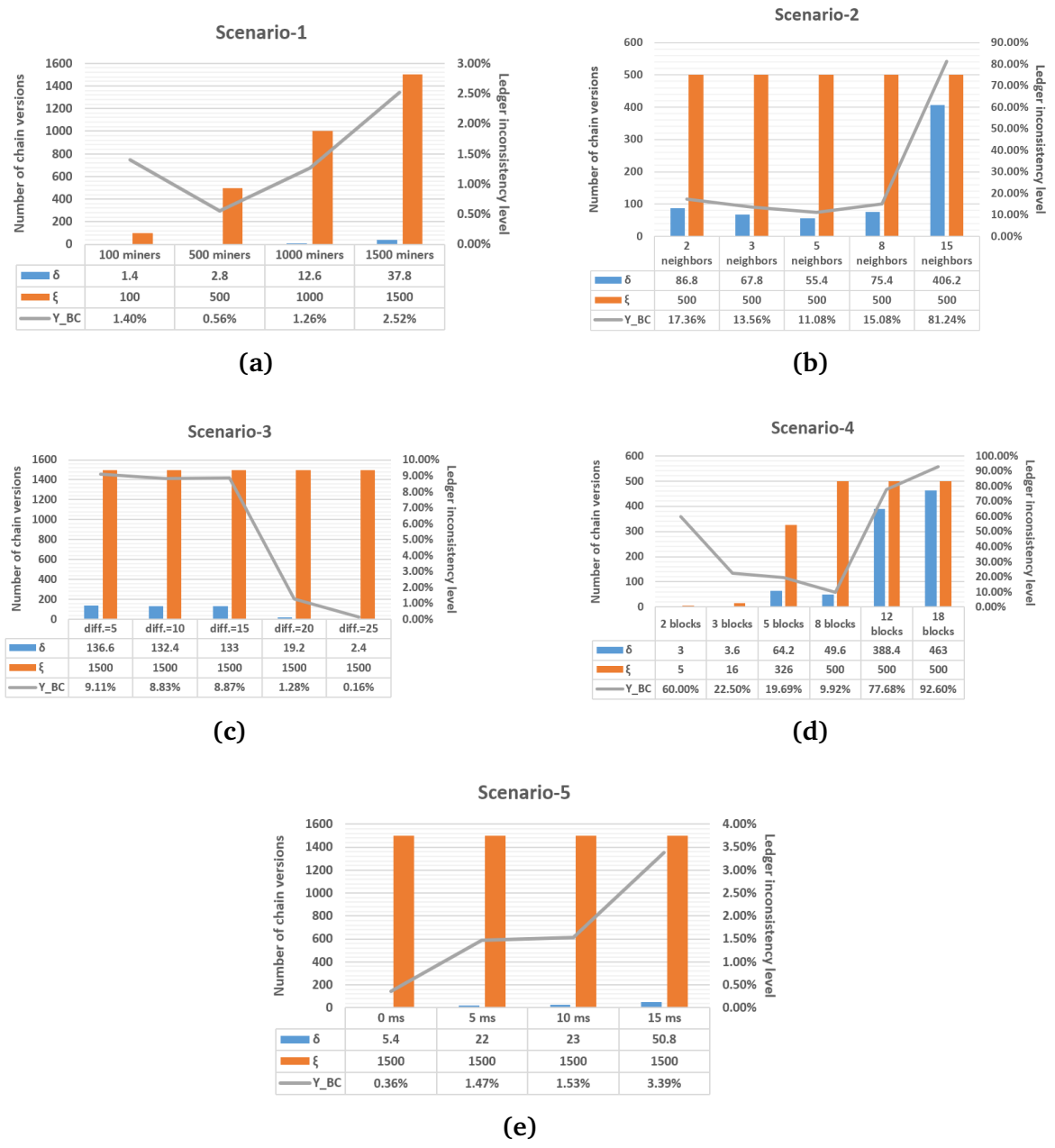


Figure 4.1: *Average number of chain versions in Scenarios 1–5, maximum possible number of chain versions that could appear during each simulation, and the percentage value of Ledger inconsistency*

namely weight ($w_{i,j}$ or w_e), which represents the transmission time needed to deliver 1 bit of data from node i to node j or vice versa, computed in μs.

The weight of any given graph is the sum of the weights of all its edges. Let the set of neighbors of a node $i \in V$ be $m_i = (m_{i,1}, ..m_{i,j})$. I assume that every node

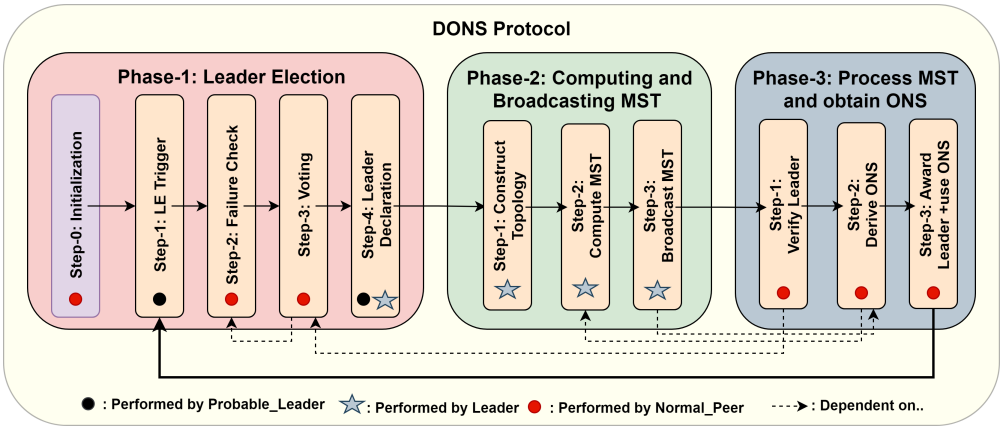


Figure 4.2: *Phases and steps of the proposed DONS protocol. Each step is performed by one (or more) system entity(s), and may depend on a preceding steps' results (in previous or current round*

$i \in G$ is initially aware of its m_i, and is aware of the weight associated with each edge $e_{i,j} \forall j \in m_i$. To mathematically represent a graph, one can use the adjacency matrix, which is a matrix of size V $\times$ V whose elements are the weights $w_{i,j}$ if there is $e_{i,j}$.

A sub-graph of G is any graph $G'(V', E', w')$, such that $V' \subseteq V$ and $E' \subseteq E$. G' is also connected, undirected, and weighted as it inherits the properties of the original graph. A Spanning Tree of G is G' where $V' = V$ and $E' = V - 1$. A MST of G (with distinct $w_e \ \forall e \in E'$) is a unique Spanning Tree where the weight of MST is minimum compared to all Spanning Trees of G.

A hashing function, or a one-way encryption function, $h(.)$ is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence that is computationally difficult to invert [249]. A hashing function enables the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest[250].

Next, I attempt to find the Optimum Neighbor Selection (ONS) which I define as the subset $k_i = (m_{i,s}, ..m_{i,t}) \in m_i$, $\forall i, s, t \in V$, such that $e_{i,k} \in MST_G \ \forall k \in k_i$. I aim at solving this problem using a protocol that fulfills the following privacy condition:

$$\forall i \in V \Rightarrow \sigma'_i = \sigma_i + ONS_i \tag{4.8}$$

where σ_i is the total knowledge of miner i before starting the protocol and σ'_i is the total knowledge of miner i after the protocol is terminated.

4.5.2 The Proposed DONS Protocol

Next, I propose a Dynamic Optimized Neighbor Selection protocol, namely DONS, and I discuss the proposed algorithms and methods for each phase of the protocol. The generalized framework addresses a public permissionless BC with no TTP, and initially assumes all network entities are honest. However, I discuss counter assumptions where applicable. The phases and steps of the DONS protocol are demonstrated in Figure 4.2. Due to the page number limitations of this thesis, only major descriptive discussions are provided here. More comprehensive details are provided in [234]. Due to the limited page number of this thesis, I omitted the full evaluations and discussions and only present major evaluation results in terms of complexity, privacy, finality time [195] and fidelity [180]. The detailed evaluation discussion and results, along with the algorithms of the proposed protocols, are provided in [234].

Phase-1: Leader Election

The protocol requires a global view of the underlying BC network, so that the MST can be computed. Miners joining and leaving the network implies that this global view, and accordingly the computed MST, shall be regularly updated. In my studied BC model, all miners have the same access permissions and the same level of abstraction. One (or a committee) of these miners may perform the MST computations for all other miners. This way, the network decides best practices regarding networking and gossiping without administrative interference, which leads to Smart Networking [251].

To fulfill this condition, I propose the Anonymous Leader Election (AnoLE) protocol which shall not violate any of the comprehensive NS solution criteria discussed previously. The elected leader shall collect non-private local views from all peers, construct a network demonstration, solve the DMSTP of the network graph, and finally broadcast an anonymized MST. The recipient nodes shall only be able to read their own, and their neighbors' IDs. Thus, neither the leader nor any other network entity can deduce miners' private data throughout the run of the protocol. Note that this implies that a miner does not know the identity of the leader, unless the miner itself (or one of its neighbors) was the leader. The generalized workflow of the proposed AnoLE protocol is depicted in Figure 4.3. Different system entities utilize the AnoLE protocol as follows:

- Step-0 (Initialization): All nodes know their neighbors identities and the corresponding RTT expected when communicating with each of them. All nodes use this protocol honestly, with default status 'Normal_Peer', Default Required Confirmations (DRCs) equals the average number of neighbors per peer, 'Current_Leader' = null, default round time T, and MST set to empty array.

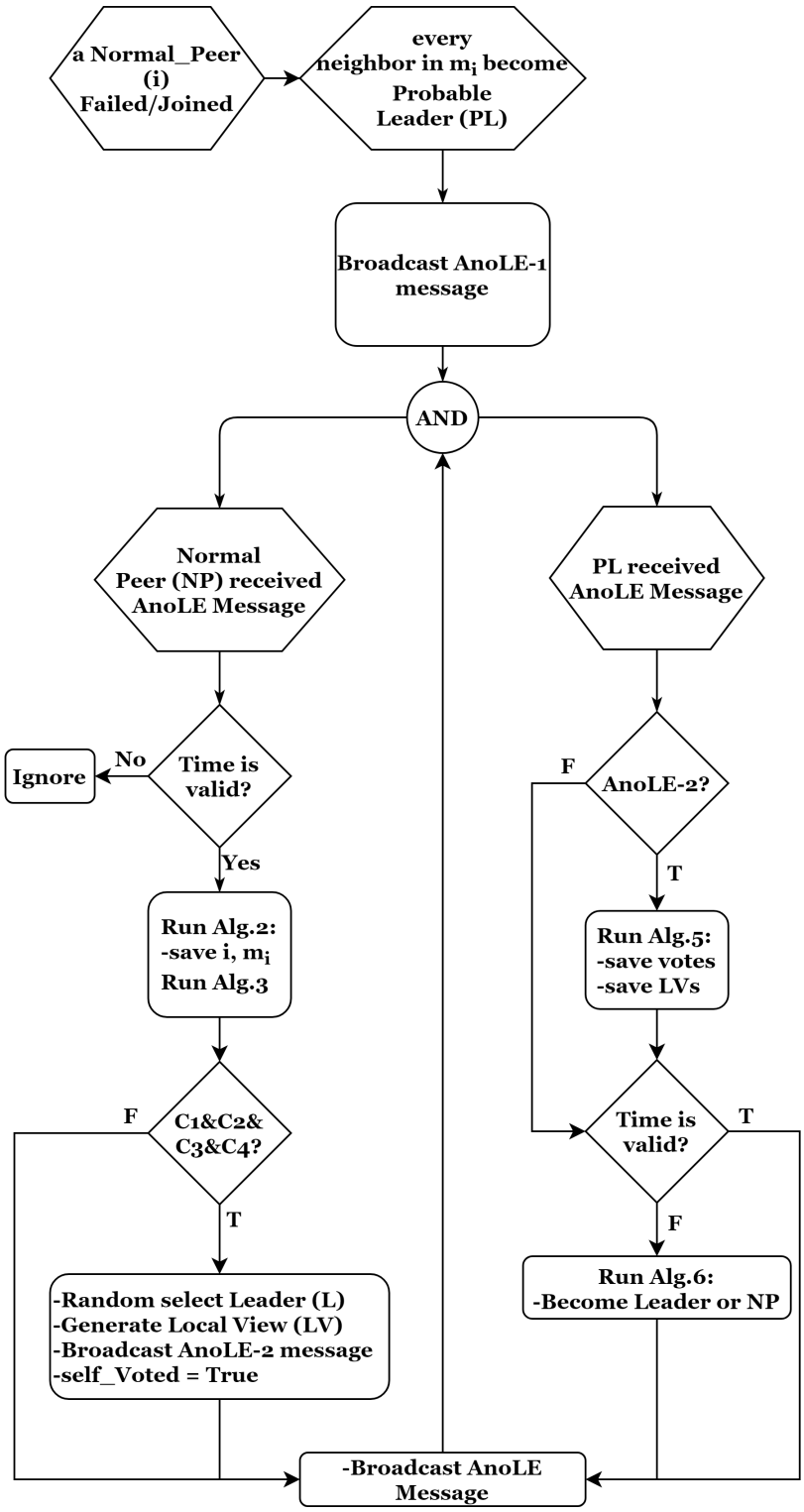


Figure 4.3: *Workflow of the proposed Anonymous Leader Election (AnoLE) protocol*

- Step-1 (LE trigger): Once a node i fails/joins the network, its neighbors, denoted $m_i = \{m_{i,1}, ..m_{i,j}\}$, are triggered to start the AnoLE protocol. Each neighbor $m_{i,k} \forall k \in 1, ..j$ sets its status to 'Probable_Leader', sleeps for an arbitrary time (default setting: waiting time is randomly selected between 0 and $T/2$), and sends 'AnoLE-1' message to all neighbors. The 'AnoLE-1' message contains $h(i)$ and $h(m_{i,k})$, along with timestamp t. votes = Dict{} and LVs = list[] are initiated to later save the responses of the AnoLE-1 message.

- Step-2 (Failure check): all nodes that receive 'AnoLE' messages wait to receive the Required Confirmations (RCs), a list of hashes of nodes in the set m_i, and the Local View (LV), respectively. These recipient nodes wait until they receive a number of distinct AnoLE messages equal to RCs (if T time units passed with no sufficient AnoLE messages, the node does not vote). The recipient checks three conditions to consider the node failure/joining reports correct: 1) Every distinct AnoLE message shall contain similar $h(i)$ and different $h(m_{i,k})$. These messages represent failure/joining proofs 2) and each $h(m_{i,k})$ should belong to the list of neighbors returned by an algorithm that assures reports are only sent by genuine neighbors and 3) All neighbors in this list shall send an AnoLE-1 message. This represents a consensus among neighbors on the honesty of the AnoLE protocol trigger (i.e. those neighbors do not know or trust each other by assumption).

- Step-3 (Voting): Once a recipient node receives a sufficient number of 'AnoLE' messages that fulfill the conditions in Step-2, the recipient can be sure that i has indeed failed/joined as all its neighbors witnessed. The NP then selects one of the received $h(m_{i,k})$s according to a predefined criteria (e.g. randomized, first sender, highest hash value, etc.) and modifies its 'Current_Leader' to the selected $h(m_{i,k})$. After that, NPs broadcast 'AnoLE-2' messages to all its neighbors, which contain their hashed IDs along with the contents of 'AnoLE-1'. AnoLE-2 messages, then, declare that NPs who generated them vote for, specifically, the candidate leader whose hash is included in their AnoLE-2 message. In the context of the DONS protocol, NPs also deposit their current LVs of the network into their generated AnoLE-2 messages. NPs who have a previous version of the MST (i.e. obtained from previous AnoLE protocol run), may utilize it to share their AnoLE messages with their ONSs. A condition to be fulfilled in order to utilize the previous ONS, however, is that non of the ONS members has an ID whose hash is equal to $h(i)$.

- Step-4 (Leader Declaration): Whenever a message of type 'AnoLE-2' is received by a 'Probable_Leader', it runs an algorithm specifically implemented for saving new votes and LVs. Once a 'Probable_Leader' finds that: current_time - t $\geq T$,

it counts the votes received so far and converts the node's status into either 'Leader' or 'Normal_Peer'.

Note that a generated/received AnoLE message might be sent to all neighbors, which implies that all probable leaders shall eventually know the winner leader if T was sufficient. However, a subset of the network might not have enough time to vote for a leader and receive its MST. This seems OK as nodes use their ONS if available, and broadcast otherwise. With several runs of the protocol, and dynamic modification technique of T, T would become more precisely adequate/sufficient. A simple modification technique of T can be defined according to application requirements. For example, nodes may assume that not receiving the MST from the leader they voted for indicates insufficient T. Thus, those nodes may double T for next rounds. Receiving the MST sooner than the end of T, on the other hand, indicates that T is bigger than needed. Thus, nodes may compute the average of T and the time elapsed from voting till receiving the MST.

The AnoLE protocol utilizes the Epoch time which implies that the location of miners, and distinct transmission delays would not impose a synchronization problem. All nodes thus use the same reference time and all nodes will track T accurately. Hence, all nodes will initiate/terminate the protocol according to unified timestamps.

Phase-2: Computing and Broadcasting MST

Assuming T was sufficient for all NPs to vote and for all PLs to receive those votes, I anticipate that by the end of Phase-1, the Leader (L) is recognized by all PLs and by the majority of network miners. Each PL returns to the state 'NP' except for L. Consequently, Phase-2 is triggered and is performed by L as follows:

- Step-1 (Construct NT): L uses its locally saved LVs to construct the anonymized global network topology (NT), represented by an adjacency matrix NT.
- Step-2 (Compute MST): L utilizes Prim's approach [229] to find MST_{NT}. Note that any other (perhaps better) approach can be utilized here, e.g. [252, 253].
- Step-3 (Broadcast MST): Lastly, the Leader derives its own ONS from the MST it built, as described in Step-2 of Phase-3, and uses it to send the MST to its neighbors in its ONS. The MST is encapsulated in an AnoLE-3 message, which also contains $h(L)$ and the time of MST generation.

Phase-3: Processing received MST to get ONS

- Step-1 (Verify L): Once a NP receives an AnoLE-3 message, it checks whether this message was generated by the leader it previously voted for. The assumption of an adversary node impersonating the real leader is valid. However, such

impersonation probability may be solved using asymmetric encryption techniques, where the leader couples a public key with its AnoLE-1 message. Later, the leader can sign the AnoLE-3 message using its private key. This step also implies that the recipient NP is expecting to be present within the proposed MST. Otherwise, this NP will not accept the MST despite it was sent by the leader the NP elected.

- Step-2 (Derive ONS$_i$): Every miner (including current L and previous PLs) that receives a verified MST (within an AnoLE-3 message) derives its own ONS, which is utilized then to optimally select and share data.

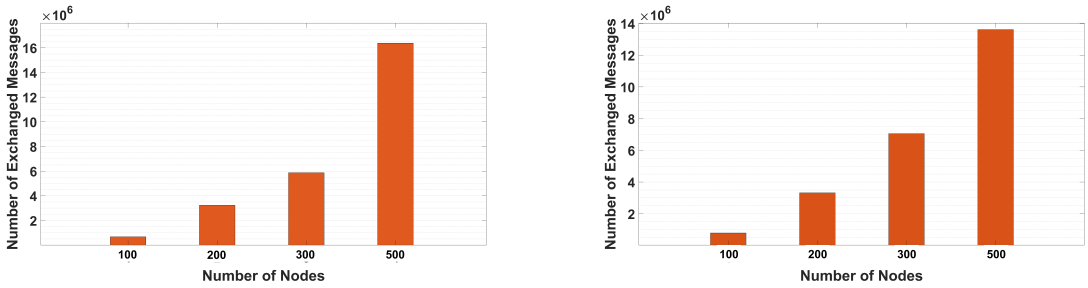
- Step-3 (Award L and utilize ONS$_i$): In case the leader shall be incentivized for its work, the leader may include its wallet id within its AnoLE-1 message. The leader then includes this piece of information within its signature, which adds another layer of verification. Miners which receive the AnoLE-3 message award the leader by adding a predefined amount of digital assets into the leader's wallet. Note that in this case, the AnoLE-3 message also represents a TX that needs not to contain the leader's wallet ID nor its public key because they have already been shared within the AnoLE-1 message.

4.5.3 Evaluation

I have calculated the computation and message complexities of the AnoLE protocol and found them both require a maximum of $O(|V|^3)$. I have implemented the AnoLE protocol using Python 3.8 with utilization of popular packages such as multiprocessing, threading, networkx, hashlib, among others. My implementation is publicly available at GitHub[1]. To evaluate my implementation of AnoLE, I performed several experiments utilizing two random network models, namely Erdős-Rényi (ER) [205] model and Barabási-Albert (BA) [206] model. I oscillated the number of nodes to capture the protocol behaviour within different network sizes. The experiments were carried out on a DELL PC with an Intel i5-8265U CPU (8-Cores, 3.8 GHz) with 12 GB DDR4-SDRAM, 500 GB of SSD and Windows-10 OS. The simulation results are depicted in Figures 4.4 and 4.5. Detailed configuration I used for running the experiments, along with exact simulation results, are presented in Table 4.3.

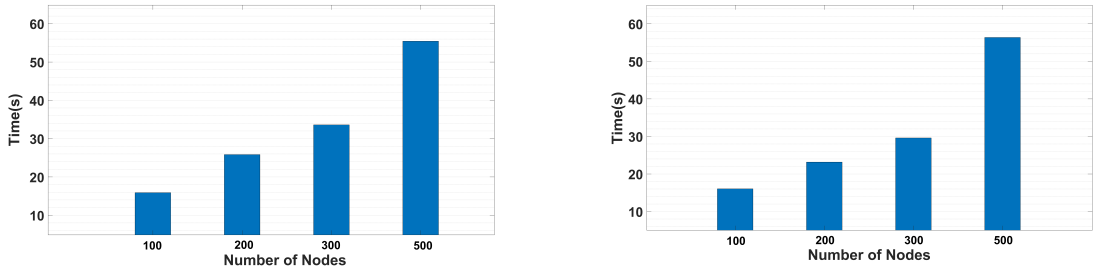
To highlight the outperformance of the DONS protocol, I compare experimental results with two previously discussed NS approaches, namely RNS and RTT-NS. Similarly to the AnoLE evaluation approach, I oscillated the configuration of network size and average number of neighbors per miner, to demonstrate the consistency of my previous analysis with real-life scenarios.

[1]https://github.com/HamzaBaniata/AnoLE_Protocol



(a) *Barabási-Albert* **(b)** *Erdős-Rényi*

Figure 4.4: *Total number of exchanged messages for running the AnoLE protocol until delivering a connected MST to nodes of two random network models with different sizes*



(a) *Barabasi-Albert* **(b)** *Erdos-Renyi*

Figure 4.5: *Required time (seconds) for running the AnoLE protocol until delivering a connected MST to nodes of two different random network models with different sizes*

Specifically, I developed a Python v:3.8 network simulator, where a random BC network is built and a randomly selected miner represents the source node of a block of data. The generated block is then shared by the source node with a group of neighbors, each of which shares the block similarly with a group of its neighbors, etc. The simulation terminates once the block reaches all nodes of the network, mimicking the push-based gossiping approach generally adopted by all BC applications. The compared three NS methods are utilized consequently using the same network for the same block being generated by the same source node. At the termination of each simulated scenario, total finality time and number of redundant messages are calculated. As miners in public-permissionless BC networks are randomly connected, analyzing the results obtained from running our developed simulator indicates the best NS approach in terms of the evaluation criteria. Consequently, I could experimentally prove that the adoption of such NS approach leads to enhanced DL consistency. My implementation workflow is demonstrated in Figure 4.6.

Finality is the assurance or guarantee that data cannot be altered, reversed, or

Table 4.3: *Results of the AnoLE protocol simulation experiments on two random network models with different sizes (N: number of neighbors per miner, p: connection probability, DRT: default round time, X Msgs: Number of exchanged messages)*

Network	size	parameter	DRT	DRC	**time(s)**	**X Msgs**
BA	100	$N = 2$	30	2	**15.9**	**676,071**
	200	$N = 2$	40	2	**25.84**	**3,237,133**
	300	$N = 2$	60	1	**33.6**	**5,871,463**
	500	$N = 5$	60	2	**55.44**	**16,380,174**
ER	100	$p = 0.05$	30	2	**16.05**	**788,139**
	200	$p = 0.02$	40	2	**23.18**	**3,321,310**
	300	$p = 0.015$	50	2	**29.6**	**7,054,606**
	500	$p = 0.01$	60	2	**56.38**	**13,617,211**

canceled after they are completed [254]. To achieve optimal finality in a given BC, shared data needs to be spread as soon as possible through the BC network, so that miners can adopt this data before a new piece of data is generated. The latency level of a BC shall, then, ultimately affect its finality rate. Fidelity, on the other hand, is the degree to which a technique can provide consistency guarantees [180]. To evaluate DONS in terms of fidelity, I designed my implementation to count the number of cycles a generated data walks when utilizing DONS, RNS and RTT-NS by counting the number of replicated messages they receive. More cycles indicate the overhead on network connection links and overhead in computation at the node level, resulting in higher overall finality time.

I made my implemented simulator code publicly available at a GitHub repository[2]. I run several simulation scenarios, as described in Table 4.4. The results of the experiments are presented in Table 4.4 and Figures 4.7 and 4.8.

4.5.4 Privacy analysis

Next, I discuss the identity privacy preservation [255] provided by the proposed AnoLE and DONS protocols. As detailed earlier, the proposed methods must guarantee the privacy condition (4.8). Following the description of the proposed protocols in the previous sections, one can notice that data shared between network nodes are exchanged in the form of AnoLE messages. For any given node a, it can receive the three types of AnoLE messages generated by all, or a subset of, network nodes.

The AnoLE-1 message includes the hash of the node i id that left/joined the network, the hash of its neighbor j id, and the timestamp of the message. According to the definition of a hash function, node a cannot obtain any private information about

[2]https://github.com/HamzaBaniata/DONS_simulator

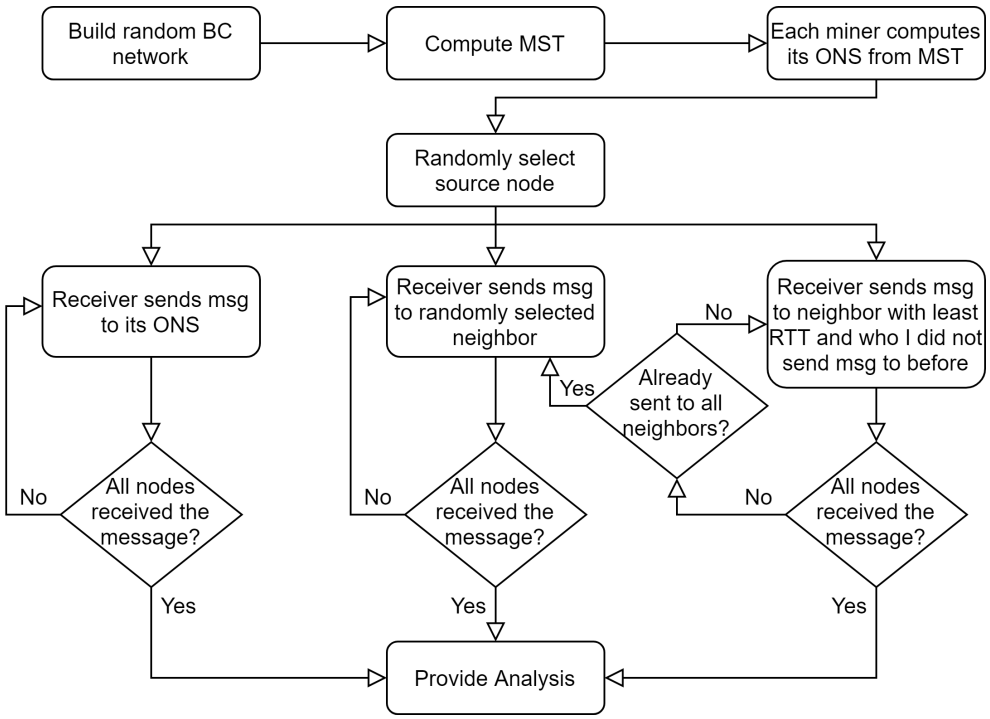
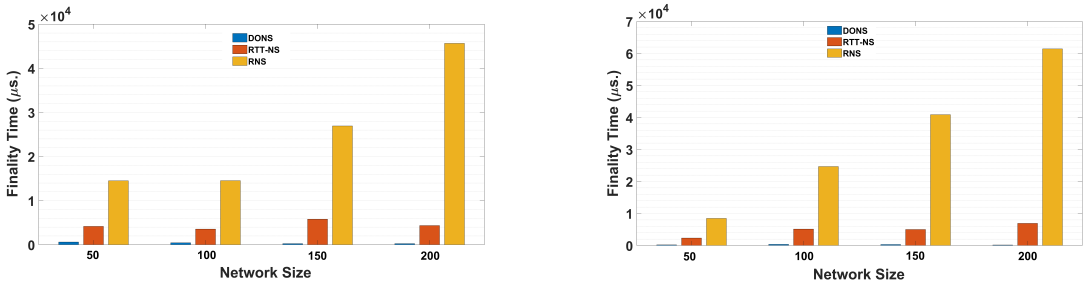


Figure 4.6: *Simulation workflow for push-based gossiping in BCs utilizing DONS, RNS and RTT-NS protocols*



(a) *Erdos-Renyi network model with connection probability = 0.1*

(b) *Barabasi-Albert network model with avg. no. of neighbors per miner ∈ [5,10]*

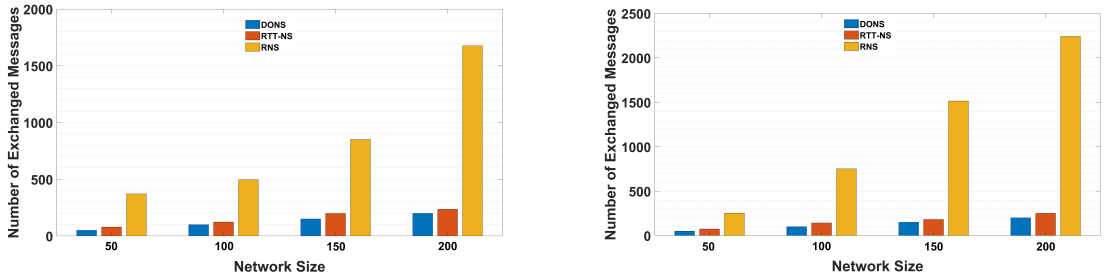
Figure 4.7: *Total Finality time of a randomly generated block by a randomly selected miner, in two random network models*

i or j from AnoLE-1 messages.

The AnoLE-2 message is similar to the AnoLE-1 message, with the addition of the hash of a's id, the hash of the elected leader id, and the Local View of a (LV_a). LV_a consists of hashed ids belonging to the neighbors of a, in addition to the RTT a has measured between itself and its neighbors. Thus, any other node b can see that a node with id hash $h(a)$ is connected to a number of neighbors with id hashes

Table 4.4: *Performance of the DONS protocol against RTT-NS and RNS protocols, on two random network models with different sizes (N: Number of Neighbors, p: connection probability)*

network	size	Parameter	Finality time (ms)			Exchanged messages		
			DONS	RTT-NS	RNS	DONS	RTT-NS	RNS
BA	50	$N = 5$	204	2,343	8,463	50	74	253
	100	$N = 5$	367	5,139	24,667	100	142	751
	150	$N = 7$	310	5,008	40,870	150	182	1,514
	200	$N = 10$	176	6,927	61,429	200	251	2,241
ER	50	$p = 0.1$	629	4,176	14,510	50	79	371
	100	$p = 0.1$	450	3,558	14,533	100	123	496
	150	$p = 0.1$	257	5,795	26,932	150	199	851
	200	$p = 0.1$	260	4,363	45,656	200	236	1,676



(a) *Erdos-Renyi (ER) network model (with connection probability = 0.1)*

(b) *Barabasi-Albert (BA) network model with avg. no. of neighbors per miner in [5,10]*

Figure 4.8: *Total number of exchanged messages until a randomly generated block, by a randomly selected miner, is delivered to all network nodes*

$h(1), ..h(n)$ with links of some given weights. However, b cannot determine the true identities of a nor its neighbors, which makes the knowledge of weights on the links useless. In the case where a, or any of its neighbors, is a neighbor to b, b can only determine the true identity of its neighbors.

The last type of exchanged messages is the AnoLE-3 message, which includes the hash of the leader id $h(L)$ and the MST. Note that the MST is a collection of reduced LVs and, thus, what applies to the LV knowledge deduction applies to the MST. Note also, that network nodes accept the anonymized MST and deduce their ONSs by comparison and not by decryption. That is, each node constructs a list of hashes of its neighbors and compares these hashes with hashes in the MST. Additionally, nodes compare the hash of the leader they voted for, with the hash of the AnoLE-3

message generator. If the two were compatible, the MST within the AnoLE-3 message is accepted.

As can be noted from this description, by the end of the protocol run, network nodes can only read the true identities of their neighbors. Additionally, even the leader cannot read any true identity in the MST it builds unless it was for itself or for one of its neighbors. Network nodes further vote for leaders, and accept leaders' MSTs without any knowledge of true identities of leaders.

4.5.5 Discussion

Based on the results presented thus far, I could state that my proposed protocols significantly enhance the levels of Finality and Fidelity in the studied networks, due to the provision of globally optimal NS techniques.

Although the time and message complexities for the AnoLE protocol are higher than those of the recently proposed DMSTP solution, I argue that this shall not be problematic in real-life scenarios. That is, the first the AnoLE protocol is only triggered when a node joins or leaves the network. With average miner active time extending to weeks, and even months for many cases, average complexities of the DONS protocol, through long periods of time, shall be decreased. That is, the significant enhancement in terms of Finality and Fidelity once the network nodes utilize the MST, shall positively compensate for the rarely performed high complexities of the AnoLE protocol. Note that in public-permissionless BCs, a new block is generated every few seconds (e.g. Ethereum) or minutes (e.g. Bitcoin), while a node is probably staying online for weeks or months [256]. Furthermore, many miners cooperate in mining pools and warehouses that never leave the network [257].

The adoption of the proposed protocols, then, shall consider the average active time of nodes in the network. That is, relatively rare node failure (e.g. an average active time of nodes equals to a month) implies that the AnoLE protocol is rarely triggered. Accordingly, triggering this protocol may indeed cost much exchanged messages, yet the MST proposed afterwards would definitely enhance data propagation through the network for a long period of time, until a new trigger appears.

For example, let G be a BA network with size 200 nodes, where the average active time of nodes equals to one week (i.e. 168 hours), and a new block is generated every minute (i.e. 1440 block per 24 hours). It can be seen in Table 4.3 that triggering the AnoLE protocol in G would cost nearly 3.2×10^6 exchanged messages. Once the MST is available to G, data can be shared with a total number of exchanged messages equals to 200 per block of data (Table 4.4). As a result, the network would exchange a total of $200 \times 1440 \times 7 \approx 2 \times 10^6$ messages per week. Adding this to the 3.2×10^6 exchanged messages to obtain the MST gives $\approx 5.2 \times 10^6$ messages per week. If this network uses the RNS method to share data, the total number of exchanged messages

per week would be $1440 \times 2,241 \times 7 \approx 22.6 \times 10^6$ messages per week.

The AnoLE protocol is one component within the DONS protocol and can be optimized as well as replaced with a better protocol whenever available, leading to even better results. Nevertheless, the experimental results presented earlier represent the complexities of the AnoLE protocol utilized for the first time. This means that nodes broadcast the messages they receive from their neighbors. However, with multiple leader election rounds run consequently, resulting in larger number of nodes finding their ONSs, total round-time and number of exchanged messages will be significantly decreased. Simply put, the results presented are the upper bound complexities of the AnoLE protocol.

Additional discussions regarding the potential behind the proposed protocols, initial configuration recommendations, and future enhancement directions are discussed in [234].

4.6 Deployment Decision

BC technology can provide services that map to some of the services that Fog-enhanced systems are expected to provide. Depending on the expected characteristics of the fog layer, one can evaluate the benefits of deploying BC in the fog, so that FC services can be extended beyond data pre-processing, monitoring and storage. For example, fog components are expected to be directly connected to each other, which lowers the transmission delay, while the BC nodes, deployed in the end-user layer, are connected through the internet, in a P2P fashion. Thus, it is trivial to expect the enhancement of a BC application in terms of ledger consistency, when deployed in the fog layer instead of the end-user layer.

To evaluate these expectations, we assume two cases, where probabilistic finality based BC is deployed in the end-user layer, and in the fog layer. In both cases, $M = 500$, $N = 5$ and $\Omega = 20$. Meanwhile, the size of the message to be sent to neighbors is unified. Recalling the results presented in [191] and [192], the average transmission delay τ can be set between miners in the first case to 1000 ms, and between miners in the second case to 12 ms, while no jitter is considered in both cases.

The computational power C of fog nodes and end-user miners need to be also considered as described in [211]. That is, computers used for mining by end users have more computational power than it is expected for fog nodes. This is due to the usage of Graphics Processing Units (GPUs). On the other hand, fog nodes usually use their CPUs to perform tasks and provide fog services.

Since M, N and Ω are equal in both cases, relation 4.7 states that τ and C are the remaining effective factors on Y in both deployments. The current state-of-the-art is unclear regarding the exact fog node architecture and whether they have built-in

GPUs, the comparison results remain dependant on the individual case parameters at the time of application. Additionally, some PoW versions, that are described as Memory Hard Puzzles [258], may indeed present fogs to have higher ability to solve the puzzle. That is, even if they actually have less computational power than the average computational power of end-user miners, they may offer higher memory (e.g. *Chia* [259]). In such a case, C represents memory capacity, rather than computational capacity, of compared miners.

To help making the right deployment decision in such situations, I propose the evaluation in terms of Y. That is, one can find the ratio between Y_f and Y_e, where f and e notates the fog and the end-user layers, respectively. This shall result in a positive value Ψ. As C is, by definition, proportional with β, and inversely proportional with Ω [15], a modified version of Equation 4.7 gives:

$$\Psi = \frac{Y_f}{Y_e} = \frac{T_f \times C_f}{T_e \times C_e} \tag{4.9}$$

Where $T = \varsigma \times \tau$ is the average total time of propagation and ς is a constant that reflects different communications criteria between miners. Those criteria may cause additional delays related to average processing/routing delay, average queuing delay, link distance/type/quality, block/packet size, resources allocation, etc.

Equation 4.9 can be used then to make a reliable decision regarding the deployment of the BC in e or f. Assuming all factors other than τ and C are equal, if the value of Ψ is less than 1, then the BC is better be deployed in f, because such deployment guarantees higher DL consistency. Otherwise, the BC is better be deployed in e. The single critical point, where both e and f deployments are expected to provide equivalent Y measures, can be utilized as follows:

$$1 = \frac{12 \times C_f}{1000 \times C_e} \Rightarrow 12 \times C_f = 1000 \times C_e \Rightarrow C_e = \frac{1.2}{100} \times C_f$$

Consequently, if average C_e is less than 1.2% of average C_f, then it is better, in terms of Y, to deploy the BC in e. Otherwise, the BC is better be deployed in f.

On one hand, Equation 4.9 describes a trade-off between T and C, as T is expected to decrease, which strengthens DL consistency, while C is expected to increase, which weakens DL consistency (or at least triggers reactions for maintaining it, such as increasing Ω). Similar exclusive trade-off observations were discussed in [203], where the relation between mining costs and queuing delays was discussed. On the other hand, Equation 4.9 describes a race condition between the technology enhancement in e devices against f devices, which shall boost the optimization of BC deployment in terms of DL consistency.

4.7 Conclusions

In this chapter, I have discussed and analyzed the concepts and effective factors of the consistency of Distributed Ledgers (DLs) in Blockchain (BC) systems. I designed various simulation scenarios to accurately capture why and how data in a DL becomes consistent or inconsistent. I used the FoBSim tool to simulate my proposed scenarios and measure maximum possible number of chain versions in a given BC. Accordingly, I proposed a quantitative method to describe the DL inconsistency using the principles of enumerative combinatorics in probability theory. I further deployed this method to contribute to a decision making model, which can determine the optimal deployment features of a BC in a fog-enhanced system, depending on information regarding the average computational power or memory capacity C, and the average transmission delay between miners T. The proposed model describes the trade-off between T and C, and the race between technologies deployed in the fog layer versus the end-user layer.

I have also addressed the Neighbor Selection Problem (NSP) for public permissionless BCs by proposing a Dynamic Optimized Neighbor Selection protocol called DONS. As a first step of the DONS protocol, a leader needs to be elected in order to perform additional computations. To this end, I proposed an Anonymous Leader Election protocol called AnoLE, that aims at electing a leader in a distributed fashion, without any previous knowledge of the network size or nodes private identities. By the end of the AnoLE protocol, a leader is announced to perform the following computations including the construction of an anonymized network topology, from which it computes the Minimum Spanning Tree (MST) of the network. An Optimum Neighbor Selection (ONS) is then privately derived from the MST by the leader and the rest of network nodes. Each node utilizes its derived ONS to communicate with the least number of neighbors, but with optimized communications paths. I analyzed the security and privacy of the proposed protocols, and provided the time and message complexities of their algorithms. Additionally, I provided publicly available implementations of both protocols, which I used to experimentally validate my proposals. The experiments showed significant enhancement of message propagation for different network models and sizes, in terms of finality and fidelity, compared to similar networks utilizing state-of-the-art methods.

Future plans include the investigation of multi-leader scenario and its implications on the security and the efficiency of the DONS protocol. Additionally, my current proposal of the AnoLE protocol does not utilize a compatible privacy-aware leader *incentivization* mechanism. Thus, I plan to investigate and deploy a suitable mechanism, and evaluate the trade-offs that need to be tuned. I will focus on some interesting previous works solving similar challenges, e.g. [260, 261, 262, 263, 264]. Finally, I

will research the possibility of upgrading the purpose of the DONS and AnoLE protocols into a comprehensive consensus protocol for public-permissionless BCs, turning a PoW-based BC into a PoUW-based BC.

I am responsible for the following contributions presented in this chapter:

II/1. I formalized and quantified the concepts of Blockchain network consistency and reliability.

II/2. I developed two novel protocols (DONS and AnoLE) for the neighbor selection problem, targeting optimal block finality and privacy-preserving data propagation in public and permissionless Blockchains.

II/3. I experimentally proved that Blockchain and Fog Computing integration is advantageous.

II/4. I contributed to a decision-making model to facilitate making the right deployment decision of BCs in fog-enhanced systems.

Chapter 5

Optimization Methods for Fog-Blockchain integrated systems

In this chapter, I propose two solutions for enhancing BC-based systems using FC, and for enhancing FC-enabled systems using BC technology.

5.1 Introduction

A challenge that generally appears in cloud-based systems is task scheduling. In smart city applications, for instance, there is a large number of tasks and/or Virtual Resources (VRs), which exposes task scheduling as an NP-Hard problem. Consequently, it is preferred and more efficient to use a task scheduling automation technique. As there are many automated scheduling solutions proposed, new possibilities arise with the advent of Fog Computing (FC) and Blockchain (BC) technologies. In order to accomplish the required effectiveness of data management in an IoT-Fog-Cloud environment, the deployment of highly efficient task scheduling schemes is essential.

Task scheduling is the method by which tasks are assigned to resources that perform the computations [265]. This assignment problem has been shown to be NP-Hard problem [266, 267], because the more tasks assigned, the higher the number of probable assignment schedules [268]. To find the best assignment of requested tasks to the available resources, one needs to generate $n!$ assignment schedules, where n is the number of tasks, and find the optimum assignment out of them. For example, a bakery production line that processes 40 products on 26 production stages, leads to 8.2×10^{47} different possible schedules, which is not solvable in reasonable time using exact methods [269]. In such cases, optimization algorithms are used to find optimal solutions. The assignment problem had been discussed in many books and papers due to the wide variety of problem versions got when considering different criteria in different execution environments. Deeper and detailed explanation of those versions,

criteria and systems, can be found in [270] [271], and [272].

Ant Colony Optimization (ACO) [273] was introduced as a Computational Intelligence meta-heuristic technique for optimizing a wide set of different problems, such as the assignment problem discussed here. ACO takes inspiration from the social behaviour of some ant species, who deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. In ACO, a number of artificial ants build solutions to the considered optimization problem at hand, and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants.

In this chapter, I propose an ACO algorithm in a fog-enabled BC-assisted scheduling model, namely PF-BTS. The protocol and algorithms of PF-BTS exploit BC miners for generating efficient assignment of tasks to be performed in the cloud's VRs using ACO, and award miner nodes for their contribution in generating the best schedule. PF-BTS further allows the fog to process, manage, and perform the tasks to enhance latency measurements. Meanwhile, the fog is enforced to respect the privacy of system components, and assure that data, location, identity, and usage information are not exposed. I evaluate and compare PF-BTS performance, with a recently proposed BC-based task scheduling protocol, in a simulated environment. My evaluation and experiments show reliable privacy awareness of PF-BTS, along with noticeable enhancement in execution time and network load.

I also propose a novel BC validation mechanism (PF-BVM) that maintains an equivalent consensus feature, where trusted fog nodes are able to validate TXs on behalf of BC nodes authenticated with them. Meanwhile, all nodes are deployed for the block confirmation and mining. The proposed mechanism is an approach for reducing the heavy load on the BC network, represented by extended validation time, and high energy consumption. On the other hand, the privacy-awareness property is preserved in PF-BVM by limiting the number of network nodes verifying a TX generator.

5.2 Related Works

5.2.1 Task Scheduling Optimization

Ala'a Al-Shaikh et al. [274] investigated the resource utilization problem in cloud computing, and proved that it is an NP-Complete problem. Accordingly, they implemented and analyzed a greedy algorithm to address such issue. Panda et al. [265] proposed three allocation-aware task scheduling algorithms for a multi-cloud environment. In their work, different cloud service providers were assigned to different tasks. The First-Come-First-Served (FCFS) approach was used in different algorithms, in which one was shown to be better than the others.

The origin of the ACO algorithm goes back to the Ant algorithm proposed in the early nineties by Colorni et al. [275]. Various variations have been proposed since then, such as Ant Colony System (ACS), Elitist Ant System (EAS), Max-Min Ant System (MMAS), and Rank-based Ant System (ASrank). Tuba et al. [276], provided deep analysis and comparison between different variations of the Ant algorithms. More information on different variations can be found in [277].

Wilczyński and Kołodziej [26] developed a Proof-of-Schedule (PoSch) algorithm in their proposed BC-based Scheduling, namely (BS) approach, for solving the same problem discussed in this chapter. In their proposed approach, task schedulers in the cloud layer are treated as BC miners which were categorized into four different groups, each group find its optimal solution using a different algorithm. The algorithms used in the four groups are the FCFS, Shortest-Job-First (SJF), Round-Robin (RR), and Hybrid Heuristic based on Genetic Algorithm (HSGA). The cloud then chooses the least time/energy consuming assignment. Zhou et al. [278] proposed MGAS; a modified Genetic Algorithm combined with greedy strategy, which was evaluated in terms of average response time, maximum QoS, and total execution time of VRs. MGAS was compared to the classical GA, FCFS, and Min-Min algorithms, and was shown to outperform them in most cases. Umarani Srikanth et al. [279] used ACO algorithm to achieve a feasible assignment of tasks to heterogeneous processors. In their research, it was also experimentally proven that optimizing task scheduling using ACO guarantees better task assignment than the results of FCFS approach.

Several other works have proposed the utilization of ACO or new methods to improve its results. Examples include Merkle and Middendorf [280] who analyzed a case where an ant forgets the best-found solution and Mirtaheri and Shirzad [281] who used ACO to solve the Traveling Salesman Problem. For improvements, Li and Wu [282] proposed a modified variation of the ACO algorithm that performs even better than original, while Hussein and Mousa [283] proposed two improved variants of ACO and Particle Swarm Optimization for task scheduling.

5.2.2 Validation in Blockchain-based solutions

TX validation is the process of checking whether the generator of the TX (i.e. sender) has sufficient amount of money to spend, or determining whether the new TX conforms to the network or Consensus Algorithm (CA) rules. This is usually performed at the node level by checking the companion signatures of a TX; if the signature is valid, the TX is accepted. The checking process includes comparing the amount of money the sender is willing to spend, to the amount of money registered in the sender's wallet in the BC, which is held locally within the nodes' memory [238]. All recipient nodes then *validate all* TXs within the new block again [284], and check if the solution of the puzzle was correct. If the block is valid in terms of TXs and puzzle

solution, it is *confirmed* and added as the head of the locally saved BC.

The term '*Verification*' indicates the recognition of a TX generator by other network entities through the linkage with the generator's public/private keys.

Despite the fact that most previous works considered miners to be computationally-strong devices relatively to other nodes, some researchers proposed ideas where miners can be moderately strong mobile devices too [285].

Axon [286][287] discussed the public key infrastructure (PKI) concepts, and proposed a privacy-aware BC-based PKI architecture. The trust in BC is typically gained by the majority consensus on the validity of a piece of information, and a user verifyability [288]. However, the proposed architecture in [286] limits the necessity of verification by all nodes of the network. That is, public keys and private keys -online and offline versions- are only registered and verified by few previously-verified and trusted neighbors. Those keys are timestamped and have an expiration so that they should be regularly replaced. Also, a user-controlled identity disclosure mechanism is built, where each user chooses whether and when to disclose their identities or past public keys.

Li et al. [6] proposed a privacy-preserving carpooling system that uses a private BC in a vehicular Fog Computing context. Silva et al. [289] proposed an approach for using BC to ensure the privacy of patient medical data in Fog environment by limiting authorized users accessing the patient's medical information .

Debe et al. [9] proposed a reputation system for fog nodes that are delivering services to IoT devices, using BC Ethereum Smart Contractss (SCs). Here, IoT devices rate fog nodes according to specific modifiable criteria. IoT devices' credibility is also computed, according to specific contributions.

5.3 Enhancing Fog efficiency using Blockchain

Here, I propose exploiting BC miners (e.g. peers of Ethereum BC), who usually perform computational puzzle solving in order to receive digital coins from the BC network. By generating a SC to the platform, I aim to exploit BC nodes to assign end users' task into the cloud VRs. I deploy the fog nodes to control the communication among end users, BC network, and the cloud. The fog node in my proposed system is responsible for choosing the best schedule according to specific criteria, which reduces the latency compared with the case of BC network generating the best schedule by consensus. That is, a BC node picks the SC generated by the fog node, and sends its proposed schedule back to the fog node. The fog node then chooses the best schedule among the received ones and sends it to the cloud. The cloud accordingly performs the tasks using the assigned VRs and returns the results to the fog node, which forwards them to end users. The direct incentive for BC miners who run the SCs is provided in the form of GAS, while incentives for generating new blocks

are given by the BC network itself. Notably, the results are saved on the chain by BC peers consensus (e.g. PoW in the case of Ethereum) for future reference and analysis. However, the time needed for reaching a consensus in order to save the data on the chain is not included in the time required to generate the optimal schedule. Thus, the type of CA deployed in the BC network does not affect the latency. If some related information are needed, by the system administrators, the cloud, or the fog, it can be easily retrieved as the result of each SC is immutably saved on the chain.

To evaluate my proposed system, I adopt two different approaches of parallel computing [290]. The first is the Single Instruction Multiple Data (SIMD), used to enforce BC miners who pick the generated-by-fog SCs to run the required code and computations. The second is the Multiple Instruction Multiple Data (MIMD), which I use to simulate the BC Scheduling (BS) system proposed in [26]. The latter is used for validating my experimental results by comparison. I performed the experiments on eight VRs, five of them ran the ACO assignment optimization presented in the following subsection, while the other three performed SJF, FCFS, and RR assignment. I excluded the fourth group studied in [26] since it presented the worst performance in most cases that were originally evaluated in the BS proposal.

5.3.1 Preliminaries and problem statement

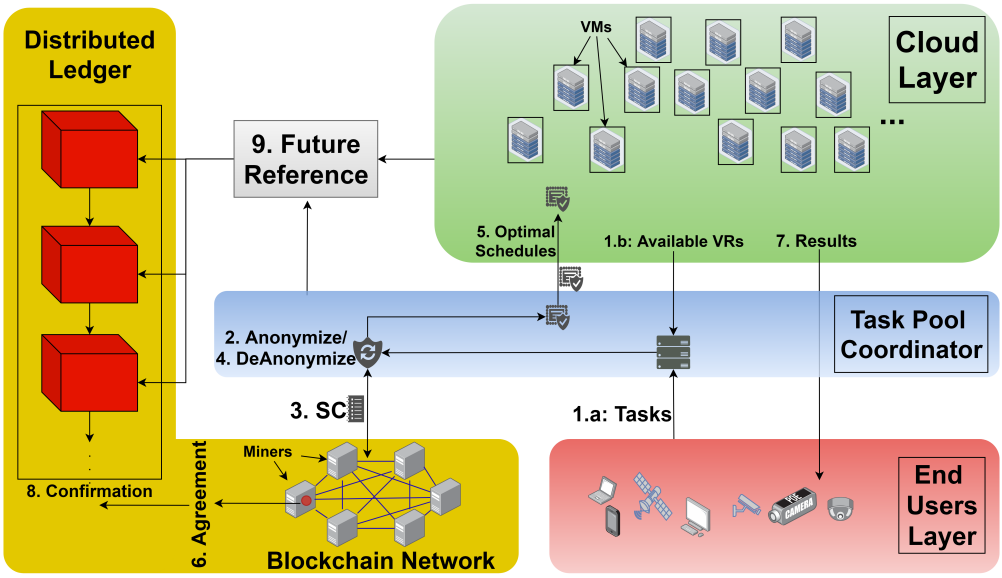


Figure 5.1: *The Fog-Cloud architecture in which the fog and BC are deployed to perform the proposed PF-BTS protocol*

In the ACO approach, ants of the colony communicate by depositing chemical pheromones along the paths they take from home colony to food sources. Such

behaviour provokes other ants of the colony to follow the paths that are most recommended by the majority of ants, indicated by high levels of pheromones deposited. Implementing this nature-inspired methodology was shown to provide high ability of solving different versions of the assignment problem [291, 292]. There are several proposed versions of ACO depending on the problem to be solved. Generally, ACO works over the following phases:

1. Construct Ant Solutions: each ant stochastically constructs its solution from a finite set of available solutions.

2. Update Pheromone: In this phase the pheromone values are modified to indicate promising/bad solutions. Moreover, all pheromone values are equally decreased every predefined number of iterations due to the pheromone evaporation concept.

3. The final solution is biased by the pheromone values deposited by all ants.

Here, I adopt a Multi-Objective ACO with Global Pheromone Evaluation, improved by a greedy optimal assignment approach to get even better solutions. Detailed technical details of the ACO version I adopted, its equations, the proposed improvement, and the final solution algorithm were not presented in this thesis due to the limitation regarding pages number. However, such details, among others, can be found in [48].

The research problem here is the assignment of n tasks, denoted by T_i where $i = 1...n$, to m machines denoted by M_j where $j = 1..m$, such that each task is assigned to one machine. The objective function is to find a schedule where the time consumption, required by the cloud VRs to perform the computational tasks, is minimal. The utilization of the machines, denoted by u_{ij}, relates to two parameters. First, the computing power of M_j, measured by MFLOPS (Million FLoating point Operations Performed in one Second). Second, the length of T_i, which is the number of operations or instructions needed to execute the task T_i expressed in MFLO (milion floating point operations).

A schedule can be presented using a $n \times m$ binary matrix called schedule matrix S whose entries are denoted by s_{ij}. That is, $s_{ij} = 1$ if task i is assigned to machine j, and equals zero otherwise. The proposed algorithm makes sure that no more than one element is in the same row to make sure that each task is assigned for exactly one machine. However, there can be several tasks performed by one machine, depending on the best assignment criteria. The state space under such condition is decreased from $n!$ to m^n, hence the optimum assignment is more likely to appear during the search.

5.3.2 System Characteristics and Protocol

The ACO algorithm, implemented in a form of a SC, optimizes the assignment of tasks to the available cloud VRs, while fog nodes control the process and guarantees the anonymity of end-users and tasks' information. Figure 5.1 clarifies the framework of my proposed protocol in a FC architecture. The key steps of the proposed protocol are as follows:

1. End-users send their computation tasks to the fog node they are authenticated with.

2. Fog nodes, termed as Task Pool Coordinator (TPC) component, receive the tasks from end-users, and receive information about the capacities of available VRs from the cloud.

3. Unless the TPC is capable of performing the computational tasks, it randomly assigns each task, or group of tasks, a unique ID (anonymization). The abilities of the TPC to perform the tasks is determined by different criteria, such as the CPU computational power, the length of each task, the network load balancing, etc.

4. After that, the TPC generates a number of SCs into the BC network which include: the optimization algorithm (i.e. ACO), a list of tasks IDs each coupled with its task length, a list of VRs capacities, the public address of the fog node, and some appropriate amount of GAS.

5. As the SCs are successfully generated, each SC is picked and run by a BC miner. According to a defined criteria (e.g. the least time consumption/ the least cost) each miner proposes a different optimal schedule that is immediately sent back to the TPC. All miners are expected to finish their iterations and send the results at approximately the same time because the SIMD principle applies.

6. Each time the TPC receives a better schedule than a previously proposed one, the best assignment is updated.

7. TPC waits for a pre-determined time to check if it receives a better schedule. If the time passes and no other better schedule appears, the tasks are deanonymized and sent to the cloud. Otherwise, TPC replaces the schedule with the better one and waits again. However, if the number of received schedules equals the number of generated SCs, the best is chosen and sent to the cloud regardless of the waiting time.

8. After the cloud receives the optimal schedule from TPC, the cloud performs the tasks and sends the results back to the TPC, which forwards the results back to end-users.

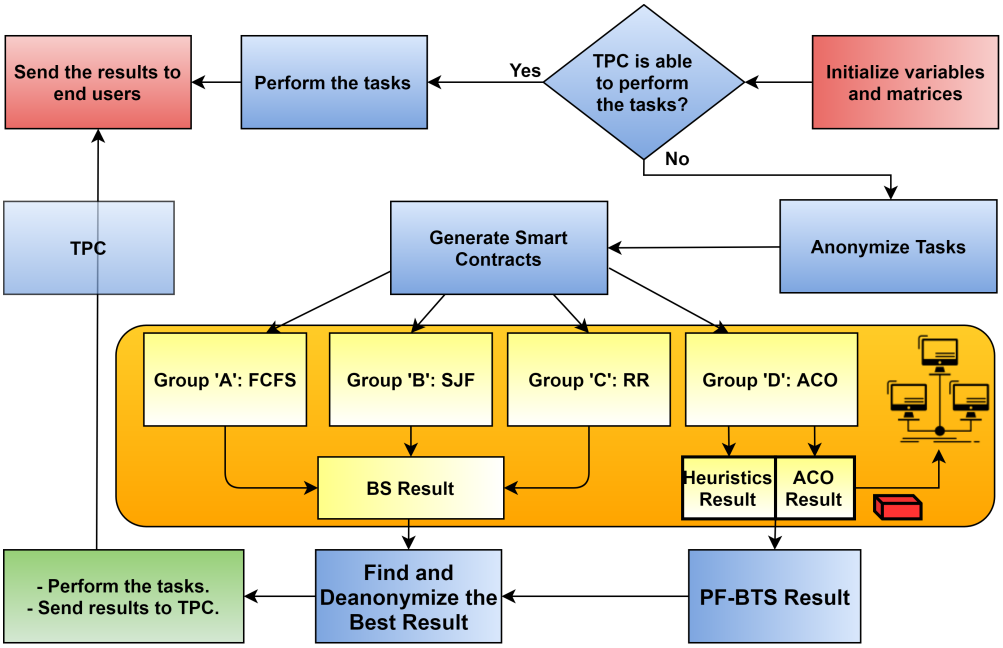


Figure 5.2: *The simulation workflow of PF-BTM experimentation*

9. Later, the results of the SCs are saved On-Chain using any CA (e.g. PoW in the case of Ethereum) for future reference. Those saved results are immutable and private. That is, all information are anonymized and can not be used/analyzed from any party other than the trusted ones (i.e. the cloud and TPC).

5.3.3 Evaluation

Here, I will present the major experiment setup and results. For full evaluation and discussion of the proposed protocol in terms of Privacy, Total Execution Time, and Network Load, the reader may refer to [48].

It is proven that the BS scheduling technique, proposed in [26], is able to provide more optimal solutions than FCFS, SJF, and RR approaches. Consequently, for 't' tasks to be performed on 'v' resources, BC miners are expected to provide better assignment schedules [293, 294] than FCFS, SJF, RR, or a random assignment on a single computer as shown in [295, 296]. However, the results presented in [297] and [298] indicated an equivalency of performance of ACO compared to SJF in most cases. Depending on these results, along with the transitivity relation of the two homogeneous problems, I could draw a conclusion of an expected superiority of my proposed PF-BTS protocol over BS. This is because BS takes the best proposed assignment output from different miner nodes that perform FCFS, SJF, HSGA, and RR. The validity I am seeking later is that PF-BTS outperforms BS in providing a more

optimal schedule with respect to the computation power provided by the same tested tasks and VRs, or in providing equivalently optimal schedule using less resources. This argument can be formalized as follows:

Definition:

An Optimal Schedule ($OS_{t,v}$) is the best assignment of requested computational tasks 't' to be performed on a set of available VRs 'v', in terms of minimal execution time of 'v' to perform 't'.

Hypothesis:

$OS_{PF-BTS(t,v)} < OS_{BS(t,v)}$

proof:

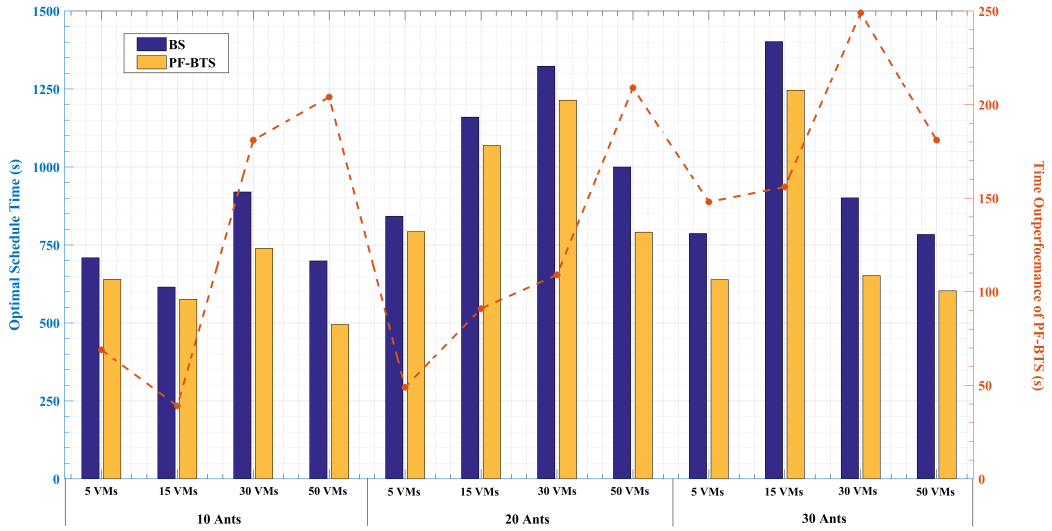
$OS_{ACO(t,v)} < OS_{Random(t,v)}$
$OS_{ACO(t,v)} <= Min(OS_{FCFS,SJF,RR(t,v)})$
$OS_{BS(t,v)} = Min(OS_{FCFS,SJF,RR(t,v)})$
$OS_{PF-BTS(t,v)} < OS_{ACO(t,v)}$
$\therefore OS_{PF-BTS(t,v)} < OS_{BS(t,v)}$ $\square$

To experimentally prove these expectations, I implemented a simulation environment using Python 3.8 [1]. I ran the code on Google Cloud Platform (GCP) using a C2-standard-8 (8 vCPUs, 3.8 GHz, 32 GB memory). Here, I dedicated the first core for the FCFS computations, the second for the SJF computations, the third for the RR computations, and the remaining five cores were dedicated to the ACO computations.

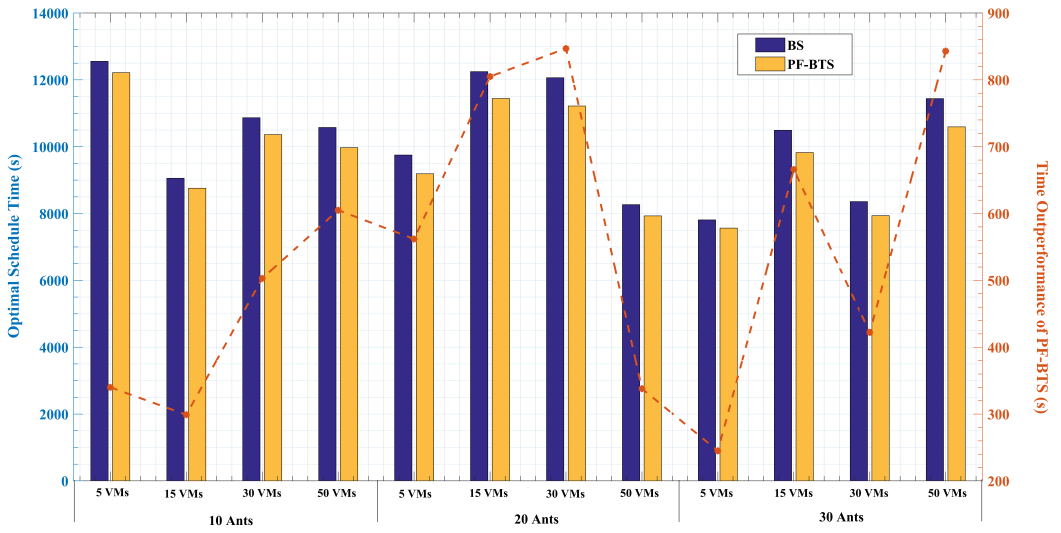
Note that there was no need to recruit more than one miner handling each of the scheduling algorithms of BS (i.e. FCFS, SJF, and RR) since all cores would produce the same result. Only because the whole simulation code was implemented using python, the SCs in my simulation used the same language so that unnecessary higher complexity is avoided. Figure 5.2 depicts the simulation steps within a network consisting of 8 miners, and grouped into four different groups as described above. The best assignment of received results from Group D is considered the PF-BTS result. The TPC compares BS result with PF-BTS result, re-acknowledges the tasks and sends the best assignment to the cloud and then the protocol continues as described earlier. The experimental results are presented in Figure 5.3.

Specifically, I conducted 33 simulation runs in groups of 12, 12, and 9 runs, for simulating 30, 300, and 3000 tasks, respectively, that need to be computed by 5, 15,

[1] https://github.com/HamzaBaniata/PF-BTS



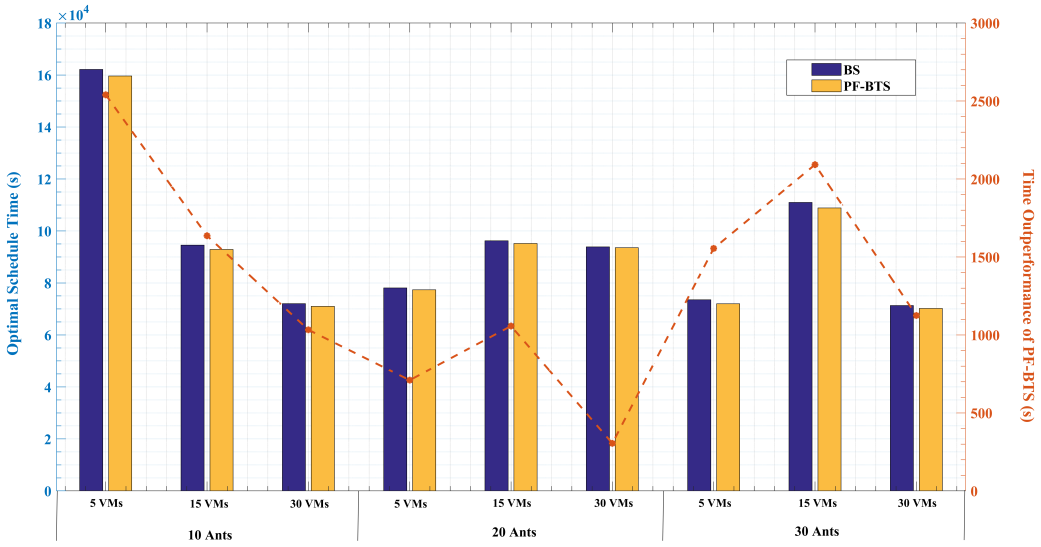
(a) *30 tasks*



(b) *300 tasks*

Figure 5.3: *Time consumption of BS and PF-BTS (blue and yellow bars, respectively) is correlated with the primary y-axis on the left. Out-performance of PF-BTS over BS (dotted red line) is correlated with the secondary y-axis on the right*

30, and 50 VRs. In each simulation run, same tasks' length and VRs' computing capacities were input into all generated SCs. Each VR is assigned a random computing capacity that ranges from 4 to 48 MFLOPS. Each task is assigned a random length that ranges from 100 to 1000 MFLO. The TPC was assigned a computing capacity



(c) *3000 tasks*

Figure 5.3: ... *Continue*

of 20 MFLOPS. Accordingly, a maximum total execution time of generated tasks in TPC was configured to be 80 seconds. If this condition was met, the tasks were performed in the fog layer (i.e. the BC miners, the SCs, and the Cloud are not exploited). Evaporation factor is sit to 0.3. One minute cost of VR is sit to 0.1 cent/minute.

To further highlight the time, in a PF-BTS system, needed from sending the tasks and VRs capacities to the BC network until getting the schedules back from them, I conducted a simulation run where eight tasks need to be assigned into four VRs, the assignment job was distributed to 16 BC miner nodes (16 SCs were generated) with each running 10 ants. The 16 miners sent their assignment suggestions to the TPC within 0.03 second. Obviously, this is a better result than the one gained by BS (0.07 second), despite that there were 16 miner nodes involved instead of 4, and the schedules provided are optimal rather than random.

5.4 Enhancing Blockchain efficiency using Fog Computing

By understanding the several steps and procedures performed by a BC miner once a new TX or block received, we can trivially state that the less validation time spent, the more time dedicated for processing new TXs [299]. This absolutely means higher overall operational efficiency of the system in terms of throughput. In Chapter 3, I

showed how deploying a BC in the fog layer outperforms its performance against deploying it in the end-user layer, in terms of block finality and storage cost. I also showed in Chapter 4 how to independently improve a BC-based system efficiency, in terms of block finality and network overhead, by optimizing the utilized Neighbor Selection mechanism. In this section, I attempt to show how to enhance a BC efficiency, in terms of energy consumption and storage cost, by allowing the BC to depend on trusted fog nodes to perform the validation step. Thus, providing more time for BC miners to carry on with other tasks.

To clarify the motivations behind this proposal, I used the FoBSim tool to experimentally measure the time consumed by BC miners for, specifically, validating newly received blocks. Simulators like BlockSim [159], iFogsim [149], and PeerSim [136] simulate the validation time with a delay without actually performing the validation as it is in reality, hence all TXs in these simulators are considered valid. On the other hand, TXs are actually checked and validated similarly to Bitcoin and Ethereum in FoBSim. Once the number of TX/B –or gas limit– is reached the block is mined.

I performed this experiment using an Intel i5-8265U CPU, backed up by 12 GB of DDR4 SDRAM and 45 GB vRAM. Figure 5.4 shows the results of this experiment in which I tested four TX/B scenarios (i.e. 100, 1000, 10k and 100k). I performed the four scenarios on eight groups of randomly generated TXs; 10–3000 blocks. Finally, I computed the average block validation time for each group by dividing the summation of validation time values by the number of processed blocks.

It can be seen in the figure that the average validation time consumed for blocks holding 1000 TXs or less evaluates to almost zero. However, the validation time is proportional to the increment of TX/B ratio.

Accordingly, proposing a system where TXs and blocks be validated outside BC nodes, would save much processing time for them. Consequently, I analyze, as suggested in [300], the average validation time consumed by two scenarios similar to those shown in Figure 5.5.

Studying a case where three rich fog nodes [301] are connected to each other, let each fog node be connected to 10 BC nodes (both miners and non-miners), with total of 30 BC nodes. Once a TX is generated by a node, it is broadcast throughout the network. Then, all nodes locally perform the validation. If the time needed to validate this TX equals x, then the total processing time consumed to validate this TX equals 30x. Equation 5.1 generalizes such calculations, where $Time_c$ is the total processing time needed to validate a TX, n is the total number of network nodes, x is time needed to locally validate a TX on one node. The second scenario will be discussed shortly.

$$Time_c = nx \tag{5.1}$$

Note that the maximum time consumption in this experiment is 0.6 sec. This

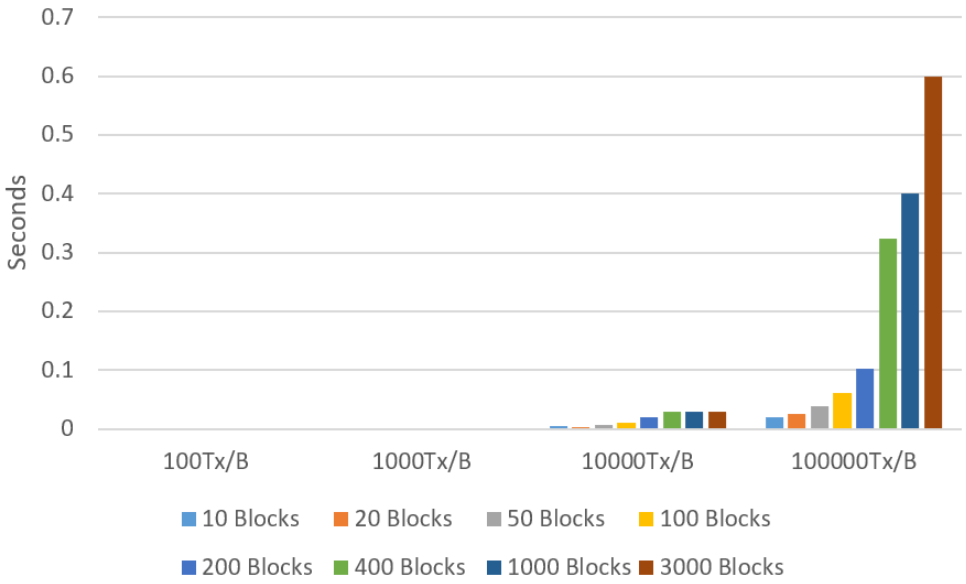


Figure 5.4: *Average time consumption for block validation in a BC node*

contributes to the usage of the main memory for data storage, which is closer to the CPU. To check if the pattern of time consumption remains, I have implemented my code using Apache HTTP Server 2.4.41, with the valid TXs being saved in a MySQL database on the secondary memory[2]. I obtained similar results of exponential increasing in time consumption with higher ranges. For example, using SSD disk, the scenarios of 100 TX/B, 1000 TX/B, and 10k TX/B applied on 10 Blocks resulted an average block validation time of 0.112, 4.9, and 419.9 seconds respectively.

Maintaining security, reliability, and trust in Proof-based BCs generally depends on very high total power consumption rates [302]. A single bitcoin TX confirmation, for instance, consumes more power than an average U.S. household in 21 days [303]. I was motivated by these facts, and by the primary results presented thus far, to propose a Privacy-aware Fog-enhanced Blockchain Validation Mechanism (PF-BVM). This method shall maintain the decentralization and reliability of a proof-based BC, yet is more energy and storage efficient.

5.4.1 Framework and principles of the proposed PF-BVM

PF-BVM deploys some of the trust management concepts proposed in [9]. In addition, it aims to provide privacy awareness, and enhanced BC validation using fog nodes. In order to achieve this, a trust management scheme is needed. PF-BVM is a combination of different services provided by the fog, which is enforced to prove

[2]https://github.com/HamzaBaniata/BlockChainValidation/blob/master/Second%20Code

trustfulness in order to be allowed to validate blocks instead of network nodes. The following principles present my proposed PF-BVM:

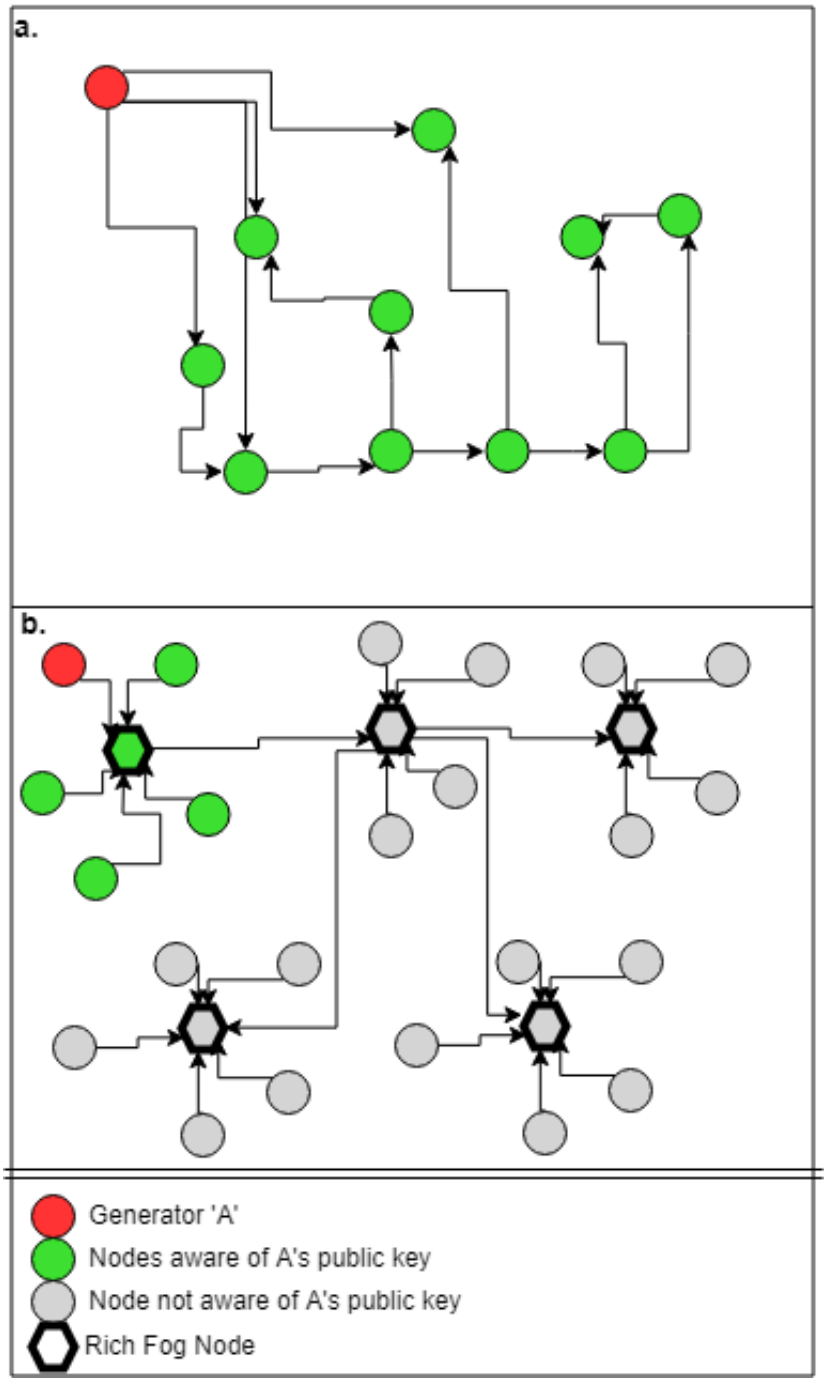


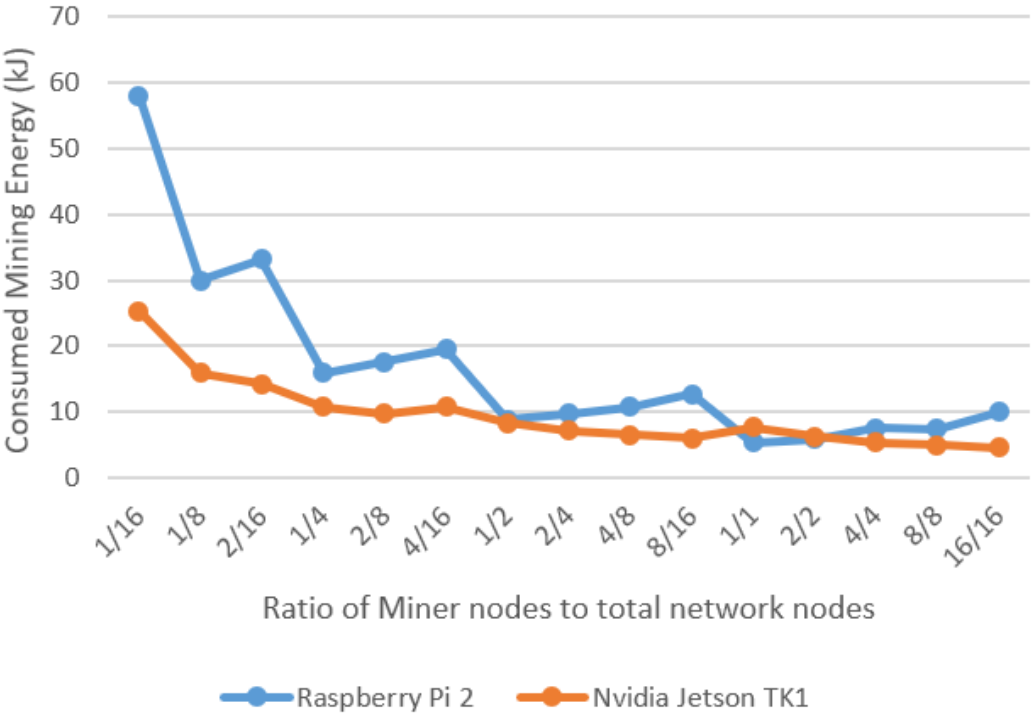
Figure 5.5: *a PF-BVM conceptual network compared to the currently used network. a. all BC nodes are connected and responsible for a TX validation. b. all BC nodes are connected yet few are responsible for a transaction validation*

- *Trusted* fog nodes are authorized to validate new TXs on behalf of the nodes authenticated with it. A fog node is considered *trusted* as long as its acceptance/rejection decisions regarding new TXs match the decisions made by nodes authenticated with it in a percentage of 100% .
- A default status of a fog node is *not Trusted*. The level of trust increases through time by comparing decisions made by the fog node to decisions made by the BC nodes regarding new TXs. If the match percentage stays 100% for named number of TXs, the fog node status is switched to *Trusted*.
- A *Trusted* fog node is randomly, yet regularly, tested by nodes authenticated with it. In contrast, a newly trusted fog node should be tested more frequently than an older trusted ones.
- The longer the fog node is *Trusted*, the fewer times it is tested for maintaining the 100% match. Yet the frequency of testing should never reach ZERO times per named number of TXs.
- The privacy awareness of the system is preserved by applying the PB-PKI architecture proposed in [287]. Using this architecture, real identities of nodes shall be only known to the least number of users in the network. That is, a node reveals its user's identity only to neighbours that share the same fog-node domain, while the public keys are kept in the fog node's local memory. When a new TX is generated, the node's public key is replaced with the fog-node's public key. Hence, the TX can only be related to all nodes authenticated with that fog node. Consequently, the probability of disclosing the generator's real identity shall be totally minimized. This idea is demonstrated in Figure 5.5. In scenario 'a', all nodes are connected and all nodes should *validate* a TX, hence all network nodes shall *verify* the generator. In scenario 'b', only the fog node and the BC nodes connected to it are able to validate the TX, yet the TX is generated to the network as valid and referred to by the public key of the fog node domain.

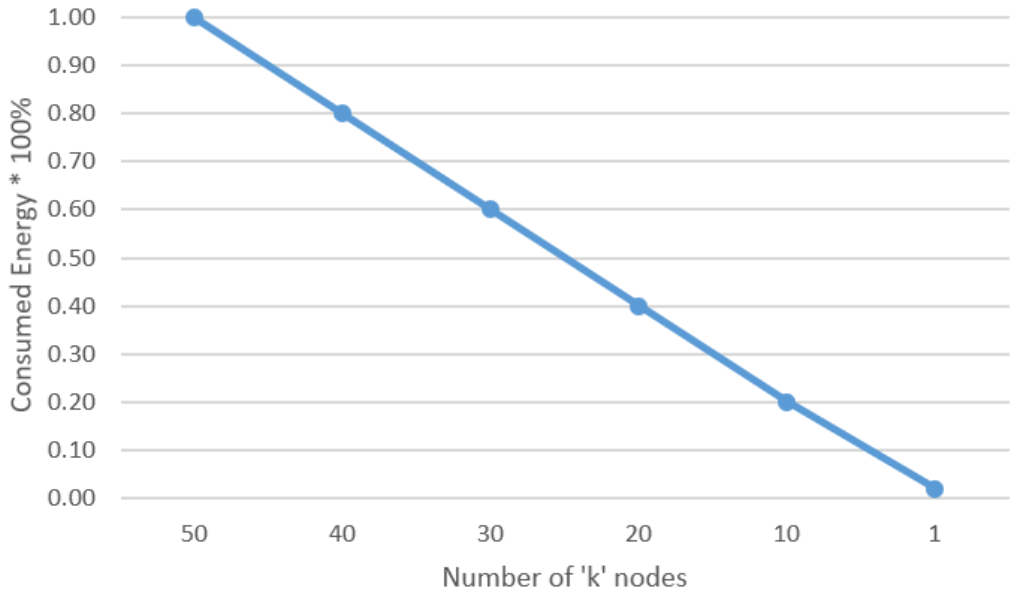
5.4.2 Evaluation

Time Consumption

Carrying on with the analysis of time consumption scenarios, the total processing time consumed to validate a single TX in PF-BVM equals 3x instead of 30x. That is, only the three fog nodes would spend time on validation, while the rest of network nodes are free to perform other BC related tasks. Equation 5.2 generalizes these computations, where $Time_{PF-BVM}$ is the total processing time needed to validate a TX when using PF-BVM, and k is the number of authorized nodes to validate TXs.



(a)



(b)

Figure 5.6: *Energy consumption: a) when mining 400 blocks by two different BC execution platforms and b) when validating a block using PF-BVM*

$$Time_{PF-BVM} = kx \tag{5.2}$$

Energy Consumption

When there are too many nodes, the communication performed to exchange agreements between them would be very complicated [20] and energy consuming [140]. Kreku et al. [140] demonstrated the energy consumption for mining 400 blocks by two BC execution platforms: 'Raspberry Pi 2' and 'Nvidia Jetson TK1' (Figure 5.6a). These results suggested that the more the miner nodes relative to the total number of nodes in the network, the less energy consuming the block confirmation is. Further, least energy consumption was gained when using only one miner node in a network that contains only one node in total.

Following these insights, the wasted amount of energy consumed for a TX or block validation shall decrease when decreasing the number of validators. Nearly all BC platforms do not differentiate between a validation role and a mining role during energy consumption evaluation.

That is, although the total amount of energy and storage consumed for confirmation may not change, the lower the number of nodes validating a TX, the lower the validation energy consumed as a whole in the system. To clarify, lets consider five nodes of have received a block, each will consume equal amount of energy 'x' to validate this block, depending on the TX/B ratio and the number of blocks in the locally saved chain [285]. The total amount of energy consumed to validate this block equals to $5x$. If a validation protocol -such as the proposed one- in which only one of the five nodes is trusted and authorized to validate the block on behalf of the other four nodes, then the total amount of consumed energy evaluates to $1/5$ relative to the energy consumed by the first system. Equation 5.3 generalizes the computation method of the consumed energy in PF-BVM, where E is the consumed energy percentage by PF-BVM compared to current protocol.

$$E = \frac{k}{n}\% \tag{5.3}$$

To present the effect of the proposed PF-BVM, I simulated the energy consumption, using equation 5.3, in my validation simulator (see Figure 5.6). In the simulation experiment, the value of k changes, while the value of n equals 50 nodes.

As it can be seen in Figure 5.6, the less the number of nodes validating a TX, the less the total energy consumed to validate a TX, hence, the more efficient the system is. It is also worth noting that the same approach of calculations can be applied for evaluating the storage efficiency proposed by PF-BVM. For example, $n*$ 150 and 46 GB storage, used to locally save Bitcoin and Ethereum chains respectively [74], can be reduced to k/n % needed storage capacity for the whole system.

5.5 Conclusions

In this chapter, proposed solutions for enhancing BC efficiency using FC, and vice versa. First, I proposed a Privacy-aware Fog-enhanced BC-assisted Task Scheduling protocol, called PF-BTS. The proposed protocol deploys the ACO algorithm to find the best assignment schedule of tasks into the cloud's VRs in a secure manner. The cloud utilizing PF-BTS receives the best assignment of tasks into its VRs, in terms of total execution time of the VRs. I have shown how the proposed protocol outperforms several previously proposed approaches for solving similar problems, in terms of privacy, efficient scheduling of the VRs, and minimal exchanged messages. Moreover, PF-BTS allows the fog node, as an extension of the cloud, to perform the computational tasks at the edge of the network.

Second, I proposed a Privacy-aware Fog-enhanced BC Validation Mechanism (PF-BVM), which contributes to the integration of Fog Computing, the internet of things and BC techniques. As a conceptual criterion, PF-BVM allows trusted rich fog nodes to perform TX validation on behalf of BC miners, who gains trust by randomly running matching tests. To evaluate the proposed mechanism, I simulated it using FoB-Sim and the experimental results showed that PF-BVM can significantly enhance a BC system validation in terms of time consumption, energy efficiency, and storage capacity.

Future work will be directed towards developing PF-BVM in order to measure the dynamics of trust evolution over time for BC nodes. Additionally, I plan to experiment other optimization approaches that are competitive with ACO, such as the ACTS-LB [282], MO-ACO [304], or the Black Hole [305] algorithms. I will also investigate the output of BC nodes in terms of assignment solutions quality and time consumption, by enforcing multiprocessing and multi-threading techniques in each miner node.

I am responsible for the following contributions presented in this chapter:

III/1. I designed and developed a Privacy-aware Fog-enhanced Blockchain-assisted Task Scheduling protocol (PF-BTS) which allows the fog layer to exploit BC in a privacy-aware manner to provide optimal task schedules for cloud infrastructures.

III/2. I designed and developed a Privacy-aware Fog-enabled Blockchain Validation Mechanism (PF-BVM) which allows BC networks to exploit fog nodes for faster privacy-aware block validation.

Chapter 6

Integrated Fog-Blockchain applications

The unprecedented pace of technological development in smart systems, incorporating sensing, actuation, and control functions, have the following properties and needs: (i) they are interconnected and need scalable, virtualized resources to run, store and process data, (ii) they are mobile and can potentially access and build on user data made available by smartphones and tablets, and (iii) they are getting smarter, so they may get access to user data provided by connected smart devices. As the number of smart devices in smart systems grows, the vast amount of data they produce requires high-performance computational and storage services for processing and analysis, among other novel techniques and methods that enhance these services and their management. BC applications have been proposed in a wide variety of environments such as distributed voting, eHealth, Mobile Computing, Internet of Vehicles, Self-Sovereign-Identity, etc. Integrating BC technology with such smart applications for managing data of mobile devices can further enhance the privacy and security requirements of current complex systems.

Specifically, trusted online credential management solutions are needed for instant and practical verification. Most of the available frameworks targeting this field violate the privacy of end-users or lack sufficient solutions in terms of security and QoS. To this end, I propose a Privacy-aware FC-enhanced BC-based online credential solution, namely PriFoB. The proposed solution adopts a public permissioned BC model with different reliable encryption schemes, standardized Zero-Knowledge-Proofs (ZKPs) and Digital Signatures (DSs) within a FC-BC integrated framework, which is also GDPR compliant. I deploy both the PoA and the Signatures of Work (SoW) Consensus Algorithms (CAs) for efficient and secure handling of Verifiable Credentials (VCs) and global accreditation of VC issuers, respectively. Furthermore, I propose a novel Three-Dimensional DAG-based model of the Distributed Ledger (3DDL) and provide ready-to-deploy PriFoB implementation. I discuss insights re-

garding the utilization and the potential of PriFoB, and evaluate it in terms of security, privacy, latency, throughput and power utilization. I analyze its performance in different layers of FC-enabled cloud architecture with simulation and emulation, and I show that PriFoB outperforms several BC-based solutions utilizing Ethereum, Hyperledger Fabric, Hyperledger Besu and Hyperledger Indy platforms.

I further discuss BC-integration possibilities for smart systems to support efficient, secure, and privacy-aware execution of smart applications. For that, I propose a design space where issues need to be addressed at different layers of such integrated systems.

6.1 Introduction

A Smart Systems (SS) incorporates the functions of sensing, actuation, and control in order to describe and analyze a situation and make decisions based on the available data in a predictive or adaptive manner, thereby performing smart actions [306]. A Smart Device is a fundamental component of a SS generally connected to other devices or networks via different wireless protocols (such as Bluetooth, Zigbee, NFC, Wi-Fi, LiFi, 5G, etc.). They can operate to some extent interactively and autonomously [307]. SSs address environmental, societal, and economic challenges like limited resources, climate change, population aging, and globalization. They are for this reason increasingly used in a large number of sectors, such as education, transportation, healthcare, energy, safety and security, logistics, ICT, and manufacturing. One can also categorize SSs via regions by referring to smart homes and smart cities. The management of smart devices and their data in SSs require smart applications, raising many requirements and open issues.

Credential recognition is the process where a (inter)national body, called Verifier, validates the legitimacy of a document that was issued by another body, also called as Issuer. A credential is issued upon an event occurrence to certify that this event has indeed happened, such as educational credentials, vaccination certificates, governmental passports/IDs, etc. Within a country, area, or continent, one may find agreed-on regulations to recognize a named type of credentials for purposes like governmental treatment, hiring, travelling, etc. However, once a person/entity, who has been issued a legitimate credential, needs to approve it abroad, a painfully lengthy and costly process needs to be carried out. This is because credential documents generally include different types of stamps, proofs, identification numbers and other data that have to be verified individually and carefully for each credential referring to distinct, centralized, locally maintained databases. That is, no global credentialing standard is used by issuers and no constant way to prove different credentials is guaranteed. Generally, the more sensitive data in a credential, the more complicated and costly it is to validate.

Several previous studies looked for solutions to overcome the above mentioned drawbacks. The approach usually taken is to propose a central TTP that carries out international issuer recognition process [308], making it a viable reference for credential validation. The most recent example of such a solution is the EU Digital COVID Certificate [1], where authorized governmental bodies within Europe update a central database that includes vaccination personal data. Using this service, vaccinated individuals, or their agents, can prove that they have received a vaccine within a European country, allowing them to travel abroad without, e.g. quarantine, restrictions. However, private data of those agents in such a scheme is not only exposed to national, but also to international governmental personnel/systems. Additionally, locally vaccinated people need to individually register to several different platforms, before they can obtain an EU accredited vaccination certificate. Although such data management schemes are not compliant with the General Data Protection Regulations (GDPR) [309], it apparently was the only available approach to relax the pandemic's restrictions as soon as possible. In other, more sensitive cases, such as foreign educational diploma recognition or voting systems, privacy awareness is a critical factor that needs to be adopted by design in any proposed solution.

In this chapter, I utilize a public-permissioned BC and a FC layer to propose an efficient system for global institution accreditation and credential verification, namely PriFoB. The BC in PriFoB acts as a Distributed Trusted Third Party (DTTP), in which miners are national accreditation bodies (e.g. national ministry of higher education, ministry of foreign affairs or ministry of health affairs, etc.). On the other hand, fog nodes are realized by credential issuer bodies (e.g. universities, hospitals, vaccination centers, etc.). I designed PriFoB to guarantee privacy and security of system entities, by deploying the robust PoA [139] CA, a realized application of the secure SoW [310] CA, RSA-based encryption with DSs, and ZKP mechanisms. The SoW deployment, specifically, is an additional and optional consensus sub-layer for practical realization of PriFoB as a global accreditation solution. I also propose a relaxed and efficient multi-dimensional ledger model, where blocks are partially (not fully) immutable. Each dimension holds a different type of TXs, which enhances the overall efficiency of the validation process. Furthermore, I use an improved Directed Acyclic Graph (DAG) based block relations within each dimension, which outperforms the classical linear model in terms of total throughput and response latency [311].

6.2 Related Works

Several previous works approached BC-based solutions for realizing digital credential verification. Fauteux et al. [312] stated how BC deployment can solve the long

[1] https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate

list of challenges when an academic credential needs to be validated. The BC solution was discussed along its pros and challenges, and some technical details were further presented. Only in the years of 2019 and 2020, some few testable implementations were proposed to address credential verification problem using BC technology. Recently launched BC-based credential verification projects, namely BlockCerts [313], OpenCerts [314], and trustED [315] were surveyed in [316], where the most mature and suitable consensus methods, BC architectures, and BC platforms were presented and discussed. Consequently, the authors proposed AcaChain, which is a private, permissioned BC system that allows issuers to track the achievements of their agents, and then issue the graduation proof once all conditions of the credential are fulfilled. As the three BC-based credential systems, namely AcaChain, BlockCerts and OpenCerts, save all credentials information, along with relevant personal identifiers of students on the immutable chain, they are non GDPR-compliant. Furthermore, saving all data on the chain is considered an inefficient approach of using a BC system, as this can rapidly drain storage and computation resources. Thus, these projects may not be considered practical in industrial deployment. Similar approach was utilized by other solutions as well, e.g. [317].

Anant et al. [318] proposed deploying BC for SRM Institute of Science and Technology digital credential validation, using Ethereum SCs. This approach also saves all students credentials and data on the public chain. Similarly, Ahammad MS Tomal MH [319] proposed a BC-based system that saves only the signed hash of each certificate instead of the certificate itself, while deployed the inefficient PoW CA to maintain the consistency of the DL. In [320], a BC-based privacy preserving protocol is proposed so that users can access on-chain services. The proposed protocol suggests that a verifier is needed to personally verify the correctness of a claimed credential. Furthermore, in the generality of this protocol, the assumption of the user having to be a member of the BC raises some questions regarding deployment feasibility.

Hyperledger Indy [321] is an open source project, administered by the Linux foundation, which aims at providing a BC-based VC system. The project implementation provides a platform for Decentralized Identifier (DID) rooted on BCs or other DLs so that they are inter-operable across administrative domains, applications, etc. The project deploys privacy preserving mechanisms such as ZKPs, and takes good care on what data is saved on-chain. Indy uses the Plenum CA which is a special purpose RBFT CA [322]. The Sovrin BC [323] is built on top of Indy project for providing a general purpose, global DIDs. After years of development, this BC was officially launched in September 2019. Similarly, The GraphChain [324] was proposed for exploiting the advantages of Indy and facilitate the issuance of Legal Entity Identifiers.

Tariq et al. [325] proposed Cerberus, where Ethereum-based, private-permissioned BC architecture was utilized. Here, VCs and (if applicable) revoke TXs are saved

on-chain. A QR code-scanning approach is deployed then to verify a VC is indeed on-chain and that it is not revoked. The network in Cerberus is monitored and maintained by a single accreditation body that refers to its local DB for a full list of issuers. Thus, an issuer is automatically accredited by the system only if it was accredited through a legacy accreditation channel. Although the architecture seemed very promising, the authors have only discussed their proposal theoretically while no implementation or experimental comparisons were provided.

In [326], a strategy for modelling, designing and developing BC-based healthcare accreditation and verification solutions was discussed. Although the paper aimed at e-Health applications, it provided rich insights on design principles, trade-offs, and critical terminology definitions. Similarly to most proposals for BC-based accreditation and verification solutions, this proposal was discussed only in theory.

In November 2019, the World Wide Web Consortium[2] (W3C) published a standard recommendation for solutions targeting VC solutions, including data models, system concepts, approved methods, privacy and security challenges with proposed solutions, and validation. EBSI[3] and BCDiploma[4] are examples of services built according to this standard with the BC as the TTP where system entities save their data. However, the standard recommendation of W3C[5] and the solutions following it did not consider the issuer accreditation challenge assuming any entity should be able to own a DID and issue any type of VCs.

EBSI utilizes the general purpose Hyperledger Fabric[6] which uses the Raft CA. VC issuers in EBSI apply to an accreditation body for accreditation outside EBSI scope. Once an issuer is accredited in a legacy approach, the accreditation body issues a VC on the EBSI network certifying that this issuer is eligible to issue credentials. Miners in EBSI refer to bodies who attain accreditation confirmation. The BCDiploma platform directly saves all issued VC hashes on the BC which means that they can never be deleted, and it uses a PoW-based BC with linear DL model. Additionally, issuer accreditation service is not provided.

in the late 2020, Wang et al. [327] have comprehensively investigated the state-of-the-art regarding DAG-based DL systems. They provided a general mathematical model of such systems, and categorized existing structures into six types. They could then systematically define potential applications and drawbacks of those structures as they were found either rough in summaries, superficial in analysis, or incomplete in evaluations. The main performance bottleneck of linear DL structure, which is not the case in DAG-based ones, was identified to be the utilized consensus mechanisms. Specifically, the competition among a group of miners for the right of block packaging

[2]https://www.w3.org/
[3]https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EBSI
[4]https://www.bcdiploma.com/en-GB
[5]https://www.w3.org/TR/vc-data-model/#introduction
[6]https://www.hyperledger.org/use/fabric

does not appear in DAG-based DLs as each newly added block is allowed to refer to more than one parent (many-to-many cardinality model of the BC). To solve this issue in linear DLs, block confirmation must be artificially suppressed (e.g. adjust the puzzle difficulty in the consensus method) so that each block is fully attached before the next one's arrival (resulting in one-to-one cardinality model of the BC). DAG-based DLs, on the other hand, support concurrent operations as multiple nodes can simultaneously add TXs/blocks to the DL, thereby significantly improving the throughput.

Type-II DAGs, specifically, have been utilized in several previous works including Spectre [328], Phantom [329] and Meshcash [330]. However, Spectre uses it temporarily for swift processing but the final DL is linearized by a majority voting. Phantom utilizes a PoW consensus and enforces a strict linear ordering over blocks and TXs in the network, resulting in a DL with probabilistic finality. Meshcash utilizes a PoW consensus as well, with blocks pointing for necessity to *every* block confirmed in the previous round. None of those solutions offer a Tree-like multi-dimensional DL as PriFoB does. Specifically, all previous works adopt a many-to-many cardinality model; each newly added block may refer to more than one parent, and each parent may have several children. As will be discussed later, my proposed 3DDL adopts a one-to-many cardinality model; each newly added block refers to specifically one parent, while each parent may have several children. To the best of my knowledge, such 3DDL modelling is the first to propose a permanent, multi-dimensional and PoA- and SoW-based DL with deterministic finality.

6.3 Global Accreditation and Credential Verification

6.3.1 System Modelling

The main objective of the proposed system in this chapter is the simultaneous provision of two major services: i) Institution Accreditation and ii) Credential Verification. The proposed system must be privacy-preserving by design, meaning that the deployed communications protocols and interaction/processing methods shall allow no window for private data leakage. The general architecture of the proposed PriFoB, and a simplified control flow scheme within, are depicted in Figure 6.1. The source-code of the PriFoB solution, along with a tutorial on its setup and deployment, is publicly available at Github[7]. Due to the limited page number of this thesis, I skip the presentation of less important technical details, data layer configurations, and evaluation results. However, complete data and evaluation analysis are provided in [331]. As demonstrated in the figure, PriFoB consists of three major layers:

[7]https://github.com/sed-inf-u-szeged/PriFoB

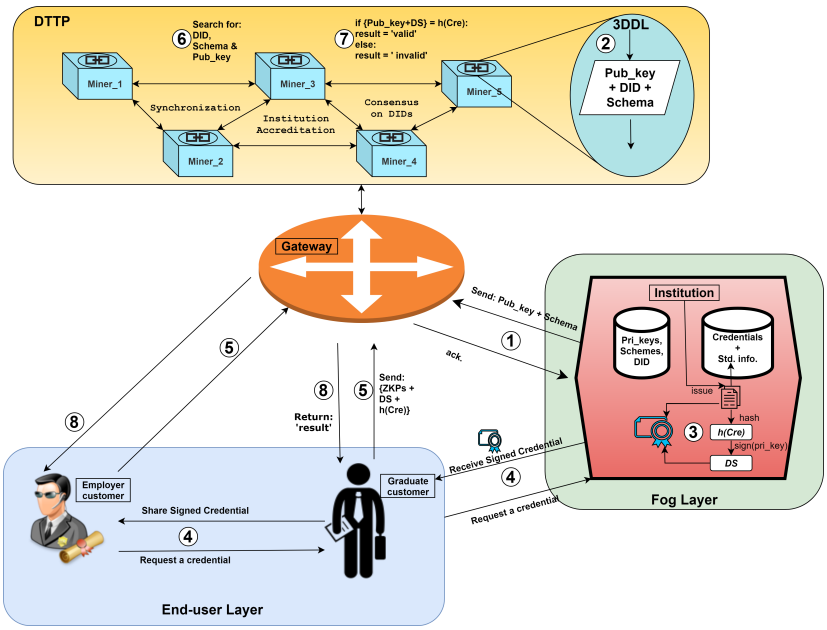


Figure 6.1: *The general architecture and framework of entities in a PriFoB system. The circled numbers indicates the order of steps to remotely accredit an issuer, publish new schemes, issue a new VC by an accredited issuer and verify that VC*

1. The DTTP layer: which consists of the Gateway (GW) and Miners. The GW connects the BC network with the issuers and end-users. Furthermore, it is responsible for bootstrapping new miners with randomly selected peers. Miners, on the other hand are responsible for verifying new blocks and maintaining the consistency of the DL. Furthermore, miners are responsible for validating VCs using DSs and ZKPs.

2. End-user layer: consists of regular end-users requesting to be issued VCs. Those end-users can request, validate, share, or rename their issued VCs. Additionally, those end-users can download the whole or part of the BC. Entities belonging to this layer are not allowed to write on-chain.

3. The Fog layer: consists of issuer(s). An issuer is an extended end-user entity, which is responsible for issuing new VCs to other qualifying end-users. To do so, an issuer is initially required to publish its unique Decentralized Identifier (DID) and a schema(s) (which is a VC template) into the DTTP through the GW. Once both are published, it can issue as many VCs as it needs.

Note that in addition to the functions a non-issuer end-user can perform, an issuer entity can also write on-chain (DID and Schemes), and it can revoke a VC that it has previously issued. Issuers are set in the fog layer because they can be directly connected to regular end-users without a middling element, and they do provide

several types of services to them. However, they are still considered end-users from the DTTP point of view, since the DTTP provides services for them. Some issuers can belong to both end-user layer and fog layer at the same time, since they can request VCs from other issuers as well.

6.3.2 PriFoB Protocol

Knowing that using asymmetric encryption is computationally expensive and bounded, Ionly use it in PriFoB when necessary. In the following subsections, detailed steps of the PriFoB protocol are discussed to clarify the simplified framework depicted in Figure 6.1.

Accreditation

As depicted in Figure 6.1, the first step for an issuer to be able to issue new VCs is to be accredited. To do that, the issuer generates a public key and sends it to the DTTP along with non-private issuer information (e.g. official name, IP-address, etc.). The corresponding private key is then saved and managed locally by the issuer. No encryption is needed in this step, other than ordinary symmetric encryption performed within the frame of the TCP/IP protocol. The combination of data sent by an issuer to request accreditation is called a DID request. Once the request is accepted and the DID is published on-chain (i.e. as a DID block), we say the issuer is accredited and can issue VC schemes. Miners perform the SoW CA on DID blocks (detailed later) in order to maintain the DL consistency.

Schema Publication

A schema is a VC template that is saved on-chain to refer to later when a VC is to be issued/verified. In addition to the DID definitions to which a schema relates to, a schema consists of its own public key and the fields that needs to be filled for each VC of this type. That is, each schema corresponds to a unique type of VCs.

Publishing a new schema upon issuer accreditation includes sending non-private information (e.g. Schema title, public key, etc.). Miners perform the PoA CA on schema blocks (detailed later) in order to maintain the DL consistency. Once the issuer is accredited and its schemes have been published, it can generate new VCs to its clients.

Issuing a Verifiable Credential

As depicted in Figure 6.1, the third step of the PriFoB protocol after publishing a schema is issuing new VCs with its clients' private data (e.g. name, grade, birth-

info, etc.) and perhaps with its own shareable private data as well (e.g. courses, professors' names, admin and registrar's signatures, etc.). Thus, this VC is only saved locally, as I assume that, trivially, customers of an issuer do trust that issuer with their private data. An end-user (e.g. student or hospital patient, I also interchangeably use the terms client, customer, or agent) sends a VC request to the issuer, which also includes the client's public key. The request shall include some private identifying information (e.g. full name, SSN, year of credential issuing, etc.), so that the issuer can share a VC representing the original credential with high confidence that the requester is the client herself. Mandatory identifying data are declared in the schema. Because of that, the client connects with the DTTP to ask for the issuer's public key (which was initially saved on-chain in the first step of the protocol) and the mandatory data that needs to be submitted. Using this public key, the client can encrypt her VC request that includes the mandatory information. Consequently, no entity but the issuer can read private data within the request, as only its private key can decipher the request. The issuer can then use the client's public key to encrypt its response.

If a new type of VCs to be issued, a new schema needs to be submitted and, only after saved on-chain, the new type of VCs can be issued. Note that old schemes remain saved on-chain as the BC provides immutable storage, thus old fashioned VCs remain verifiable despite a new schema application. This is both beneficial and critical. It is beneficial for old scholars who are guaranteed they will not lose their credibility even if the issuer's system is changed. However, it is critical if, for some reason, the issuer decided that some of its previously issued VCs should be considered invalid. In my proposed PriFoB system, I solve these issues by utilizing a novel 3DDL as discussed later.

Mainly, the issuer response shall include two parts (assuming the requested credential was indeed issued) the digital credential, and Sig (using the schema's private key). The encrypted response, which is in fact the VC, can only be then read by the client, as only her private key can decipher it. This is the fourth step of the protocol in Figure 6.1. Once the response is decrypted, the client is free to share and verify the obtained VC (repeat step 4).

Credential Verification

The client may send a verification request to the DTTP (step 5 in Figure 6.1). The verification request includes only non-private data, including: the DID block identifier and index (e.g. issuer official name and its index on-chain), the schema block identifier and index (which might be similar for different issuers but unique for each issuer), the hash of the credential to be validated, and the signature originally provided by the issuer within the VC.

Note that none of these data can reveal any private information about the client.

Accordingly, asymmetric encryption is not required here. Once the BC receives the verification request, a miner, randomly selected by the GW according to the implemented load-balancing criteria, performs a VC verification (steps 6 and 7, technically described in Sections 6.3.3 and 6.3.3). The output of this step defines the response from the DTTP to the client (step 8). That is, the response is either Valid or Invalid, yet the reason for considering a VC invalid shall be also provided. Reasons for considering a VC invalid include: the DID or Schema has not been published on-chain, the hash of the credential is not equivalent to the decrypted Sig, or the VC has been revoked by the issuer.

Miners search for a DID with a claimed identifier and index. If found, it searches within the schemes chain within this DID block for the schema identifier and index. Otherwise, a response is sent to the requester that the issuer/schema is not accredited/registered. Once the schema is found, it searches within the revoke chain within this schema block for the hash of the credential provided within the request. If not found, then the signature is verified using the public key of the schema found. If the signature is valid, and the hash is not in the revoke chain, a response is sent to the requester that the credential is valid. If the signature is valid but the hash is in the revoke chain, a response is sent to the requester that the credential is revoked.

Note that no private data are saved on-chain, or provided for validation. This is the ZKPs scheme I use as the VC is validated without any knowledge requirement. The requester need not to expose any private data other than its address to which the response should be send. The VC validation is performed automatically and publicly without any restrictions or conditions. If a client decides to download the publicly available DL, it can verify any VC without referring to the DTTP in PriFoB.

Revoking a Credential

If an issuer decides to revoke a credential, a revoke request needs to be sent to the DTTP. The revoke request consists of DID and Schema data and the hash of the VC to be revoked. The request should be signed using both the DID private key and the Schema private key and thus the miner handling the request is assured the VC to be revoked is placed correctly and the request is legit. Once the issuer's signatures are verified, a revoke block is added on-chain. Later on, if a client attempts to verify this VC, the DTTP will respond with Revoked instead of Valid. Miners perform the PoA CA on revoke blocks in order to maintain the DL consistency.

6.3.3 Data layer modelling

In this subsection, I discuss how and when different types of data can be encrypted, decrypted, signed, and verified. Furthermore, I explain types of messages and ZKPs exchanged between different entities of PriFoB.

One-way, Symmetric and Asymmetric encryption

In cryptography, there are many types of encryption used to hide shared information. A hashing function $h(.)$, or a one-way encryption function, is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence that is computationally difficult to invert [249]. A hashing function enables the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest [250]. In PriFoB, I use the SHA-256 [332] function for one-way encryption.

Symmetric encryption is the process of turning a readable text (plain text P) into a non-understandable text (cipher C) using an encryption function $E(.)$ [333]. The input of a symmetric encryption function is P or C, and a key k, leading to $E(P, k) = C$. The processes performed by $E(.)$ shall be traversed, using k, if P to be derived from C. That is, $E^{-1}(C, k) = P$. In PriFoB, I use the AES methods [334] for the symmetric encryption.

Asymmetric encryption is a secure method $S(M, k)$ used to ensure that only the receiver is able to decrypt $D(C, g)$ a cipher and read the original message M. I deploy this type of encryption in PriFoB, in addition to one-way and symmetric encryption methods, so that the security of exchanged messages that include private data is guaranteed. This type of encryption implies the generation of two keys, g which decrypts C that was originally encrypted by k. A message M that was encrypted using k is computationally hard to be decrypted unless g is known. One of the most secure and famous asymmetric encryption algorithms is the RSA algorithm [335].

Digital signature and verification

The RSA keys generated above can be swapped without the loss of generality. That is, g may be used to decrypt a cipher C that was encrypted using k, or to encrypt a message M to verify the credibility of M's origin. Specifically:

1. The original sender of M computes:

   $S(h(M), g) \rightarrow Sig$, where $h(.)$ is an agreed on hashing function (e.g. SHA-256),

2. The sender sends the resulting Signature (Sig) along with M [M, Sig] to the receiver,

3. The receiver computes $h(M)$,

4. The receiver computes $D(Sig, k) \rightarrow h'(M)$,

5. if $h(M) = h'(M)$, the receiver shall be confident that M was sent by the original sender who is the only one that can read g.

Because this method is typically used to prove the sender credibility, while anyone can decrypt Sig using the publicly available k, it is called Signing and Verification rather than Encryption and Decryption. Following this remark, both M and Sig shall be encrypted at the sender side using symmetric encryption with a shared key, or asymmetric encryption with the public key k of the *receiver*. As described in the previous subsection, only the receiver then shall be able to $read$ and $verify$ the contents of M and Sig, respectively.

Zero-Knowledge-Proofs (ZKPs)

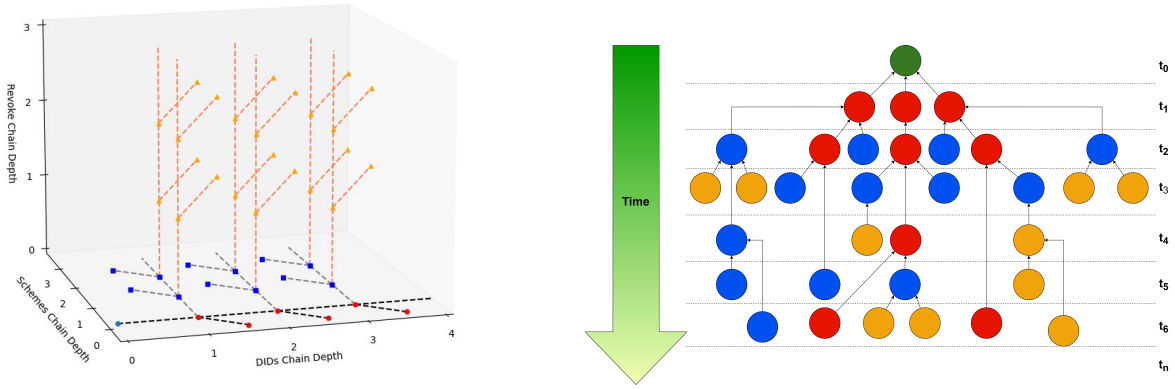
A ZKP is a verification technique which, using cryptography, allows one substance to prove to another component that it knows a specific data or fulfills a particular requirement without disclosing any actual data that supports that evidence [336]. I deploy the ZKP technique in PriFoB with the goal of end-users being able to verify VCs without disclosing any private data within. PriFoB implies that each VC is coupled with a Sig which is the encrypted hash of the issued VC $h(VC)$. Referring to the definitions presented in [337], BC miners and end-users in PriFoB can be considered verifiers and provers, respectively. Following the notations described in subsection 6.3.3, a prover sends the $h(VC)$ and the received Sig accompanied with the VC, which could only be generated by the VC issuer. A verifier then only performs step 5 of subsection 6.3.3. The ZKPs in PriFoB fulfill all the properties of a successful ZKP deployment as follows:

1. **Completeness**: an honest prover can always convince an honest verifier. In PriFoB, No system entity is able to generate a *correct* Sig other than the original issuer of the VC because only the issuer knows the private key used for signing VCs.

2. **Proof of knowledge**: a malicious prover not knowing the secret cannot convince the verifier, except with negligible small probability. This is true in PriFoB as a malicious prover needs to know both $h(VC)$ and g in order to generate a correct Sig, which is not the case according to the PriFoB protocol.

3. **Zero-knowledge**: an honest verifier that follows the protocol cannot learn additional information about the secret. According to the previously presented definition of hashing functions, any alteration within the input of $h(.)$ results in an unexpected output. Additionally, the input can not be known from the hash string (hence, the name one-way-encryption). By allowing the verifier to read $h(VC)$, $h'(VC)$, Sig and k the verifier shall not be able to read/deduce any private data that belongs to the prover nor to the issuer.

More in-depth details on how the ZKPs work and their level of security and privacy can be found in [338].



(a) (b)

Figure 6.2: *A simplified view of the proposed DAG-based 3DDL (dashed links in (a) and arrows in (b) represent the usage of higher depth block of the signature of the linked lower depth block. Green node: Genesis Block, red nodes: DID blocks, blue nodes: schema blocks and orange nodes: revoke blocks)*

The Distributed Ledger

The BC as a PriFoB system element, including all its components, represents a DTTP for different types of agents to hold their public information and/or to securely validate issued VCs. On the other hand, many verifiers are required to perform tasks instead of a single central entity. Thus, it is recommended, in order to fulfil the system globalization feature, to have this DTTP designed as a BC. Additionally, those verifiers need to be granted equal provisioning and maintenance rights of the DL, as their roles in their territories are alike. As general criteria of a system that needs a BC include several equal participants, performing similar tasks, and maintaining a DL using an agreed-on CA, this description perfectly fits the scenario of the DTTP in PriFoB.

The DAG-based DL proposed in PriFoB has three dimensions, each dimension is used for a specific type of blocks. Simplified views of my proposed DL are depicted in Figure 6.2. In Figure 6.2a, the order of confirmed blocks, present at a given time slot within the DL, is demonstrated with reference to the depth of each dimension up to the genesis block. It is noticeable here that each child block is pointing to specifically one parent while each parent block is allowed to have several children blocks. Additionally, several blocks at a given dimension can have similar index.

In Figure 6.2b, mature blocks (will be discussed later) appearing in the DTTP are demonstrated with reference to the time they appear. It is noticeable here that DID blocks can either point to the genesis block or to other DID blocks. Schema blocks can either point to the DID block of, specifically, their issuer, or to other schema blocks generated earlier by their issuer. Revoke blocks can either point to the schema block

that was used to issue the revoked VC, or to any other revoke block that was generated earlier by the same issuer using the same schema block. It is also noticeable here that the time at which a mature block appears does not necessarily define the index of the block nor the position at which it is placed within the DL.

Although both demonstrations within Figure 6.2 are captured from one miner's point of view, other honest miners within the DTTP will have exact similar views. The index and the previous signature of a mature block are decided by the miner who generated that block. In comparison with linear DL models, if a miner receives a valid block with a previous signature that is not of the, specifically, previous block, the new block is rejected. Additionally, the index of a received new valid block is determined by the receiver not the generator of the block.

According to the classification of DAG-based DLs, detailed in [327], each dimension in my proposed DL is of Type-II DAG, where TXs need to be organized in blocks for packaging and the topology is a natural graph. However, the proposed DL model allows for parent blocks being pointed to by several child blocks, while each child block is allowed to point to only one parent block. This results in a tree-like DL, with several blocks potentially having similar index but unique identifiers (i.e. official name of issuer, type of schema, and hash of revoked VC, respectively). To efficiently allocate a block, an orthogonal parameter (*identifier*, *index*) needs to be provided. The identifier of every newly advertised block is checked and repetitions are not allowed. In a rare temporary case where an orthogonal parameter is similar for two different blocks within the same dimension, the longest-chain extension method [339] is used resulting in a DL with no orphaned blocks and no repetitions. This concludes that the proposed DL provides a deterministic finality [254]. The consensus mechanisms utilized to add new blocks will be detailed in later sections.

The DTTP maintains a 3DDL, where each dimension is a DAG structured. The first dimension holds confirmed DID blocks, each of those blocks includes:

1. an IMMUTABLE Header consisting the block type and miner signature,
2. an IMMUTABLE Body used in miner signature generation, consisting:
    - the DID TX sent by the issuer,
    - signatures of active miners (each indicates whether this issuer is accredited or not by the signer),
    - and the signature in the Header of the previous block,
3. a MUTABLE independent chain of schemes (i.e. The second dimension), where future schema blocks issued by this specific issuer shall be saved.

Similar to the first dimension design, the second dimension holds a DAG of confirmed schema blocks each includes:

1. an IMMUTABLE Header,
2. an IMMUTABLE Body (with or without the accreditation signatures of all miners according to application specifications),
3. and a MUTABLE independent chain of revoked credentials (i.e. the third dimension), where future confirmed revoke blocks issued by the DID owner shall be saved.

The third dimension of the proposed 3DDL is dedicated to saving the hashes of revoked credentials. This would allow BC miners to check if the VC was revoked by the issuer, without actually reading any private data within the VC.

In all the three dimensions, the previous signature of a given block is not necessarily the signature in the header of the *last* confirmed block, yet the claimed previous signature must exist in one of the previously confirmed blocks for credibility. The proposed design allows for flexible data confirmation and non-linear chaining, leading to higher block confirmation rates, higher throughput rates, and lower response latency.

To this end, there is no need to implement a time synchronization method between miners (which typically appears in PoA-based BCs) as there is no restrictions on the order of confirmed blocks. Several blocks can simultaneously appear in the network, instead of a single block per time slot, without a consistency problem as they are all considered valid by all miners who confirmed their claimed previous blocks.

The immutability property of confirmed blocks is still preserved as changing a block, and all its consequent confirmed blocks for a successful attack, requires the attacker to know all private keys of all miners who mined the consequent blocks.

6.3.4 Consensus modelling

In PriFoB, I utilize two types of CAs for two different layers of consensus regarding new blocks, namely PoA and SoW. It is assumed that a successful verification of a new schema TX or a new revoke TX means a successful verification of the TX issuer (among other things to validate). Thus, those two types of TXs are processed by only one miner leading to mining a new block using the signature of this miner (i.e. PoA). DID TXs, on the other hand, are propagated throughout the BC network asking each miner to declare whether the DID requester is accredited or not in its country. The decision can be performed both manually by the miner admin, or automatically referring to a local/remote DB (both approaches are implemented).

Once a miner's declaration is ready, the miner's signature using its private key is added to the declaration and the signed declaration is added to the TX. The TX is propagated throughout the network, in the state of unready/immature TX, until the

following strict conditions are met. When a miner receives a DID TX that meets all these conditions, the TX is considered ready or mature to be mined:

1. All active miners (periodically pinged by the GW, e.g. every 5 seconds) have signed the TX
2. Recipient miner has already signed the TX
3. All signatures are correct
4. TX is not found in any previously confirmed DID block (i.e. the claimed DID is unique)

Unlike classical PoA CA, no restrictions are enforced in PriFoB regarding the number of blocks that can be generated within a single time slot. The miner who finds a mature DID TX mines and broadcasts it immediately.

All miners are authorized to mine new blocks, according to the data layer definitions, in any time slot by signing the block body, and adding the signature to the Header of this new block. Note that the accumulated and complete group of correct miners' signatures (i.e. the SoW) represents a trigger for the last miner to start the mining process. In other words, the trigger to mine a new DID block in PriFoB is pulled by the network (instead of a single leader node in typical PoA-based BCs). Consequently, the miner adds a PoA to the new block and propagates it throughout the network. The PoA is then verified, as well as the block body, by all recipient miners, and is added upon successful validation and verification.

Table 6.1: *Computational complexities required to generate new blocks referring to different types of requests, and the expected appearance rates of different types of TXs throughout the life-cycle of an accredited institution*

Request type	appearance	computational complexity (wrt. no. Miners)
DID TX	Few	(n+1) signing + (2n-1) verification
Schema TX	Moderate	1 signing + n verification
VC validation	Most	1 verification
Revoke TX	Rare	1 signing + (2n - 1) verification

Table 6.1 presents the computational complexities expected, for each type of TXs, from the time it is delivered to the GW, until it is confirmed and responded to. Note that the expected relative appearance of different TX types can be deduced logically. That is, each issuer is allowed only one unique DID, while it is expected to issue unlimited number of VCs referring to a limited number of schemes. Additionally, it is relatively rather rare that an institution would revoke a VC it issued. This being said, an expiration data of a VC can be injected within by the issuer, and consequently

the accompanying signature, so that it can be checked once the VC is found valid. A similar approach can be used for DIDs and schemes as well.

Compared to different consensus complexities presented in [340], PriFoB performs optimally for schema and revoke TXs as both require O(1) computational complexity by the system (i.e. signing). For DID blocks, PriFoB requires O(n) computational complexity by the system, which is better than the PBFT and RBFT CAs with complexities $O(n^2)$ and $O(n^3)$, respectively.

It can be observed from the values presented in the table that the verification is used much more for all types of TXs than signing. It can also be observed that the most appearing type of TXs is the VC validation request which requires no signing by any BC entity as it is not saved on-chain. This, in fact, was the motivation to adopt the RSA encryption methods. For more details, a thorough comparison between the mostly used ECC and the adopted RSA encryption methods is presented in the published paper.

6.3.5 Evaluation

Next, I asses the privacy-awareness of PriFoB, and compare real measures of PriFoB deployed in the cloud against well documented implementations of Ethereum and different Hyperledger platforms, including Indy, Besu and Fabric. Furthermore, I parameterize a Discrete Event Simulation environment, using the real data I obtained, to predict the throughput and power utilization of PriFoB in different settings.

Privacy

As PriFoB complies with the W3C standard and further does not allow for any private date to be saved on the DL, PriFoB can be trivially considered a GDPR-compliant system. As had been described earlier, VCs are only saved locally at the end-user layer and are never sent to the DTTP. Additionally, the deployed PKI schemes, the ZKPs with collision resistant hash functions, along with the utilization of DSs, all lead to a Privacy-by-Design implementation.

Nevertheless, I referred to the GDPR checklist for data controllers available at [341]. I have conducted an information audit to determine what information PriFoB processes and who has access to it. Thus, I provided clear information about PriFoB data processing and legal justification in its privacy policy (PriFoB-based solutions will have a legal justification for its data processing activities as end-users voluntarily sign up into the system). Encryption, pseudonymization, and anonymization of personal data wherever possible is performed in PriFoB. Additionally, it is easy for PriFoB customers to request and receive all the information it has about them. That is, the whole BC is directly downloadable via the application interface. Finally, different types of TXs can be revoked and regenerated, while private data is only saved

locally, making the private data controllable only by its owners (there is no need to request private data deletion).

Latency

I have previously analyzed the latency of BC-based solutions utilizing Hyperledger Indy in [342]. Urbančok [343] described and compared four open-source BC platforms, namely Ethereum, Hyperledger's Fabric, Besu, and Iroha, in different terms including latency. Xu et al. [344] analyzed the latency of BC-based solutions utilizing Hyperledger Fabric. Zhang et al. [345] experimentally evaluated the performance of Ethereum testnets in terms of Account balance query latency, Block generation time and End-to-end TX acceptance latency. Härer and Fill [346] utilized the main net of Ethereum platform and proposed a credential verification solution, on which they performed latency assessment. Bampatsikos et al. [347] proposed a solution for mitigating probable Computational Denial of Service (CDoS) attacks when utilizing Ethereum for credential verification purposes, and provided latency assessment for their solution. The differences and similarities, along with brief comparison of the results, for those works and PriFoB, in Table 6.2.

Some of these works evaluated their solutions using simulation (all miner nodes run on a single machine), while others evaluated their solutions using real test-beds (each miner is allocated a different machine). I compare the latency of PriFoB, however, with all of them, to prove PriFoB outperformance and provide insights regarding expected latency with different scalability measures. Additionally, I run several test scenarios in order to comprehensively compare PriFoB with all of them. The results of the scenarios I tested, along with each scenario parameters, are detailed in Figure 6.3. To facilitate reading and comparing the performance results, I have color-coded the tables' cells according to the scale provided within the figure.

The performance of PriFoB was evaluated using a Proof of Concept (PoC) system composed of a single GW, a network of miners with different sizes (2, 4 and 6 miners), and a script I implemented, that emulates several issuers and agents simultaneously communicating with the DTTP. I deployed each of those entities on a separate VM at the Google Cloud Platform, each is of type E2-medium (2 Intel(R) Xeon(R) vCPUs clocked at 2.30GHz with 10GB of RAM), running Linux 20.10 OS. Alternatively, issuer and agent implementations are available and tested within the project repository but they can not be used to 'stress-test' the DTTP in PriFoB.

I tested PriFoB using one GW to obtain the worst results possible. The bottleneck effect can trivially justify the observable proportional relation, in Figure 6.3, between the number of miners and the average latency. However, the Read latency in PriFoB should not be affected by the number of miners when deploying a computationally powerful GW. As a result, my containerized implementation of PriFoB elements, including the GW, shall show better measurements than those presented

Table 6.2: *PriFoB comparison with previous related works, that proposed solutions for distributed credential verification systems, in terms of utilized Blockchain platform, granting institution accreditation services, number of Miners (M), assisted request type (T), lower and upper bounds of request per second rates (req/s) and the lower and upper bounds of response Latency measured as second per request (s/req)*

Solution	CA	w/Accreditation?	M	T	req/s	Latency (s/req)
Indy [342]	Plenum (PBFT)	NO	4	DID/Schema (write)	1–250	2–6
				Any (read)		0.08–1.6
			8	DID/Schema (write)		2–10
				Any (read)		0.1–2.5
Besu [343]	PoA	NO	4	Any (write)	10–100	3.34–4.60
				Any (read)		0.04–0,56
Fabric [344]	RAFT	NO	2	Any (wirte)	50–250	0.6–0.8
			4	Any (write)		0.7–0.95
Fabric [343]		NO	2	Any (read)	10–100	0.6–0.8
Ethereum [343]	PoW	NO	2	Any (write)	10–100	5.03–5.58
			2	Any (read)		0.02–0.06
Ethereum [345]	PoA	NO	N/A	Any (write)	25–100	5–34
			N/A	Any (read)		0.2–0.4
Ethereum [346]	PoW	YES	Main Net	DID (write)	1–100	47–114
Ethereum [347]	PoW	YES	Main Net	DID (write)	N/A	5–40
PriFoB	SoW+PoA	YES	**2–6**	**DID (write)**	**1–250**	**0.013–1.09**
				Schema (write)		**0.006–0.6**
				Revoke (write)		**0.005–0.09**
				Any (read)		**0.003–0.14**

here, if more powerful machines and/or more GWs were deployed using e.g. Kubernetes orchestrator. It is worth noting here that despite the apparent bottleneck effect in the experiments, PriFoB still outperforms all the compared related systems.

Throughput

Next, I mathematically model PriFoB so its general behaviour becomes predictable, while the rate of requests per second (λ) and the number of miners (n) increase. To do so, I refer to the Queuing Theory [348] and characterize PriFoB using the Kendall's notation as (λ=M/D=M/n):(FCFS/∞/∞), where M, D and FCFS stand for Poisson distribution, output distribution and the First-Come-First-Served discipline, respectively. Utilizing such approach to model BC-based systems has been predominantly used in the literature [349]. To compute the expected average processing time (W_s) in PriFoB, I use Equation 6.1:

$$W_s = \frac{\mu(\lambda/\mu)^n}{(n-1)!(n\mu-\lambda)^2}P_0 + \frac{1}{\mu} \tag{6.1}$$

Solution	no. Miners	request / second								
		1	10	20	25	50	100	150	200	250
Indy	4	0.08				0.3	0.6	0.9	1.1	1.6
	8	0.1				0.4	0.5	0.8	1.1	2.5
Besu	4		0.04	0.1	0.3	0.4	0.56			
Fabric	2		0.6	0.65		0.7	0.8			
Ethereum	2		0.02	0.03		0.06	0.06			
Ethereum	N/A				0.2	0.3	0.4			
PriFoB (read)	2	0.003	0.006	0.009	0.009	0.015	0.02	0.03	0.05	0.05
	4	0.003	0.008	0.01	0.012	0.02	0.03	0.06	0.08	0.14
	6	0.004	0.006	0.016	0.02	0.036	0.057	0.08	0.1	0.13

Solution	no. Miners	request / second								
		1	10	20	25	50	100	150	200	250
Indy	4	2.7				2.5	2.2	4.5	5.4	6.4
	8	2.6				2.6	2.5	4.9	5.6	10.4
Besu	4		3.34	3.37		4.6	4.32			
Fabric	2					0.65	0.7	0.72	0.77	0.8
Fabric	4					0.7	0.8	0.82	0.85	0.9
Ethereum	2		5.03	5.04		5.58	5.07			
Ethereum	N/A	7			12	15	20			
Ethereum	main net	47					114			
Ethereum	main net	5		25		35	40			
PriFoB DID	2	0.013	0.032	0.03	0.08	0.15	0.3	0.5	0.75	0.9
	4	0.02	0.04	0.037	0.08	0.17	0.4	0.5	0.72	0.89
	6	0.046	0.06	0.12	0.14	0.26	0.47	0.71	0.98	1.09
PriFoB schema	2	0.006	0.02	0.03	0.05	0.1	0.2	0.3	0.45	0.6
	4	0.007	0.02	0.04	0.04	0.09	0.18	0.28	0.38	0.46
	6	0.008	0.016	0.04	0.054	0.095	0.19	0.26	0.36	0.45
PriFoB revoke	2	0.005	0.01	0.01	0.01	0.02	0.03	0.04	0.06	0.09
	4	0.005	0.009	0.01	0.012	0.02	0.05	0.06	0.07	0.08
	6	0.006	0.016	0.016	0.013	0.02	0.03	0.05	0.07	0.08

Figure 6.3: *Color-coded average response latency for PriFoB verification requests per second (upper table), and average response latency for PriFoB DID, Schema and Revoke write requests per second (lower table) against average read and write latency measurements, reported in the literature, for major Blockchain solutions utilized for objectives similar to PriFoB*

where μ and P_0 are the mean service rate per busy server (request per second i.e. $\mu = 1/\text{Latency}$) and the probability that there are Zero requests in the system, respectively. For all $n \times \mu > \lambda$, I compute P_0 using Equation 6.2:

$$P_0 = \frac{1}{[\sum_{i=0}^{n-1} \frac{1}{i!}(\frac{\lambda}{\mu})^i] + \frac{1}{n!}(\frac{\lambda}{\mu})^n \frac{n\mu}{n\mu-\lambda}} \tag{6.2}$$

All the following experiments were run using the Discrete Event Simulation tool provided at [350], with suitable modifications and tuning referring to my model and real parameterization. I measured W_s for $\lambda \in [1, 250]$ with a static $n = 10$ and using $\mu = 335$ referring to the real READ results detailed previously. Accordingly, I could simulate the general effect of increasing λ and compare it with the real data to validate the simulation tool. The results obtained for the first scenario are depicted in Figure 6.4. As can be noticed, the simulation results comply with the real measures obtained as increasing λ results in linear increment of W_s. However, average W_s is

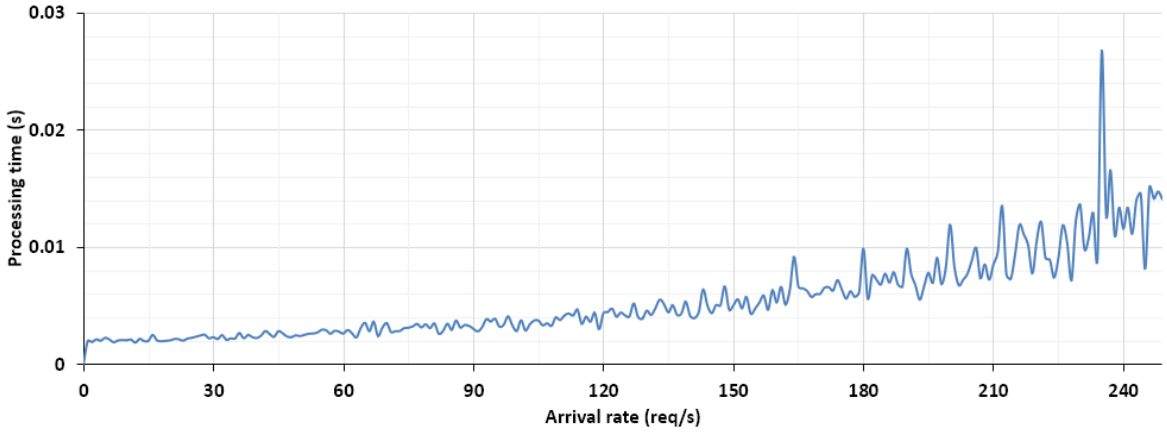


Figure 6.4: *Processing time per READ request in PriFoB for* $\lambda \in [1, 250]$*,* $n = 10$ *and* $\mu = 335$

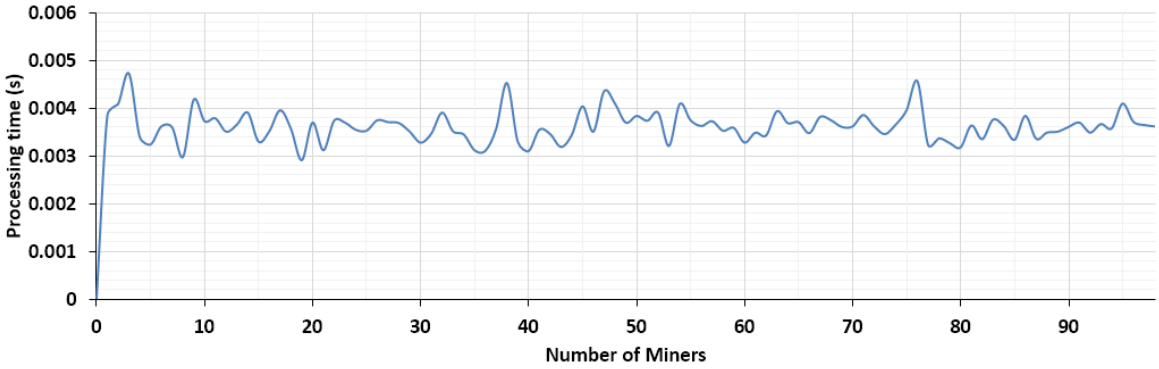


Figure 6.5: *Processing time per READ request in PriFoB for* $\lambda = 10$*,* $n \in [1, 100]$ *and* $\mu = 335$

less than the real data presented earlier, as expected, due to the deployment of more mining nodes. This indeed shall increase the overall throughput of the system as more servers are able to process an equivalent λ.

To capture the general effect on W_s when increasing n, I tested $\lambda = 10$, $n \in$ [1–100] and using $\mu = 335$. The results obtained are depicted in Figure 6.5. Here, the simulation assumes that all incoming requests are immediately distributed among available miners (i.e. no bottleneck effect). Accordingly, the latency is nearly constant between 3–5 millisecond per request. This proves that the proportional relation between n and the Read latency is mainly attributed to the limited computational power of the GW. Uncontrollable communication delays and/or unpredicted hit ratios when searching the BC can also cause such effect. For all of these reasons, I used a static μ in both simulated scenarios so far, as μ is not related to n for all types of TXs, except for DID TXs.

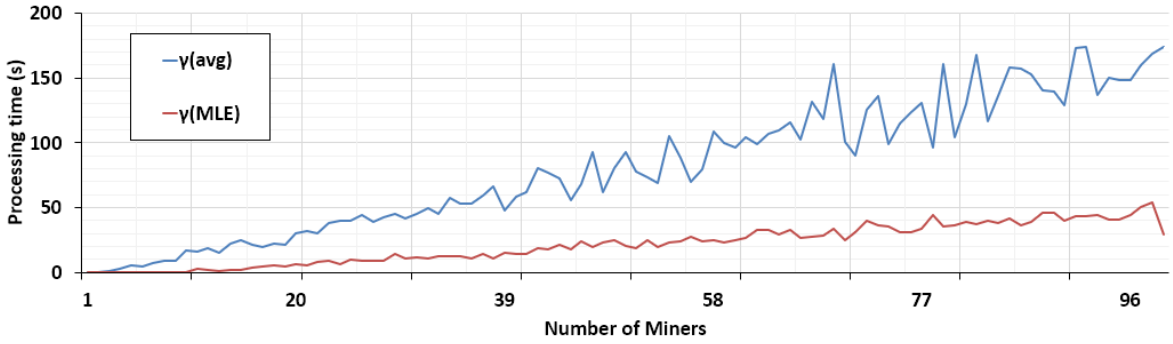


Figure 6.6: *Processing time per DID request in PriFoB for $\lambda = 10$ DID TXs, $n \in [1, 100]$, $\gamma_{avg} = 0.03$ and $\gamma_{MLE} = 0.01$*

For DID TXs, however, μ is affected by changing n as more miners in the system implies more signature generations. Consequently, increasing n should, theoretically, increase W_s for DID TXs. For this reason, the results depicted in Figure 6.5 do not represent the expected behaviour of PriFoB for DID TXs. To address this, I estimate the values of μ for increased n using the latency observations obtained previously. Let $X_{i,l}$ be the real DID TX latency for $i \in Y$ = [2, 4, 6], $l \in Z$ = [1–250]. The increase in latency can then be calculated per each added miner, denoted by $\gamma_{i,j,l}$, using Equation 6.3:

$$\gamma_{i,j,l} = \frac{X_{i,l} - X_{j,l}}{j - i} \; \forall \, i, j \in Y, \, l \in Z \, and \, i < j \tag{6.3}$$

Using data in the set Γ, which consists of all γ values, $\gamma_{avg} = 0.03$ second per request per added miner. The Maximum Likelihood Estimation (MLE) $\gamma_{MLE} = 0.01$ second per request per added miner. The MLE was obtained from a Poisson Distribution obtained from the set Γ. Note that MLE is generally more accurate than the average [351]. Both γ_{avg} and γ_{MLE} were injected into Equation 6.1 resulting Equation 6.4:

$$W_{s_{DID}} = \frac{\mu(\lambda/\mu)^n}{(n-1)!(n\mu - \lambda)^2} P_0 + \frac{1}{\mu} + (n \times \gamma) \tag{6.4}$$

Predicting $W_{s_{DID}}$ values for different n values is now realistic using both of γ values. I tested $\lambda = 10$ DID TXs and $n \in [1, 100]$. The results are depicted in Figure 6.6.

Power Utilization

Next, I attempt to further evaluate PriFoB in terms of power utilization. To achieve this, I injected several variables into my mathematical model implementation. I could

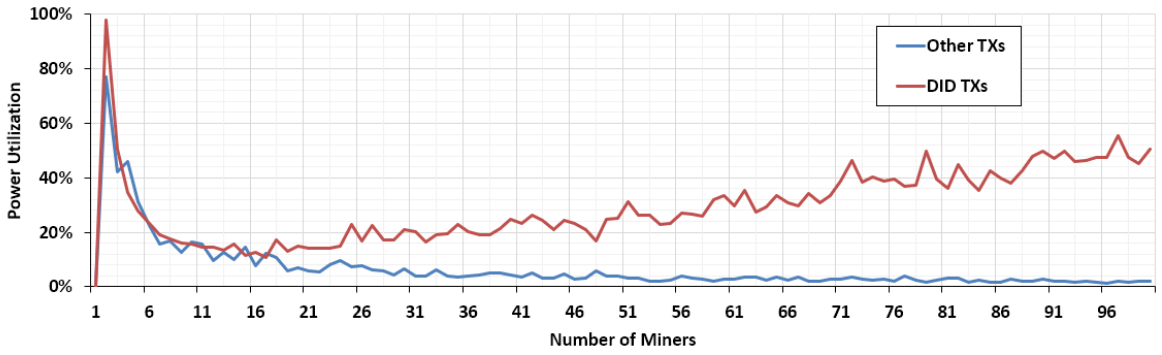


Figure 6.7: *Power utilization with $\lambda = 250$, $n \in [1, 100]$ for DID TXs ($\gamma_{MLE} = 0.01$) and all other types of TXs ($\mu = 335$)*

then assess the percentage of system power utilization for known λ and n. I tested for $\lambda = 250$, $n \in [1, 100]$, a dynamic μ for DID TXs with $\gamma_{MLE} = 0.01$, and a static $\mu = 335$ for all other types of TXs. Figure 6.7 describes the percentage of computational power consumption out of the total available computational power, to process all received requests. For all types of TXs other than DID TXs, it is clear from the obtained results that adding more miners to the system shall enhance the overall efficiency in terms of computational power consumption per request. However, increasing n implies higher power consumption rates due to the required SoW computational complexity. Note that if the optional accreditation service in PriFoB is deactivated, DID TXs shall consume as much energy as any other type of TXs. Also note that the power utilization here is inversely proportional with μ which is rather low in the presented experiments, due to the low computational capacity of test-bed infrastructure.

6.4 Discussion and concluding remarks

BC deployment in a wide range of applications was proven as an enhancement factor in terms of security [352], decentralization [353], reliability [354], and optimization of multi-party decision making [355]. Generally, these criteria enhanced by successful BC integration, are considered the main challenges in IoT, FC, and cloud based SSs.

Figure 6.8 depicts the design space of a vision and research methodology for performing research in BC technology integration. The main entities are: (i) BCs, (ii) SSs and (iii) applications. Each of the entities in the demonstrated design space has unique identification layers, where different services and protocols can be placed and investigated, be it the end-user devices, servers/APs in the FC tier, or VMs in the cloud. The infrastructure, specifically, can be studied according to different P2P con-

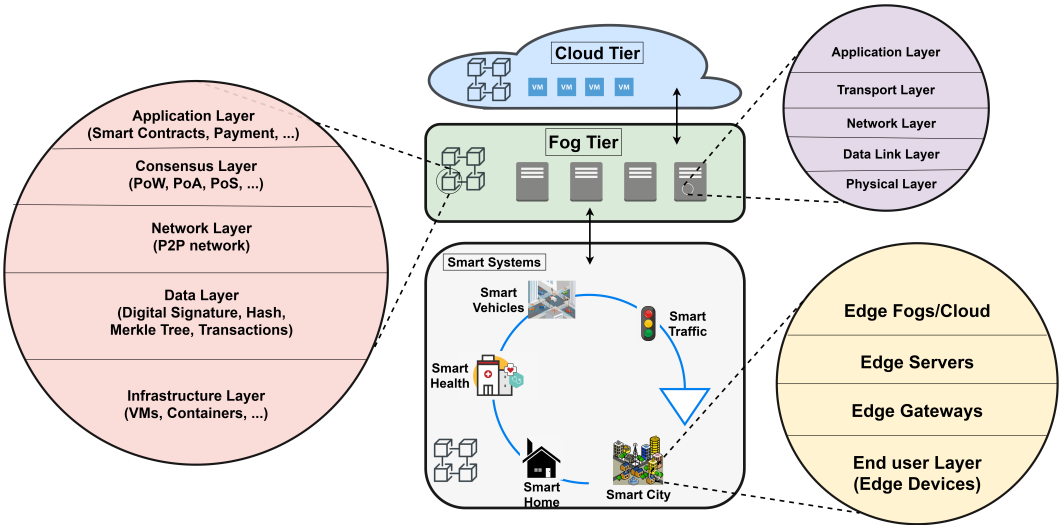


Figure 6.8: *Design Space for Blockchain and Smart Systems integration within a fog-enhanced cloud architecture*

nection models with technical consideration referred to by the OSI network model. Finally, a SS is defined by edge devices (mainly corresponding to end-user devices in an IoT enabled system), edge gateways that locally control and secure communications among edge devices and with upper layers, edge servers (corresponding to the fog nodes in the lowest layer of the fog), and global servers, clouds or upper FC layers.

In this chapter, I have proposed PriFoB: a Privacy-aware FC-enhanced BC-based solution for global accreditation and digital credential verification. I used ZKPs, DSs, SHA-256, AES and RSA encryption schemes, and two different CAs, namely PoA and SoW. Furthermore, I have proposed a novel Three-Dimensional DAG-based model of the Distributed Ledger (3DDL) with efficient deterministic finality. I evaluated PriFoB in terms of security, privacy, latency, throughput and power utilization. Additionally, I compared a realized cloud-based deployment of PriFoB with similar BC-based solutions, for various system parameterization. PriFoB outperformed all of the recently proposed solutions utilizing Ethereum and Hyperledger projects (Besu, Fabric and Indy). I provided a ready-to-deploy implementation of PriFoB, and I made it available at a public, open-source repository.

Future directions include the investigation of deploying new techniques into PriFoB, such as Sharding and Merkle Trees with light nodes. Specifically, I will investigate available opportunities for increasing the block size in PriFoB, as it is currently set to 1 TX/B. In the future, I also plan to provide a web-based implementation of PriFoB entities and a wallet application for mobile devices. To optimize the block finality time and message propagation, I plan to enhance future versions of PriFoB by deploying the DONS protocol [234] into the DTTP. The following points summarizes

my contributions presented in this chapter:

IV/1. I designed and tested a novel hybrid PoA-SoW consensus algorithm that outperformed PoW, PoA, pBFT, and RAFT consensus algorithms.

IV/2. I designed and tested a novel DAG-based three-Dimensional Distributed Ledger model (3DDL) which was proven secure and efficient compared to state-of-the-art DL models.

IV/3. I deployed the proposed 3DDL model and PoA-SoW consensus mechanism in a novel Privacy-aware Fog-enhanced Blockchain-based application for global accreditation and credential verification, which was proven more efficient compared to major state-of-the-art solutions.

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[2] Mayra Samaniego and Ralph Deters. Blockchain as a service for iot. In *2016 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)*, pages 433–436. IEEE, 2016.

[3] Victor Garcia-Font. Socialblock: An architecture for decentralized user-centric data management applications for communications in smart cities. *Journal of Parallel and Distributed Computing*, 2020.

[4] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[5] Hoang Tam Vo, Ashish Kundu, and Mukesh K Mohania. Research directions in blockchain data management and analytics. In *EDBT*, pages 445–448, 2018.

[6] Meng Li, Liehuang Zhu, and Xiaodong Lin. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4573–4584, 2018.

[7] Benedikt Notheisen, Jacob Benjamin Cholewa, and Arun Prasad Shanmugam. Trading real-world assets on blockchain. *Business & Information Systems Engineering*, 59(6):425–440, 2017.

[8] Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 131–138. IEEE, 2015.

[9] Mazin Debe, Khaled Salah, Muhammad Habib Ur Rehman, and Davor Svetinovic. Iot public fog nodes reputation system: A decentralized solution using ethereum blockchain. *IEEE Access*, 7:178082–178093, 2019.

[10] Nir Kshetri and Jeffrey Voas. Blockchain-enabled e-voting. *IEEE Software*, 35 (4):95–99, 2018.

[11] Paul Dunphy and Fabien AP Petitcolas. A first look at identity management schemes on the blockchain. *IEEE security & privacy*, 16(4):20–29, 2018.

[12] Xiaoyang Zhu and Youakim Badr. Fog computing security architecture for the internet of things using blockchain-based social networks. 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)*, pages 1361–1366. IEEE, 2018.

[13] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

[14] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer, 2015.

[15] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9(2):397–413, 2016.

[16] Ethereum. Ethereum average block time chart, June 2020. URL `https://etherscan.io/chart/blocktime`. [Online; accessed 25-June-2020].

[17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[18] Andrea Coladangelo and Or Sattath. A quantum money solution to the blockchain scalability problem. *arXiv preprint arXiv:2002.11998*, 2020.

[19] Jörg Becker, Dominic Breuker, Tobias Heide, Justus Holler, Hans Peter Rauer, and Rainer Böhme. Can we afford integrity by proof-of-work? scenarios inspired by the bitcoin currency. In *The economics of information security and privacy*, pages 135–156. Springer, 2013.

[20] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1), 2018.

[21] S King and S Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stack, august 19, 2012, 2014.

[22] Tengfei Xue, Yuyu Yuan, Zahir Ahmed, Krishna Moniz, Ganyuan Cao, and Cong Wang. Proof of contribution: A modification of proof of work to increase mining efficiency. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 636–644. IEEE, 2018.

[23] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*, pages 1–6, 2016.

[24] JP Buntinx. What is proof of elapsed time. *The Merkle Hash. Available online: https://themerkle. com/what-is-proof-of-elapsed-time/(accessed on 8 August 2020)*, 2017.

[25] Anushree A Avasthi and Ankur Saxena. Two hop blockchain model: Resonating between proof of work (pow) and proof of authority (poa). *International Journal of Information Systems & Management Science*, 1(1), 2018.

[26] Andrzej Wilczyński and Joanna Kołodziej. Modelling and simulation of security-aware task scheduling in cloud computing based on blockchain technology. *Simulation Modelling Practice and Theory*, 99:102038, 2020.

[27] Zee Ali. A simple introduction to blockchain algorithms. `https://blog.goodaudience.com/a-simple-introduction-to-blockchain-algorithms-ca05b9bcc32f`, 2019. [Online; accessed 26-October-2019].

[28] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1, 2012.

[29] Daniel Larimer. Transactions as proof-of-stake. *Nov-2013*, 2013.

[30] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International conference on financial cryptography and data security*, pages 142–157. Springer, 2016.

[31] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. A survey on long-range attacks for proof of stake protocols. *IEEE Access*, 7:28712–28725, 2019.

[32] Rong Zhang and Wai Kin Victor Chan. Evaluation of energy consumption in block-chains with proof of work and proof of stake. In *Journal of Physics: Conference Series*, volume 1584(1), page 012023. IOP Publishing, 2020.

[33] Keke Gai, Meikang Qiu, and Hui Zhao. Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. *Journal of Parallel and Distributed Computing*, 111:126–135, 2018.

[34] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78. IEEE, 2015.

[35] Evangelos K Markakis, Kimon Karras, Nikolaos Zotos, Anargyros Sideris, Theoharris Moysiadis, Angelo Corsaro, George Alexiou, Charalabos Skianis, George Mastorakis, Constandinos X Mavromoustakis, et al. Exegesis: Extreme edge resource harvesting for a virtualized fog environment. *IEEE Communications Magazine*, 55(7):173–179, 2017.

[36] Leandro Loffi, Carla Merkle Westphall, Lukas Derner Grüdtner, and Carlos Becker Westphall. Mutual authentication with multi-factor in iot-fog-cloud environment. *Journal of Network and Computer Applications*, 176:102932, 2021.

[37] Pooyan Habibi, Mohammad Farhoudi, Sepehr Kazemian, Siavash Khorsandi, and Alberto Leon-Garcia. Fog computing: A comprehensive architectural survey. *IEEE Access*, 8:69105–69133, 2020.

[38] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya. Fog computing: Principles, architectures, and applications. In *Internet of things*, pages 61–75. Elsevier, 2016.

[39] OpenFog Consortium. Openfog reference architecture for fog computing. *Architecture Working Group*, pages 1–162, 2017.

[40] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*, pages 169–186. Springer, 2014.

[41] Ammar Awad Mutlag, Mohd Khanapi Abd Ghani, Mazin Abed Mohammed, Mashael S Maashi, Othman Mohd, Salama A Mostafa, Karrar Hameed Abdulkareem, Gonçalo Marques, and Isabel de la Torre Díez. Mafc: Multi-agent fog computing model for healthcare critical tasks management. *Sensors*, 20 (7):1853, 2020.

[42] Fariba Khosroabadi, Faranak Fotouhi-Ghazvini, and Hossein Fotouhi. Scatter: Service placement in real-time fog-assisted iot networks. *Journal of Sensor and Actuator Networks*, 10(2):26, 2021.

[43] Tian Wang, Jiyuan Zhou, Anfeng Liu, Md Zakirul Alam Bhuiyan, Guojun Wang, and Weijia Jia. Fog-based computing and storage offloading for data synchronization in iot. *IEEE Internet of Things Journal*, 6(3):4272–4282, 2018.

[44] Muhammad Arif, Guojun Wang, Valentina Emilia Balas, Oana Geman, Aniello Castiglione, and Jianer Chen. Sdn based communications privacy-preserving architecture for vanets using fog computing. *Vehicular Communications*, 26: 100265, 2020.

[45] Mozhdeh Farhadi, Jean-Louis Lanet, Guillaume Pierre, and Daniele Miorandi. A systematic approach toward security in fog computing: Assets, vulnerabilities, possible countermeasures. *Software: Practice and Experience*, 50(6):973–997, 2020.

[46] Hamza Baniata and Attila Kertesz. A survey on blockchain-fog integration approaches. *IEEE Access*, 8:102657–102668, 2020.

[47] Hamza Baniata and Attila Kertész. Pf-bvm: A privacy-aware fog-enhanced blockchain validation mechanism. In *CLOSER*, pages 430–439, 2020.

[48] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. Pf-bts: A privacy-aware fog-enhanced blockchain-assisted task scheduling. *Information Processing and Management*, 58, 2021.

[49] Deloitte. Deloitte's 2019 global blockchain survey. 2019.

[50] Sz Varadi, G Gultekin Varkonyi, and Attila Kertész. Legal issues of social iot services: The effects of using clouds, fogs and ai. In *Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications*, pages 123–138. Springer, 2020.

[51] Barbara Guttman and Edward A Roback. *An introduction to computer security: the NIST handbook*. DIANE Publishing, 1995.

[52] Jianbing Ni, Aiqing Zhang, Xiaodong Lin, and Xuemin Sherman Shen. Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine*, 55(6):146–152, 2017.

[53] Binara NB Ekanayake, Malka N Halgamuge, and Ali Syed. Security and privacy issues of fog computing for the internet of things (iot). In *Cognitive Computing for Big Data Systems Over IoT*, pages 139–174. Springer, 2018.

[54] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications*, pages 685–695. Springer, 2015.

[55] Mohamed Amine Ferrag, Abdelouahid Derhab, Leandros Maglaras, Mithun Mukherjee, and Helge Janicke. Privacy-preserving schemes for fog-based iot applications: Threat models, solutions, and challenges. In *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pages 37–42. IEEE, 2018.

[56] Mohammad Aminul Hoque and Ragib Hasan. Towards an analysis of the architecture, security, and privacy issues in vehicular fog computing. In *2019 SoutheastCon*, pages 1–8. IEEE, 2019.

[57] Geetha Kurikala, K Gurnadha Gupta, and A Swapna. Fog computing: Implementation of security and privacy to comprehensive approach for avoiding knowledge thieving attack exploitation decoy technology. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 2(4):176–181, 2017.

[58] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

[59] Seyedakbar Mostafavi and Waswa Shafik. Fog computing architectures, privacy and security solutions. *Journal of Communications Technology, Electronics and Computer Science*, 24:1–14, 2019.

[60] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304, 2017.

[61] Neha Shrikant Dhande. Fog computing: review of privacy and security issues. *Intern. J. of Engineering Research and General Science*, 3(2):864, 2015.

[62] Khalid A Fakeeh. Privacy and security problems in fog computing. *Commun Appl Electron*, 4(7), 2016.

[63] Hyun-Jae Nam, Ho-Yeol Choi, Hyung-June Shin, Hyun-Soo Kwon, Jong-Min Jeong, Chang-Hee Hahn, and Jun-Beom Hur. Security and privacy issues of fog computing. *The Journal of Korean Institute of Communications and Information Sciences*, 42(1):257–267, 2017.

[64] T Veerraju and K Kiran Kumar. A survey on fog computing: research challenges in security and privacy issues. *International Journal of Engineering & Technology*, 7(2.7):335–340, 2018.

[65] Tian Wang, Jiyuan Zhou, Xinlei Chen, Guojun Wang, Anfeng Liu, and Yang Liu. A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):3–12, 2018.

[66] Rajani Sharma and Rajender Kumar Trivedi. Literature review: cloud computing–security issues, solution and technologies. *International Journal of Engineering Research*, 3(4):221–225, 2014.

[67] Zheng Yan, Xueyun Li, and Raimo Kantola. Heterogeneous data access control based on trust and reputation in mobile cloud computing. In *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, pages 65–113. Springer, 2017.

[68] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H Glitho, Monique J Morrow, and Paul A Polakos. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 20(1):416–464, 2017.

[69] Muhammad Baqer Mollah, Md Abul Kalam Azad, and Athanasios Vasilakos. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, 84:38–54, 2017.

[70] Lingjuan Lyu, Karthik Nandakumar, Ben Rubinstein, Jiong Jin, Justin Bedo, and Marimuthu Palaniswami. Ppfa: privacy preserving fog-enabled aggregation in smart grid. *IEEE Transactions on Industrial Informatics*, 14(8):3733–3744, 2018.

[71] Mianxiong Dong, Kaoru Ota, and Anfeng Liu. Preserving source-location privacy through redundant fog loop for wireless sensor networks. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1835–1842. IEEE, 2015.

[72] Rupeng Yang, Qiuliang Xu, Man Ho Au, Zuoxia Yu, Hao Wang, and Lu Zhou. Position based cryptography with location privacy: A step for fog computing. *Future Generation Computer Systems*, 78:799–806, 2018.

[73] Cisco. Cisco fog data services. `https://www.cisco.com/c/en/us/products/\cloud-systems-management/fog-data-services/index.html?dtid=osscdc000283`, 2019. [Online; accessed 15-January-2020].

[74] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.

[75] Raheel Ahmed Memon, Jian Ping Li, Muhammad Irshad Nazeer, Ahmad Neyaz Khan, and Junaid Ahmed. Dualfog-iot: Additional fog layer for solving blockchain integration problem in internet of things. *IEEE Access*, 7:169073–169093, 2019.

[76] NN Pokrovskaia. Tax, financial and social regulatory mechanisms within the knowledge-driven economy. blockchain algorithms and fog computing for the efficient regulation. In *2017 XX IEEE International Conference on Soft Computing and Measurements (SCM)*, pages 709–712. IEEE, 2017.

[77] Paul Rimba, An Binh Tran, Ingo Weber, Mark Staples, Alexander Ponomarev, and Xiwei Xu. Comparing blockchain and cloud services for business process execution. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 257–260. IEEE, 2017.

[78] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Blockchain and iot integration: A systematic survey. *Sensors*, 18(8):2575, 2018.

[79] Rafael Brundo Uriarte and Rocco De Nicola. Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards. *IEEE Communications Standards Magazine*, 2(3):22–28, 2018.

[80] Tiago M Fernández-Caramés and Paula Fraga-Lamas. Towards next generation teaching, learning, and context-aware applications for higher education: A review on blockchain, iot, fog and edge computing enabled smart campuses and universities. *Applied Sciences*, 9(21):4479, 2019.

[81] Noshina Tariq, Muhammad Asim, Feras Al-Obeidat, Muhammad Zubair Farooqi, Thar Baker, Mohammad Hammoudeh, and Ibrahim Ghafir. The security of big data in fog-enabled iot applications including blockchain: a survey. *Sensors*, 19(8):1788, 2019.

[82] Geogen George and Suresh Sankaranarayanan. Light weight cryptographic solutions for fog based blockchain. In *2019 International Conference on Smart Structures and Systems (ICSSS)*, pages 1–5. IEEE, 2019.

[83] Razi Iqbal, Talal Ashraf Butt, Muhammad Afzaal, and Khaled Salah. Trust management in social internet of vehicles: Factors, challenges, blockchain, and fog solutions. *International Journal of Distributed Sensor Networks*, 15(1): 1550147719825820, 2019.

[84] Mohammed Amine Bouras, Qinghua Lu, Fan Zhang, Yueliang Wan, Tao Zhang, and Huansheng Ning. Distributed ledger technology for ehealth identity privacy: State of the art and future perspective. *Sensors*, 20(2):483, 2020.

[85] Adam A Alli and Mugigayi Fahadi. Chapter four blockchain and fog computing: Fog-blockchain concept, opportunities, and challenges. *Blockchain in Data Analytics*, page 75, 2020.

[86] Anjani Barhanpure, Paaras Belandor, and Bhaskarjyoti Das. Proof of stack consensus for blockchain networks. In *International Symposium on Security in Computing and Communication*, pages 104–116. Springer, 2018.

[87] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 2019.

[88] Pradip Kumar Sharma, Mu-Yen Chen, and Jong Hyuk Park. A software defined fog node based distributed blockchain cloud architecture for iot. *IEEE Access*, 6:115–124, 2017.

[89] Moon Yong Jung, Won-Suk Kim, Sang-Hwa Chung, and Ju Wook Jang. A blockchain-based id/ip mapping and user-friendly fog computing for hyper-connected iot architecture. *IJICTDC*, 2(2):12–19, 2017.

[90] Zehui Xiong, Shaohan Feng, Wenbo Wang, Dusit Niyato, Ping Wang, and Zhu Han. Cloud/fog computing resource management and pricing for blockchain networks. *IEEE Internet of Things Journal*, 2018.

[91] Randa Almadhoun, Maha Kadadha, Maya Alhemeiri, Maryam Alshehhi, and Khaled Salah. A user authentication scheme of iot devices using blockchain-enabled fog nodes. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2018.

[92] Danco Davcev, Ljupco Kocarev, Anna Carbone, Vlado Stankovski, and Kosta Mitresk. Blockchain-based distributed cloud/fog platform for iot supply chain management. In *Eighth international conference on advances in computing, electronics and electrical technology (CEET)*, pages 51–58, 2018.

[93] Xin Gu, Jun Peng, Wentao Yu, Yijun Cheng, Fu Jiang, Xiaoyong Zhang, Zhiwu Huang, and Lin Cai. Using blockchain to enhance the security of fog-assisted crowdsensing systems. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 1859–1864. IEEE, 2019.

[94] Hendrik L Cech, Marcel Großmann, and Udo R Krieger. A fog computing architecture to share sensor data by means of blockchain functionality. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 31–40. IEEE, 2019.

[95] Michael Herbert Ziegler, Marcel Groβmann, and Udo R Krieger. Integration of fog computing and blockchain technology using the plasma framework. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 120–123. IEEE, 2019.

[96] Shreshth Tuli, Redowan Mahmud, Shikhar Tuli, and Rajkumar Buyya. Fogbus: A blockchain-based lightweight framework for edge and fog computing. *Journal of Systems and Software*, 2019.

[97] Ammar Muthanna, Abdelhamied A Ateya, Abdukodir Khakimov, Irina Gudkova, Abdelrahman Abuarqoub, Konstantin Samouylov, and Andrey Koucheryavy. Secure and reliable iot networks using fog computing with software-defined networking and blockchain. *Journal of Sensor and Actuator Networks*, 8(1):15, 2019.

[98] Said El Kafhali, Chorouk Chahir, Mohamed Hanini, and Khaled Salah. Architecture to manage internet of things data using blockchain and fog computing. In *Proceedings of the 4th International Conference on Big Data and Internet of Things*, pages 1–8, 2019.

[99] Muhammad Yanuar Ary Saputro and Riri Fitri Sari. Securing iot network using lightweight multi-fog (lmf) blockchain model. In *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 183–188. IEEE, 2019.

[100] Haoyu Wang, Lina Wang, Zhichao Zhou, Xueqiang Tao, Giovanni Pau, and Fabio Arena. Blockchain-based resource allocation model in fog computing. *Applied Sciences*, 9(24):5538, 2019.

[101] Bjoern Holste, Vlado Stankovski, Petar Kochovski, Antonio Puliafito, and Philippe Massonet. Blockchain based variability management solutions for fog native open source software. In *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–6. IEEE, 2019.

[102] Marcello Cinque, Christian Esposito, and Stefano Russo. Trust management in fog/edge computing by means of blockchain technologies. 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)*, pages 1433–1439. IEEE, 2018.

[103] Vishal Sharma, Ilsun You, Francesco Palmieri, Dushantha Nalin K Jayakody, and Jun Li. Secure and energy-efficient handover in fog networks using blockchain-based dmm. *IEEE Communications Magazine*, 56(5):22–31, 2018.

[104] Wenda Tang, Xuan Zhao, Wajid Rafique, and Wanchun Dou. A blockchain-based offloading approach in fog computing environment. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 308–315. IEEE, 2018.

[105] Yutao Jiao, Ping Wang, Dusit Niyato, and Kongrath Suankaewmanee. Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks. *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[106] Ivan Podsevalov, Oleg Iakushkin, Ruslan Kurbangaliev, and Vladimir Korkhov. Blockchain as a platform for fog computing. In *International Conference on Computational Science and Its Applications*, pages 596–605. Springer, 2019.

[107] Meng Li, Liehuang Zhu, and Xiaodong Lin. Coride: A privacy-preserving collaborative-ride hailing service using blockchain-assisted vehicular fog computing. In *International Conference on Security and Privacy in Communication Systems*, pages 408–422. Springer, 2019.

[108] Alessio Bonadio, Francesco Chiti, Romano Fantacci, and Vincenzo Vespri. An integrated framework for blockchain inspired fog communications and computing in internet of vehicles. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–8, 2019.

[109] Jianbin Gao, Kwame Opuni-Boachie Obour Agyekum, Emmanuel Boateng Sifah, Kingsley Nketia Acheampong, Qi Xia, Xiaojiang Du, Moshen Guizani, and Hu Xia. A blockchain-sdn enabled internet of vehicles environment for fog computing and 5g networks. *IEEE Internet of Things Journal*, 2019.

[110] Sara Nadeem, Muhammad Rizwan, Fahad Ahmad, and Jaweria Manzoor. Securing cognitive radio vehicular ad hoc network with fog node based distributed blockchain cloud architecture. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 10(1):288–295, 2019.

[111] Wei Ou, Mingwei Deng, and Entao Luo. A decentralized and anonymous data transaction scheme based on blockchain and zero-knowledge proof in vehicle networking (workshop paper). In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 712–726. Springer, 2019.

[112] Yingying Yao, Xiaolin Chang, Jelena Mišić, Vojislav B Mišić, and Lin Li. Bla: blockchain-assisted lightweight anonymous authentication for distributed vehicular fog services. *IEEE Internet of Things Journal*, 6(2):3775–3784, 2019.

[113] Kuljeet Kaur, Sahil Garg, Georges Kaddoum, François Gagnon, and Syed Hassan Ahmed. Blockchain-based lightweight authentication mechanism for vehicular fog infrastructure. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2019.

[114] Naveed Islam, Yasir Faheem, Ikram Ud Din, Muhammad Talha, Mohsen Guizani, and Mudassir Khalil. A blockchain-based fog computing framework for activity recognition as an application to e-healthcare services. *Future Generation Computer Systems*, 100:569–578, 2019.

[115] Tiago M Fernández-Caramés, Iván Froiz-Míguez, Oscar Blanco-Novoa, and Paula Fraga-Lamas. Enabling the internet of mobile crowdsourcing health things: A mobile fog computing, blockchain and iot based continuous glucose monitoring system for diabetes mellitus research and care. *Sensors*, 19(15): 3319, 2019.

[116] Andreas Seitz, Dominic Henze, Daniel Miehle, Bernd Bruegge, Jochen Nickles, and Markus Sauer. Fog computing as enabler for blockchain-based iiot app marketplaces-a case study. In *2018 Fifth international conference on internet of things: systems, management and security*, pages 182–188. IEEE, 2018.

[117] Su-Hwan Jang, Jo Guejong, Jongpil Jeong, and Bae Sangmin. Fog computing architecture based blockchain for industrial iot. In *International Conference on Computational Science*, pages 593–606. Springer, 2019.

[118] Efthimios N Lallas, Apostolos Xenakis, and Georgios Stamoulis. A generic framework for a peer to peer blockchain based fog architecture in industrial automation. In *2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–5. IEEE, 2019.

[119] Petar Kochovski, Sandi Gec, Vlado Stankovski, Marko Bajec, and Pavel D Drobintsev. Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Generation Computer Systems*, 101: 747–759, 2019.

[120] Hui Huang, Kuan-Ching Li, and Xiaofeng Chen. Blockchain-based fair three-party contract signing protocol for fog computing. *Concurrency and Computation: Practice and Experience*, 31(22):e4469, 2019.

[121] Gaurav Deep, Rajni Mohana, Anand Nayyar, P Sanjeevikumar, and Eklas Hossain. Authentication protocol for cloud databases using blockchain mechanism. *Sensors*, 19(20):4444, 2019.

[122] Gulshan Kumar, Rahul Saha, Mritunjay Kumar Rai, Reji Thomas, and Tai-Hoon Kim. Proof-of-work consensus approach in blockchain technology for cloud and fog computing using maximization-factorization statistics. *IEEE Internet of Things Journal*, 6(4):6835–6842, 2019.

[123] Marco Savi, Daniele Santoro, Katarzyna Di Meo, Daniele Pizzolli, Miguel Pincheira, Raffaele Giaffreda, Silvio Cretti, Seung-woo Kum, and Domenico Siracusa. A blockchain-based brokerage platform for fog computing resource federation. In *Conference on Innovation in Clouds, Internet and Networks*, 2020.

[124] Mohammed Alshehri and Brajendra Panda. A blockchain-encryption-based approach to protect fog federations from rogue nodes. *arXiv preprint arXiv:2001.04490*, 2020.

[125] Mazin Debe, Khaled Salah, Muhammad Habib ur Rehman, and Davor Svetinovic. Monetization of services provided by public fog nodes using blockchain and smart contracts. *IEEE Access*, 2020.

[126] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Lsb: A lightweight scalable blockchain for iot security and privacy. *arXiv preprint arXiv:1712.02969*, 2017.

[127] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 282–297. Springer, 2017.

[128] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[129] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99(1999), pages 173–186, 1999.

[130] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285. Springer, 2005.

[131] Deepak Puthal and Saraju P Mohanty. Proof of authentication: Iot-friendly blockchains. *IEEE Potentials*, 38(1):26–29, 2018.

[132] Mohammed Alshehri and Brajendra Panda. An encryption-based approach to protect fog federations from rogue nodes. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pages 225–243. Springer, 2019.

[133] Smart Dubai Department. *BLOCKCHAIN*, 2020 (accessed October, 27, 2020). URL `https://www.smartdubai.ae/initiatives/blockchain`.

[134] Global Times. *China launches blockchain-based smart city identification system*, 2019 (accessed October, 27, 2020). URL `https://www.globaltimes.cn/content/1168878.shtml`.

[135] Smartcity Press. *China Taking A Big Leap With Blockchain*, 2019 (accessed October, 27, 2020). URL `https://www.smartcity.press/blockchain-technology-china/`.

[136] Alberto Montresor and Márk Jelasity. Peersim: A scalable p2p simulator. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 99–100. IEEE, 2009.

[137] Ioan Petri, Masoud Barati, Yacine Rezgui, and Omer F Rana. Blockchain for energy sharing and trading in distributed prosumer communities. *Computers in Industry*, 123:103282, 2020.

[138] J Manning. Proof-of-work vs. proof-of-stake explained. *ETHNews, November. Available at: https://www. ethnews. com/proof-of-work-vs-proofof-stake-explained (Accessed: 6 January 2018)*, 2016.

[139] Pranav Kumar Singh, Roshan Singh, Sunit Kumar Nandi, and Sukumar Nandi. Managing smart home appliances with proof of authority and blockchain. In *International Conference on Innovations for Community Services*, pages 221–232. Springer, 2019.

[140] Jari Kreku, Visa Antero Vallivaara, Kimmo Halunen, Jani Suomalainen, M Ramachandran, VM Muñoz, V Kantere, G Wills, and RJ Walters. Evaluating the efficiency of blockchains in iot with simulations. In *IoTBDS*, pages 216–223, 2017.

[141] Sotirios Liaskos, Tarun Anand, and Nahid Alimohammadi. Architecting blockchain network simulators: a model-driven perspective. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE, 2020.

[142] Tanesh Kumar, Erkki Harjula, Muneeb Ejaz, Ahsan Manzoor, Pawani Porambage, Ijaz Ahmad, Madhusanka Liyanage, An Braeken, and Mika Ylianttila. Blockedge: Blockchain-edge framework for industrial iot networks. *IEEE Access*, 2020.

[143] Andras Markus and Attila Kertesz. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*, 101:102042, 2020.

[144] Zahra Nikdel, Bing Gao, and Stephen W Neville. Dockersim: Full-stack simulation of container-based software-as-a-service (saas) cloud deployments and environments. In *2017 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 1–6. IEEE, 2017.

[145] Tariq Qayyum, Asad Waqar Malik, Muazzam A Khan Khattak, Osman Khalid, and Samee U Khan. Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583, 2018.

[146] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.

[147] Ubaid Ur Rahman, Kashif Bilal, Aiman Erbad, Osman Khalid, and Samee U Khan. Nutshell–simulation toolkit for modeling data center networks and cloud computing. *IEEE Access*, 7:19922–19942, 2019.

[148] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[149] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[150] Mohammed Islam Naas, Jalil Boukhobza, Philippe Raipin Parvedy, and Laurent Lemarchand. An extension to ifogsim to enable the design of data placement strategies. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–8. IEEE, 2018.

[151] Ruben Mayer, Leon Graser, Harshit Gupta, Enrique Saurez, and Umakishore Ramachandran. Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures. In *2017 IEEE Fog World Congress (FWC)*, pages 1–6. IEEE, 2017.

[152] Antonio Coutinho, Fabiola Greve, Cassio Prazeres, and Joao Cardoso. Fogbed: A rapid-prototyping emulation environment for fog computing. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[153] Márcio Moraes Lopes, Wilson A Higashino, Miriam AM Capretz, and Luiz Fernando Bittencourt. Myifogsim: A simulator for virtual machine migration in fog computing. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 47–52, 2017.

[154] Vysakh Anilkumar, Joseph Antony Joji, Asif Afzal, and Reshma Sheik. Blockchain simulation and development platforms: Survey, issues and challenges. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 935–939. IEEE, 2019.

[155] Ethereum. *Remix Platform*, 2020 (accessed October, 27, 2020). URL `https://remix.ethereum.org/`.

[156] Truffle Blockchain Group. *TRUFFLE OVERVIEW*, 2020 (accessed October, 27, 2020). URL `https://www.trufflesuite.com/docs/truffle/overview`.

[157] Arshdeep Bahga and Vijay Madisetti. *Blockchain applications: a hands-on approach*. Vpt, 2017.

[158] Bruno. *Explaining Ethereum Tools: What Are Geth and Mist?*, 2018 (accessed October, 27, 2020). URL `https://bitfalls.com/2018/02/12/explaining-ethereum-tools-geth-mist/`.

[159] Maher Alharby and Aad van Moorsel. Blocksim: a simulation framework for blockchain systems. *ACM SIGMETRICS Performance Evaluation Review*, 46(3): 135–138, 2019.

[160] Carlos Faria and Miguel Correia. Blocksim: Blockchain simulator. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 439–446. IEEE, 2019.

[161] Bozhi Wang, Shiping Chen, Lina Yao, Bin Liu, Xiwei Xu, and Liming Zhu. A simulation approach for studying behavior and quality of blockchain networks. In *International Conference on Blockchain*, pages 18–31. Springer, 2018.

[162] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[163] Raheel Ahmed Memon, Jianping Li, Junaid Ahmed, Asif Khan, M Irshad Nazir, and M Ismail Mangrio. Modeling of blockchain based systems using queuing theory simulation. In *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 107–111. IEEE, 2018.

[164] Fangyuan Zhao, Xin Guo, and Wai Kin Victor Chan. Individual green certificates on blockchain: A simulation approach. *Sustainability*, 12(9):3942, 2020.

[165] Pierre-Yves Piriou and Jean-Francois Dumas. Simulation of stochastic blockchain models. In *2018 14th European Dependable Computing Conference (EDCC)*, pages 150–157. IEEE, 2018.

[166] Aditya Deshpande, Pezhman Nasirifard, and Hans-Arno Jacobsen. evibes: Configurable and interactive ethereum blockchain simulation framework. In *Proceedings of the 19th International Middleware Conference (Posters)*, pages 11–12, 2018.

[167] Bozhi Wang, Qin Wang, Shiping Chen, and Yang Xiang. Security analysis on tangle-based blockchain through simulation. In *Australasian Conference on Information Security and Privacy*, pages 653–663. Springer, 2020.

[168] Ravi Kiran Raman, Roman Vaculin, Michael Hind, Sekou L Remy, Eleftheria K Pissadaki, Nelson Kibichii Bore, Roozbeh Daneshvar, Biplav Srivastava, and Kush R Varshney. A scalable blockchain approach for trusted computation and verifiable simulation in multi-party collaborations. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 277–284. IEEE, 2019.

[169] The Linux Foundation. *What is Hyperledger?*, 2020 (accessed October, 27, 2020). URL `https://www.hyperledger.org/`.

[170] Hamza Baniata and Attila Kertesz. *FoBSim public repository*, 2020 (accessed March, 18, 2022). URL `https://github.com/sed-szeged/FobSim`.

[171] Hamza Baniata and Attila Kertesz. Fobsim: An extensible open-source simulation tool for integrated fog-blockchain systems, Nov 2020. URL `DOI: 10.36227/techrxiv.13148390`.

[172] Hamza Baniata, Wesam Almobaideen, and Attila Kertesz. A privacy preserving model for fog-enabled mcc systems using 5g connection. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 223–230. IEEE, 2020.

[173] Piotr Fröhlich, Erol Gelenbe, and Mateusz P Nowak. Smart sdn management of fog services. In *2020 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2020.

[174] Bastian Blywis, Mesut Günes, Felix Juraschek, Oliver Hahm, and Nicolai Schmittberger. A survey of flooding, gossip routing, and related schemes for wireless multi-hop networks. Technical report, Freie Universitat, Berlin, Germany, 2011. URL `https://refubium.fu-berlin.de/bitstream/handle/fub188/18401/2010-Gossip-Routing.pdf?sequence=1`.

[175] Xiaowei He, Yiju Cui, and Yunchao Jiang. An improved gossip algorithm based on semi-distributed blockchain network. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 24–27. IEEE, 2019.

[176] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003.

[177] Wilfried N Gansterer, Gerhard Niederbrucker, Hana Straková, and Stefan Schulze Grotthoff. Scalable and fault tolerant orthogonalization based on randomized distributed data aggregation. *Journal of Computational Science*, 4 (6):480–488, 2013.

[178] Gábor Danner and Márk Jelasity. Robust decentralized mean estimation with limited communication. In *European Conference on Parallel Processing*, pages 447–461. Springer, 2018.

[179] Caixiang Fan, Sara Ghaemi, Hamzeh Khazaei, and Petr Musilek. Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950, 2020.

[180] Jiang Lan, Xiaotao Liu, Prashant Shenoy, and Krithi Ramamritham. Consistency maintenance in peer-to-peer file sharing networks. In *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*, pages 90–94. IEEE, 2003.

[181] Nabhendra Bisnik and Alhussein A Abouzeid. Optimizing random walk search algorithms in p2p networks. *Computer networks*, 51(6):1499–1514, 2007.

[182] The_Python_Software_Foundation. *The Python Standard Library*, 2020 (Last accessed September 14, 2020). `https://docs.python.org/2/library/multiprocessing.html`.

[183] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917, 2012.

[184] Kevin Alarcón Negy, Peter R Rizun, and Emin Gün Sirer. Selfish mining re-examined. In *International Conference on Financial Cryptography and Data Security*, pages 61–78. Springer, 2020.

[185] Hamza Baniata and Attila Kertesz. Fobsim: an extensible open-source simulation tool for integrated fog-blockchain systems. *PeerJ Computer Science*, 7: e431, 2021.

[186] Robert G Sargent. Verification and validation of simulation models. *Journal of simulation*, 7(1):12–24, 2013.

[187] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. 2018.

[188] Binance Academy. *Proof of Authority Explained*, 2020 (accessed October, 27, 2020). URL `https://academy.binance.com/en/articles/proof-of-authority-explained`.

[189] Jelena Misic, Vojislav B Misic, and Xiaolin Chang. Performance of bitcoin network with synchronizing nodes and a mix of regular and compact blocks. *IEEE Transactions on Network Science and Engineering*, 2020.

[190] Bin Cao, Zhenghui Zhang, Daquan Feng, Shengli Zhang, Lei Zhang, Mugen Peng, and Yun Li. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 2020.

[191] Wei Bi, Huawei Yang, and Maolin Zheng. An accelerated method for message propagation in blockchain networks. *arXiv preprint arXiv:1809.00455*, 2018.

[192] Jianhua Li, Tiehua Zhang, Jiong Jin, Yingying Yang, Dong Yuan, and Longxiang Gao. Latency estimation for fog-based internet of things. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2017.

[193] Muntadher Fadhil Sallal. *Evaluation of Security and Performance of Clustering in the Bitcoin Network, with the Aim of Improving the Consistency of the Blockchain*. PhD thesis, University of Portsmouth, 2018.

[194] Gabriel R Carrara, Leonardo M Burle, Dianne SV Medeiros, Célio Vinicius N de Albuquerque, and Diogo MF Mattos. Consistency, availability, and partition tolerance in blockchain: a survey on the consensus mechanism over peer-to-peer networking. *Annals of Telecommunications*, pages 1–12, 2020.

[195] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[196] Furqan Jameel, Muhammad Nabeel, Muhammad Ali Jamshed, and Riku Jäntti. Minimizing forking in blockchain-based iot networks. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[197] Zakwan Jaroucheh, Baraq Ghaleb, and William J Buchanan. Sklcoin: Toward a scalable proof-of-stake and collective signature based consensus protocol for strong consistency in blockchain. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 143–150. IEEE, 2020.

[198] Simplified payment verification, Nov 2020. URL `https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification`.

[199] Robbert Van Renesse, Kenneth P Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM transactions on computer systems (TOCS)*, 21(2): 164–206, 2003.

[200] Jon Foss Mikalsen. Firechain: An efficient blockchain protocol using secure gossip. Master's thesis, UiT Norges arktiske universitet, 2018.

[201] Calvin Newport and Alex Weaver. Random gossip processes in smartphone peer-to-peer networks. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 139–146. IEEE, 2019.

[202] Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. A theoretical model for fork analysis in the bitcoin network. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 237–244. IEEE, 2019.

[203] Minghong Fang and Jia Liu. Toward low-cost and stable blockchain networks. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[204] Petrică C Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 283(1):1–15, 2020.

[205] Erdos P. and Renyi A. On random graphs,. *Publicationes Mathematicae*, 6: 290–297, 1959.

[206] Albert R. and Barabasi A.L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[207] AKM Bahalul Haque and Bharat Bhushan. Blockchain in a nutshell: State-of-the-art applications and future research directions. In *Blockchain and AI Technology in the Industrial Internet of Things*, pages 124–143. IGI Global, 2021.

[208] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time-and message-optimal distributed algorithm for minimum spanning trees. *ACM Transactions on Algorithms*, 16(1), 2019.

[209] Eric Brewer. Cap twelve years later: How the" rules" have changed. *Computer*, 45(2):23–29, 2012.

[210] Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 729–744, 2018.

[211] Jun Zhao, Jing Tang, Li Zengxiang, Huaxiong Wang, Kwok-Yan Lam, and Kaiping Xue. An analysis of blockchain consistency in asynchronous networks: Deriving a neat bound. *arXiv preprint arXiv:1909.06587*, 2019.

[212] Zhijun Wang, Anwitaman Datta, Sajal K Das, and Mohan Kumar. Cmv: File consistency maintenance through virtual servers in peer-to-peer systems. *Journal of parallel and distributed computing*, 69(4):360–372, 2009.

[213] Narjes Nikzad Khasmakhi, Shahram Jamali, and Meysam Asgari Chenaghlu. A solution for replica consistency maintenance in unstructured peer-to-peer networks. *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS*, 2016.

[214] Gao Song. *Dynamic data consistency maintenance in peer-to-peer caching system*. PhD thesis, SCHOOL OF COMPUTING, NATIONAL UNIVERSITY OF SINGAPORE, 2004.

[215] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

[216] Jia Kan, Lingyi Zou, Bella Liu, and Xin Huang. Boost blockchain broadcast propagation with tree routing. In *International Conference on Smart Blockchain*, pages 77–85. Springer, 2018.

[217] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[218] Huakun Liu, Xin Wei, Ruliang Xiao, Lifei Chen, Xin Du, and Shi Zhang. Oprcp: approximate nearest neighbor binary search algorithm for hybrid data over wmsn blockchain. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):1–14, 2018.

[219] Maleq Khan, Gopal Pandurangan, and VS Anil Kumar. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(1):124–139, 2008.

[220] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[221] Kaushik Ayinala, Baek-Young Choi, and Sejun Song. Pichu: Accelerating block broadcasting in blockchain networks with pipelining and chunking. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 221–228. IEEE, 2020.

[222] Robbert van Renesse. A blockchain based on gossip?-a position paper. *Cornell University*, 2016.

[223] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[224] Ke Wang and Hyong S Kim. Fastchain: Scaling blockchain system with informed neighbor selection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 376–383. IEEE, 2019.

[225] Yusuke Aoki and Kazuyuki Shudo. Proximity neighbor selection in blockchain networks. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 52–58. IEEE, 2019.

[226] Ming Jin, Xiaojiao Chen, and Sian-Jheng Lin. Reducing the bandwidth of block propagation in bitcoin network with erasure coding. *IEEE Access*, 7: 175606–175613, 2019.

[227] Bin Yu, Xiaofeng Li, He Zhao, and Tong Zhou. A scalable blockchain network model with transmission paths and neighbor node subareas. *Computing*, pages 1–25, 2021.

[228] Jiao Li. Data transmission scheme considering node failure for blockchain. *Wireless Personal Communications*, 103(1):179–194, 2018.

[229] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[230] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7 (1):48–50, 1956.

[231] Gopal Pandurangan, Peter Robinson, Michele Scquizzato, et al. The distributed minimum spanning tree problem. *Bulletin of EATCS*, 2(125), 2018.

[232] Otakar Boruvka. O jistém problému minimálním. 1926.

[233] Farhad Soleimanian Gharehchopogh and Hassan Arjang. A survey and taxonomy of leader election algorithms in distributed systems. *Indian Journal of Science and Technology*, 7(6):815, 2014.

[234] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. Dons: Dynamic optimized neighbor selection for smart blockchain networks. *Future Generation Computer Systems*, 130:75–90, 2022.

[235] Sachin Babar, Parikshit Mahalle, Antonietta Stango, Neeli Prasad, and Ramjee Prasad. Proposed security model and threat taxonomy for the internet of things (iot). In *International Conference on Network Security and Applications*, pages 420–429. Springer, 2010.

[236] Victoria L Lemieux. Blockchain and distributed ledgers as trusted recordkeeping systems. In *Future Technologies Conference (FTC)*, volume 2017, 2017.

[237] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[238] Ting Chen, Yuxiao Zhu, Zihao Li, Jiachi Chen, Xiaoqi Li, Xiapu Luo, Xiaodong Lin, and Xiaosong Zhange. Understanding ethereum via graph analysis. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1484–1492. IEEE, 2018.

[239] Mitchell Moos. Bitcoin transactions per block at all-time highs. `https://cryptoslate.com/bitcoin-transactions-per-block-at-all-time-highs/`, 2019. [Online; accessed 03-November-2019].

[240] Murch. What is the probability of forking in blockchain? `https://bitcoin.stackexchange.com/questions/42649/what-is-the-probability-of-forking-in-blockchain`, 2019. [Online; accessed 03-November-2019].

[241] BlockChain.com. Blockchain charts. `https://www.blockchain.com/en/charts`, 2019. [Online; accessed 03-November-2019].

[242] Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Blockchain for internet of things: A survey. *arXiv preprint arXiv:1906.00245*, 2019.

[243] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[244] Carlos Faria and Miguel Correia. Blocksim: blockchain simulator. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 439–446. IEEE, 2019.

[245] John Shalf. The future of computing beyond moore's law. *Philosophical Transactions of the Royal Society A*, 378(2166):20190061, 2020.

[246] Zhijie Ma, Qinglin Zhao, Jianwen Yuan, Xiaobo Zhou, and Li Feng. Fork probability analysis of pouw consensus mechanism. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 333–337. IEEE, 2020.

[247] Qin Hu, Minghui Xu, Shengling Wang, and Shaoyong Guo. Sync or fork: Node-level synchronization analysis of blockchain. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 170–181. Springer, 2020.

[248] Daniel A Marcus. *Graph theory*, volume 53. American Mathematical Soc., 2020.

[249] CISSP Thomas Porter, CCDA CCNP, Michael Gough, et al. *How to cheat at VoIP security*. Syngress, 2011.

[250] Leighton Johnson. *Security controls evaluation, testing, and assessment handbook*. Academic Press, 2019.

[251] Anichur Rahman, Md Jahidul Islam, Antonio Montieri, Mostofa Kamal Nasir, Md Mahfuz Reza, Shahab S Band, Antonio Pescape, Mahedi Hasan, Mehdi Sookhak, and Amir Mosavi. Smartblock-sdn: An optimized blockchain-sdn framework for resource management in iot. *IEEE Access*, 9:28361–28376, 2021.

[252] Bernard ME Moret and Henry D Shapiro. An empirical analysis of algorithms for constructing a minimum spanning tree. In *Workshop on Algorithms and Data Structures*, pages 400–411. Springer, 1991.

[253] Michael L Fredman, Robert Sedgewick, Daniel D Sleator, and Robert E Tarjan. The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1(1-4): 111–129, 1986.

[254] Emmanuelle Anceaume, Antonella Pozzo, Thibault Rieutord, and Sara Tucci-Piergiovanni. On finality in blockchains. *arXiv preprint arXiv:2012.10172*, 2020.

[255] Dev Arora, Siddharth Gautham, Harshit Gupta, and Bharat Bhushan. Blockchain-based security solutions to preserve data privacy and integrity. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 468–472. IEEE, 2019.

[256] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 358–367. IEEE, 2016.

[257] Behavioral mining: Blockchain's new rocket fuel (part 1), Sep 2018. URL `https://medium.com/the-notice-board/behavioral-mining-blockchains-new-rocket-fuel-part-1-ca65a7d82d22`.

[258] Zonghao Feng and Qiong Luo. Evaluating memory-hard proof-of-work algorithms on three processors. *Proceedings of the VLDB Endowment*, 13(6):898–911, 2020.

[259] Gene Hoffman. The chia business whitepaper. Technical report, Chia Network, February 2021. URL `chia.net/assets/Chia-Business-Whitepaper-2021-02-09-v1.0.pdf`.

[260] Huiying Xu, Xiaoyu Qiu, Weikun Zhang, Kang Liu, Shuo Liu, and Wuhui Chen. Privacy-preserving incentive mechanism for multi-leader multi-follower iot-edge computing market: A reinforcement learning approach. *Journal of Systems Architecture*, 114:101932, 2021.

[261] Xin Wang, Jianping He, Peng Cheng, and Jiming Chen. Privacy preserving collaborative computing: Heterogeneous privacy guarantee and efficient incentive mechanism. *IEEE Transactions on Signal Processing*, 67(1):221–233, 2018.

[262] G Suriya Praba Devi and JC Miraclin Joyce Pamila. Accident alert system application using a privacy-preserving blockchain-based incentive mechanism. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 390–394. IEEE, 2019.

[263] Jiejun Hu, Kun Yang, Kezhi Wang, and Kai Zhang. A blockchain-based reward mechanism for mobile crowdsensing. *IEEE Transactions on Computational Social Systems*, 7(1):178–191, 2020.

[264] Osamah Ibrahim Khalaf and Ghaida Muttashar Abdulsahib. Optimized dynamic storage of data (odsd) in iot based on blockchain for wireless sensor networks. *Peer-to-Peer Networking and Applications*, pages 1–16, 2021.

[265] Sanjaya K Panda, Indrajeet Gupta, and Prasanta K Jana. Task scheduling algorithms for multi-cloud systems: allocation-aware approach. *Information Systems Frontiers*, 21(2):241–259, 2019.

[266] Jeffrey D. Ullman. Np-complete scheduling problems. *Journal of Computer and System sciences*, 10(3):384–393, 1975.

[267] Mala Kalra and Sarbjeet Singh. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian informatics journal*, 16(3):275–295, 2015.

[268] Vincent T'kindt and Jean-Charles Billaut. *Multicriteria scheduling: theory, models and algorithms*. Springer Science & Business Media, 2006.

[269] Florian T Hecker, Marc Stanke, Thomas Becker, and Bernd Hitzmann. Application of a modified ga, aco and a random search procedure to solve the production scheduling of a case study bakery. *Expert Systems with Applications*, 41(13):5882–5891, 2014.

[270] Shahid H Bokhari. *Assignment problems in parallel and distributed computing*, volume 32. Springer Science & Business Media, 2012.

[271] David W Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

[272] Gábor Bacsó, Tamás Kis, Ádám Visegrádi, Attila Kertész, and Zsolt Németh. A set of successive job allocation models in distributed computing infrastructures. *Journal of Grid Computing*, 14(2):347–358, 2016.

[273] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[274] Hebatallah Khattab Ala'a Al-Shaikh, Ahmad Sharieh, and Azzam Sleit. Resource utilization in cloud computing as an optimization problem. *Resource*, 7 (6), 2016.

[275] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. An investigation of some properties of an" ant algorithm". In *Ppsn*, volume 92, 1992.

[276] Milan Tuba, Raka Jovanovic, and SERBIA SERBIA. An analysis of different variations of ant colony optimization to the minimum weight vertex cover problem. *WSEAS Transactions on Information Science and Applications*, 6(6): 936–945, 2009.

[277] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[278] Zhou Zhou, Fangmin Li, Huaxi Zhu, Houliang Xie, Jemal H Abawajy, and Morshed U Chowdhury. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications*, pages 1–11, 2019.

[279] G Umarani Srikanth, V Uma Maheswari, P Shanthi, and Arul Siromoney. Tasks scheduling using ant colony optimization. 2012.

[280] Daniel Merkle and Martin Middendorf. Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, 18(1):105–111, 2003.

[281] Seyedeh Leili Mirtaheri and Hamid Reza Shirzad. Optimized distributed resource management in fog computing by using ant-olony optimization c. *Future Trends of HPC in a Disruptive Scenario*, 34:206, 2019.

[282] Gang Li and Zhijun Wu. Ant colony optimization task scheduling algorithm for swim based on load balancing. *Future Internet*, 11(4):90, 2019.

[283] Mohamed K Hussein and Mohamed H Mousa. Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access*, 8:37191–37201, 2020.

[284] Satoshi Nakamoto. Bitcoin whitepaper, 2008.

[285] Kongrath Suankaewmanee, Dinh Thai Hoang, Dusit Niyato, Suttinee Sawadsitang, Ping Wang, and Zhu Han. Performance analysis and application of mobile blockchain. In *2018 international conference on computing, networking and communications (ICNC)*, pages 642–646. IEEE, 2018.

[286] Louise Axon. Privacy-awareness in blockchain-based pki. *Cdt technical paper series*, 2015.

[287] LM Axon and Michael Goldsmith. Pb-pki: A privacy-aware blockchain-based pki. In *14th International Conference on Security and Cryptography (SECRYPT 2017)*, volume 6. SCITEPRESS, 2016.

[288] Steem. Steem an incentivized, blockchain-based, public content platform. `https://steem.io/SteemWhitePaper.pdf`, 2017. [Online; accessed 21-November-2019].

[289] Cícero Alves da Silva, Gibeon Soares de Aquino Júnior, and Sávio Rennan Menêzes Melo. A blockchain-based approach for privacy control of patient's medical records in the fog layer. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, WebMedia '19, pages 133–136, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6763-9. doi: 10.1145/3323503.3360640. URL `http://doi.acm.org/10.1145/3323503.3360640`.

[290] Dietrich Stauffer, Friedrich W Hehl, Nobuyasu Ito, Volker Winkelmann, and John G Zabolitzky. *Computer simulation and computer algebra: lectures for beginners*. Springer Science & Business Media, 2012.

[291] Hua Chen and Albert MK Cheng. Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors. *ACM SIGBED Review*, 2(2):11–14, 2005.

[292] Umarani Srikanth, Uma Maheswari, Shanthi Palaniswami, and Arul Siromoney. Task scheduling using probabilistic ant colony heuristics. *International Arab Journal of Information Technology (IAJIT)*, 13(4), 2016.

[293] Medhat A Tawfeek, Ashraf El-Sisi, Arabi E Keshk, and Fawzy A Torkey. Cloud task scheduling based on ant colony optimization. In *2013 8th international conference on computer engineering & systems (ICCES)*, pages 64–69. IEEE, 2013.

[294] Kadarla Kavitha and SC Sharma. Performance analysis of aco-based improved virtual machine allocation in cloud for iot-enabled healthcare. *Concurrency and Computation: Practice and Experience*, page e5613, 2019.

[295] Marc Reimann and Jose Eugenio Leal. Single line train scheduling with aco. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 226–237. Springer, 2013.

[296] Ulysse Rugwiro, Chunhua Gu, and Weichao Ding. Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning. *Journal of Internet Technology*, 20(5):1463–1475, 2019.

[297] Sunil Kumar and Subhash Chander. Comparative analysis of task scheduling algorithms in cloud environment in term of their future prospective and risk. *Webology (ISSN: 1735-188X)*, 18(5), 2021.

[298] Fariha Nosheen, Sadia Bibi, and Salabat Khan. Ant colony optimization based scheduling algorithm. In *2013 International Conference on Open Source Systems and Technologies*, pages 18–22. IEEE, 2013.

[299] Ciprian Pungila and Viorel Negru. Improving blockchain security validation and transaction processing through heterogeneous computing. In *International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019)*, pages 132–140. Springer, 2019.

[300] Sergej Svorobej, Patricia Takako Endo, Malika Bendechache, Christos Filelis-Papadopoulos, Konstantinos M Giannoutakis, George A Gravvanis, Dimitrios Tzovaras, James Byrne, and Theo Lynn. Simulating fog and edge computing scenarios: An overview and research challenges. *Future Internet*, 11(3):55, 2019.

[301] Quanqing Xu, Khin Mi Mi Aung, Yongqing Zhu, and Khai Leong Yong. A blockchain-based storage system for data analytics in the internet of things. In *New Advances in the Internet of Things*, pages 119–138. Springer, 2018.

[302] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

[303] digiconomist. Bitcoin energy consumption index. `https://digiconomist.net/bitcoin-energy-consumption`, 2019. [Online; accessed 14-November-2019].

[304] Qiang Guo. Task scheduling based on ant colony optimization in cloud environment. In *AIP Conference Proceedings*, volume 1834(1), page 040039. AIP Publishing LLC, 2017.

[305] Abdolreza Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information sciences*, 222:175–184, 2013.

[306] Georges Akhras. Smart materials and smart systems for the future. *Canadian Military Journal*, 1(3):25–31, 2000.

[307] Seonghyeon Gong, Erzhena Tcydenova, Jeonghoon Jo, Younghun Lee, and Jong Hyuk Park. Blockchain-based secure device management framework for an internet of things network in a smart city. *Sustainability*, 11(14):3889, 2019.

[308] Borhene Chakroun and James Keevy. Digital credentialing: implications for the recognition of learning across borders. *United Nations Educational, Scientific and Cultural Organization*, 7, 2018.

[309] European Commission. 2018 reform of eu data protection rules. `ec.europa.eu/commission/sites/beta-political/\files/data-protection-factsheet-changes_en.pdf`, 2020.

[310] Juan A Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Consensus from signatures of work. In *Cryptographers' Track at the RSA Conference*, pages 319–344. Springer, 2020.

[311] Huma Pervez, Muhammad Muneeb, Muhammad Usama Irfan, and Irfan Ul Haq. A comparative analysis of dag-based blockchain architectures. In *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, pages 27–34. IEEE, 2018.

[312] Lauzon Fauteux, Pierre Alexandre, Leclerc Maxime, and Morin Guillaume. Blockchain–tackling the academic credential verification problem. Technical report, Tampere University, Faculty of Management and Business, 2020.

[313] Blockcerts home page. `https://www.blockcerts.org/`, 2021. Accessed: 2021-12-03.

[314] Opencerts home page. `https://opencerts.io/`, 2021. Accessed: 2021-12-04.

[315] Kosta Batzavalis, Raghu Bala, Alex Norta, and OÜ Norta Partners. A platform for leveraging blockchain technology for the storage, issuance and authentication of academic credentials. `https://www.trusteducation.io/`, 2021. Accessed: 2021-12-16.

[316] Kiratijuta Bhumichitr and Songsak Channarukul. Acachain: Academic credential attestation system using blockchain. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–8, 2020.

[317] Zecheng Li, Haotian Wu, Lap Hou Lao, Songtao Guo, Yuanyuan Yang, and Bin Xiao. Pistis: Issuing trusted and authorized certificates with distributed ledger and tee. *IEEE Transactions on Parallel and Distributed Systems*, 33(7): 1636–1649, 2021.

[318] Shukla Anant, Indra Sneha, Trivedi Tanisha J., and Singh Ujjwala. Academic credential verification technique using blockchain. *International Journal of Advanced Science and Technology*, 29:4244–4254, 2020.

[319] Islam MJ Ahammad MS Tomal MH, Rahman A. Distb-cvs: A distributed secure blockchain based online certificate verification system from bangladesh perspective. In *2nd International Conference on Advanced Information and Communication Technology 2020 (ICAICT 2020)*, pages 1–6. IEEE, 2020.

[320] Kalpana Singh, Omar Dib, Clément Huyart, and Khalifa Toumi. A novel credential protocol for protecting personal attributes in blockchain. *Computers & Electrical Engineering*, 83:106586, 2020.

[321] Linux Foundation. Hyperledger indy. `hyperledger.org/use/\hyperledger-indy`, 2020.

[322] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. Rbft: Redundant byzantine fault tolerance. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 297–306. IEEE, 2013.

[323] Phillip J Windley. How sovrin works. *Windely. com*, 2016.

[324] Mirek Sopek, Przemysaw Gradzki, Dominik Kuzinski, Rafa Trojczak, and Robert Trypuz. Legal entity identifier blockchained by a hyperledger indy implementation of graphchain. In *Research Conference on Metadata and Semantics Research*, pages 26–36. Springer, 2018.

[325] Aamna Tariq, Hina Binte Haq, and Syed Taha Ali. Cerberus: A blockchain-based accreditation and degree verification system. *arXiv preprint arXiv:1912.06812*, 2019.

[326] JIM De Souza Jr, DS De Araújo, GV Barbosa, and P Letouze. An international accreditation system for healthcare professionals based on blockchain. *Int. J. Info. Educ. Tech*, 9(7):462–469, 2019.

[327] Qin Wang, Jiangshan Yu, Shiping Chen, and Yang Xiang. Sok: Diving into dag-based blockchain systems. *arXiv preprint arXiv:2012.06128*, 2020.

[328] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: a fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.*, 2016(1159), 2016.

[329] Yonatan Sompolinsky and Aviv Zohar. Phantom. *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.

[330] Iddo Bentov, Pavel Hubácek, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptol. ePrint Arch.*, 2017:300, 2017.

[331] Hamza Baniata and Attila Kertesz. Prifob: a privacy-aware fog-enhanced blockchain-based system for global accreditation and credential verification. *Journal of Network and Computer Applications*, 205, 2022.

[332] Hirotaka Yoshida and Alex Biryukov. Analysis of a sha-256 variant. In *International Workshop on Selected Areas in Cryptography*, pages 245–260. Springer, 2005.

[333] Christophe De Cannière. Analysis and design of symmetric encryption algorithms. *Doctoral Dissertaion, KULeuven*, 2007.

[334] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 309–318. Springer, 2001.

[335] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[336] Ruchika Singh Malyan and Ashok Kumar Madan. Blockchain technology as a tool to manage digital identity: A conceptual study. In *Advances in Manufacturing and Industrial Engineering*, pages 635–647. Springer, 2021.

[337] José Bacelar Almeida, Manuel Barbosa, Endre Bangerter, Gilles Barthe, Stephan Krenn, and Santiago Zanella Béguelin. Full proof cryptography: verifiable compilation of efficient zero-knowledge protocols. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 488–500, 2012.

[338] Maksym Petkus. Why and how zk-snark works. *arXiv preprint arXiv:1906.07221*, 2019.

[339] Elaine Shi. Analysis of deterministic longest-chain protocols. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 122–12213. IEEE, 2019.

[340] Leo Eichhorn, Tanya Shreedhar, Aleksandr Zavodovski, and Nitinder Mohan. Distributed ledgers for distributed edge: Are we there yet? In *Proceedings of the Interdisciplinary Workshop on (de) Centralization in the Internet*, pages 26–33, 2021.

[341] Proton Technologies AG. Gdpr checklist for data controllers. `https://gdpr.eu/checklist/`, 2021. Accessed: 2021-12-12.

[342] Hamza Baniata, Tamas Pflanzner, Zoltan Feher, and Attila Kertesz. Latency assessment of blockchain-based ssi applications utilizing hyperledger indy. In *Proceedings of the 12th International Conference on Cloud Computing and Services Science - CLOSER*. SciTePress, 2022.

[343] Dávid Urbančok. Blockchain open-source software comparison. Master's thesis, Masaryk University, Faculty of Informatics, Brno, The Czech Republic, 2019.

[344] Xiaoqiong Xu, Gang Sun, Long Luo, Huilong Cao, Hongfang Yu, and Athanasios V Vasilakos. Latency performance modeling and analysis for hyperledger fabric blockchain network. *Information Processing & Management*, 58(1): 102436, 2021.

[345] Lin Zhang, Brian Lee, Yuhang Ye, and Yuansong Qiao. Ethereum transaction performance evaluation using test-nets. In *European Conference on Parallel Processing*, pages 179–190. Springer, 2019.

[346] Felix Härer and Hans-Georg Fill. Decentralized attestation of conceptual models using the ethereum blockchain. In *2019 IEEE 21st Conference on Business Informatics (CBI)*, volume 1, pages 104–113. IEEE, 2019.

[347] Michail Bampatsikos, Christoforos Ntantogian, Christos Xenakis, and Stelios CA Thomopoulos. Barrett blockchain regulated remote attestation. In *IEEE/WIC/ACM International Conference on Web Intelligence-Companion Volume*, pages 256–262, 2019.

[348] Quan-Lin Li, Jing-Yu Ma, and Yan-Xia Chang. Blockchain queue theory. In *International Conference on Computational Social Networks*, pages 25–40. Springer, 2018.

[349] Sergey Smetanin, Aleksandr Ometov, Niclas Kannengießer, Benjamin Sturm, Mikhail Komarov, and Ali Sunyaev. Modeling of distributed ledgers: Challenges and future perspectives. In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, volume 1, pages 162–171. IEEE, 2020.

[350] Miguel Angel Rizzo Gonzalez. Discrete event simulation of a queue using python. `github.com/miguelrizzog96/Queue_Simulation_Python`, 2020.

[351] Thomas A Severini. *Likelihood methods in statistics*. Oxford University Press, 2000.

[352] Saurabh Singh, ASM Sanwar Hosen, and Byungun Yoon. Blockchain security attacks, challenges, and solutions for the future distributed iot network. *IEEE Access*, 9:13938–13959, 2021.

[353] Kaifeng Yue, Yuanyuan Zhang, Yanru Chen, Yang Li, Lian Zhao, Chunming Rong, and Liangyin Chen. A survey of decentralizing applications via blockchain: The 5g and beyond perspective. *IEEE Communications Surveys & Tutorials*, 2021.

[354] Can Zhang, Liehuang Zhu, Chang Xu, Chuan Zhang, Kashif Sharif, Huishu Wu, and Hannes Westermann. Bsfp: blockchain-enabled smart parking with fairness, reliability and privacy protection. *IEEE Transactions on Vehicular Technology*, 69(6):6578–6591, 2020.

[355] Bin Cao, Xuesong Wang, Weizheng Zhang, Houbing Song, and Zhihan Lv. A many-objective optimization model of industrial internet of things based on private blockchain. *IEEE Network*, 34(5):78–83, 2020.

Summary

This PhD dissertation concludes a three-year long research journey on the integration of Fog Computing and Blockchain technologies. The main aim of such integration is to address the challenges of each of these technologies, by integrating it with the other. Blockchain technology (BC) is a distributed ledger technology in the form of a distributed transactional database, secured by cryptography, and governed by a consensus mechanism. It was initially proposed for decentralized cryptocurrency applications with practically proven high robustness. Fog Computing (FC) is a geographically distributed computing architecture, in which various heterogeneous devices at the edge of network are ubiquitously connected to collaboratively provide elastic computation services. FC provides enhanced services closer to end-users in terms of time, energy, and network load. The integration of FC with BC can result in more efficient services, in terms of latency and privacy, mostly required by Internet of Things systems.

Five main integration challenges were targeted when the research started. The dissertation consists of four theses distributed among five chapters (and an Introduction chapter in addition). I made the following contributions as per thesis and per chapter.

Contributions of the theses

In the **first thesis group**, the contributions are related to the study of state-of-the-art FC-BC integration solutions and simulation tools. Accordingly, a fine-tuned FC-BC simulation tool was implemented mimicking realized FC-BC integrated applications. Detailed discussion can be found in Chapters 2 and 3.

I/1. I performed a comprehensive literature review related to integrated Fog Computing and Blockchain solutions, tools and applications.

I/2. I categorized the studied papers according to the year of publication, domain, used algorithms, BC roles, and the placement of BCs in FC-enhanced solutions.

I/3. I provided concluding observations, characteristics and challenges of BC-FC integrated solutions.

I/4. I developed FoBSim, a novel simulation tool that allows for reliable and realistic FC-BC integration simulation. It facilitates the simulation of different consensus algorithms and different applications, and allows to deploy the BC at different layers of an FC-enabled cloud system, with the advantage of easy parameterization of simulation scenarios.

I/5. Using FoBSim, I experimentally proved how integrating FC and BC can provide enhancement in terms of latency and cost. Additionally, I analyzed different factors affecting distributed ledger consistency and trust, which motivated the development of novel methods for quantifying the consistency and reliability of BCs. Using these methods, I could introduce a decision-making model resulting in better integration potential of FC and BC technologies.

In the **second thesis group**, the contributions are related to the efficiency of Blockchain-based solutions in general terms. Concepts for quantifying the trust and consistency of blockchains with probabilistic finality were proposed. Additionally, protocols for enhancing blockchain-based systems were designed and evaluated. Detailed discussion can be found in Chapter 4.

II/1. I formalized and quantified the concepts of Blockchain network consistency and reliability.

II/2. I developed two novel protocols (DONS and AnoLE) for the neighbor selection problem, targeting optimal block finality and privacy-preserving data propagation in public and permissionless Blockchains.

II/3. I experimentally proved that Blockchain and Fog Computing integration is advantageous.

In the **third thesis group**, the contributions are related to addressing the challenges of utilizing FC and BC technologies, by novel FC-BC integrated solutions. It was experimentally discussed how FC can be exploited to enhance BCs and vice versa. Detailed discussion can be found in Chapter 5.

III/1. I designed and developed PF-BTS, which allows the fog layer to exploit BC in a privacy-aware manner to provide optimal task schedules for cloud infrastructures.

III/2. I proposed a Privacy-aware Fog-enabled Blockchain Validation Mechanism (PF-BVM) which exploits the Fog Computing technology to enhance block finality of Blockchain-based systems.

In the **fourth thesis group**, the contributions are related to applying the integration approaches, studied and proposed thus far, into realized FC-BC applications, including Smart Systems and Global Accreditation and Credential solutions. Detailed discussion can be found in Chapter 6.

IV/1. I designed and tested a novel hybrid PoA-SoW consensus algorithm that outperformed PoW, PoA, pBFT, and RAFT consensus algorithms in terms of latency and throughput.

IV/2. I designed and tested a novel DAG-based three-Dimensional Distributed Ledger model (3DDL) which was proven secure and efficient compared to state-of-the-art DL models.

IV/3. I deployed the proposed 3DDL model and PoA-SoW consensus mechanism in a novel Privacy-aware Fog-enhanced Blockchain-based application for global accreditation and credential verification, which was proven more privacy-aware and efficient compared to major state-of-the-art solutions.

Summary in Hungarian

Ez a doktori disszertáció a Köd Számítási és a Blokklánc technológiák integrációjához kapcsolódó, hároméves kutatói utat zárja le. Kutatásom fő célja az egyes technológiák kihívásainak megoldása, a másikkal való integráció révén. A blokklánc (BL) technológia egy elosztott főkönyvi technológia, mely egy elosztott tranzakciós adatbázist valósít meg, kriptográfia és konszenzusos mechanizmusok alkalmazásával, melyet eredetileg decentralizált kriptovaluta alkalmazásokhoz hoztak létre. A Köd Számítások olyan földrajzilag elosztott számítási architektúrát definiálnak, melyben a hálózat szélén található különféle eszközök összekapcsolásával rugalmas számítási szolgáltatásokat nyújtanak. A Köd ezáltal olyan szolgáltatásokat nyújt a végfelhasználókhoz közel, melyek csökkentik a végrehajtási időt, az energiát és a hálózati terhelést. A Köd és a BL integrációja hatékonyabb szolgáltatásokat eredményezhet a késleltetés és az adatvédelem tekintetében, melyet leginkább a Dolgok Internete alkalmazásai igényelnek.

A kutatás megkezdésekor öt fő integrációs kihívást határoztam meg. A disszertáció négy tézisből áll, amelyek öt fejezetben vannak elosztva (és ezen felül tartalmaz egy Bevezetés fejezetet). Az alábbi eredményeket értem el és mutattam be tézisenként és fejezetenként.

A tézisek kontribúciói

Az **első téziscsoport** eredményei a legkorszerűbb Köd-BL integrációs megoldások és szimulációs eszközök tanulmányozásához kapcsolódnak. Ennek megfelelően egy finomhangolt Köd-BL szimulációs eszközt valósítottam meg, mellyel integrált Köd-BL alkalmazásokat modellezhetünk. Részletes bemutatás a 2. és 3. fejezetekben található.

I/1. Részletes szakirodalmi áttekintést végeztem a Köd Számítási és a Blokklánc technológiákhoz kapcsolódó szimulációs eszközökről és integrációs megközelítésekről.

I/2. A vizsgált publikációkat a megjelenés éve, a kutatás területe, a használt algoritmusok, a BL szerepek, valamint a BL-ok ködökkel bővített megoldásaiban való

elhelyezése szerint csoportosítottam.

I/3. Következtetéseket vontam le a Köd-BL integrációs lehetőségek és megoldások tulajdonságaival és kihívásaival kapcsolatban.

I/4. Kifejlesztettem egy új szimulációs eszközt a FoBSim-et, mely részletesen paraméterezhető Köd-BL integrációs szimulációkat tesz lehetővé. Megkönnyíti a különböző konszenzusos algoritmusok és alkalmazások vizsgálatát, és lehetővé teszi a BL telepítésének modellezését Köd- és Felhő-alapú rendszerek különböző rétegeiben.

I/5. A FoBSim szimulátor Segítségével igazoltam, hogy a Köd és a BL integráció miként javíthat a különféle alkalmazások végrehajtásán késleltetés és hatékonyság tekintetében. Elemeztem az elosztott főkönyvi konzisztenciát és a bizalmat befolyásoló különböző tényezőket, melyekhez új módszereket dolgoztam ki a BL-ok konzisztenciájának és megbízhatóságának számszerűsítésére. Így olyan döntéshozatali modellt tudtam kifejleszteni, mely a Köd és a BL technológiák hatékonyabb integrációját eredményezi.

A **második téziscsoportban** a hozzájárulások a blokklánc-alapú megoldások hatékonysági vizsgálataihoz kapcsolódnak, melyekben egy új megközelítést javasoltam a blokkláncok megbízhatóságának és konzisztenciájának számszerűsítésére. Ezentúl új protokollokat dolgoztam és értékeltem ki a blokklánc-alapú rendszerek fejlesztéséhez. Az eredmények részletes bemutatása a 4. fejezetben található.

II/1. Formalizáltam és számszerűsítettem a BL hálózatok konzisztenciájának és megbízhatóságának fogalmait.

II/2. Kidolgoztam két új vezérlési protokollt (DONS és AnoLE), melyek célja a Köd-alapú BL rendszerek hatékonyságának növelése optimalizált szomszéd választással. Ezen protokollok alkalmazásával optimális blokk lezárást és magánszféra-védelmet támogató adatkezelést valósíthatunk meg nyilvános, engedély nélküli blokkláncokban.

II/3. Kísérleteimmel igazoltam, hogy a Köd és a BL technológiák integrációja előnyös.

A **harmadik téziscsoportban** a kontribúciók a Köd és BL alkalmazások kihívásainak megválaszolásához kapcsolódnak, új Köd-BL integrált megoldásokkal. Kísérleteimmel igazoltam, hogyan lehet Köd rendszerek alkalmazásával hatékonyabb BL működést elérni. A részletes ismertetés az 5. fejezetben olvasható.

III/1. Megterveztem és kifejlesztettem a PF-BTS módszert, mely optimális feladatütemezést valósít meg felhő infrastruktúrák számára köd csomópontok alkalmazásával, a magánszféra védelem elősegítésére.

III/2. Javasoltam egy Magánszféra-védelmet elősegítő, Köd-alapú blokklánc-ellenőrzési mechanizmust (PF-BVM), mely Köd csomópontok alkalmazását használja fel a blokklánc-alapú rendszerek blokk lezárásának javítására.

A **negyedik téziscsoport** hozzájárulásai az eddig tanulmányozott és javasolt integrációs megközelítések felhasználását célozzák valós Köd-BL alkalmazásokban, mint pld. okos rendszerek és akkreditációs és hitelesítési szolgáltatások. A részletes bemutatás a 6. fejezetben található.

IV/1. Megterveztem és kiértékeltem egy új, hibrid PoA-SoW konszenzus algoritmust, amely átviteli sebesség és késleltetés tekintetében hatékonyabbnak bizonyult a PoW, PoA, pBFT és RAFT konszenzus algoritmusoknál.

IV/2. Megterveztem és kiértékeltem egy új, DAG-alapú háromdimenziós elosztott főkönyvi modellt (3DDL), mely biztonságosabbnak és hatékonyabbnak bizonyult a legmodernebb DL modellekhez képest.

IV/3. A 3DDL-modell és a PoA-SoW kozenzus algoritmus felhasználásával kidolgoztam egy új, Magánszféra-védelmet elősegítő, Köd- és BL-alapú globális akkreditációs és hitelesítő rendszert, a PriFoB-ot, mely biztonságosabb és hatékonyabb tanúsítványkezelést tesz lehetővé a szakirodalomban fellelhető megoldásokkal szemben.