ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

Ingegneria Elettronica, Telecomunicazioni e Tecnologie dell'Informazione

Ciclo: **XXXII**

Settore Concorsuale: **09 / F2**

Settore Scientifico Disciplinare: **ING-INF / 03**

ARCHITECTING A BLOCKCHAIN-BASED FRAMEWORK FOR THE INTERNET OF THINGS

Presentata da:
Muhammad Salek Ali

Supervisore:
Fabio Antonelli

Co-Supervisore:
Prof. Massimo Vecchio

Coordinatore Dottorato:
Prof.ssa Alessandra Costanzo

Esame finale anno 2020

"... O my Lord! Increase me in knowledge."

(Quran 20:114)

Dedicated to my father.

Abstract

Architecting a Blockchain-Based Framework for the Internet of Things

by Muhammad Salek Ali

While the Internet of Things (IoT) continues to connect an exponentially increasing number of devices to the Internet, handling massive amounts of sensitive data economically and securely remains a challenge. Current IoT solutions are mostly based on centralized infrastructures, which necessitate high-end servers for handling and transferring data. Centralized solutions incur high costs associated to maintaining centralized servers, and do not provide built-in guarantees against security threats and trust issues. Therefore, it is an essential research problem to mitigate the aforementioned problems by developing new methods for IoT decentralisation.

In recent years, blockchain technology, the underlying technology of Bitcoin, has attracted research interest as the potential missing link towards building a truly decentralized, trustless and secure environment for the IoT. Nevertheless, employing blockchain in the IoT has significant issues and challenges. While designing blockchain-based solutions, one must contend with an auditability/privacy trade-off. Blockchains inherently provide a publicly verifiable record of all transactions that occur in a network, which can be a hindrance to user privacy, especially in IoT applications. Meanwhile, transactions in blockchain networks are finalized only after achieving algorithmic consensus among blockchain peers. With the high volume of data generation events taking place, and the latency introduced through consensus mechanisms, blockchains cannot scale up to meet the requirements of the IoT.

This thesis presents the design and implementation of a blockchain-based decentralized IoT framework that can leverage the inherent security characteristics of blockchains, while addressing the challenges associated with developing such a framework. This decentralized IoT framework aims to employ blockchains in combination with other peer-to-peer mechanisms to provide: access control; secure IoT data transfer; peer-to-peer data-sharing business models; and secure end-to-end IoT communications, without depending upon a centralized intermediary for authentication or data handling. This framework uses a multi-tiered blockchain architecture with a control-plane/data-plane split, in that the bulk data is transferred through peer-to-peer data transfer mechanisms, and blockchains are used to enforce terms and conditions and store relevant timestamped metadata. Implementations of the blockchain-based framework have been presented in a multitude of use-cases, to observe the framework's viability and adaptability in real-world scenarios. In its agricultural use-case, we learned which consensus mechanisms would be better suited for the multiple blockchain tiers. We

also learned how the framework can be used to enable traceability in agricultural supply chains, and to foster cooperation among agricultural farms to pump groundwater in a sustainable fashion.

We applied the blockchain-based framework in two use-cases that provide monetary services in exchange for IoT data. We demonstrated the malleability of the framework in multiple e-business models, and through our performance analysis we highlighted the feasibility of users to engage with the framework. Finally, we implemented a modified version of the framework to a remote health monitoring use-case to demonstrate the capability of the framework in securing end-to-end IoT data communications. With all the potential applications of the blockchain-based framework within the IoT, this thesis takes a step towards the goal of a truly decentralized IoT.

Acknowledgements

All my praise is to the Almighty, who has given me good fortune and has shown me far more kindness than I deserve. During my time as a PhD candidate, I have travelled far and wide, and have had experiences I will cherish for all my days to come. I have developed a fondness for the city of Trento, its picturesque scenery, its mountains, lakes and rivers. Though I had some growing pains getting accustomed to some of the paperwork involved in living there, the time I spent in Trento will remain with me as nothing but happy memories.

I thank my advisor, Fabio Antonelli, for the freedom he gave me to conduct my research work in the direction I wanted. Conversely, I want to thank my co-advisor Massimo Vecchio, for always pointing me in the right direction, for helping me find my niche, and for helping me consolidate the theme of my PhD research. This Yin-Yang in supervision has helped me grow in my capacity as an engineer and researcher.

I thank my collaborators and colleagues at CREATE-NET, FBK. Working with you all has been a privilege and I hope we can continue our collaboration. I am thankful to my dear friends during this time, Koustabh, Nicola, Abdullah, Waqas, Husein, and the Friday prayer group. My friends have all made my time in Trento truly memorable. My best friend of all, Ahmad Katal may have been oceans away from me during this time, but thankfully our friendship has pulled me from the brink multiple times.

I thank Salil Kanhere, for hosting me at UNSW, Sydney, as a visiting researcher, and to the blockchain research group at UNSW. This was my first time in Australia, and the entire group made me feel very welcomed. I truly appreciate the collaborative energy exuded by the group, especially Guntur, my collaborator in the remote health monitoring project.

Now, my family. I thank my wife, in all honesty, for her patience in dealing with me during this time. She has helped me cope with my anxieties and has provided me support in times when I felt at my lowest. I hope I prove to be a good husband, and a good father. I thank my brothers for our unbreakable bond, us three together. Saleh brings me pride with his professionalism, and has never yet failed to make me laugh. Saad is my shining pride, and my close confidant.

I am profoundly thankful to my parents, for all that they have done to raise me, to cultivate my curiosity, and to make me capable of taking on this journey. I thank my mother for her unquestioning support and for always being by my side, no matter how difficult I may be as a child. You deserve the world. I thank my father for his untiring work ethic, the sweat on his brow and his fortitude in the face of all the challenges he has faced in life. I cannot pay him back for all that he has done for me, but I will strive to make his efforts worthwhile. This work is dedicated to his name: Sir Muhammad Ali.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The 'Internet of Things' (IoT), as defined by the IEEE IoT Initiative is an "application domain that integrates different technological and social fields" [1]. The IoT is made up of internetworked objects which collect sensory data and automate various tasks within localized systems, as well as in large global systems, which comprise geographically distributed subsystems. The IoT has found its applications within various segments of the industry, and through the introduction of wearable devices, expanding Internet access and the affordability of embedded computers, the IoT is steadily growing worldwide [2].

In current IoT (and the Internet in general) architectures, users implicitly trust third party entities and hidden services for handling data collected by IoT devices, and for issuing security certificates. Over the Internet during the last decade, a shift has occurred from local desktop applications to remote web services which store data remotely. Indeed, many security threats have emerged that seek to exploit and compromise these centralized points of trust [3]. These attacks were seen not only in the IoT through botnets [4, 5], but also the Internet at large through adversarial attacks on centralized servers [6]. The data collected by IoT devices contains confidential and private information, therefore there are acute privacy implications in the event of a security leak within centralized services.

1.1 "Trust" in Traditional IoT Architectures

"The City's central computer told you? R2-D2, you know better than to trust a strange computer."
C3PO - The Empire Strikes Back, 1980

In traditionally centralized IoT infrastructures, all IoT data collected is uploaded to the cloud, and applications hosted on the cloud process the data and send commands back to IoT devices. Subsequently, this drives up the need for high-performance networking resources, which may still not keep up with the increasing volume of data, and prove to be a bottleneck for the growth of the IoT.

Existing centralized infrastructures necessitate the maintenance of high-end servers, under the control of third-party entities. In order to avail services

over centralized network architectures, IoT users are made to trust third-party entities to handle their data, and are forced to risk their privacy being compromised. Centralized IoT architectures face the following challenges:

- The entire network risks being paralyzed with centralized servers being single points of failure [7].
- IoT users have no insight in how their IoT data is used by third-party entities, and have no accountability mechanisms for exercising control over their data. Data stored in centralized servers are subject to analysis and customer behaviour profiling [8]. Data stored in centralized servers can also be tampered with, either for misrepresentation or censorship of IoT users.
- A centralized approach to the IoT is not efficient in time-critical applications where large amounts of data are to be sent for IoT automation functions.

To improve the responsivity of the IoT, computational capabilities are being brought closer to the IoT edge through fog and edge computing. A computationally capable IoT edge, due to its proximity to the IoT users, can reduce latency in IoT data processing and end-to-end communications, and can thus provide an improved Quality of Service (QoS). Despite "trust" having moved closer to the IoT edge, trust is still not fully decentralized, especially in cases where the edge computing services are owned or managed by multiple different entities. In extension to this, another challenge is providing uniform decentralized security mechanisms that span the entirety of the IoT edge. The movement of trust from third party service providers to the end users is also called decentralisation, which is the end-goal in fundamentally rethinking how network architectures are designed. In recent years, decentralized blockchain networks have gained attention towards developing a truly decentralized and secure fabric for the IoT.

1.2 Blockchains and Decentralized Services

"Online identity and reputation will be decentralized. We will own the data that belongs to us."
William Mougayar

Blockchain technology was formally theorized by S. Nakamoto [9] and implemented for the Bitcoin cryptocurrency. Digital currency, unlike traditional currency falls under the risk of being spent twice by the same stakeholder, therefore cryptocurrency networks have an inherent need for preventative measures. Blockchains provide a solution to double-spending without centralized authorities. Blockchains are decentralized, peer-to-peer ledgers that store logs of transactions that happen in a network. These logs are grouped into blocks, which are linked together through their hash, in a chain of blocks. Blockchains are replicated over all the peers in a network, and a shared state is maintained through algorithmic consensus. Blockchain networks do not

have central points of failure, and all peers can verify the validity of transactions being carried out (hence rooting out double-spending). Blockchains are an append-only data structure, and through the virtue of their replicated shared state, their contents cannot be altered or tampered with.

Beyond exchanging digital tokens, blockchains have found their applications within an array of industries including, but not limited to, logistics, auditing, insurance and digital governance systems. Blockchains offer a promising paradigm shift towards providing decentralized services without the need to "trust" any third-party entities. Research is being actively conducted to apply blockchains to decentralize specific services within the IoT [10]. The idea of a "blockchain-based" IoT is picking up steam within research, due to the following potential benefits:

- A decentralized blockchain-based IoT enhances fault tolerance and eliminates singular points of failures [11].
- With decentralized identity management, authorisation and authentication features, decentralized peer to peer network architectures foster IoT device autonomy.
- Logs stored in the blockchains are immutable, with guaranteed auditability. Thus blockchains create a "trustless" enviroment for data exchange in the IoT. Blockchain network peers can verify the integrity of the data or software they receive through blockchain transactions.
- Through smart contracts, blockchains gain the functionality of programmable logic [12], and can enforce terms and conditions over IoT interactions. Smart contracts form the basis of decentralized applications in blockchain networks.
- Blockchains enable data and device monetisation in a truly democratic fashion, whether by renting out device utility, monetizing data, or even through microtransactions, all without centralized banks or governing bodies.

Despite these benefits, blockchains are in their early stages of development, and there is mounting research attention being paid towards design tradeoffs, to combat the scalability and performance issues that blockchains can incur (Chapter 3). Our work in developing a blockchain-based IoT framework aims to demonstrate that blockchains offer an important piece to the puzzle of decentralizing the IoT, and providing secure, decentralized IoT services.

1.3 A Blockchain-Based Framework for the IoT

The benefits of decentralized blockchain-based services motivated our design of a blockchain-based IoT framework which removes all centralized points of trust from the core of IoT service provision. The challenges associated with the blockchain-IoT convergence are mainly those of privacy and scalability [10, 13]. The framework provides one step further towards the goal of a decentralized networking medium for the IoT.

The IoT generates data in high volumes and with high frequencies. If all the IoT data generation events were to be logged onto a blockchain, the storage requirements for hosting replicas of the blockchain would explode. Furthermore, reaching consensus while recording every IoT data transaction log would add latency high enough to choke up the network. Not only can a single blockchain solution scale up to meet the needs of the IoT, it also has privacy implications since all timestamped logs of IoT data generation events will be accessible to all the peers hosting a replica of the blockchain. Furthermore, some of the mainstream blockchain consensus mechanisms are computationally expensive, and IoT devices are resource-constrained, so they are not capable enough to partipate in blockchain consensus.

To combat these challenges, we have designed a blockchain-based IoT framework, built on blockchains, in tandem with other decentralized data storage and transfer mechanisms. The framework was built with the following specifications in mind:

- Users should be able to privately share or monetize their data with entities of their choosing, without trusting any third-party intermediaries for auditability and authentication.
- Users should be able to transfer data using decentralized data transfer mechanisms with cryptographic guarantees in confidentiality.
- Users should be able to feasibly engage with the framework, with low networking overheads and transaction processing fees. The framework should be able to scale to a large number of geographically distributed users.

In this thesis, we present the basic working principles of the blockchain, along with the various consensus algorithms being used (Chapter 2). We discuss the challenges involved in integrating blockchains into the IoT, and review the existing literature so far that aims to contribute to the goal of blockchain-IoT convergence (Chapter 3).

Our understanding of blockchain technology in general, and of recent research efforts, informed the design and implementation of our blockchain-based IoT framework. The design involves a two-tier blockchain architecture, with permissionless and permissioned blockchains working together to privately log IoT data within specific verticals, and conducting confidential transactions of batches of IoT data. The IoT data itself is transferred to intended destinations using "off-chain" decentralized data storage and transfer

mechanisms, and none of the data is made accessible to members of the public.

For secure end-to-end IoT data streaming, we modify the architecture and use Tor hidden services while enforcing terms and conditions on the blockchain. Following the same design principle as before, the framework splits its data plane and control plane functionalities. It relies on off-chain data transfer mechanisms for handling the bulk data streaming, and uses the blockchain for control functionalities, including negotiations and logging.

1.4 Contributions

In this thesis, we have made the following contributions:

- We have presented the design of a decentralized, private-by-design framework for the IoT using blockchains and peer-to-peer data storage and transfer mechanisms. Our design removes centralized trust points in IoT service provision in a scalable manner. Through our design, we observe that blockchains provide trustless networking environments, however, a separation between control plane and data plane functionalities is essential for practical implementations (Chapter 4).
- We have taken theory into implementation, and have demonstrated a real-world implementation over an agri-food supply chain for traceability, along with performance metrics we observed in our implementation. Implementing the framework over multiple agri-food supply chains, the framework is used to apply game-theoretical cooperation for sustainable groundwater irrigation in smart farming applications (Chapter 4).
- We have demonstrated the malleability of the framework in various IoT business models through implementing it in two use-cases: an open IoT data marketplace, and a connected-vehicle insurance system. Both these use-cases make use of the tiered blockchain architecture and decentralized data storage and transfer mechanisms to exchange batches of data, in exchange for monetary services and benefits (Chapter 5). We conducted a performance analysis on real-world implementations to highlight the feasibility of the framework in terms of computational and networking overheads, along with transaction fees.
- We have presented a use-case of the framework in remote health monitoring. The use-case involves sending streams of data (instead of batches) from patients to their doctors, with privacy built into the solution by design (Chapter 6). Our performance analysis highlights the minimal delay in transferring real-time health data over diametrically opposite geographical locations, all while not relying on any centralized authorities.

Part I

Blockchain-IoT Concepts

Chapter 2

Blockchain Working Principles

This chapter contains text taken from the published work:
"Applications of blockchains in the Internet of Things: A comprehensive survey."
IEEE Communications Surveys & Tutorials 21, no. 2 (2018): 1676-1717.

Blockchain-based systems are the product of cryptography, public key infrastructure and economic modelling, applied upon peer-to-peer networking and distributed consensus to achieve distributed database synchronization [14] [15]. The blockchain is a distributed data structure, and is dubbed a distributed "ledger" in its utility of recording transactions occurring within a network [9]. With cryptocurrencies being one application of the record-keeping feature of blockchains, the distributed ledger has the potential to be applied in networks where any form of data exchange takes place. In a peer-to-peer blockchain-based network, all participating peers maintain identical copies of the ledger. A canonical shared state of the blockchain is maintained through decentralized consensus among the peers. Some of the core features of blockchain-based systems are as follows:

1. **Decentralization**: in centralized network infrastructures, data exchanges (i.e., *transactions*) are validated and authorized by trusted central third-party entities. This incurs costs in terms of centralized server maintenance, as well as performance cost bottlenecks. In blockchain-based infrastructures, two nodes can engage in transactions with each other without the need to place trust upon a central entity to maintain records or perform authorization.
2. **Immutability**: since all new entries made in the blockchain are agreed upon by peers via decentralized consensus, the blockchain is censorship-resistant and any attempt to alter the contents of the blockchain in a peer node is easily detected and corrected.
3. **Auditability**: all peers hold a copy of the blockchain, and can thus verify the validity of all timestamped transaction records.
4. **Fault tolerance**: All blockchain peers contain identical replicas of the ledger records. Any faults that occur in the blockchain network can be identified through decentralized consensus, and data leakages can be mitigated using the replicas stored in blockchain peers.

In this chapter, we will discuss the core working principles of a blockchain-based network ecosystem. These include the building blocks of the blockchain itself as a data structure, as well as the algorithms used to achieve decentralized consensus. Having a comprehensive idea of these principles will enable further in-depth discussion on the recent advancements of blockchain in the IoT research space. This section will start out with a general view of how blockchains work, followed by specifics on how decentralized consensus is achieved in different blockchain platforms.

2.1 Blockchain Structure

The blockchain is made up of blocks containing details of transactions that have occurred within the network. The transaction information can be regarding token transfers occurring in a network, or any manner of data exchange. Each block is divided in two parts, the header and the body. Transactions are stored within the body of the block. The header of each block contains the identifier of the previous block, therefore the blocks are connected in a chain similar to a linked list, as shown in Figure 2.1. The first block in the chain is called the "genesis" block [10].

The identifier of each block is obtained by taking its cryptographic hash, which is why having each block linked to the previous block helps the blockchain achieve immutability of its contents. If a hacker were to alter the contents of a past block, its identifier would no longer be valid, and a domino effect would render the parent block hashes in the subsequent blocks invalid as well. Therefore, to successfully alter the contents of a single block, the attacker would have to alter the headers in all successive blocks, and have this alteration take place in the majority of the nodes in the network, so as to have the peers reach consensus on this altered blockchain.

Other than the block's own identifier, and the identifier of the previous block, the header contains a timestamp of when the block was published, and the Merkle tree root [16] for all the transactions stored within the body of the block. The Merkle tree root significantly reduces the effort required to verify transactions within a block. The blockchain is a linearly growing data

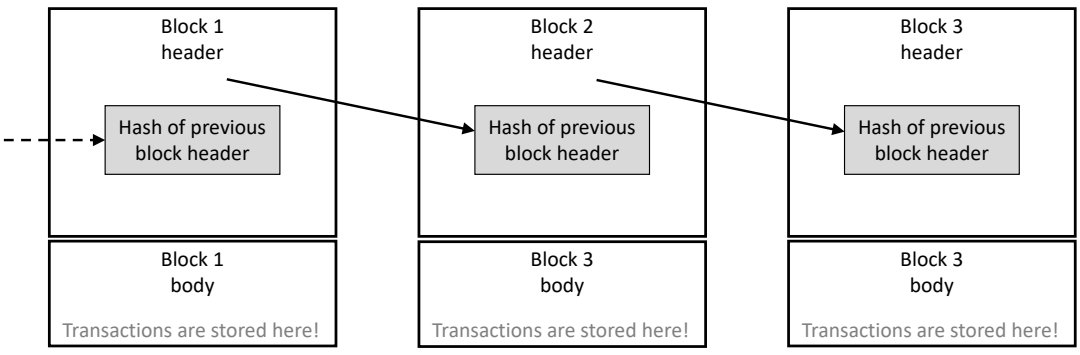


FIGURE 2.1: Graphical representation of the blockchain.

structure, with higher transaction activity inflating the sizes of newer blocks. As part of all consensus algorithms, peers verify transactions recorded in a newly published block. The transactions within a block all have a transaction ID, whereby each transaction ID is the cryptographic hash of the corresponding transaction's information stored in the block. The transaction IDs are hashed together in pairs and a hash tree is built within the block, as shown in Figure 2.2. The root of this tree is stored in the block header. To verify a transaction, a local copy of all the transactions is not required, and verification can be carried out by simply using the Merkle tree branch containing the transaction in question. A tampered transaction would produce altered hashes within its branch and would be detected without much computational effort.

In the event of multiple nodes in the blockchain network producing valid blocks at the same time, the blockchain can fork, and maintaining a single canonical version of the blockchain becomes an issue. Mainstream blockchain networks [9] [17] resolve this issue by only considering the longest fork as canon, and all orphaned forks are discarded. Other fields included in the block header contain information specific to the consensus algorithm used within the blockchain network.

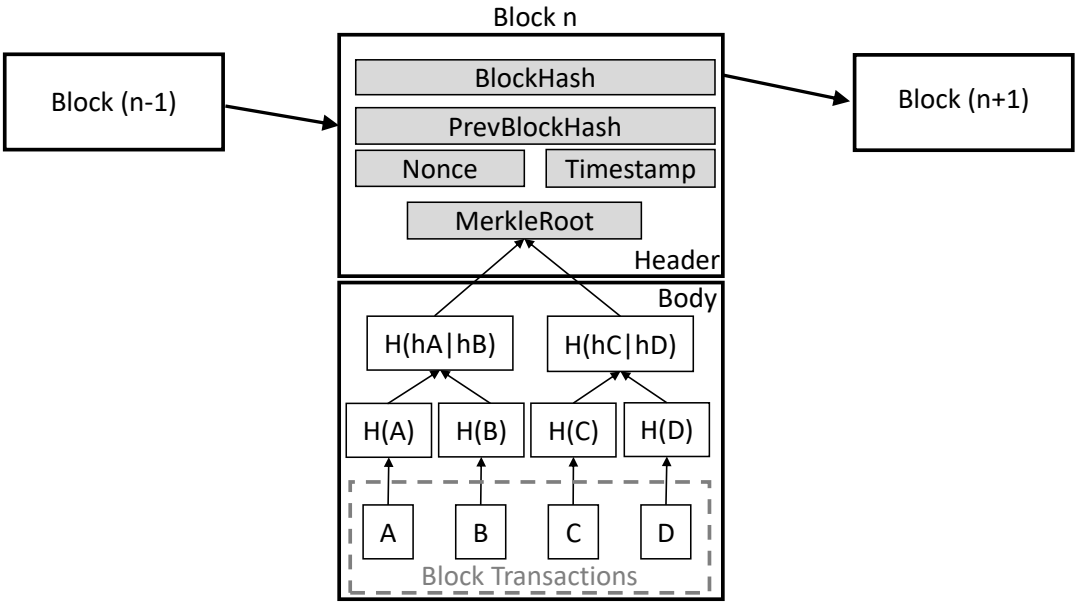


FIGURE 2.2: Block header including Merkle tree of transactions

2.2 Transactions and Digital Signatures

Each peer y_r on the blockchain has a public/private keypair $\{k_p^r, k_s^r\}$ which is used for addressing, and creating digital signatures on each transaction, for guaranteed non-repudiation. Since these keypairs are not associated to real-life identities, blockchains offer *"pseudonymity"* to its users [13]. Signed transactions are made for transferring cryptocurrency tokens, or interacting with the Application Binary Interface (ABI) of functions written in deployed

smart contracts. Each peer y_r on the blockchain has a public/private keypair $\{k_p^r, k_s^r\}$ which is used for addressing, and creating digital signatures on each transaction, for guaranteed non-repudiation. Since these keypairs are not associated to real-life identities, blockchains offer *"pseudonymity"* to its users [13]. Signed transactions are made for transferring cryptocurrency tokens, or interacting with the Application Binary Interface (ABI) of functions written in deployed smart contracts.

Transactions can also take place in between two separate blockchains via sidechaining. Sidechains [10] are blockchains synchronized with and running in parallel to an existing blockchain, referred to as the "main chain". Tokens can be transferred from the main chain to the sidechain and back, whereby the sidechain uses the tokens it has in an isolated use-case scenario. Sidechains therefore enhance functionality of the main chain and provide a testing ground for blockchain application development.

2.3 Smart Contracts

The term smart contract was coined by Nick Szabo, with the objective of "securing relationships on public networks" [12]. In blockchain networks, smart contracts perform the function of carrying out transactions in a predetermined fashion, agreed upon by parties participating in the contract. As such, smart contracts are the digital equivalent of traditional economic contracts between various engaging entities, but without the need for authorising intermediaries.

When deployed, smart contract code is stored in the blockchain, and the functions written in the smart contract can be invoked by authorized participants. Invoking functions in smart contracts incurs an execution fee, since an invocation itself is considered a transaction that is logged in the blockchain. The amount of the execution fee depends upon the amount of computation carried out by a smart contract function. Execution fees incentivize peers publishing new blocks and mitigate flooding attacks on the network.

Smart contracts can be utilized to perform a variety of functions within a blockchain network, such as:

1. Allowing 'multi-signature' transactions, whereby a transaction requires a specified amount of signatures to be issued [18].
2. Enabling automated transactions triggered by specific events. This facilitates request-response type transactions, for decentralized data access within a blockchain-based system. A smart contract can also be triggered when a message is sent to the smart contract's address [19].
3. Providing utility to other smart contracts. For example, in Ethereum, inheritance can be written into smart contracts, where one contract can invoke functions written in another contract.

4. Allowing storage space for application-specific information, such as membership records, lists or boolean states.

While Bitcoin had very limited scripting capabilities [20], newer blockchain platforms like Ethereum [17] and Hyperledger Fabric [21] use more flexible and Turing-complete smart contract scripting languages. Deployed smart contracts are stored within the blockchain, so they are visible to all participants in the network. Security lapses can occur if a participant exploits any bugs or loopholes in a deployed contract, therefore it becomes critical to practice stringency in the design process. Most notably, in June 2016, the DAO attack in the Ethereum network resulted in the attacker unlawfully siphoning off Ether worth 60 Million USD, with transactions that were valid according to the exploited smart contract [22].

With secure and well-written smart contracts, many applications provide various functionalities, utilities and algorithmic processing in blockchain networks. For example, Hawk is a smart contract-based platform designed to anonymize transactions [23], while RootStock (RSK)[1] uses smart contracts within sidechains connected to the main Bitcoin blockchain.

2.4 Consensus Algorithms

Consensus algorithms aim to securely update replicated shared states and are the essential piece of the puzzle in the working principles of the blockchain. In the blockchain, a system based on "state machine replication", consensus protocols ensure all replicas of the shared state are synchronized and in agreement at any given point in time.

According to [24] and [25], deterministic consensus in fully asynchronous communication models cannot tolerate any faults, thus assumptions for partial synchrony are required, with maximum thresholds for the latency of propagating transactions. Earlier works on consensus protocols [26] involved cryptography and partial synchrony [27], and precursor designs and proposals of digital currency (including BitGold[2] and BitMoney[3] were the building blocks that went into developing "decentralized" consensus algorithms used in blockchain networks.

Core principles applied in designing consensus algorithms are safety, liveness and fault tolerance. Safety is the extent to which a system can tolerate failures, say in an (n, f) fault tolerant system, where n represents the total number of processes, the system should be able to tolerate at most f faults. Safety is the ability to mitigate corrupted or out-of-order messages so that all non-faulty nodes reach consensus on results that are valid to the rules of the state machine. Liveness of a fault tolerant system means that in despite the

[1] https://www.rsk.co/
[2] http://unenumerated.blogspot.de/2005/12/bit-gold.html
[3] http://www.weidai.com/bmoney.txt

presence of f faults, all correctly participating nodes should be able to move forward with their distributed processes.

Maintaining fault tolerance in a consensus protocol becomes difficult in scenarios where it is possible for nodes to stop participating at any moment, or by nodes acting maliciously. This fault is termed the "Byzantine Generals Problem" [28], using the example of generals taking command of different parts of the Byzantine army. The generals rely on messengers to maintain a synchronized battle plan. The messengers can be caught by the enemy, causing the messages to be lost. More importantly, the messengers or even some of the generals may be corrupted and may cause to maliciously sabotage the battle plan. Therefore, the problem is, how do the generals maintain a synchronized battle plan without traitorous participants getting the upper hand? Similarly, in a distributed system running a consensus protocol, a node can fall under a Byzantine fault as a result of software bugs, or by being compromised. Byzantine faults occur when a node sends false messages and misleads the other nodes participating in the consensus protocol. A number of algorithms are proposed in literature [29], and in use today, that address Byzantine faults, by making varying assumptions on specific use-cases, network performance and maliciousness of compromised nodes.

2.4.1 Permissionless Blockchains

Publicly deployed blockchains that accommodate anonymous participants are termed "permissionless", and reaching consensus using votes in a permissionless blockchain is problematic. If a permissionless blockchain were to use voting to achieve consensus, participants can use multiple accounts on the blockchain to launch a Sybil attack [30], and can drive decisions in their favour. Therefore, in permissionless blockchain implementations, the consensus algorithms are based on a lottery-based selection of a single node that publishes a new block onto the blockchain. To ensure security in public blockchains where anonymous participants are required to transact in a trustless manner, block creation needs to be "expensive" so that the resources of one entity are insufficient to bias the consensus decisions in its favour. In public blockchains, the throughput can be evaluated using the parameters of block-size and block-interval, where block-interval is the average time required to publish a new block [31]. The transaction throughput can thus be theoretically calculated by:

$$\Omega(S_b, T_i) = \frac{[S_b \,/\, \chi]}{T_i} \tag{2.1}$$

where S_b represents block-size, T_i represents the average block-interval, and χ is the average transaction size.

Proof of Work

The first public blockchain consensus protocol was the Proof-of-Work (PoW) consensus, as seen in Bitcoin [9]. In the Bitcoin network, any node can participate in publishing new blocks to the blockchain, by showing that it has performed a computationally expensive amount of work, the proof of which forms the basis of the PoW consensus algorithm. Publishing new blocks under the proof of work algorithm is called "mining", and miners engage in a race to find a nonce that, when hashed with the hash of a block, produces a resultant smaller than a predefined threshold. The proportional inverse of this threshold is called the *"difficulty level"*, which is stored in the block header, and gets adjusted with increasing number of participants, to maintain an average block processing time [20][32]. Figure 2.3 shows the adjustment in difficulty level for the Bitcoin network in the last eight years[4]. Here, the calculated nonce is the proof of work a miner does, which the miner adds to the block header, and broadcasts their block to the network. This enables all participating nodes to verify the block published by the miner. Subsequently, the miner claims the processing fees associated with the transactions stored within the block as a reward for mining. In PoW consensus, the computationally expensive block creation and transaction fees secure the network against DDoS attacks and false block creation.

In a fully synchronized system, it would be easier to maintain the correct block sequence in the case of two nodes publishing a block almost concurrently [**bmoney**]. Such a system is not feasible in geographically spread-out networks since total synchrony cannot be assumed or guaranteed. Consider the case where after a block n, a node in Australia mines a valid block $n+1$, and at the same time, a node in Sweden mines another valid block $n+1'$.

[4] https://blockchain.info/charts/difficulty

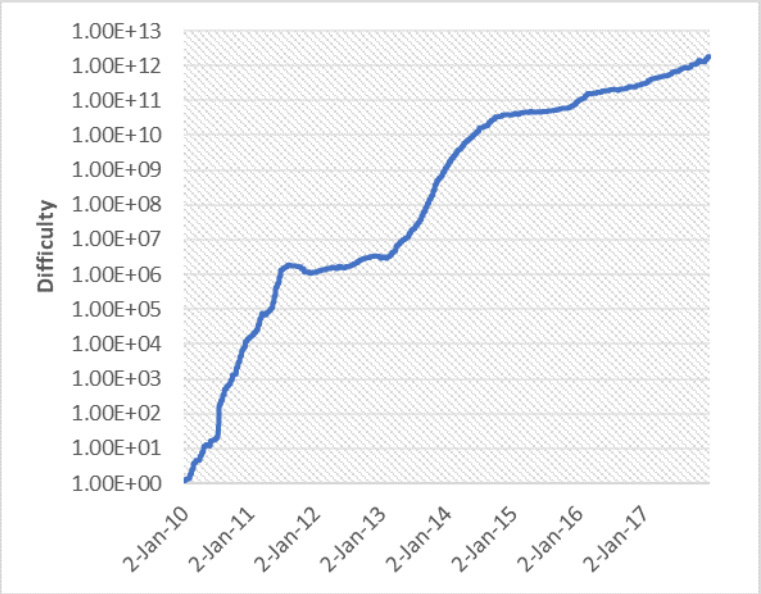


FIGURE 2.3: Adjustment of Difficulty Level in the Bitcoin Blockchain

This creates a temporary fork, where one fork has $n+1$ after n, and the other has $(n+1)'$ after n. Beyond this point, more blocks are added to these forks, and the fork with the most work committed to it is hence considered canon, and the other fork is orphaned.

Proof of work based consensus is, however, vulnerable in scenarios where a user takes control of 51% of processing power in the network [33]. Therefore, proof of work consensus provides fault tolerance as long as the total computational power is $n \geq 2f+1$ where f is the computational power occupied by a single malicious user.

PoW blockchains like Bitcoin and Ethereum delay the 'finality' of a block decision, so the blockchain can be rolled back to a past block height in the event of a 51% attack. After a block is 'finalized' it is considered irreversible. In both Ethereum and Bitcoin blockchains, a transaction is finalized after 6 confirmations. 6 confirmations take 60 minutes in Bitcoin [20], and 2 minutes in Ethereum [32].

Proof of Stake

The Proof-of-Stake (PoS) algorithm aims to cut back on the ever-increasing electricity consumption of PoW blockchain networks [34]. As an alternative to computationally expensive puzzle solving, proof of stake aims to stake peers' economic share in the network (e.g., as seen in Peercoin[5]). Here, the term "miners" is replaced with "validators," and similar to the proof of work algorithm, one of the validators is chosen to publish a block onto the blockchain. The difference lies in how the validator is chosen. In proof of stake, a validator is selected in a pseudorandom fashion, with the probability of being selected proportional to the validator's share in the network (as seen in NXT[6] and Blackcoin[7]). Naive Proof of Stake consensus mechanisms are prone to attacks like the "nothing at stake" attack, and require further considerations for it to be consensus-safe [35]. Block finality in PoS blockchains is faster compared to PoW blockchains, since there is no computational puzzle solving involved in choosing the validator.

Naive implementations of proof of stake are vulnerable to the infamous "nothing at stake" attack [35]. This form of attack arises from the fact that in the event of a fork, it costs nothing to create new blocks on both branches of the fork. In proof of work, it is computationally expensive to create a block, so any forks that occur do not last long, since it is outgrown by the valid chain and orphaned. In naive proof of stake however, a fork does not get settled easily, since it costs nothing to create new blocks, and validators can validate both branches of the fork to claim validation rewards. In this situation, even if 1% of the validators is malicious, they can tip the balance in their favour and choose to validate the branch of their choosing, either to censor transactions or commit double-spending.

[5] https://peercoin.net
[6] https://nxtwiki.org/wiki/Whitepaper:Nxt
[7] http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf

In Ethereum's Slasher[8] implementation of proof of stake, peers join the validator pool by locking up some or all of their tokens in a deposit. Therefore, the nodes become bonded validators and show commitment towards maintaining the integrity of the blockchain. Here, the blockchain keeps track of the share validators stake to publish new blocks. Ethereum's Casper [36], the most advanced implementation of the proof of stake algorithm, punishes malicious nodes for validating multiple branches in the fork by subtracting the funds they stake. As an asynchronous consensus protocol, Ethereum's Casper achieves a fault tolerance of $n \geq 3f + 1$.

Proof of X

Further alternative consensus algorithms for public blockchain deployments came about, and are classified as "Proof of X". In [37], the author present an exhaustive study of these algorithms.

Proof of activity [38] was proposed as an alternative to Bitcoin mining, designed to deliver consensus by combining aspects of the proof of work and proof of stake. The objective is to reward stakeholders that actively participate in the network. Peers start off with mining potential blocks, similar to proof of work. Decred [9] uses proof of activity to achieve distributed consensus. Computational puzzle solving in proof of activity only involves finding a proof of work against the block header, without the transactions in the block. Beyond this point, a random group of validators are chosen to vote on the validity of the mined block header. Similar to proof of stake, the probability of the validators of being chosen is proportional to their share in the network. The block is considered valid if all the validators vouch for its validity. If some of the validators are offline, the next mined block is chosen, along with a new set of validators, till a block is voted as valid. Transaction fees in this case are split between the miner and validators. Criticism of proof of activity includes concerns pertinent to both proof of work and proof of stake. It requires higher computational power, and a naive implementation can be prone to nothing at stake attacks.

Hyperledger Sawtooth[10] is an open-source project with its own consensus algorithm called ***proof of elapsed time***. Proof of elapsed time runs in a Trusted Execution Environment (TEE), like Intel's Software Guard Extensions (SGX) [39]. A trusted voting model built on the SGX helps elect a validator for publishing a new block. Proof of elapsed time is another lottery based consensus algorithm, however it foregoes the need for expensive computational puzzle solving. Nodes in the Sawtooth blockchain network request for a wait time from a trusted function within the SGX. The validator with the shortest wait time is selected the leader as soon as its waiting time runs out. Another trusted function attests to the fact that the validator did indeed wait an allotted amount of time before publishing a new block. This second function

[8] https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm

[9] https://docs.decred.org/research/overview/

[10] https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html

thus provides a proof of the validator being chosen after its allotted time had elapsed. The probability of being elected here is proportional to the resources (general-purpose processors running TEE) contributed to the network.

2.4.2 Permissioned Blockchains

In "permissioned" blockchain deployments such as private and consortium blockchains, only a limited number of known participants carry a copy of the entire blockchain [40]. Maintaining consensus therefore is much straightforward and doesn't require costly proofs for publishing a new block. Since participants are known, there is no risk of a Sybil attack, therefore voting mechanisms are used to achieve consensus. By this virtue, permissioned blockchains have a much higher performance than permissionless blockchains.

Practical Byzantine Fault Tolerance

The Practical Byzantine Fault Tolerance (PBFT) algorithm, as described in [41] involves multiple rounds of voting by all nodes of the network, in order to commit state changes. The PBFT algorithm includes an optimized, encrypted message exchange for making global voting more practical. To solve the Byzantine Generals problem via multiple rounds of voting, this algorithm requires $n \geq 3f + 1$ nodes to tolerate f failing nodes. Hyperledger Fabric [42] is a permissioned blockchain application platform being developed under the Linux Foundation's Hyperledger project. Hyperledger Fabric is designed for private or consortium blockchains, and supports smart contracts written in multiple programming languages, called chaincode. In PBFT consensus, one node is chosen to be the "leader," who assembles a set of ordered transactions into a block and broadcasts it to the network. The validating peers in the network calculate a hash of the block and broadcast it. Validating peers observe the hashes they receive from the rest of the network, which can be seen as "votes," over multiple rounds. If 2/3 votes are in favour of the candidate block, the peers add it to their copy of the blockchain. PBFT consensus provides high throughput and low latency in validating transactions, however, the overhead incurred by broadcasting blocks and votes in PBFT consensus makes it unable to scale beyond a network with tens of validators. Hyperledger Fabric also uses a variation of PBFT called Sieve[11], which is designed to perform consensus while executing non-deterministic chaincode. In scenarios involving non-deterministic chaincode, Sieve runs the chaincode first and speculates the outputs. Sieve then gets rid of minor divergences in the chaincode's output, or gets rid of entire processes resulting in greatly diverging outputs. Subsequently, Sieve maintains consensus in state-changes to the blockchain as was the case in PBFT.

[11] https://github.com/diegomasini/hyperledger-fabric/blob/master/docs/FAQ/consensus_FAQ.md

Tendermint

Tendermint [43] is a Byzantine Fault Tolerant consensus algorithm, which, similar to PBFT, provides an $n \geq 3f + 1$ fault tolerance. Tendermint uses proof of stake in combination with principles of PBFT to provide security, high throughput and low block processing times of 1-3 seconds. While in PBFT, a leader node used to get chosen pseudorandomly, Tendermint uses the lottery based properties of proof of stake, and chooses the leader node with probability proportional to the stakeholders' share in the network. After leader selection, Tendermint performs multiple rounds of voting to reach consensus on a new block. Tendermint requires a supermajority, or $2/3$ of its validators to maintain 100% uptime, and if more than $1/3$ go offline, the network may stop progressing and lose liveness. Transactions are ordered, and assuming if less than a third of all validators are faulty, Tendermint provides a safety guarantee that no conflicting blocks are created and no forks appear in the blockchain. Tendermint is compatible for public or private chains, however, it does not enjoy the same level of scalability as proof of work or proof of stake blockchains. Transaction finality in Tendermint is approximately 1 second [43].

Federated BFT

Blockchain implementations in Ripple [44] and Stellar [45] extended the traditional Byzantine Fault Tolerance and made it open-ended for participation in scenarios involving a consortium or federation of nodes.

Ripple consensus begins with a unique node list (UNL), which is a list of active validator nodes in the network. Each node has a UNL with 100+ nodes in it, and each UNL has to overlap by at least 40% with the UNLs stored by other nodes. Ripple carries out multiple rounds of voting, where nodes assemble transactions into candidate blocks and broadcast them to the nodes in their UNL. Nodes then broadcast votes on each candidate block. Each round of voting helps nodes refine their candidate block, and a new block is added to the ledger once it receives a supermajority vote of 80%. Thus, even though Ripple carries out multiple rounds of votes, it provides a fault tolerance of $n \geq 5f + 1$. Consensus in the entire network is based on consensus within subnetworks, so Ripple allows open-ended participation of users, market entities and gateways to other subnetworks.

Stellar introduces the idea of quorums in blockchain networks, where a quorum is a set of nodes used to reach consensus. A node in such a network can be part of multiple quorum slices, where each quorum slice securely reaches consensus through voting. Stellar opts for a safety over liveness property, in the event of malicious behaviour in the network, the blockchain does not progress till the malicious behaviour is resolved.

2.4.3 Performance and Scalability in Consensus Algorithms

Permissionless blockchains are forced to have slower block creation speeds, in order to take into account the propagation speeds of nodes within the network. On the other hand, permissioned blockchains have much lower latency, but suffer from a severe scalability issue. The networking overhead incurred from voting mechanisms limits permissioned blockchains to scale to only hundreds of nodes. The worst case complexity for permissioned blockchains is $O(N^2)$ compared to the $O(N)$ worst case complexity of permissionless blockchains. This limits the usability of permissioned blockchains for the IoT. Therefore, there is a steep trade-off between performance and scalability from PoW consensus to PBFT consensus [46].

Through the virtues of publicly anonymous accessibility and decentralization, permissionless blockchains are better suited to industry-wide IoT applications. Permissioned blockchains are more suited to enterprise solutions due to their higher degree of control and permission granting capabilities. Sharding[12] mechanisms in Ethereum and Tendermint can lead to leveraging the benefits of higher performance and scalability for IoT applications.

[12] https://github.com/ethereum/wiki/wiki/Sharding-FAQ

Chapter 3

The Blockchain-IoT Convergence

This chapter contains text taken from the published work:
"Applications of blockchains in the Internet of Things: A comprehensive survey."
IEEE Communications Surveys & Tutorials 21, no. 2 (2018): 1676-1717.

The term "Internet of Things" was coined in 1999 by K. Ashton as a bridge to link supply chain RFID's to the Internet. However, according to another authoritative source, the first proof-of-concept for the IoT came to life in 1982, when a group of students turned a Coke machine installed at the Carnegie Mellon University into what may be considered the first smart, connected appliance [47].

Today, the term is used as an umbrella keyword for covering various aspects related to the extension of the Internet and the Web into the physical realm, by means of the widespread deployment of spatially distributed devices with embedded identification, sensing and/or actuation capabilities [48]. However, the IoT is far more than a marketable label, rather it can be seen as a technology that is, sometimes drastically, transforming all industries and markets, enhancing and extending the digitalization enabled by information and communication technology (ICT) towards the broader impact offered by the capability to sense, communicate and actuate on the whole physical environment where such IoT devices and applications are deployed.

3.1 Issues and Challenges In the IoT

During the last two years, IoT platforms themselves are proliferating; These span from horizontal platforms able to accommodate quite generic use cases within different domains to vertical approaches able to address very specific market needs (e.g., cities, spaces, manufacturing, etc.). Clearly, the combinations of functional specializations offered by such platforms are also variegated: device management, enabling applications, data analytics, cloud storage, connectivity, only to mention a few examples. Last but not least, they come with different licensing models, either proprietary or open source. The result of this Babylon is an over-crowded and fragmented market. Moreover, while there is a common understanding on the fact that the IoT technology could play the role of enabler for several business opportunities, there exists

a set of technical challenges that, despite being already identified, are slowing down a truly global IoT adoption. The following are brief introductions to these challenges:

Cybersecurity it is considered the most critical and challenging barrier for the IoT. With respect to typical Web security, IoT security is subject to several new factors and conditions that amplify potential threats. First of all, IoT devices are commonly isolated hardware solutions that, depending on their deployment conditions, are subject to tampering in ways that may be unpredictable by manufacturers. IoT devices are then typically interconnected with other devices making it complex to manage device-to-device interactions and to protect them from malicious data manipulation. Moreover, IoT devices have typically limited computational power: this limitation hinders the adoption of highly sophisticated security frameworks. Once IoT devices are connected with each other and with the Internet, they become an interconnected and complex system which is difficult to immunize against modern security threats. For this reason, such systems become exponentially exposed to several web attacks (password security attacks, message spoofing/alteration, traffic analysis, Distributed Denial of Service, Sybil attack, eavesdropping, etc.). On the other hand, a generic "one-size-fits-all" security model is difficult to implement. To properly address security in IoT there is a need for novel security models foreseeing the development of specific policies and best practices capable of combining security-by-design approaches with specific technical countermeasures designed at different technological stacks, as well as novel organizational processes capable of addressing information security for IoT in a more holistic way [49].

Privacy the huge amount of data generated by IoT devices may offer detailed information about the context where device owners/users live, and about their habits. This data may be collected without any explicit user consent and exposed to third parties when shared by supporting IoT platforms, depriving users about control on which data and to whom his personal data is given access [50]. While administrative policies exist for providing privacy to IoT users, the challenge is to develop solutions that ensure privacy by design.

Massive Data Management the volume of data generated by IoT devices can be enormous and difficult to manage in terms of elaboration, communication/transmission, and storage. Scalable infrastructures are necessary to efficiently handle this massive growing volume of data [51].

Lack of Standardization and Interoperability the landscape of standards for the IoT is full of open solutions, supported by independent and multinational governance bodies, alliances or organizations (e.g., IEEE, ETSI, IETF,

W3C, OMG, OneM2M, ITU-T, OASIS IEC, etc.). These standards cover different aspects of IoT products, services, systems, from communication technologies to architectures. Some of them follow a neutral, cross-domain approach, while others are applicable only to specific vertical domains. Unfortunately, the uncontrolled proliferation of standards, further exacerbated by the lack of commonly accepted standards, only leads to fragmentation and can even become a real barrier for the IoT adoption and for the possibility of performing real integration in multiple application domains [52].

Lack of Skills the complexity and the heterogeneity of the technologies involved in an IoT domain require specific skills for the design, implementation, but also for the operations of the deployed solutions. Such skills are typically difficult to build or acquire by organizations. In this case, the *IoT ecosystem* plays a critical role, as it could guarantee that the right skills are offered and acquired in a proper and effective way [53].

3.2 Decentralizing the IoT through Blockchains

Simplifying the concept as much as possible, the aim of the IoT is to have smart objects communicate over the Internet to collect comprehensive data and provide personalized automation services, with little deliberate human interaction [53]. Towards this aim, current IoT platforms are built on a centralized model where a central server or broker provides services like data handling, device coordination, and authorization. This approach necessitates high-end servers and proves to be unsuitable for scenarios where objects are required to autonomously exchange data. In a centralized model, centralized servers authorize objects to communicate with each other, so the increasing number of devices communicating with each other over the Internet steadily increase set requirements for the servers. Other issues associated with a centralized model are of security [54, 55], data privacy [56] and the trust inherently required in using centralized servers [57].

Following the recognition of the opportunities blockchains offer and their potential impact, researchers and developers have taken to create decentralized applications for the IoT. The inherent features of blockchains as discussed previously, make them a natural fit to developing a secure distributed fabric for the Internet of Things and distributed cloud computing in general. Based on these features, the following are the potential benefits and motivations for developing a blockchain-based decentralized IoT framework:

- *Resilience*: IoT applications require integrity in the data being transferred and analyzed, therefore IoT frameworks need to be resilient to data leaks and breakage. Blockchain networks store redundant replicas of records over blockchain peers, which help maintain data integrity and can provide resilience to IoT frameworks.

- *Adaptability*: Currently, the heterogeneity of IoT devices and protocols limit their interoperability, and since blockchains are semantics independent distributed databases, using blockchains as the network control mechanism for the IoT will add a greater degree of adaptability to it. Blockchains are proven to work over heterogeneous hardware platforms, and a blockchain-based IoT framework holds the promise to adapt to varying environments and use cases to meet the growing needs and demands of IoT users.

- *Fault tolerance*: The Internet of Things represents a proliferation of always available smart devices that collect data and provide automated functionality. Network control mechanisms for the IoT require high availability, which may not always be the case in architectures involving centralized servers. Blockchains are Byzantine fault tolerant record-keeping mechanisms that can identify failures through distributed consensus protocols.

- *Security and privacy*: One of the most important challenges faced by the IoT, as discussed before is network security. To ensure confidentiality and data protection, blockchains have pseudonymity in its addressing and distributed consensus for record immutability. Data modification attacks cannot be mounted in public blockchains since the blockchain does not exist in a singular location. Furthermore, the cost added to making new transactions (either monetary or computational) protect the network against flooding attacks and DDoS attacks.

- *Trust*: Blockchains enable trust between transacting parties. The "trustless" features of blockchains remove the need for users to trust centralized entities to handle their IoT data, thus preventing malicious third party entities from accumulating users' private data. Blockchains allow faster settlements for automated contracts without the need for trusted intermediaries.

- *Reduced maintenance costs*: Centralized IoT infrastructures face a significant disadvantage in their high server maintenance costs, which not only add monetary cost, but also adds to the communication costs in device-to-device communications. Centralized cloud storage services use geographically spaced data centers which are large central points of failure. Centralized cloud services introduced much lower prices for storage and computing, which led to their widespread adoption. However, blockchains have the potential to significantly reduce costs incurred by maintaining dedicated servers. Public blockchains applications do not require dedicated servers, and utilize the computational and storage capabilities of its participants. Since the participants receive incentives for their contributions, blockchains stand to be the next step in democratizing the IoT. Blockchain-based data storage platforms like Sia[1] demonstrate the reduced costs in storing data using blockchains. In Sia, instead of using dedicated servers, users rent out any

[1] https://sia.tech/technology

available storage space they have, which others utilize to store data. While the cost for storing 1 Terabyte per month on Amazon S3[2] is $25, the cost of blockchain-based data storage in Sia is $2 per Terabyte per month.

- *IoT e-business models*: In current IoT service provision, users surrender their data to centralized service providers in exchange for IoT services, however, data being exchanged over public blockchains can have the added benefit of enabling users to engage in a new data marketplace and monetize their IoT data. Blockchain-based solutions also incentivize users to make IoT resources available for others to use on demand, in exchange for cryptocurrency.

Blockchains show promise in several industry verticals, and startups are locked in a race to develop blockchain-based distributed applications for different use case scenarios. As discussed before, a significant part of these applications have direct link to the IoT. An example of these applications can be seen in the insurance industry.

An example of the numerous industry verticals for a blockchain-based IoT are smart grids. Blockchains have the potential to facilitate trade of energy between producers and consumers. In a blockchain-based smart-grid system, each participant has a unique identifier which can be authenticated without relying on a third-party service provider, thus bridging transacting entities in a democratic fashion. Blockchain-based records and cryptocurrency can be used for negotiating, effectuating trade and maintaining records, as proposed in [58, 59, 60, 61, 62]. .

Another use case for integrating blockchains with IoT is in smart-insurance. In the insurance sector, many companies have taken up IoT applications to collect data for aiding them in calculating insurance premiums and processing insurance claims. Several management processes within insurance can be automated using smart contracts, while maintaining compliance to legal requirements. Considering the benefits of the combination of the IoT and blockchains, eventually insurance use cases will migrate from telematics to real-time IoT cryptocurrency applications [3].

Other industry verticals where blockchains and IoT can bring potential benefits include healthcare, supply chains, energy trading smart-grids, smart home applications, connected vehicle fleet management and robot swarm coordination. Peer-to-peer decentralized applications in these areas can bring about a revolution in ubiquitous service provision and distributed oversight of all IoT data transactions.

[2] https://aws.amazon.com/s3/

[3] https://www.ibm.com/blockchain/industries/insurance

3.3 Integration Schemes for Blockchains and IoT

Centralized cloud services have made major contributions in the growth of IoT, but in data transparency, there is an inherent need of trust and a lack of absolute confidence. Centralized cloud services act much like a black box for IoT services, and IoT users do not have control and total confidence in how the data they share will be used. Furthermore, centralized cloud services are vulnerable to faults and lethal security attacks. In the evolution of IoT, the network edge is getting more functionality as compared to the cloud, as seen in fog and mist architectures [63]. The IoT can benefit from the decentralized network paradigms offered by blockchains, so further developments to the IoT can continue while eliminating the need for trust in centralized services. However, blockchains are still in their early stages of research and development, and there are still multiple research challenges towards integrating IoT and blockchains in a seamless manner.

Achieving absolute decentralisation in the IoT using blockchains is problematic, considering the vastly varying devices involved in the IoT. Most devices on the IoT edge have resource constraints, and cannot host a copy of the blockchain or engage in validating new blocks for the blockchain. Therefore, it is important to decide upon what roles the different entities in the IoT edge (devices, gateways, etc) will take.

Table 3.1 indicates the possible roles the participants of a blockchain network can assume. Full nodes are participants in the blockchain network that host the entire copy of the blockchain. Full nodes can issue transactions to the blockchain, and can choose to act as a validator for adding new blocks onto the blockchain. Light nodes running a "light-client" application can issue transactions to the blockchain, and can host a copy of the block headers from the blockchain. Light nodes can verify the validity of transactions through the block headers, however they do not publish new blocks to the blockchain. Light nodes are used as an easier entry point to the blockchain, using limited computational resources. A transaction-issuer running a "light wallet" application is a participant that does not maintain a copy of the blockchain or engage in block validation, however it simply issues transactions to the blockchain. In some blockchain platforms, the potential downside of having a light wallet transaction-issuer is that it performs transactions through a light or full node. This can be a node in the same local network as the

TABLE 3.1: Node Types in Blockchain Networks

Node Type	Storage	Validator
Full Node	Full Blockchain	Yes
Light Node	Block headers	No
Transaction Issuer	None	No

transaction-issuer, or in the case of the Ethereum platform, a third party service like Infura [4] and Metamask[5]. The former is a more suitable choice since using third party services nullifies the point of decentralization.

Choosing the right consensus algorithm can prove to be detrimental in integrating blockchains with the IoT. Proof-of-Work based mining remains unfeasible in context of the IoT due to its high computational requirements and high block processing time. In some cases, researchers have attempted to relax the validation requirements of PoW based consensus [64], however, this can lead to compromises in the security afforded to IoT networks by blockchains. PoW consensus with relaxed requirements can be securely implemented in consortium blockchain deployments, since all members of the blockchain are known. In single-enterprise solutions, or use-cases where the blockchain-connected nodes or gateways are known and in the order of hundreds, voting-based consensus like PBFT can be used, to maintain security and low block processing times. For public blockchain deployments, alternate consensus algorithms including Proof-of-Stake and other Proof-of-X algorithms are seen as more suitable for blockchain deployments within the context of the IoT.

Keeping in mind the resource constraints faced by IoT devices, it becomes necessary to employ some design considerations about the extent of their involvement in a blockchain network. Most IoT devices do not have cryptographic capabilities, and do not meet the computational and storage requirements for engaging in blockchain consensus algorithms. To account for these limitations, IoT edge devices only take on the role of simple transaction issuers. Even in the case of light-nodes, most IoT edge devices do not carry sufficient storage capabilities to host the "headers only" version of the blockchain. IoT edge devices or gateways running as simple transaction-issuers have verifiable blockchain-identities without the need to host an entire copy of the blockchain. Therefore, such edge devices are more manageable within blockchain networks and can continue making contributions to the blockchain, while other full nodes in the blockchain network can carry out decentralized consensus and block validation.

In recent literature, we have surveyed a variety of integration schemes that aim to account for IoT edge device constraints in a blockchain-based IoT, with varying requirements of cryptographic capabilities for the IoT edge devices. The following is a discussion of the alternate paradigms as seen in recent literature for integrating blockchains and IoT:

- **Gateway devices as end-points to the blockchain**: in this integration scheme, all communications go through the blockchain, while the IoT gateways act as end-points to the blockchain network. In this case, the IoT devices will be registered to the gateway device, and the gateway

[4] www.infura.io
[5] www.metamask.io

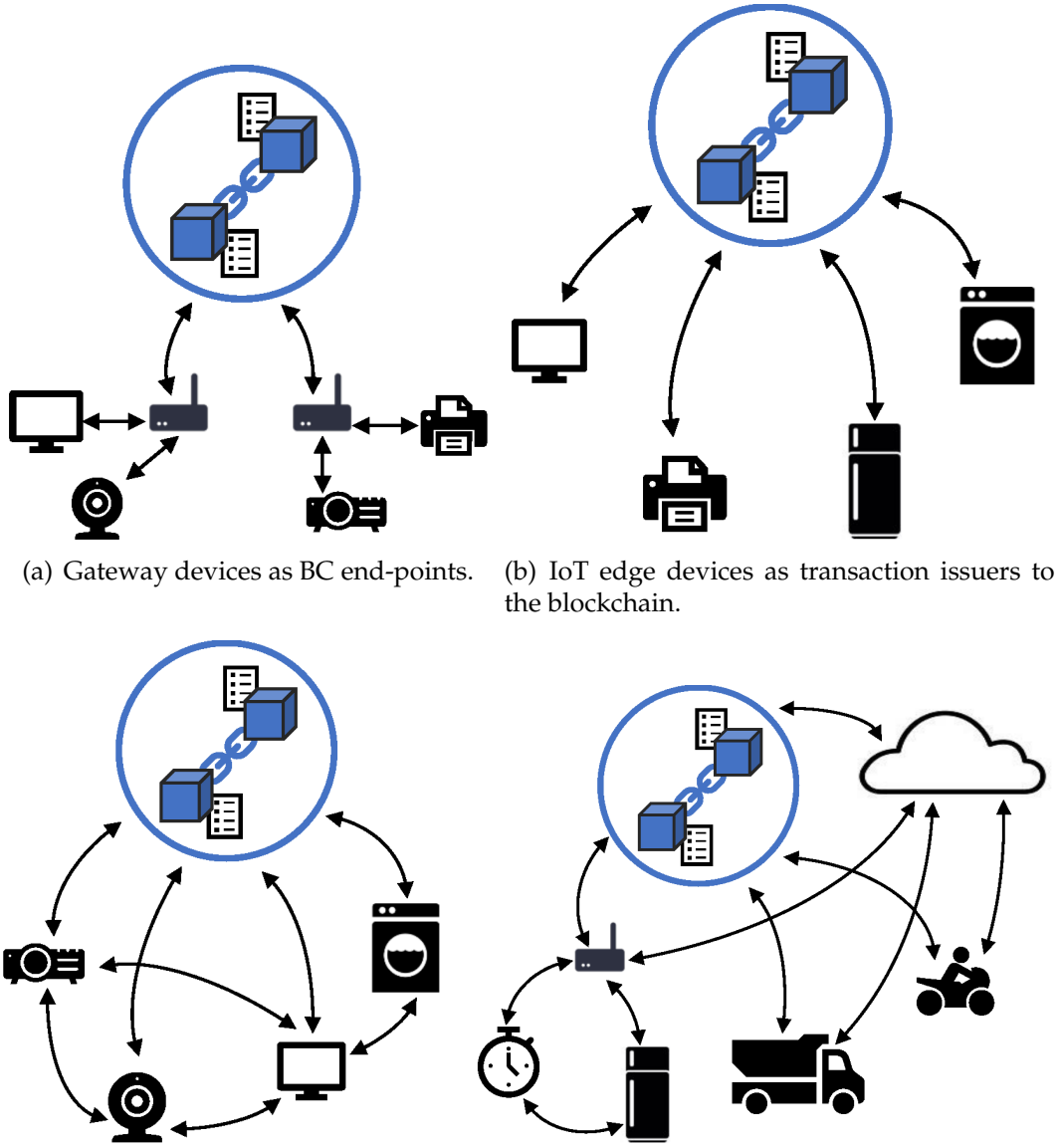
issues transactions to the blockchain. This approach enables traceability of all communications involving a specific IoT gateway and IoT service. This integration scheme can also be used to authenticate communications between devices connected to separate blockchain-enabled gateways [65]. In this approach, not all of the data transferred needs to be stored on the blockchain. The blockchain itself can be used as a control mechanism, with smart contracts acting as programmable logic, while data transfer can occur over peer-to-peer technologies like BitTorrent and IPFS [6]. However, recording all IoT interaction events on the blockchain will increase bandwidth and storage requirements, and currently scalability is a well known research challenge towards the integration of blockchains and IoT. Fig. 3.1(a) is an illustration of this approach. The degree of decentralization achieved through this approach is not as fine-grained as in the case where devices issue transactions directly to the blockchain.

- **Devices as transaction-issuers to the blockchain**: this integration scheme is seen in [66], however, in our discussion we are assuming that the IoT devices are not in fact carrying a copy of the blockchain, but are simply issuing transactions to the blockchain, as shown in Fig. 3.1(b). Similar to the previous approach, all IoT interaction events are logged onto the blockchain for secure accountability. In this approach, IoT devices can be provided with cryptographic functionality. The trade-off here is higher degree of autonomy of IoT devices and applications, versus increased computational complexity of IoT hardware.

- **Interconnected edge devices as end-points to the blockchain**: in this approach [66], IoT gateways and devices issue transactions to the blockchain and can communicate with eachother off-chain, as seen in Fig. 3.1(c). While introducing the need for routing and discovery protocols, this approach ensures low latency between the IoT devices and the choice to log specific interactions on the blockchain. This integration scheme would be more suited to scenarios where interactions are much more frequent and high throughput, low latency, reliable IoT data is required.

- **Cloud-blockchain hybrid with the IoT edge**: this approach is an extension to the previous integration scheme, whereby IoT users have a choice to use the blockchain for certain IoT interaction events, and the remaining events occur directly between IoT devices [66]. This approach leverages the benefits of decentralized record-keeping through blockchains as well as real time IoT communication. Fig. 3.1(d) is an illustration of this hybrid integration scheme.The challenge posed by this approach is to optimize the split between the interactions that occur in real-time and the ones that go through the blockchain. Hybrid approaches can utilize fog computing to overcome the limitations of blockchain-based IoT networks.

[6] www.ipfs.io



(a) Gateway devices as BC end-points.

(b) IoT edge devices as transaction issuers to the blockchain.

(c) Interconnected edge devices as blockchain end-points.

(d) A hybrid cloud/blockchain approach.

FIGURE 3.1: Blockchain integration schemes for the IoT. All arrows indicate interactions.

Which integration scheme to implement depends upon the requirements of the IoT application. For instance, when there is a need for immutable record-keeping and relatively lower number of interactions are taking place, the first two interaction schemes make more sense. In applications that require higher performance, using a blockchain alone may not be adequate, and it would make sense to use a hybrid integration scheme. In IoT use-cases neither IoT devices or gateways should ever be used as full-nodes, since the storage and computational overheads will not be able to justify the potential benefits. Furthermore, in the case of some applications, an integration with blockchains may not be necessary. In order to ascertain which application scenarios justify a blockchain integration, the methodology presented in [67]

can be used.

Current centralized IoT models are linked to specific drawbacks and limitations that can be canceled or mitigated by the decentralization properties of the blockchains [68]. Blockchains lay the groundwork for developing decentralized IoT platforms that enable secure data exchanges, and trustless record keeping of the messages exchanged between devices without the need for maintaining high-end servers. In the following sections, we will see how the blockchain technology can play a relevant role in addressing and overcoming some of the aforementioned challenges in different areas of the IoT.

3.4 Related Work

The aim of IoT is to have smart objects communicate over the Internet to collect comprehensive data and provide personalized automation services with little deliberate human interaction [69]. Towards this aim, current IoT platforms are built on a centralized model where a central server or broker provides services like data handling, device coordination and authorization. This approach necessitates high-end servers and proves to be unsuitable for scenarios where objects are required to autonomously exchange data. In a centralized model, centralized servers authorize objects to communicate with each other, so the increasing number of devices communicating with each other over the Internet steadily increase set requirements for the servers. Other issues associated with a centralized model are of security [54, 55], data privacy [56, 7] and the trust inherently required in using centralized servers [57].

The specific challenges posed by the current centralized model of IoT make it a suitable challenge domain where the decentralization properties of blockchains can yield relevant benefits. Blockchains lay the groundwork for developing decentralized IoT platforms that enable secure data exchanges, and trust-less record keeping of the messages exchanges between devices without the need for maintaining high-end servers.

This section will outline the issues and disadvantages of centralized IoT architectures, and make a case for the benefits of decentralizing IoT architectures using blockchains. Following that, we will address different research directions for blockchain use cases in IoT, and discuss the solutions proposed for each use case in recent research efforts.

3.4.1 Privacy in IoT

Within the centralized and hyperconnected nature of homes and cities today, we see concerns related to user data privacy. The privacy issues in IoT are immense, considering the sheer amount of data being collected, transferred, stored, and undoubtedly sold. Data collection in IoT has diverse purposes, for example, an organization may lease equipment and collect usage data for

billing purposes. The organization can draw inferences about user's preferences and habits from the data itself as well as the associated metadata [70]. Customers in this position place their trust in the organizations providing the Internet-based applications, and have little knowledge of what data is being transmitted, or if their data is being shared or sold to third party entities [71]. The worst-case scenario here would be mass-surveillance programs [8], whereby entities collecting user data can collaborate with 'Big Brother' entities and collect data not relevant to the provided service.

Apart from authentication and secure cloud computing, in order to prevent violations of privacy, the challenges involved are implementing policies that ensure data confidentiality, integrity, ownership and governance [72]. [73] advocates for "privacy-by-design," and emphasizes the need for empowering users, and giving them the ability to control the data that is collected and shared. Such a design aims to implement access control policies to evaluate requests and decide whether to allow access to data or not. Towards meeting the challenge of data ownership, [74] proposes transparency in data transfer. Transparency enables users to keep track of the parties that utilize the data collected by their IoT devices. To combat the privacy violation by a rogue sensor network, current solutions in privacy involve users going through a privacy broker [75], which itself if an intermediary entity between the user and the sensor network. Similar to advances promoting anonymity in blockchains, group signatures [76] and ring signatures [77] are a proposed solution for IoT privacy. [78, 79, 80, 81] apply group signatures to mobile and vehicular ad-hoc networks (MANETs and VANETs). The concept remains the same: the user transfers data as part of a group, so as to mask the user's identity. Another proposed solution for privacy and anonymity in IoT is k-anonymity [82], which is an approach meant to prevent identity disclosure by anonymizing data transmitted. K-anonymity however, does not prevent attribute disclosure and has received critique [83]. To address this, an attribute-based signature was proposed to further the idea of data anonymization [84].

For issues related to privacy, decentralization is being explored, and [85] proposes a decentralized anonymous authentication protocol, based on cryptographic Zero-Knowledge Proof of Knowledge (ZKPK). However, this protocol has received criticism: the protocol is susceptible to attack when an adversary impersonates an actual user in the data collection aspect of the protocol [86]. Blockchains lay down the foundations of decentralizing networks, and carrying out data transfers securely, without the need of any authorizing and authenticating intermediaries. The immutable record-keeping attributes of blockchains provide a viable solution for governing IoT micropayments and data sharing, so privacy-preserving network design for IoT using blockchain and smart contracts is a fertile and active area of research.

In enabling microtransactions over the Internet, blockchains provide pseudoanonymity for IoT users and devices to engage in data transactions or device commissioning [87]. FairAccess [88] is a distributed authorization framework that leverages pseudonymity in blockchain networks. Huckle et al. discuss the possibilities of resource sharing over IoT using blockchains [89].

Furthering that idea, Hardjono et al. [90] propose privacy preservation in commissioning IoT devices over the cloud, using permissioned blockchains. Even though the proposed architecture uses permissioned blockchains, users have the ability to commission IoT devices anonymously without the need for an authenticating intermediary. PlaTIBART [91] is a platform that uses off-chain computation to decrease latency in blockchain-based transactive applications for the IoT, while maintaining privacy using private blockchain implementation. [92] uses attribite-based encryption for sensor data to enable privacy in IoT ecosystems. PISCES [93] is a framework for monetizing IoT user data with privacy-by-design, using Privacy Validation Chain (PVC), which allows users to control and monitor where their data is shared, and for what purpose.

For cloud computing, the proposed solution outlined in [94] introduces software-defined cloud computing with blockchain based access control for a distributed solution for privacy. Another privacy-preserving access model is described in [95] where blockchains and fine-grained access-control policies allow users to govern their own data.

The most promising solution for private-by-design IoT data transfer is inter-blockchain communications. The ICON loopchain[7] and COSMOS[8] project aim to interconnect blockchains to open them up to multiple use-cases, including IoT. Dorri et al. [64] introduce a privacy-preserving architecture where smart home owners can log IoT events in a private sidechain and use cloud storage for IoT data. Users can then choose to share any amount of their encrypted data with others over a public overlay blockchain, according to access-control policies written into the block headers. Smart contracts for access-control was an idea introduced in Hawk [23], which implements programmatic access-control mechanisms via smart contracts. [96] uses peer-to-peer storage to address storage-based scalability issues of blockchain, however, they do not implement interconnected blockchains for horizontal scalability. [97] expands upon these proposals by using IPFS as a distributed storage medium for IoT data, and smart contracts to enforce access-control.

While power grids are experiencing changes due to a boom in renewable energy solutions, decentralized IoT applications are emerging that help manage transactive microgrids. Here, blockchains are being researched for use in smart grid applications, where energy sharing applications require privacy, decentralized control and monetization mechanisms. [58] uses group signatures and off-chain encrypted anonymous message streams to provide privacy in energy trading applications. More recently, [59] was proposed as a solution towards enabling energy prosumers to tokenize and trade units of energy with consumers, while protecting the prosumers' personal information. [60] uses smart contracts to enable privacy and decide tariffs for energy sharing within smart grids.

[7] https://icon.foundation/
[8] https://cosmos.network/

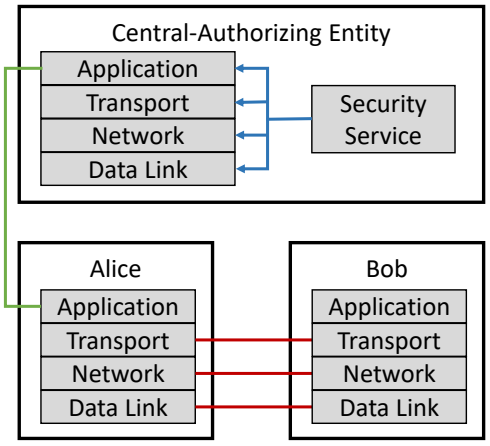


FIGURE 3.2: Traditional Security Mechanism Architecture in IoT.

3.4.2 Security in IoT via Blockchains

As the physical world joins the Internet, the attack surface from known and new threats expands exponentially, resulting in complex security implications [98]. The goal of the IoT is to automate functions while maintaining protection against the threat of a varying range of security attacks. An essential security challenge of the IoT comes from its ever expanding edge. In an IoT network, nodes at the edge are potential points of failure where attacks such as Distributed Denial-of-Service (DDoS) can be launched [99]. Within the IoT edge, a set of corrupted nodes and devices can act together to collapse the IoT service provision, as seen recently in botnet attacks [4].

Another threat to the availability of IoT service provisioning comes from its heavily centralized configuration [100]. A central point of failure not only is a threat to availability, but also to confidentiality and authorization [101]. Furthermore, confidentiality attacks arise from identity spoofing and analyzing routing and traffic information.

The IoT faces confidentiality attacks that arise from identity spoofing and analyzing routing and traffic information, as well as integrity attacks such as modification attacks and Byzantine routing information attacks [102]. Data integrity in the centralized IoT configuration is challenged by injection attacks in applications where decision making is based on incoming data streams. IoT data alteration, data theft and downtime can result in varying degrees of loss. Traditional security solutions in the IoT are centralized, involving third party security services, as seen in Fig. 3.2. Even within fog architectures, maintaining uniform standards for security is logistically difficult. Using blockchains for security policy enforcement and maintaining publicly auditable record of IoT interactions, without depending on a third party, can prove to be highly benefitial to the IoT.

With virtues of decentralized public-key infrastructure, fault-tolerant design, auditability and inbuilt protection against DDoS attacks, blockchains have demonstrated their capabilities in delivering security to transactive networks

like Bitcoin. A blockchain-based IoT solution is resistant to false authentication since all devices issuing transactions have dedicated blockchain addresses. The consensus protocols used in public blockchains prevent malicious actors from launching denial of service attacks since making multiple empty transactions incurs transaction fees [103]. Thus blockchains have the potential to disrupt IoT security mechanisms and provide improved security solutions to the IoT stack.

Providing Access Control Through Blockchains

Recent research has seen several proposed solutions for enforcing access control policies in the IoT without relying on a third party service. Solutions like [104] provide a secure public key infrastructure that is more fault tolerant than centralized solutions.

Zhang et al. [87] introduce a tokenized approach to performing access control in the IoT through blockchains and smart contracts. The main idea is to use customized cryptocurrency to buy temporary access privileges for physical or digital assets. Another tokenized approach to access control is outlined in [105], where users are assigned different roles, and access control policies written into smart contracts can be used to grant or revoke access privileges for an IoT user's data. Similarly, [106] and Enigma [107] store chunks of encrypted data in the blockchain and uses a tokenized approach and smart contract policies for allowing and revoking access to stored IoT data. Another access control model is proposed in [108], whereby IoT users can grant and revoke access to stored IoT data through smart contracts. Hamza et al. [109] use an overlay blockchain to provide an access control mechanism for big data. They use programmable smart contracts to inform authorization decisions for big data access requests.

Shafagh et al. [110] propose an blockchain-based access control solution for data stored in off-chain Decentralized Hash Tables (DHT). The blockchain in this solution stores access privileges for different users for any stored data in the DHT. DHT nodes lookup the blockchain records to make access control decisions. Dorri et al. [111] provide a lightweight scalable blockchain consensus mechanism for securing and anonymizing IoT users, and allowing them to monitor their data remotely.

Maintaining Data Integrity Through Blockchains

To launch a modification attack in a blockchain-enabled IoT architecture, an adversary would attempt to alter the records in the blockchain, or create false blocks in the blockchain, either containing false transactions, or censoring transactions that have occurred. This is near impossible in public implementations of the blockchain, where canonical records of the blockchain are maintained by means of distributed consensus. This further makes the case for decentralizing the IoT using blockchains, since properties inherent to the blockchain prevent attacks that compromise data integrity [112].

Biswas et al. [113] propose a blockchain-based smart city solution whereby the integrity of the stored data is guaranteed through the blockchain's inherent immutability features. Enigma [107] and Shafagh et al. [110] propose data storage solutions based on Decentralized Hash Tables (DHT) and immutable blockchain records. Data requests go to DHT nodes while the blockchain ensures integrity of access control policies and the stored data itself.

A blockchain-based data integrity service is outlined in [114], where query-based integrity checks can be performed without third-party verification. Here, the blockchain is used as an added layer for providing security and integrity to data objects stored on the cloud. Issuing queries and verifying the blockchain records are used to detect any loss of data integrity.

Secure software updates for the IoT by applying blockchains in IoT is receiving research attention. In [115], embedded IoT devices receive secure firmware updates in a blockchain network. The proposed scheme uses peer-to-peer technology for delivering firmware updates and ensures the integrity of the firmware installed in embedded devices. Similarly, the authors of [116] and [117] use permissioned blockchains to store software updates within transactions, so IoT devices can receive updates in a secure, peer-to-peer fashion.

Ensuring Confidentiality Through Blockchains

The blockchain has inherent addressing involving public/private key pairs, therefore, blockchain-based applications have built in authorization and confidentiality features since each transaction is signed by the issuer's private key. Axon et al. [104] leverage a blockchain-based PKI to manage IoT devices. They used smart contracts that issued commands to the IoT devices using their blockchain addresses. These commands range from changing working policies, to recording energy usage information onto the blockchain.

Aitzan et al. [58] propose a confidentiality solution for energy transacting smart grids. The aim is to not only keep the information shared between two parties confidential, but to also hide the identity of the energy producers. In this regard, the authors suggest a mechanism for generating and altering blockchain addresses for the energy producers, so as to hide the producer's identity altogether. The solution does not aim for complete decentralization because it uses Distribution System Operators (DSOs) to manage security among the producers and consumers as an automated intermediary.

Alphand et al. [118] proposed a solution which is a platform for IoT security management. It is built on a blockchain that enforces authorization policies and maintains interaction records, and the OSCAR (Object Security Architecture for the IoT) [119] security model, using a group key scheme. The authors use OSCAR to set up authorized multi-signature groups, and the blockchain for flexibly setting authorization rules and maintaining an immutable records of all access events.

Cha et al. [65] use an Ethereum blockchain to maintain confidentiality between IoT gateways within private blockchains. The gateway maintains information pertinent to the devices and all interactions with the IoT remain confidential under blockchain-based signatures.

[90] is an approach for commissioning cloud-based IoT resources. It uses a permissioned blockchain, and all data transferred to and from a commissioning party are kept confidential under blockchain-based PKI.

Improving IoT Availability With Blockchains

The proposed blockchain-based security solutions discussed above provide improved availability in the IoT by decentralization properties inherent in blockchains. Solutions that provide on-chain data storage have built-in features for availability, since there are no central points of failure. Off-chain storage solutions have improved availability of its interaction records, however the availability of the stored data is dependent upon the off-chain storage mechanisms used.

Chakraborty et al. [120] proposed a blockchain solution to handle security issues with resource-constrained IoT devices. The authors have computationally capable gateways participate as validators in a blockchain network, which has inherent decentralization and fault-tolerance properties that guarantee liveness of the solution.

Bahga et al. [121] has IoT devices with blockchain addresses in a blockchain network. The aim is to develop a blockchain-based manufacturing and smart factory system. Since each device is on the blockchain, users can issue manufacturing commands directly to the devices as transactions. The authors present a machine maintenance and diagnostics use case. The decentralized nature of connected devices help the network stay live in the event of multiple faults in machines, and in the event of a fault, the remaining live devices can report it.

3.4.3 ID Management

In the traditional Internet, identity management solutions such as SAML [122] and OpenID [123] incorporate authentication methods, to prove identities and to provide secure channels. Open ID and SAML provide a decentralized method for authentication, but do not enable two parties to engage without an authorizing third party. A SAML or OpenID identity provider is required so that users can sign up for online services. While there is no single central authority for OpenID or SAML, third party identity providers perform authentication and therefore, users are mandated to place their trust on third party entities for authentication.

In cases where users are not involved, devices authenticate themselves with tokens or security certificates. Furthermore, in many cases, the protocols used in IoT do not necessarily fit the TCP/IP stack. Over the course of the

development of IoT, certain protections have been put in place to prevent identity abuse. OAuth 2.0 [124] is an open authorization framework that has been widely used for IoT applications. OAuth uses tokens to grant or revoke access to specific online applications. Despite its merits in managing IoT device identities, the common issue of traditional identity management solutions is the lack of guaranteed trust and reliance upon third party authorizing entities. In the case of OAuth, this is the Authorization Server, that controls the issuance and revocation of tokens.

For current identity management protocols in the IoT, interoperability is an ongoing challenge. Interoperability becomes difficult in the presence of multiple protocol options, cross-platform architectures, and variations in semantics and conformance.

A blockchain-based IoT ecosystem would provide identification for every device, that can be used as a watermark over all the transactions a device makes. The IoT, and as an extension, the Internet, can benefit greatly by blockchain identity management solutions. The most pronounced benefits are distributed trust and security since blockchains render centralized authenticating servers irrelevant.

While multiple startup companies have identity management applications in varying stages of development, proposed solutions have emerged in recent research publications for managing identities of connected devices in the IoT. [104] highlights the potential benefits of PKI without single points of failure by using blockchains. This study demonstrates varying levels of privacy-awareness that can be achieved with blockchain-based PKI.

The authors of [59], [125] and [126] propose identity management systems based on blockchains for transacting energy systems. Applications like these contribute to the vision of an open model energy sharing system, and to the goal of developing smart grids with renewable energy.

In [127], the proposed solution for hosting IoT devices on the cloud calls for identity management, and the authors detail their findings on performance analysis in blockchain deployment over IBM Bluemix. They use blockchain-based addressing to host virtual IoT resources, that users can transact with using their specific blockchain address. Kravitz et al. [128] use permissioned blockchains to propose a solution for distributed identity management. Since all participants in a permissioned blockchain have to be known, a participant makes their identity known and linked to their blockchain address, which can then be used for IoT interactions. This does not allow for anonymity, but for specific enterprise-level IoT applications, it is a viable decentralized identity management mechanism. Huh et al. [129] implemented an identity management system for interconnected devices using Ethereum smart contracts. They implement smart contract programmability for managing keys in a fine-grained fashion.

Lee et al. [130] propose a blockchain-based identity and authentication management system for mobile users as well as IoT devices. Their proposed solution involves generating and maintaining blockchain identities as a service, without any considerations for interactions or communications through the blockchain. The blockchain-based identities in this case are only meant for decentralized authentication purposes. Urien et al. [131] propose a unique identity management solution for a blockchain-based IoT. They developed cryptocurrency smart cards (CCSC) based on javacard secure elements. The smart card, developed on the JC3.04 standard platform provides improved security compared to 32 byte keys typically used in blockchain networks.

Identity management is a challenge being actively worked upon in blockchain research and development. Early contributions like [132] aimed to provide a distributed domain naming system for the Internet using blockchains. Several startups are developing solutions for blockchain-based identity management for online entities, including IoT devices. ShoCard [133] is an identity verification platform built on a public blockchain, where users can verify their blockchain ID simply by passing their card over a sensor. Thus, ShoCard provides an identity solution for humans by leveraging IoT and blockchains. A startup that aims to provide identity management for IoT devices is Uniquid[9], a platform for access and identity management for devices, cloud services, and humans. Furthermore, Chronicled[10] is a company that is using the IoT and blockchain to provide digital identity to physical products, while Riddle and Code[11] offers its own hardware and software stack to provide any physical object with a unique tamper-proof identity. These solutions are independent of tokens, certificates or IP addressing and instead rely on blockchain addressing that has tamper-proof logging for every interaction a specific address is involved with.

3.4.4 IoT Data Management

Research challenges in IoT remain open for storing and handling data produced by smart objects which surpass the human population. Recent research efforts have attempted to develop frameworks and mechanisms to manage the sheer volume of data generated in the IoT.

Data management in the IoT involves online data aggregation while providing event logs, auditing and storage, for offline query-processing and data analysis. Thus data management systems are required to have live dual operations in communication as well as storage. Any data management system for the IoT should be able to abstract complex semantics for high-level IoT applications, since unprocessed IoT data faces non-uniformity and weak semantics [134]. In many IoT architectures, semantic processing for data

[9] https://uniquid.com/
[10] https://chronicled.com/
[11] https://www.riddleandcode.com

is done via middleware, a layer considered between network and application layer [135]. In addition to this, many IoT application domains are time-critical, therefore processing IoT data in a timely manner is important while considering the constrained capabilities of IoT devices.

Traditional data management solutions generally follow a design trend where IoT data is handled in centralized architectures. Some proposed solutions suggest a partially decentralized architecture by using either clusters of distributed database services [136], or by using sub-servers to enable better scalability [137, 138]. These solutions do address the bottleneck of centralized data management systems, however, they do not provide trustless data management for the IoT.

While latency and scalability remain an open challenge for data storage within blockchains, using blockchains to design data management frameworks for IoT has the benefits of globally enforced data integrity and a non-dependence on semantics for logging IoT data creation events. With distributed storage mechanisms like IPFS working alongside blockchains, the bulk of IoT data can be stored off-chain, while maintaining immutable logs and links to the data within the blockchain. Blockchain based solutions are envisioned to be at least partially distributed, where the IoT data of users is kept private and secure, without third-party handling for service provision.

Multiple works in recent research leverage blockchain features to improve data management for the IoT. [139] leverage the immutability and auditability of blockchain records, while storing collected data from drones using traditional cloud service. [110] leverage auditability of blockchain records to facilitate sharing of stored data without authorizing intermediaries. Azaria et al. [140] propose a framework for storing medical records, using blockchain solely for maintaining records and querying, while using existing IoT data storage mechanisms for hosting IoT data. The motivation for this approach is to keep data storage responsibilities off-chain, so as to not bloat the blockchain.

Similar solutions with off-chain storage hold the most promise towards a distributed data management mechanism for the IoT, since taking storage off-chain bypasses the scalability issues in blockchains. FairAccess is a multi-layered framework that focuses on privacy, reliability and integrity in its design as a blockchain-enabled IoT architecture. [105] [88] For storage, they aim to add a separate storage layer where and off-chain, decentralized storage medium enables IoT data storage. [64] is a multi-tiered architecture, where private blockchains use cloud based solutions for storing and retrieving blocks.

Enigma [107] also uses off-chain storage, and utilizes a network of nodes running a distributed-hash table for storing IoT data. The data is accessible via the blockchain, with access-control policies written into the blockchain. Similarly, approaches to data storage and decentralized access-control is outlined in [95] for IoT, and in [141] for Big Data.

The proposed architecture in [97] uses the IPFS distributed storage mechanism, with the hash of the stored files recorded in the blockchain. IPFS files are addressed using the hash of the file itself, so data integrity is ensured. Nugent et al. [142] use Ethereum's Swarm[12] protocol for storing records of clinical trials off-chain, while using smart contracts to store and retrieve stored files. The aim for this approach is to enable transparency in record-keeping for clinical trials.

Researchers at CSIRO Australia propose a data integrity service powered by blockchain [114]. The service provides querying to query integrity of IoT data stored in the cloud, without the need for a third party to perform any verification. Blockchains with cloud-based persistent data storage is also being leveraged to host IoT resources on edge hosts. [127] proposes an architecture that uses IBM Bluemix and a permissioned blockchain to allow IoT resource sharing, as well as tracking usage and resource authorization.

[12] https://swarm-guide.readthedocs.io

Part II

The Proposed Blockchain-Based IoT Framework

Chapter 4

Designing the Blockchain-Based IoT Framework

This chapter contains text taken from the following published works:

"IoT Data Privacy via Blockchains and IPFS."
In Proceedings of the 7th International Conference on the Internet of Things, p. 14. ACM, 2017.

"Enabling a Blockchain-Based IoT Edge."
IEEE Internet of Things Magazine 1, no. 2 (2018): 24-29.

"Blockchain-Based Traceability in Agri-Food Supply Chain Management: A Practical Implementation."
In 2018 IoT Vertical and Topical Summit on Agriculture - Tuscany, pp. 1-4. IEEE, 2018.

A capable IoT edge allows edge nodes to meet service demands by limiting bandwidth consumption and latency. While this model allows for rapid and flexible deployment of IoT applications, the edge computation nodes operate under varying third-party entities, thus making it difficult to ensure a uniform level of security throughout the IoT edge. Fig. 4.1 illustrates a three-layer representation of edge-computing based IoT. The intelligent gateways at the edge computing layer are much closer to the end users themselves, and are interconnected to perform end-to-end IoT communications with as low latency as possible. Such gateways can even perform data handling and analytics locally, without relying heavily on cloud servers [143]. Most IoT applications have requirements in terms of transmission, computation and storage. The edge computing layer provides faster response-times for IoT applications, whereas the cloud server provides higher computation power and larger storage; thus the IoT benefits from both edge and cloud servers.

The edge is built on varied technologies like peer-to-peer systems, network virtualization and wireless networking. Virtualized container-based functionality within the IoT edge has proven to be a huge step forwards in IoT-edge application development, interoperability and adoption [144]. As containerized IoT applications become more prevalent, real-time IoT services and analytics are drawing nearer to the edge, without reliance on centralized cloud-based orchestration. As the IoT grows, the IoT and edge computing will become inseparable. In the adoption of edge-computing IoT, the most important issues that demand consideration are security and privacy.

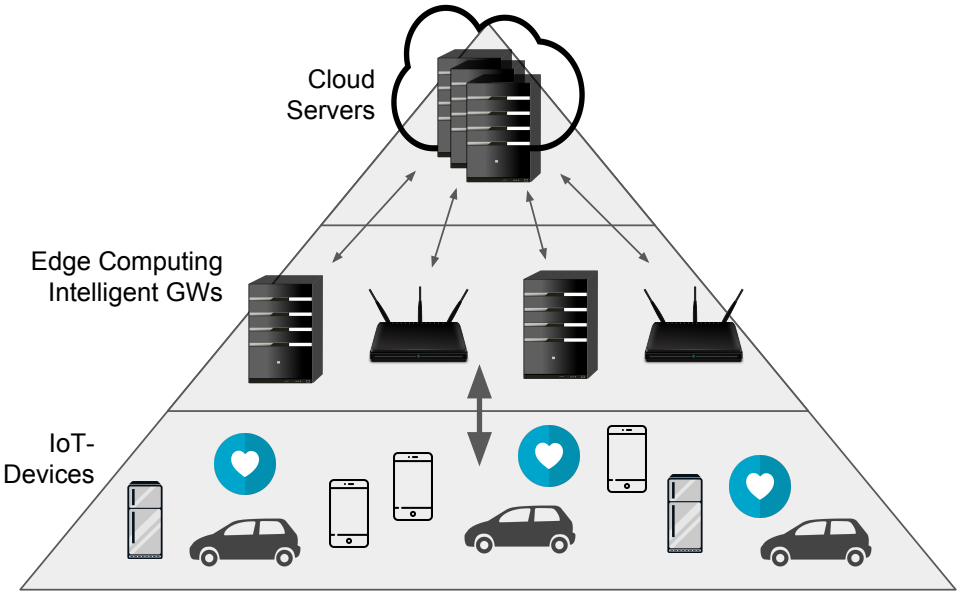


FIGURE 4.1: Traditional centralized architecture of edge computing IoT.

Despite the lofty goals of decentralization, edge computing raises some unforeseen opportunities for malicious behaviour. One of the typical security related problems in edge computing is authenticating gateways on different levels. For example, in IoT energy management systems, adversaries can compromise smart meters by falsifying data and spoofing IP addresses, thus rendering such systems vulnerable. Equivalent decentralized security strategies are difficult to enforce, considering IoT edge nodes may be managed by different entities. To truly empower the IoT users, and in extension the IoT edge, blockchains offer a fundamental rethinking of how private-by-design solutions can be implemented over decentralized networks.

In light of the challenges surrounding the blockchain-IoT convergence, there are ample research opportunities towards developing a functional and scalable blockchain-based IoT edge. To truly reap the benefits of a blockchain-based IoT edge, our contribution is a horizontally scalable blockchain framework over the IoT edge. This approach involves multiple locally deployed blockchains for multiple IoT edge segments, industry verticals, or even federated networks. Horizontally scaling blockchains and blockchain solutions is paramount for blockchains to truly arrive onto the IoT edge. The idea is to have multiple blockchains, or multiple blockchain segments, maintain communication records and enforce service level agreements on the entire surface area of the IoT edge. These multiple blockchains consist of permissioned blockchains maintaining privately held records within its edge segment, and an overlay permissionless blockchain as a negotiation medium for blockchain segments to securely exchange data with one another.

4.1 Architecture

The proposed framework is built on a tiered architecture of public and private blockchains, and is based on a decentralized application stack of smart contracts for network control, and peer-to-peer data storage and delivery mechanisms [145]. In this section, we outline the decentralized technologies used in developing the proposed blockchain-based IoT framework, as well as the network architecture designed to mitigate the privacy and scalability concerns associated with the blockchain-IoT convergence. Blockchains, along with decentralized storage mechanisms used in a multi-tiered architecture forms the basis of the proposed framework for monetizing IoT data, and securely availing IoT services involving monetary transactions.

4.1.1 Core Components of the Blockchain-IoT Framework

The proposed blockchain-based IoT framework is built on a tiered architecture of public and private blockchains, and is based on a decentralized application stack of Ethereum smart contracts [17] for network control, IPFS[1] for IoT data storage and BigchainDB [146] as an immutable and auditable database. In this section, we will discuss the basic working principles of blockchains and the key elements of a blockchain-based decentralized application stack.

For the proposed framework, the basic components required are the blockchain platforms: Ethereum and Hyperledger Burrow, along with smart contracts for programming the blockchains, and decentralized file storage for hosting the application data.

Hyperledger Burrow[2] is a Linux Foundation project under the Hyperledger umbrella of blockchains. Hyperledger Burrow received its original contributions from the developers of the Monax[3] platform. Hyperledger Burrow inherits the high throughput by virtue of the Tendermint consensus engine, and is tailored for permissioned blockchain deployments. Furthermore, Hyperledger Burrow provides a modular blockchain client with smart contract programmability built into it as per the specifications of the Ethereum Virtual Machine (EVM).

Ethereum is a blockchain platform that enables developers to deploy their own permissionless blockchain deployments, or to create distributed applications that run on already deployed Ethereum blockchains. The design choice to use Ethereum platform was based on its *"programmability"* and by-design security features. Both Ethereum and Hyperledger Burrow interpret smart contract written in Solidity, and utilize the EVM for its smart contract functionality. The core elements of the Ethereum platform are briefly described in the following, while, for a more comprehensive introduction, the interested reader is referred to [17].

[1] www.ipfs.io
[2] https://www.hyperledger.org/projects/hyperledger-burrow
[3] https://www.monax.io

- **Ethereum Virtual Machine:** the core of the Ethereum platform is the so-called Ethereum Virtual Machine (EVM), which executes smart contract code over the blockchain network. A smart contract is simply a piece of code stored in the blockchain itself and able to enforce programmatic terms and conditions over transactions occurring in the network. Every node in the Ethereum network runs the EVM, therefore, the publicly deployed main Ethereum network is considered a "world computer". In our framework, for private-by-design IoT data transactioning, we use smart contracts to enable IoT users to decide when and how much data to share with entities of their choosing, in exchange for monetary incentives and/or services.
- **Transactions and Addressing:** in Ethereum, addressing is based on hashed public-private key pairs. Since the latter are not associated to real-life identities, Ethereum is able to guarantee *"pseudonymity"* to its users [13]. Information related to transactions occurring in Ethereum are broadcasted to all nodes. Transactions can be made for transferring Ether (the official Ethereum cryptocurrency, ETH in the following) or delivering message data from one address to another, or interacting with the Application Binary Interface (ABI) of functions written in deployed smart contracts.
- **Smart Contracts:** A smart contract is simply a piece of code stored in the blockchain itself and able to enforce programmatic terms and conditions over transactions occurring in the network. In our proposed framework, for private-by-design IoT data transactioning, we use smart contracts to enable IoT users to decide when and how much data to share with entities of their choosing, in exchange for monetary incentives and/or services.
- **Consensus Algorithm:** For our framework, we used proof-of-authority (PoA), a variation of PoW, which limits mining only to delegated block validators. In the framework, the compromise for scalability is that the block validation is not fully decentralized and open to the public, however, with a sufficiently large validator pool, the immutability of the blockchain contents will remain secure. For private blockchains, we used a variation of the voting-based Practical Byzantine Fault Tolerance (PBFT) consensus algorithm [41]. Under PBFT, new candidate blocks are validated through multiple rounds of voting among block validators, instead of expending computational effort. PBFT is computationally efficient, with a fault tolerance of $n \geq 3f + 1$, where n is the total number of validating nodes, and f is the number of compromised or faulty nodes. PBFT is only suitable for private deployments, where participants are known to one another. Furthermore, PBFT blockchains are only suitable for private deployments since the high network overhead generated severely limits the scalability of the validator pool.

For applications involving data storage and transfer, relying on a blockchain as-is is highly inefficient, since storing data onto the blockchain will bloat

it, and cause high network overheads [147]. Therefore, for distributed file storage, off-chain decentralized file storage mechanisms are an important addition to the decentralized application stack. Common examples of such storage systems are the BitTorrent protocol[4], IPFS and Provable[5] .

IPFS (Interplanetary File System) is a peer-to-peer network and protocol for storing and sharing data. Using IPFS is particularly advantageous in our case, since files stored on IPFS are content-addressed. In order to deliver a file from one point to another, one needs only to send the cache-friendly hash of the file in an Ethereum transaction, so as to let the other person retrieve it through IPFS. This delivery mechanism is better suited for sharing batches of IoT data, and since the hash of the IPFS files are what identify them, this decentralized data sharing mechanism has built-in guarantees for end-to-end data integrity.

BigchainDB is a distributed database with blockchain characteristics [146]. BigchainDB boasts scalability and immutable record-keeping, and in our framework, we use BigchainDB to complement the decentralized application stack of Ethereum and IPFS. BigchainDB is essential for the framework, to maintain data integrity in private blockchains. We maintain a database of the block hashes from private blockchains that contain users' personal IoT data in BigchainDB. The entries to the database serve as an immutable record of the private blockchain contents, and users can provide these records as auditable proof of the integrity of their private blockchain contents.

The Tor Network[6] is another essential piece of the puzzle, for decentralized data transfer. While IPFS and BigchainDB together provide an accountable decentralized stack for exchanging batches of data, many blockchain applications have inherent requirements for data streaming. In near-realtime streaming applications, which are common in the IoT, incurring transactions repeatedly and retrieving data in batches over IPFS proves to be counterproductive and greatly increases the involved latency. Using the public blockchains specifically as a means for providing decentralized record-keeping and access control, while employing messaging streams over Tor adds secure end-to-end communications to the repertoire of the proposed framework's functionalities. Chapter 6 contains a detailed discussion on the usage of Tor within the proposed framework for secure end-to-end data communications.

4.1.2 Tiered Blockchain Architecture

In networks generating data, a blockchain can be used to create an immutable log of all data generation and access events, while smart contracts can be used to program responses to certain events or to enforce access control policies [97]. Then, this log can be used for transparency and accountability.

[4] http://www.bittorrent.org/beps/bep_0003.html
[5] https://provable.xyz/
[6] https://www.torproject.org/

However, this poses unique challenges in the IoT domain, in the form of a privacy/scalability trade-off. First, while permissioned blockchains retain privacy of their contents by limiting access to permissioned nodes, the consensus mechanisms involved in permissioned blockchains severely limit their scalability. Second, while permissionless blockchains have an improved scalability in terms of the number of participants, its contents are available to the public, which can impede user privacy[147]. Besides scalability in terms of the number of participants, a single monolithic blockchain cannot scale up to meet the storage requirements of a blockchain-based IoT. The IoT generates an unprecedentedly large amount of data, with a sharply high frequency of data generation events. If all the data generated by the IoT were to be encrypted and stored onto the blockchain, the storage requirements for each full node on the IoT edge would explode.

To overcome these limitations, we have designed our framework upon hierarchical, multi-tiered blockchain architecture; whereby inter-communicating permissioned and permissionless blockchains aim to bring together various segments of the IoT edge, without overwhelming each blockchain network.

The edge-tier as seen in Fig. 4.2 is composed of a federation of logically separate blockchains, deployed across multiple IoT edge segments; with IoT edge computing nodes and gateways acting as blockchain participant nodes. Thus each IoT edge segment can have its own ledger, terms of engagement and access control policies. Various IoT industry verticals can be accounted for in separate edge-tier blockchain deployments. For example, an edge-tier blockchain for a singular industry vertical, like a smart factory, would be used to keep data confidential within the blockchain network. IoT devices in this scenario would need not participate in the consensus algorithm, but would only issue transactions to the blockchain. IoT applications with mobility requirements can have IoT devices use cellular networks to issue transactions to their respective edge-tier blockchains. Implementing transaction fees within edge-tier blockchains may not cost anything in terms of money, however, it can effectively secure edge-tier blockchains from flooding attacks in the event that a mobile IoT device is compromised.

The Edge-Tier Blockchain

The edge-tier blockchain deployments are permissioned blockchains, where resource-intensive consensus protocols like Proof-of-Work are not necessary. Any nodes from outside the blockchain network only gain access after receiving permission from its existing participants. Private blockchain consensus algorithms like PBFT have voting mechanisms that generate higher network overhead than public blockchain consensus algorithms like Proof-of-Work. While Ethereum, a public blockchain platform based on Proof-of-Work requires roughly 300kbps to run a full node, private blockchain platforms like

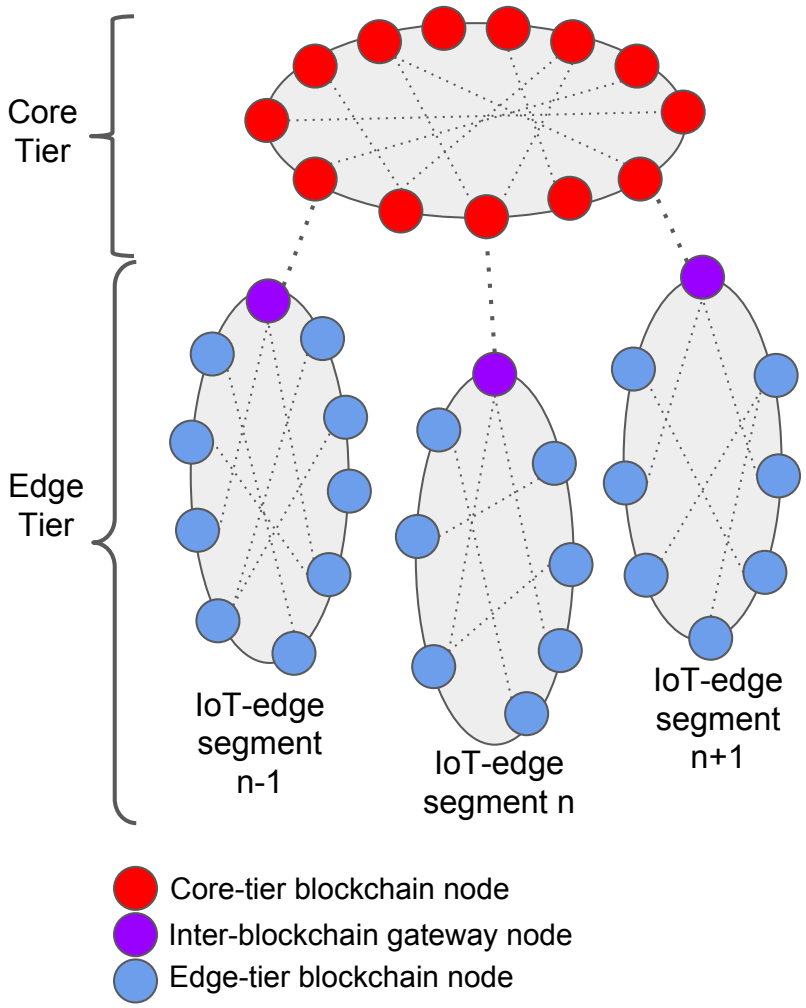


FIGURE 4.2: The two-tiered public/private blockchain architecture.

Tendermint have steadily increasing network traffic overheads with increasing number of nodes [10]. Therefore, private edge-tier networks require limits on member nodes, and need to be deployed in localised networks. Additionally, design considerations are to be made towards ensuring the integrity of the contents of edge-tier blockchains. Since private-permissioned blockchains do not achieve the same degree of decentralization and immutability as public-permissionless blockchains, it becomes necessary to be able to prove the contents of an edge-tier blockchains are unmodified. As a proof of edge-tier integrity, it is important to either use the core-tier blockchains to periodically store a hash of some recently added blocks in each edge-tier blockchain, or use a separately deployed public blockchain as a repository for hashes of all edge-tier blocks. In our framework, we use BigchainDB as a verifiable repository of edge-tier block hashes.

The IoT edge gateway devices participating in the edge-tier blockchain (with Linux operating systems including Ubuntu, Debian and Raspbian, each with support for Docker), integrate with IoT devices (sensors and actuators) and

register each data transfer and communication/actuation event into the edge-tier blockchain. In the edge-tier of this hierarchical architecture, specific nodes serve as gateways between the edge-tier and the core-tier blockchains. These inter-blockchain gateways are a building block to developing blockchain applications that span multiple edge-tier blockchains via the core-tier blockchain.

The Core-Tier Blockchain

The core-tier blockchain, as seen in Fig. 4.2, is deployed where the server/cloud side of a traditional architecture would be, however, the core-tier serves as a decentralized fabric for regulating the edge-tier itself, as well as the communications between multiple edge-tier blockchains. The core-tier blockchain can be used as an authentication and negotiation layer between two edge-tier blockchains, or even between an edge-tier blockchain and a node requesting access to the edge-tier blockchain. The core-tier blockchain is deployed publicly, in a "permissionless" fashion, such that any IoT edge segment or a member of the public could join the core-tier blockchain openly and without permission. The core-tier blockchain can employ consensus algorithms like Proof-of-Work, which are necessary to secure deployments with unknown participants. The core-tier blockchain consists of a network of nodes with a higher computational capabilities than the nodes at the edge-tier. Alternative permissionless blockchain consensus algorithms like Proof-of-Stake can ensure secure transactioning at the core-tier, without requiring high computational complexity. However, keeping node churn in mind, Proof-of-Work may indeed be a better choice, since block validating nodes are not chosen at random, and nodes actively producing computational work may validate blocks. Any nodes that leave the core-tier blockchain can rejoin and synchronize with the updated blockchain, without effecting the blockchain availability since there is no single point of failure.

Towards enabling real-time communications between two edge-tier blockchains, the core-tier blockchain can be used as an authenticating medium, following which communications can commence between two inter-blockchain gateways through off-chain communication channels. Using the core-tier blockchain for real-time applications is not feasible due to the latency and transaction processing speeds of publicly deployed permissionless blockchains in general. The transaction processing speed of permissionless blockchains does not make them suitable to handle real-time communications in higher volumes. Therefore, the idea here is to not inflate the core-tier blockchain with end-to-end communications, and to use it effectively as a decentralized security mechanism for off-chain communication channels.

The scope of this work is not to address inter-blockchain communications on a protocol-level, however, blockchain applications developed on platforms like Ethereum and Hyperledger can interact at application-level. Applications that listen for specific events in one blockchain can trigger scripts that

issue transactions onto the other blockchain. Therefore, the vision of a horizontally scaling blockchain-edge architecture is an attainable goal, for providing decentralized security to multiple IoT industry verticals.

4.2 Entities Involved

In the tiered blockchain architecture of our proposed framework, the entities involved are: the IoT gateways that users own; and requester nodes interacting with a users' IoT gateway, the smart contracts involved and finally, the decentralized data storage and transfer mechanisms. The distant node can be someone simply buying a user's IoT data, or someone providing a monetary service, or someone authorized to remotely monitor a gateway's IoT data. Between the local IoT gateway and the requester node are the Ethereum smart contract that defines the digital terms and conditions for monetizing IoT data, and IPFS for privately storing and sharing IoT data. Transactions involving cryptocurrency and IoT data are made in ETH over the Ethereum public chain.

- **Local IoT Gateway:** the local IoT gateway is a node more computationally capable than typical IoT devices. In applications where batches of edge-tier IoT data is to be exchanged over the core-tier blockchain, it is responsible for maintaining a copy of the edge-tier blockchain that contains the time-stamped logs of sensor data coming from IoT devices that were registered through it. The local IoT gateway, acting as an "**Inter-blockchain Gateway (IBGW)**" is also connected to the core-tier blockchain. Upon receiving access requests, it reads specific access privileges written in the edge-tier blockchain, and uses IPFS to transfer selected blocks from the edge-tier blockchain to a requester node in an IPFS file. For improved robustness, edge-tier blockchains can have multiple IBGW's connecting it to the core-tier blockchain. In applications where streams of IoT data are to be exchanged, the local IoT gateway engages in negotiations over the core-tier blockchain and delivers the streaming data over Tor.
- **Requester Node:** it is a peer on the core-tier blockchain that either buys data from an edge-tier blockchain or provides monetary services based on the data sent to it by an IoT user. This may be another edge-tier blockchain's IBGW, requesting data from another edge-tier industrial vertical. The requester node retrieves IPFS files sent from an edge-tier blockchain's IBGW, provided that the terms and conditions written into the core-tier smart contract are fulfilled. The terms and conditions on sharing data are decided between the requester node and the intended IoT gateway. The terms and conditions include the price, duration, volume and nature of data to be shared. In applications involving data streams, the requester node and the intended IoT gateway will engage in data transfer over Tor.

- **Smart Contract:** a smart contract code lives in the blockchain as an autonomous entity, also fitted with its own blockchain address. The latter is then used by a blockchain node (either a local gateway or a distant node) to invoke specific function methods written in the smart contract itself. Hence, the function methods are a digital parallel to the terms and conditions written in a traditional monetary contract. Through the functions of the smart contracts, both on the core-tier and edge-tier blockchains, nodes on the core-tier blockchain can issue requests and negotiate the terms, and IoT users can set their access control policies.
- **Decentralized Storage and Transfer:** the decentralized storage in the blockchain-based application stack, including IPFS and BigchainDB act as a key component for sharing data from edge-tier blockchains and providing the requester node with a mechanism for verifying the integrity of the edge-tier blockchain contents. Since permissioned blockchains do not have absolute censorship resistance, the block hash records in BigchainDB serve as a means to verify the integrity of the edge-tier blocks being shared. Similarly, in applications involving streaming data, the decentralized transfer medium will be the end-to-end messaging service on Tor.

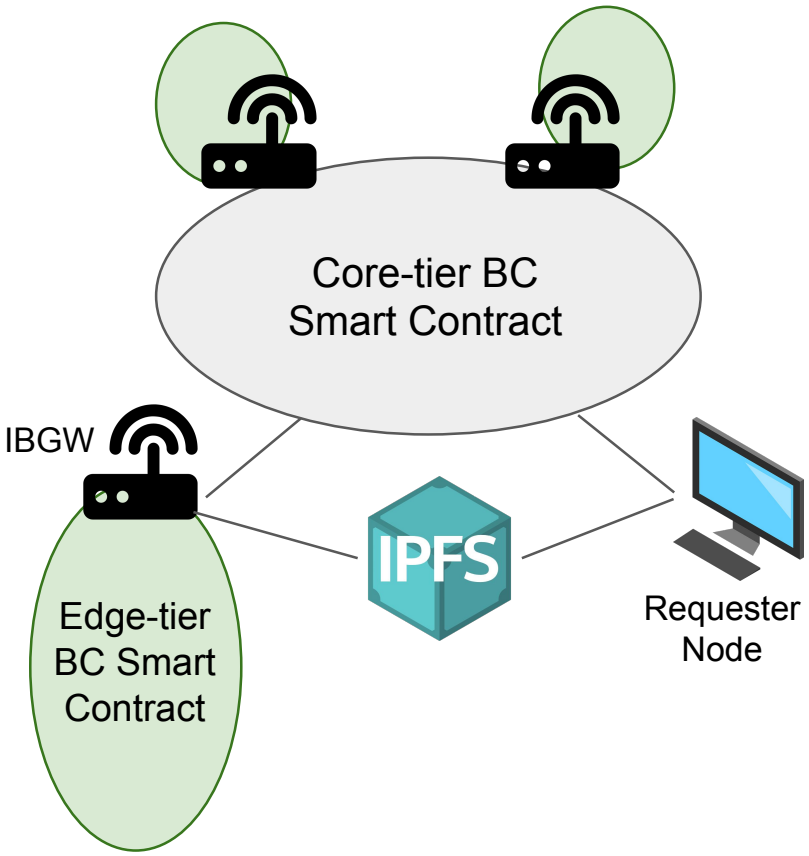


FIGURE 4.3: Entities involved in the architecture of the proposed framework.

Fig. 4.3 illustrates the entities involved in the proposed framework. Here, IPFS represents a part of the the decentralized storage and transfer stack, which can also include BigchainDB and Tor, depending upon the application.

4.3 Core-Tier and Edge-Tier Smart Contracts for Access Control

Smart contracts deployed on both the edge-tier and core-tier blockchains contain digitized terms and conditions for enforcing access control policies. How these smart contracts can be tailored specifically for different use-case scenarios are discussed in detail in Chapters 5 and 6.

IoT gateways interested in engaging with data requesters (either for data monetization or availing other data services) deploy smart contracts into the core-tier blockchain. The core-tier smart contract will be the point of contact and medium of negotiation for the requester and the intended IoT gateway. Each smart contract maintains records of requester nodes who are allowed access to the edge-tier data. These policies may allow open access to any requester willing to pay, or may have fine-grained policies on the basis of individual requesters.

To maintain requester records within the blockchain, our initial experiments involved using the array data structure within the Ethereum Virtual Machine. Simply using arrays proved to be problematic since appending objects to the array each time resulted in steadily increasing transaction fees. Therefore, the design decision to use the mapping data structure made more sense, since updates incurred a fixed transaction fee. All transaction fees were kept to a minimum since the algorithmic complexity of the functions in the smart contracts are kept at $O(1)$.

The following snippet shows how a *"Requester"* object with different attributes can be added onto the mapping structure within the smart contract. These attributes can include information on the level of access granted to the requester, or details about monetary requirements for granting access to a specific requester.

```
struct Requester {
  bool Attribute1;
  uint Attribute2;
  string Attribute3;
}

mapping (address => Requester) public Requestermap;
```

Objects such as these are maintained in the edge-tier blockchain as well, where IoT users can flexibly dictate their access control policies for each requester in particular. These fine-grained policies include the volume, time period, and nature of data that a requester is allowed to access. Requester objects on the edge-tier blockchain allow IoT users to dictate their terms for selective expression, and the requesters objects maintained in the core-tier smart contract protect requesters from issuing unauthorized payments for data access.

Requesters issue requests by invoking a function in the intended IBGW's core-tier smart contract. The core-tier smart contract can then pre-emptively prevent any payments from going through if the requester is not authorized to access the intended edge-tier blockchain's data. In case a payment does go through, the core-tier smart contract holds the payment till the agreements are met. If access is not granted, the smart contract returns the payment to the requester.

4.4 Consensus Algorithms for the Core-Tier and Edge-Tier

A major part of designing the framework was to decide which blockchain platform to use at each tier of the hierarchical architecture. Preliminary blockchain network deployments were used to study the performance metrics of various blockchain platforms. On the public blockchain tier, the clear decision was to employ the Ethereum platform, for its virtues in smart contract programmability and secure public deployments. The Ethereum cryptocurrency mainchain stands as proof of its immutability and security on the protocol level. The only weakness in a public Ethereum blockchain that an adversary can exploit are loopholes in a smart contract, therefore it is imperative to use best practices in scripting smart contracts.

For the framework to have a uniform smart contract language and interpretation, the first step was to consider the viability of Ethereum on the private blockchain tier. When deploying a custom Ethereum blockchain, a "difficulty" level must be set for the PoW consensus algorithm. A high enough difficulty level is necessary in Ethereum public deployments, to ensure that no single entity can amass significant enough computing power to launch a 51% attack. Within private blockchain networks, all peers are known, therefore there is no need for a high difficulty level.

A private Ethereum blockchain network was deployed over five core-i7 nodes, and the block processing time was observed for the first 1600 blocks. Among some preliminary results in understanding the real-world performance of Ethereum private networks, Fig. 4.4 shows very low block processing times obtained on a private Ethereum blockchain deployment. Since the difficulty level is set very low, the processing time is very fast. However, the mining process involved in Ethereum did result in a maximum CPU usage in all 5

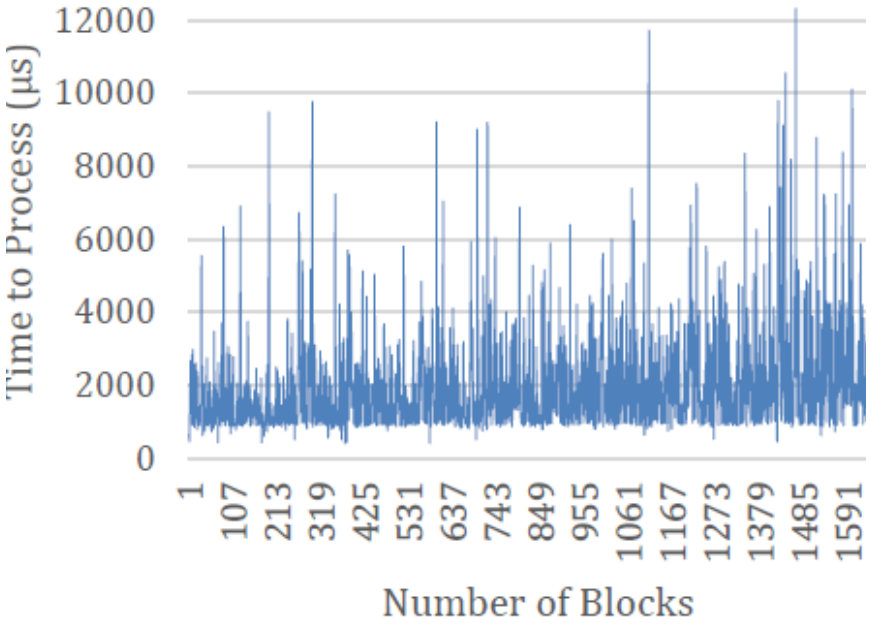


FIGURE 4.4: Block processing time in a privately deployed Ethereum blockchain.

nodes involved. With the constant usage of computational power, two major security concerns made Ethereum unsuited for the private blockchain tier.

Firstly, the blockchain is not truly permissioned, since anyone who knows the correct network ID can join the blockchain network, and only exist limited means to limit access to an Ethereum blockchain. The only means to prevent peers from mining on the Ethereum blockchain is to deploy a proof-of-authority (PoA) variant of the Ethereum blockchain, whereby only authorized nodes can mine on the blockchain and generate tokens.

On the other hand, among permissioned blockchains, the most suited consensus algorithms were PBFT, PoET, and the Tendermint consensus protocol. Sec. 4.6.1 discusses a use-case with PoET on the private blockchain tier, with Hyperledger Sawtooth, and Chapter 5 uses Hyperledger Burrow in its private edge-tier. The preference on Hyerledger Burrow is due to the fact that the smart contract scripting language is the same as Ethereum, therefore a uniform smart contract logic can be applied for both core-tier and edge-tier blockchains. Hyperledger Burrow uses Tendermint as its consensus engine, and Fig. 4.5, taken from [148], highlights the high transaction throughput of Tendermint consensus. It also shows its observably limited scalability in terms of number of validators it can accommodate. Therefore, Hyperledger Burrow blockchains are well suited for logging data generation and access events within an edge segment of the IoT.

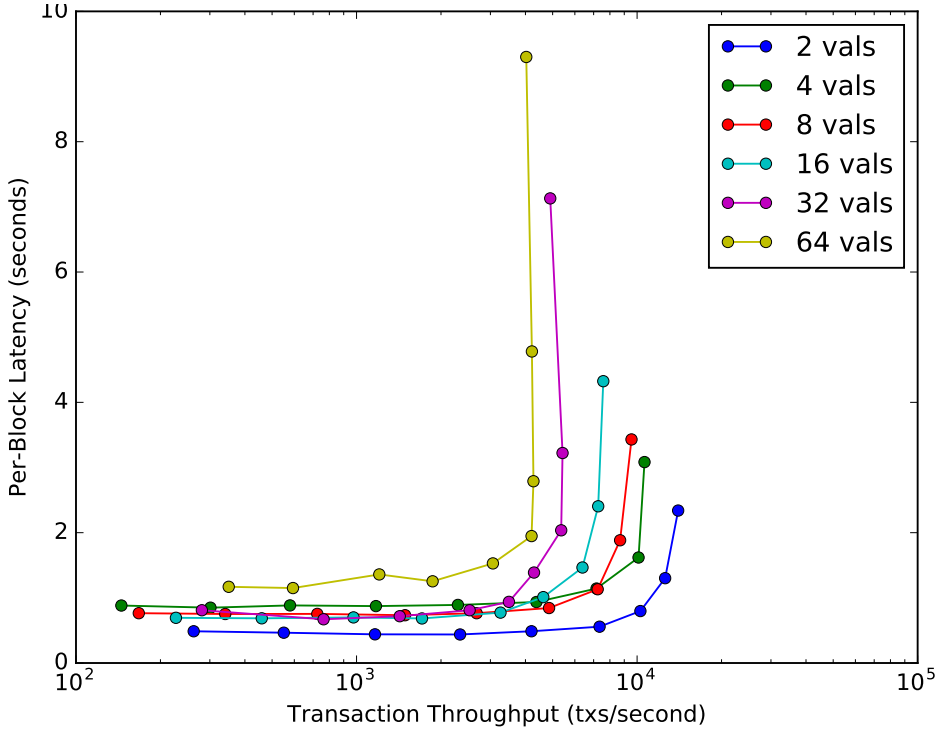


FIGURE 4.5: Despite high transaction throughput in Tendermint, there is an observable upper limit to the number of validators that can be accommodated [148].

4.5 Technical Challenges Addressed by Proposed Framework

This section outlines the technical challenges addressed through by-design features of the proposed blockchain-based IoT framework, as well as the design choices and trade-offs involved in its development.

4.5.1 Decentralized Access Control for IoT Resources

The proposed framework enables decentralized access control for IoT data. Tokenized approaches for blockchain-based access control allow any participant in the blockchain network to access digital assets in exchange for a predetermined amount of cryptocurrency tokens. In contrast, the access control technique proposed here allows users to decide whether they want their digital assets open to any paying member of the blockchain, or only allow restricted access to peers of their choosing. Empowering IoT users to exercise control over how much of their data is expressed to whom is a step in the direction of delivering IoT services with data privacy built in by design.

The access control mechanism is built on smart contracts in both the public and private tiers of blockchains. Users can choose to keep their data open to all paying blockchain peers, however, the smart contracts allow users to maintain records of all users who are allowed access to specific chunks of

IoT data. Within their edge-tier contract, IoT users can dictate policies on the extent of their selective expression of data to other parties.

The proposed framework relies on pseudonymous addressing, therefore, for added anonymity, users can utilize multiple addresses to reasonably obfuscate their identity within the core-tier blockchain network. Using multiple addresses effectively masks the identity of IoT users, without hindering the accountability features of the blockchain. In the event of an investigation, users can provide a trail of transactions signed off under all of the addresses they have chosen to use in the past.

4.5.2 Innovative IoT-Based Business Models

In developing a blockchain-based decentralized access model for the IoT, we open up room for innovation in developing innovative business models. Whereas earlier tokenized approaches form the basis for marketplaces for selling data, allowing fine-grained control of the data being exchanged in terms of money or services allows the blockchain-IoT framework to be adaptable for multiple business models. Chapter 5 demonstrates this idea, with two use-cases where smart contracts can be tailored to any business model that promises monetary services from IoT data without the users having to surrender any of their data that is irrelevant to the service being provided.

The design choice in using a separate layer for decentralized storage allows the business model to flexibly design the terms of engagement over the blockchain, without being able to access any of the IoT users' confidential data without their express permission. The smart contract within the private blockchain is where users can apply their permissions, and can decide how they engage with a particular business, while respecting their terms written in their smart contact instances deployed over the public blockchain.

4.5.3 Scalable Deployments for the IoT

Scalability is a significant challenge in the research space for integrating blockchains in the IoT. The tiered blockchain architecture allows users, or even organizations to maintain their private blockchains, with private records of their IoT devices, and engage with eachother over a public blockchain. One of the key contributions to scalability afforded by this architecture is in the split of responsibilities upon each tier. While private blockchains maintain records of data generation and internal interaction events, the public blockchain is solely responsible for effectuating and logging data interactions between two private blockchain owners. This split of responsibilities significantly reduces the overhead on the public blockchain, and makes the design of the architecture more edge-centric. Private blockchain owners can choose to join or leave the public blockchain network in a modular fashion.

4.5.4 Securing the IoT Edge

The gateway devices in the edge-tier blockchain network are responsible for registering IoT devices connected to them, and the edge-tier blockchain only allows devices registered as transaction issuers to log data to the private blockchain. Being registered simply as a transaction-issuer instead of a light client relaxes computational requirements of IoT devices in participating in the private blockchain. In the case of a private blockchain with compromised edge devices, a botnet attack becomes significantly difficult to launch because of the modular design of the hierarchical architecture. Any private blockchain network with compromised devices is safe from flooding attacks since transaction fees put a limit to how many incoming transactions are allowed from each device.

4.6 Use-Case: Smart Agriculture

4.6.1 Traceability in Agri-Food Supply Chains

For this use-case, we implemented an edge-tier blockchain as a fully decentralized traceability system for Agri-Food supply chain management [149]. Specifically, this use-case was implemented on the Hyperledger Sawtooth[7] platform. By directly producing and consuming valuable information from the IoT devices along the whole supply chain and storing such data directly in its underlying blockchain, this edge-tier use-case demonstrated transparent and auditable asset traceability. To assess the feasibility of the proposed solution, we engineered and deployed the so-called *from-farm-to-fork* agri-food use-case: a classical food traceability scenario fostering certified traceability of food along the whole supply chain, i.e., from agricultural production (the farm-side) to consumption (the fork-side). Then, we assessed the performance of the edge-tier solution through three performance metrics, namely latency, CPU load, and network usage.

The unique constrains and requirements of the modern Agri-Food industry pose some major challenges to achieve a transparent, auditable and reliable supply chain management process . Some of these challenges are the heterogeneity of the involved actors, stakeholders and business models, their different levels of confidentiality, the lack of interoperability among the involved systems and, most notably, the complete lack of a clear data governance [150]. Fig. 4.6 depicts a simplified version of such process, whose involved actors are briefly introduced in the following:

1. **provider:** providers of raw materials, such as seeds and nutrients, but also pesticides, chemicals, etc;
2. **producer:** usually the farmer, responsible of the actions from seeding/-planting to harvesting;

[7] https://www.hyperledger.org/projects/sawtooth

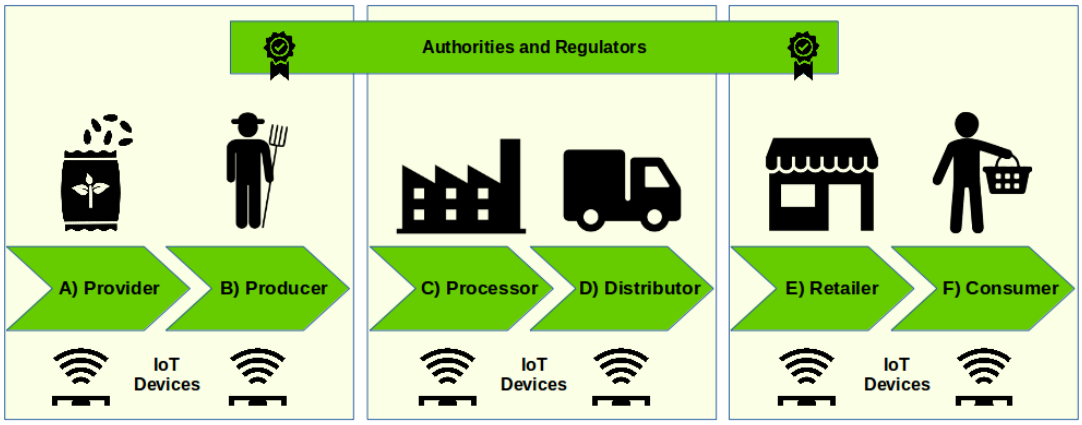


FIGURE 4.6: Simplified version of the Agri-Food supply chain management process.

3. **processor:** this actor may perform various actions, from simple packaging to more complex processes (e.g., pressing of the olives);
4. **distributor:** this actor is responsible of moving the output of the processor (e.g., the product) from processor's site to retailers;
5. **retailer:** this actor is responsible of selling the products, representing it either small local stores or big supermarkets;
6. **consumer:** the final element of the chain.

Along the whole process, authorities provide standards, regulations, laws, rules and policies that the involved actors have to comply with.

We propose an edge-tier blockchain solution (shown in Fig. 4.7) that provides a *"trustless"* environment for storing immutable supply chain logs whereas a centralized solution would have relied on cloud storage.

The proposed edge-tier solution uses gateways and mini-PC boards as full nodes, hence extending the resistance, decentralization, security and trust of the whole network. The main modules implemented in this solution are:

- **API:** a REST Application Programming Interface exposing the capabilities of the edge-tier blockchain to other applications, with a high level of abstraction, allowing easy integration with existing software systems;
- **Controller:** a component responsible of transforming the high-level function calls into the corresponding low-level calls for the blockchain layer, and viceversa (i.e., querying and converting the data records stored in the blockchain, into high-level information for the upper layer).
- **Blockchain:** The main component of the system, containing all the business logic, implemented through smart-contracts on the blockchain, as a gateway to the blockchain itself. Depending on the selected blockchain, this module will vary in complexity, according to the program capabilities of the selected blockchain, as well as the capabilities of the client interfaces for that blockchain.

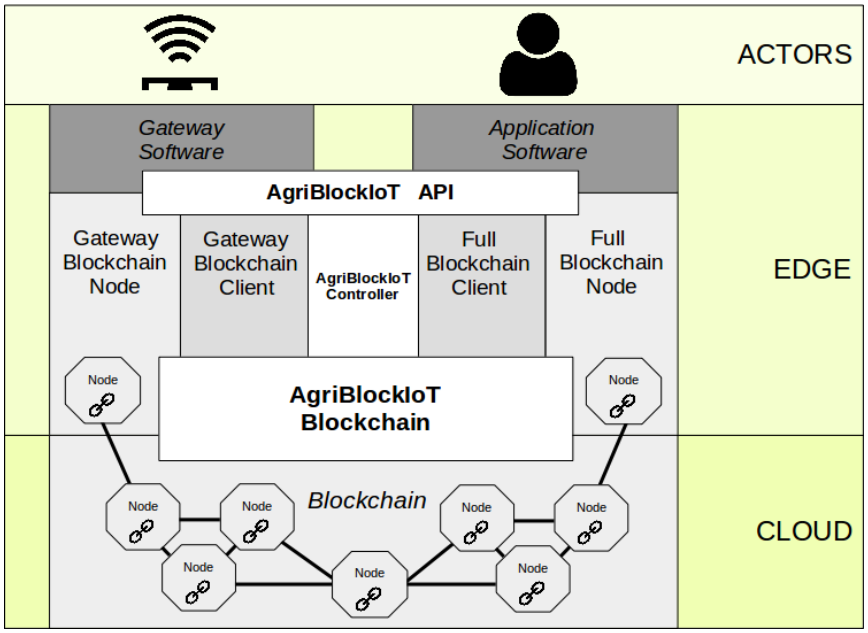


FIGURE 4.7: Architecture of the edge-tier solution. Here, the blockchain itself is backbone of the entire solution instead of a cloud storage.

Then, to coherently define the high-level functionality of our edge-tier solution, we had a bottom-up approach through which we extracted the set of requirements starting from a complete use-case, namely *from-farm-to-fork*. The latter is, indeed, a classical food traceability use-case that fosters certified traceability of food along the whole supply chain, from agricultural production to consumption. In other words, the edge-tier blockchain shall provide consumers with complete history of the food they are buying. The only precondition is that all the participants (including the IoT devices) are registered users of the edge-tier blockchain, meaning that they have the correct public/private key-pairs to digitally sign each operation on the distributed ledger.

We assessed the performance of our edge-tier Agri-Food supply chain solution implementing the functionality of an IoT sensing device producing digital values that are directly stored in the blockchain. The stored data can be then retrieved, while it is possible to implement smart-contracts that are autonomously executed upon the occurrence of certain conditions on the data produced by the sensor itself. Since the solution is blockchain-agnostic, we implemented the underlying blockchain module over two different, private, six-nodes-based implementations, namely Ethereum and Hyperledger Sawtooth to quantify the performance of both platforms.

Both blockchain networks were configured with their default settings, and deployed in dedicated virtual machines equipped with 4GB of RAM, 2 Intel(R) Core(TM) i5-6440HQ CPUs 2.60GHz and 20GB of hard disk. We opted for a Linux Ubuntu 16-04 basic distribution for the OS, only installing the packages needed to deploy the corresponding blockchain node. A series of

TABLE 4.1: Performance in terms of latency, network traffic, and CPU load.

	latency [seconds]	**network tx [bytes]**	**network rx [bytes]**	**CPU load [%]**
Ethereum	16.55	528'108	682'415	46.78
Sawtooth	0.021	19'303	20'641	6.75

100 tests where run independently for each scenario. During each test, the application simply set the value of a sensor, as done by an environmental IoT sensing device through a gateway, and issued a transaction in the blockchain. For each test we measured the time necessary to set the value in the blockchain (latency), the processing power of each node (CPU load), and the network usage (in terms of bytes transmitted and received); the average values are summarized in Table 4.1. From these results, we observe that Hyperledger Sawtooth ultimately has better performances with respect to the Ethereum counterpart.

4.6.2 Sustainability in Agricultural Groundwater Irrigation

As an extension to the agricultural edge-tier blockchain solution, an use-case over the core-tier blockchain can help mitigate selfish groundwater pumping among different farms.

Gamification comes part and parcel to the success of public blockchain networks, as is the case with the largest cryptocurrency networks, Bitcoin and Ethereum. Game theory solutions differ from conventional system optimization, since game theory takes into account the behaviours of the involved individuals and their self-interests when optimizing the outcome of the solution. In real world scenarios, results obtained from engaging participants are not always optimum as a whole for the system, since each participant makes economic decisions based pn individual information and criteria. Assuming a multi-participant solution, conventional cost or utility optimization is insufficient, since there are more human variables involved. PoW based blockchains incentivize participants to validate (or *"mine"*) new blocks on the blockchain with monetary rewards. Adding this game theoretical incentive is what contributes to the public decentralization and robustness of the Bitcoin and Ethereum networks.

In agricultural scenarios involving groundwater irrigation, regulations and costly metering solutions exist to limit *overpumping* [151], which can lead to devastating environmental and socioeconomic effects [152]. With the IoT enriching the agricultural sector, a blockchain-based IoT solution can be beneficial towards Fig. 4.8(a) shows the payoff matrix for a groundwater pumping scenario with two farmers in a Prisoner's Dilemma structure. The two farmers share an aquifer over a period of 25 years. The payoff corresponds to the revenue each farmer receives from crop sales. Each farmer must choose

		Farmer 2	
		PR1	PR2
Farmer 1	PR1	3,3	1,4
	PR2	4,1	2,2

(a) Ordinal payoffs.

		Farmer 2	
		PR1	PR2
Farmer 1	PR1	3,3	4,1
	PR2	1,4	2,2

(b) Payoffs with regulations.

FIGURE 4.8: Groundwater pumping game without cooperation regulation and with cooperation regulation.

between the cooperative pumping rate (PR1), or the higher, non-cooperative pumping rate (PR2). If farmers both pump at the cooperative rate, the groundwater levels do not drop and they can enjoy long-term groundwater irrigation with a high payoff. However, if they both pump at PR1, the groundwater level will drop, resulting in low long-term payoffs. However, for either farmer, the highest payoff comes from the scenario where he pumps at the higher rate and the other pumps at a cooperative rate. Since each farmer finds PR2 to be a dominant strategy, a (PR2, PR2) outcome is more likely based on a non-cooperative game [153], and is in fact typical in non-regulated groundwater aquifers.

The lack of trust in a groundwater aquifer system leads farmers to pump at the higher rate, leading to lower long-term payoffs, which is termed a "tragedy of the commons" [154]. The problem is compounded further with more farms pumping.

However, the scenario changes when penalties are imposed upon non-cooperative farmers. If, let's say the farmers who exceed the agreed upon pumping rate lose their pumping privileges, the groundwater pumping game changes to what is shown in Fig. 4.8(b). Here, the payoffs for non-cooperative pumping are greatly reduced based on the new terms and conditions (and their enforcement). In this new game, the dominant strategy is cooperation, and the Pareto-optimal (PR1, PR1) outcome is likely.

Fig. 4.9 visualizes the agricultural use-case as applied within the multi-tier blockchain architecture. Each edge-tier blockchain represents an agri-food supply chain, which additionally contains groundwater pumping sensor data. For accountability, the groundwater pumping data is uploaded to the core-tier blockchain through the groundwater pumping smart contract. In the event where a farmer has overpumped the groundwater, all participants in the core-tier blockchain can observe it, and can swiftly enact penalties upon the non-cooperative farmer. Besides the immutable groundwater pumping records present in the core-tier blockchain, further accountability can be added

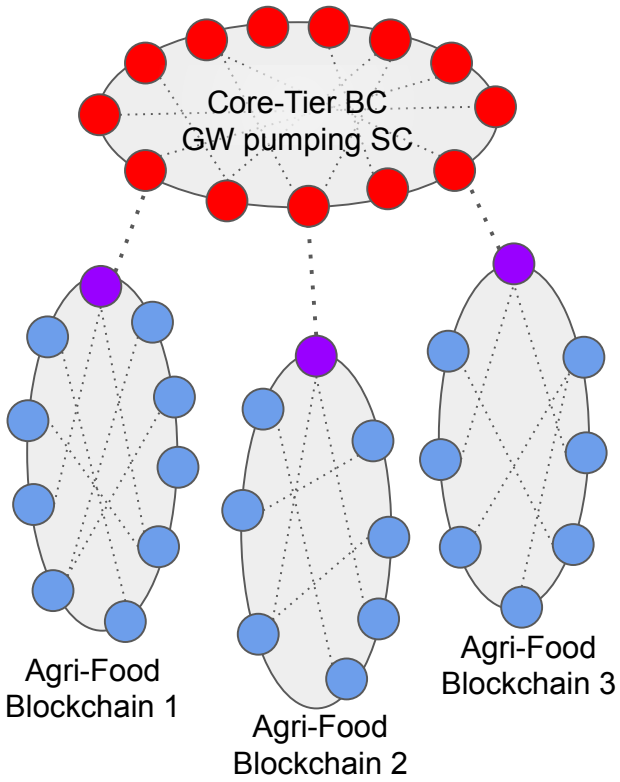


FIGURE 4.9: The multi-tiered blockchain architecture, with the core-tier blockchain for cooperative groundwater pumping.

by issuing access request to specific edge-tier blockchains to monitor groundwater pumping events. Thus, the core-tier blockchain eliminates the *need* for trust, and enables *"trustlesss"* and decentralized groundwater pumping with cooperative pumping rates.

4.7 Summary

To summarize, this chapter has discussed the design and working principles of the proposed blockchain-based IoT framework. The framework is built a horizontally scalable, multi-tiered blockchain architecture, with a decentralized stack of blockchains and other decentralized data storage solutions. While scalability on a protocol level remain an open challenge, this framework presents an architectural solution for scaling up blockchain-based security. The utilization of both permissioned and permissionless blockchains allow IoT edge users to selectively express their IoT data to third parties over a secure fabric negotiation and monetization. Smart contracts within the permissioned and permissionless blockchains work together to provide fine-grained acccss control to IoT users. These smart contracts can be flexibly modified to meet newer, innovative business models. Also discussed is a use-case scenario within the agricultural sector, where permissioned blockchains are used as a decentralized traceability system for agri-food supply

chains, and both the permissionless and permissioned blockchains work together to provide a decentralized solution for cooperative groundwater irrigation. This use-case demonstrates not only the capability of the framework to provide decentralized accountability, it also gives an overview of the access control policies that can be enforced in the framework. The following chapter discusses in great detail how these access control policies and smart contracts can be tailored to specific e-business models.

Chapter 5

Decentralized IoT Data Transfer And Monetization Services

This chapter contains text taken from the published works:

"IoT Data Privacy via Blockchains and IPFS."
In Proceedings of the 7th International Conference on the Internet of Things, p. 14. ACM, 2017.

The proposed blockchain-based framework is first and foremost an application-agnostic private-by-design solution for performing transactions involving IoT data and cryptocurrency, with fine-grained access control policies. In this section, we will outline the decentralized access control mechanism involved in the framework. We will use two use-cases to discuss the usage of the proposed framework in real-world scenarios. These use-cases further highlight how specifics of the edge-tier and core-tier smart contracts can be tailored for specific business models and IoT applications.

Typically, tokenized approaches for blockchain-based access control allow any participant in the blockchain network to access digital assets in exchange for a predetermined amount of cryptocurrency tokens [105]. In contrast, the access control technique presented in this paper empowers IoT users to decide whether they want their digital assets open to any public paying member of the blockchain network, or whether they only want to allow restricted access to requesters of their choosing.

The design trade-off in designing blockchain-based access control is between absolute anonymity and public accountability. The decentralized access control mechanism proposed here is privacy-preserving to the extent where users are in full control of how much of their data is expressed to third-party entities. The smart contracts proposed do not inherently anonymize users on the blockchain. The proposed framework relies on pseudonymous addressing, therefore, for added anonymity, users can utilize multiple addresses to reasonably obfuscate their identity within the blockchain network. Using multiple addresses effectively masks the identity of IoT users, without hindering the accountability features of the blockchain. In the event of an investigation, users can provide a trail of transactions signed off under all of the addresses they have chosen to use in the past. Here, let $Y = \{y_0, y_1, \ldots, y_n\}$ be the requester nodes, such that each requester node y_r has a public/private

keypair $\{k_p^r, k_s^r\}$. Additionally, let $C = \{c_0, c_1, \ldots, c_n\}$ be the set of edge-tier blockchains, and $X = \{x_0, x_1, \ldots, x_m\}$ be the set of IBGW nodes, such that:

$$\forall c_i \exists X^i \subset X : (c_i \ni X^i) \tag{5.1}$$

where $\ni$ denotes ownership. The IBGWs owned by a specific edge-tier blockchain c_i are the points of contact over the core-tier blockchain, between requester nodes and the edge-tier blockchain. A requester y_r would conduct transactions with c_i through x_j such that $x_j \in X^i$.

$$\forall x_j \exists s_j^r = \langle y_r, x_j \rangle : (r, j \geq 0) \wedge (r \leq p) \wedge (j \leq m) \tag{5.2}$$

Upon receiving access request from y_r, or to obtain remuneration or monetary services from y_r, an IBGW x_j is made responsible for performing data delivery through IPFS. Firstly, x_j queries access privileges written in the edge-tier smart contract. If access privileges are given, or requirements for access privileges are met, x_j will query blocks from the requested time interval stored in the edge-tier blockchain. Then, x_j adds these blocks to an IPFS file, and encrypts the IPFS file hash with y_r's public key.

$$\forall y_r \exists \langle k_p^r, k_s^r \rangle : d_{k_s^r}(e_{k_p^r}(h_t)) = h_t \tag{5.3}$$

$$h_t = IPFS(\varrho^j) \tag{5.4}$$

where ϱ^j denotes the requested blocks of the edge-tier blockchain, and h_t is the hash value returned by IPFS for transaction t. Access policies can be made further fine-grained by allowing access to specific edge-tier transactions. In this case, in addition to the edge-tier block hashes, the edge-tier blockchain owners will need to upload the private block headers (containing each block's Merkle Tree) to BigChainDB, so requesters can verify each edge-tier transaction being sent.

The following is a discussion on how the aforementioned access control and data delivery mechanism is applied in two distinct use-case scenarios. The smart contracts on both the edge-tier and core-tier are programmed to tailor the framework to these use-cases.

5.1 Decentralized IoT Data Marketplace

To illustrate how the proposed framework would be used in a real world scenario, we first draw the use case of a decentralized IoT data marketplace. The objective is to allow users to monetize (i.e., to sell) their IoT data and engage with anonymous buyers in a *"trustless"* way, without the need of intermediary parties for authentication and authorization. Besides giving IoT users the flexibility to exercise control when transacting with anonymous buyers,

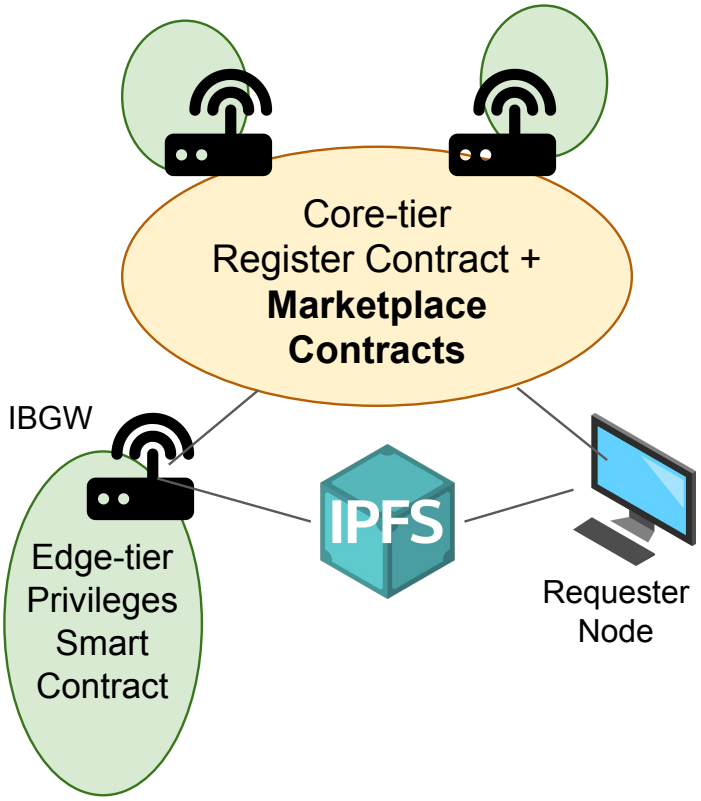


FIGURE 5.1: Components of the proposed framework, as they are used in the IoT data marketplace use case. Here, IBGW represents the seller, while the requester is the buyer.

it gives buyers a way to ensure the data they are receiving has not been doctored or tampered with. Along with providing verifiability to the buyers, the architecture can also be used to put a decentralized reputation system in place which is a blockchain-based parallel to online marketplaces of physical goods. The distributed reputation system aims to incentivize trade with reputed sellers, and discourage creating multiple seller accounts or selling fraudulent data.

On the public core-tier blockchain, the register contract is where edge-tier sellers can advertise their offerings. Through their IBGWs, edge-tier owners can deploy marketplace contracts at the core-tier, where requesters can invoke functions to interact with the IBGW. On each edge-tier blockchain, the privileges contract enable users to either let the entire contents of the private blockchain to be openly available to any paying requester, or to define fine-grained privileges and service level agreements for specific requesters. Fig. 5.1 shows the components, as they are used in this use case, while in the following we describe the two contracts in more details.

5.1.1 Edge-tier Privileges Smart Contract

The private edge-tier blockchain not only helps in keeping IoT data private within federations, or smart-city communities, they also allow users to exercise fine-grained control over whom they want to share their data with, and at what price. For edge-tier blockchains deployed over consortia with sufficiently large validator pools, the edge-tier blockchain itself provides assurity of data integrity. For smaller edge-tier deployments, the hash of each block gets stored into BigChainDB, which serves as a verifiable public repository which can be looked up to verify the integrity of the data. The privileges contract has three functions. The OPENFORALL() function sets a flag which indicates whether the edge-tier blockchain has agreed upon letting the contents of their blockchain become available to any paying requester on the core-tier blockchain. The SETPRIVILEGES() function allows participants within an edge-tier blockchain to define access privileges for specific core-tier addresses in a fine-grained fashion. The IBGW looks up these privileges and responds to incoming requests accordingly. In case a requester has no access privileges, and the OpenForAll flag is not TRUE, the IBGW does not respond and the request times out at the requester's end.

5.1.2 Core-Tier Register Contract

The register contract is where IoT data buyers go to choose which seller they want to interact with. Edge-tier owners willing to monetize their data can store their marketplace contract's addresses in the register contract as a form of advertisement, through the REGISTERMARKETPLACE() function. Here, $Y = \{y_0, y_1, \ldots, y_n\}$ are the requester nodes, and $X = \{x_0, x_1, \ldots, x_m\}$ are the IBGWs with deployed marketplace smart contracts on the core-tier blockchain. A requester y_r can invoke the VIEWSELLERS() function, look up the listings on the register contract and engage with the marketplace contract of x_j. In case the owner entities of an edge-tier blockchain turn off their OPENFORALL flag, the IBGW for that edge-tier blockchain will issue a transaction to update advertisment details on the register contract, to timely inform potential requesters. Table 5.2 is a list of the functions written in the register contract, along with their input and output parameters.

The register contract also serves the purpose of maintaining a record of decentralized reputation for the sellers. Legitimate buyers use their transaction IDs to give sellers a rating based on the data they buy. The reputation system discourages sellers to register with new accounts, since sellers with higher ratings will be favoured by buyers. To finalize each data transaction, requesters will leave a rating through the REVIEWSELLER() function.

TABLE 5.1: Functions in the Edge-Tier Privileges Contract.

Functions	Input Parameters	Output Parameters
OPENFORALL()	BOOL open, UINT price, STRING metadata	VOID
SETPRIVILEGES()	ADDRESS requester, STRING metadata	VOID

TABLE 5.2: Functions written in the Register Contract.

Functions	Input Parameters	Output Parameters
REGISTERMARKETPLACE()	ADDRESS contract, STRING metadata	VOID
REVIEWSELLER()	ADDRESS contract, STRING txID, UINT rating	VOID
VIEWSELLERS()	VOID	ARRAY sellers

TABLE 5.3: Functions written in the Marketplace Contract.

Functions	Input Parameters	Output Parameters
CHECKPRICE()	INT hours	INT price
REQUESTDATA()	STRING from, STRING to, MSG.VALUE price	VOID
SENDDATA()	STRING IPFShash	VOID
REQUESTPENDING()	VOID	BOOL pending, ADDRESS buyer
BANREQUESTER()	ADDRESS customer	VOID

5.1.3 Core-Tier Marketplace Contracts

Each seller entity has their own marketplace smart contract deployed on the core-tier blockchain, which is advertised in the register contract. Buyers engage with seller entities through their specific marketplace contracts. Table 5.3 contains details on the functions written in each of the marketplace smart contracts.

- **Requesting Data:** in a scenario where a requester y_r looks up a marketplace contract address from the register contract, y_r will invoke the REQUESTDATA() function written in the marketplace contract. If y_r is requesting data from x_j for the first time, y_r's blockchain address is registered within x_j's marketplace contract. Thus, owners of the edge-tier blockchain can maintain records of the number of sales conducted with y_r. Through input arguments of the REQUESTDATA() function, y_r can

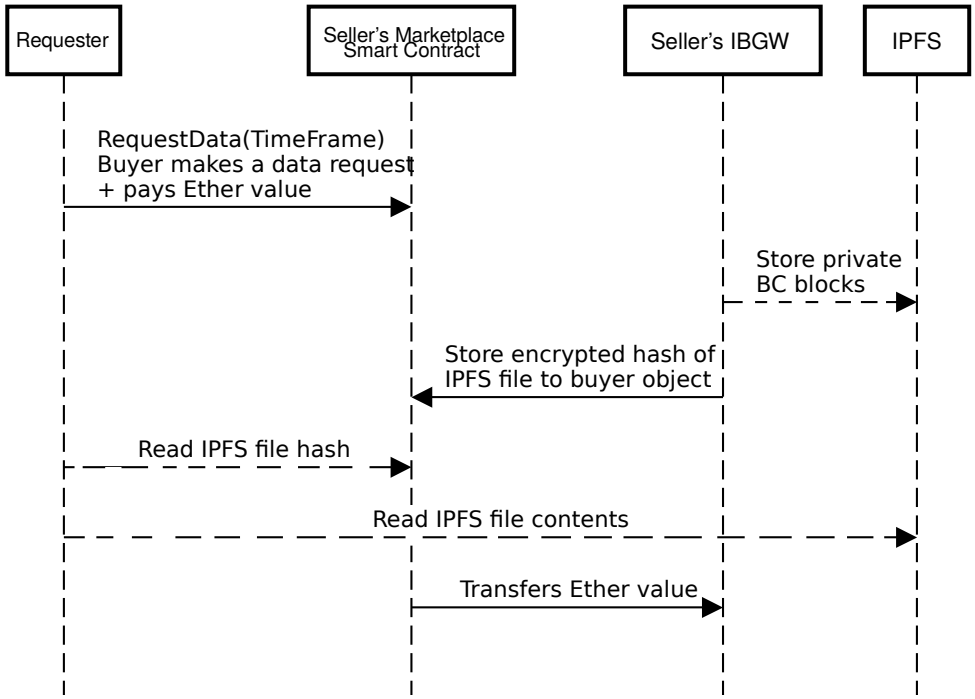


FIGURE 5.2: Sequence diagram of a buyer requesting data and a seller fulfilling the request. All interactions with the smart contract represent transactions. Dashed arrows represent interactions that are instantaneous and free of charge.

specify time intervals for which they want to request IoT data. Last, but not least, while issuing an IoT data request, y_r includes the ETH value for the requested data within the 'value' field of the method invocation.

- **Responding to Data Requests:** in the event of an incoming data request from y_r, the IBGW checks access privileges written in the edge-tier privileges smart contract to see if privileges are given and the payment is correct. If so, the IBGW invokes the SENDDATA() method in the contract and adds the encrypted IPFS file hash to y_r's customer object in the x_j smart contract, so y_r can retrieve it and decrypt it. This hash will allow y_r to access the requested data. Fig. 5.2 shows the sequence diagram of an interaction between y_r and x_j.
- **Other Functions:** other functions written in the contract are the ones that supplement the data transactioning procedure. The CHECKPRICE() function is a free-of-charge read-only function that Bob can invoke to inquire the price of the data being sold over a specific period of time, while the BANREQUESTER() function allows Alice to ban certain buyers, hence preventing them from making further requests for data.

5.2 Blockchain-Based Connected Vehicle Insurance

As connected vehicle technology continues to evolve, telematics-based insurance is gaining momentum, promising it to provide insurances based on data collected from connected vehicles [155]. In usage-based insurance (UBI), the insurance company may analyze how well a driver uses a vehicle, adjusting premium payments accordingly. UBI has recently experienced substantial growth [156], and insurance companies are experimenting with various data collection techniques. In general, data collection through mobile phones proves to be less reliable and accurate, therefore most companies providing UBI opt for proprietary solutions, where a blackbox installed inside the vehicle collects data pertaining, for instance, to the actual GPS location, acceleration and overall usage. The data collected is then analyzed and used to assess customers' driving behaviours. This approach not only requires users to involuntarily surrender their personal driving data to third-party entities but, despite the existent, severe regulatory guidelines, there is absolutely no guarantee that such a blackbox collects only the strictly necessary data required for the purpose.

Considering the privacy implications of currently existing black-box solutions, there is a need for a private-by-design solution able to, not only give

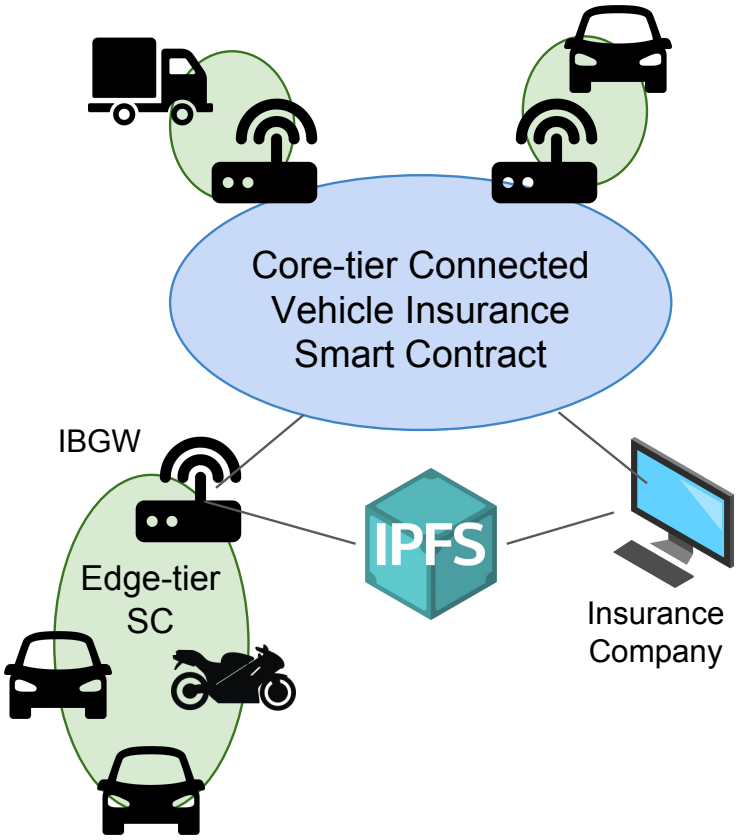


FIGURE 5.3: Components of the proposed framework, as they appear in the connected vehicle insurance use case.

users complete control over what and when they share with third-party entities, but also to facilitate fast insurance claims and premium payments. Our proposed framework is suited for this problem, since it facilitates selective expression of IoT data and ensures integrity of data sent to insurance companies. For the connected vehicle insurance use case, the architecture remains the same as before, such that $Y = \{y_0, y_1, \ldots, y_n\}$ are the insurance companies, and $X = \{x_0, x_1, \ldots, x_m\}$ are the IBGWs for edge-tier vehicle blockchain networks.

One important design consideration in the vehicle insurance use-case is that while a proof-of-work consensus may be beneficial to provide distributed security for the marketplace use-case, it would not be suitable for the vehicle insurance use-case. At the core-tier, it would be preferable to use a proof-of-authority blockchain with a significantly large validator pool, spanning multiple insurance companies. This is to prevent insurance customers from publicly validating blocks at the core-tier, since mining tokens will be a detriment to the business model. Fig. 5.3 shows the components of the framework.

Insurance companies deploy insurance smart contracts that customers can register their vehicles to. The flexibility afforded to the insurance company is that it can deploy multiple contracts in case it needs to offer various insurance premium packages. Customers can then choose which specific smart contract from an insurance company they want to interact with. In order to update insurance premiums or issuing insurance claims, edge-tier owners privately share IPFS files that contain edge-tier blocks, through the insurance smart contracts.

Table 5.4 contains details of the functions written in an insurance provider's smart contract. Together, the edge-tier and core-tier blockchains come into play while offering the services decribed as follows.

5.2.1 Edge-Tier Blockchains for Data Integrity

As discussed in Sec. 5.1, the privileges smart contract within the edge-tier blockchain is meant to set access privileges for third party data requesters. In this use-case, drivers will only want their data available to the insurance company when updating their insurance premium and making insurance claims. Here of course, since the edge-device is mobile and is uilizing mobile infrastructure for networking, the aim was to keep networking overheads at a bare minimum. Since there is only one entity inolved at the edge-tier, a bare-bones blockchain without consensus mechanisms is used, simply to locally store data in a structured manner. The privileges smart contract operates as described in Sec. 5.1, however, the main reason for structuring the data as in a blockchain, is to let the IBGW store block hashes onto BigChainDB. With BigChainDB's significantly large validator pool, hashes stored onto BigChainDB serve as a time-stamped proof of non-repudiation and integrity of the data stored locally. An insurance company can verify the integrity of the blocks a user sends to either process insurance claims or to update a user's insurance premium.

TABLE 5.4: Functions of the Connected-Vehicle Insurance Smart Contract

Functions	**Input Parameters**	**Output Parameters**
STARTCOVERAGE()	MSG.VALUE payment	VOID
SENDCLAIMDATA()	STRING IPFShash	VOID
SENDPREMIUMDATA()	STRING IPFShash	VOID
REQUESTPENDING()	ADDRESS customer	BOOL pending
ISINSURED()	ADDRESS customer	BOOL insured
CHECKPAYMENT()	ADDRESS customer	BOOL overdue
UPDATEPREMIUM()	ADDRESS customer, INT newPremium	VOID
MAKEPAYMENT()	MSG.VALUE payment	VOID
PROCESSCLAIM()	MSG.VALUE claim, ADDRESS customer	VOID

5.2.2 Functions at the Core-Tier Blockchain

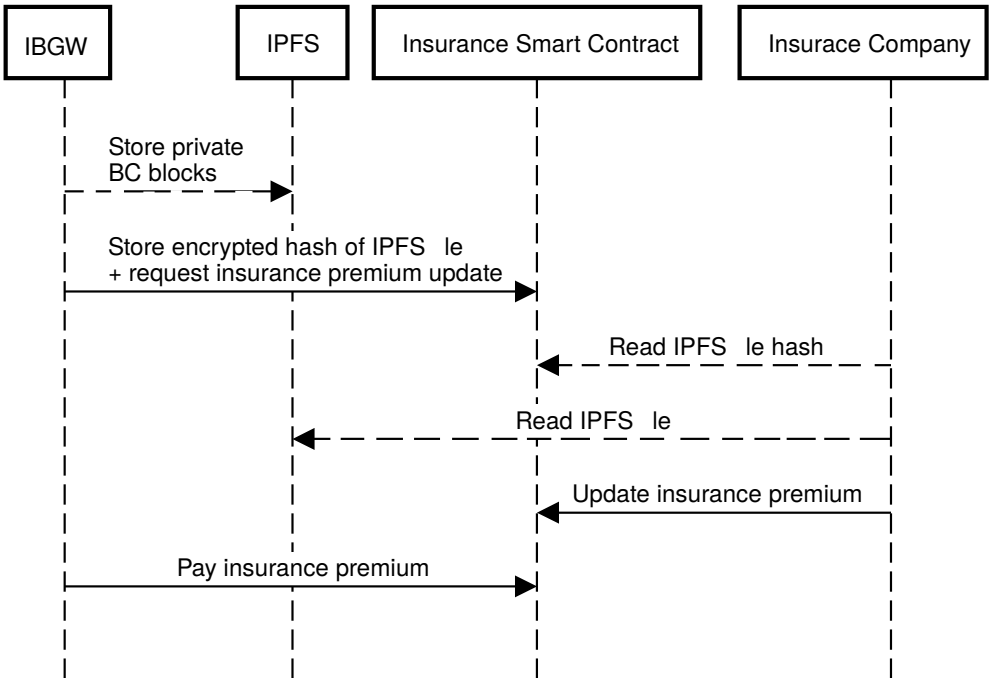
In this use-case, the core-tier blockchain will have methods to register new customers, to make insurance premiums, processing insurance claims, and more. For registration and activation of vehicle insurance coverage, a customer invokes the STARTCOVERAGE() method of the smart contract. The inputs arguments of this method are the customer's blockchain address, as well as an initial payment to start the insurance coverage. If a car-owner pays the correct amount as initial payment, he is registered as a new customer in the insurance company's smart contract. The customer object within the smart contract code comes with multiple attributes, including a string variable for storing the last hash of the IPFS file that a customer sent to the insurance company. Again, since values of stored variables are publicly accessible to all blockchain peer nodes, the IPFS file hash is encrypted with the insurance company's public address, so as to allow only the insurance company to obtain the decrypted hash.

- **Paying Insurance Premium:** usually, to encourage safe driving, connected vehicle insurances offer flexible premiums, where safe drivers have lower insurance premiums to pay. To deliver the same functionality in a decentralized private-by-design fashion, we wrote conditions in the MAKEPAYMENT() method for users to obtain an updated insurance premium value from the insurance company before making the payment. Fig. 5.4(a) shows the sequence of steps required for making an insurance premium payment. To obtain an updated insurance premium value, users select from their own private blockchain, blocks from a required time-frame to send to the insurance company. This could be, for instance, the last month, or the last year, but clearly it

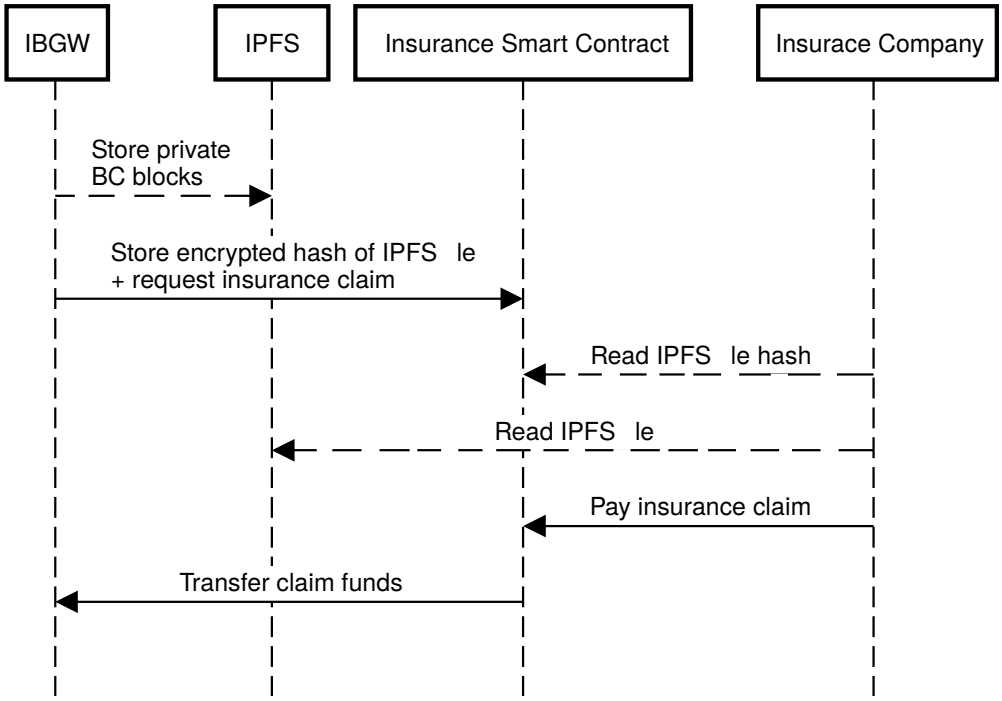
completely depends on the insurance policy. The selected blocks are added to IPFS, the file hash is encrypted and stored in the contract itself, using the SENDPREMIUMDATA() method. This time, while there is no ETH being transferred, transaction fees will be incurred, since the transaction is resulting in a state update within the blockchain. The insurance company can read the hash from the customer object within the smart contract and can then retrieve the associated IPFS file. After analyzing the contents of the file, the insurance company issues a new cost for the insurance premium, invoking the UPDATEPREMIUM() method. Upon retrieving this update, the customer can perform the appropriate payment.

- **Processing Insurance Claims:** applying for an insurance claim takes similar steps as making an insurance premium payment, as shows in Fig. 5.4(b). In the event of an accident, the customer adds blocks from its private blockchain collected during the time of the accident to IPFS. The customer applies for an insurance claim and stores the encrypted file hash into the contract, invoking the SENDCLAIMDATA() method. Then, the insurance company retrieves the IPFS file, analyzes the received data and eventually pays the adequate amount for the insurance claim to the customer. In the event of a customer not paying the insurance premium after a grace period following the due date, the insurance company has the option to deactivate the customer's insurance coverage, by setting the customer.PolicyActive attribute to 'false'.

- **Checking Payment Status:** finally, an important function is to enable insurance companies to check if customers are making timely payments for their insurance premiums. The CHECKPAYMENT() method written in the smart contract checks if the customer has not gone over the deadline for making an insurance premium payment. If the customer has indeed gone over the deadline, then the insurance company can deactivate the customer's insurance policy.



(a) Customer paying insurance premium.



(b) Customer making insurance claim.

FIGURE 5.4: Sequence of functions at the core-tier blockchain. All interactions with the smart contract represent transactions. Dashed arrows represent getter functions with no fees.

5.3 Security Analysis

In this section, we outline the security analysis we performed on the proposed blockchain-based framework using the aforementioned use-cases. In this discussion we include a threat model, as well as a discussion on the design considerations in place to guarantee security for conducting transactions in the proposed framework.

5.3.1 Security Considerations and Analysis

Here, we will discuss design considerations of DamonChainthat address three major security requirements, namely confidentiality, integrity, and availability. Briefly, *confidentiality* guarantees that only authorized users have access to the data; *integrity* ensures that a message received at the destination is the exact message sent by the source (i.e. it has not been tampered with or altered); *availability* refers to the availability of the service or data whenever is needed.

Confidentiality

As mentioned, the Ethereum blockchain uses public-private key pairs, while Daminchain relies on smart contracts to share data to distant nodes in the blockchain network. Every transaction made on the public blockchain is validated through distributed consensus and requires the transaction issuer's private-key signature, which can be verified using the issuer's public key. Moreover, the hash of the IPFS file containing the IoT user's data is encrypted with the public key of the requester, therefore only the requester is able to decrypt it and retrieve the IPFS file. Together with private-key signatures on transactions, and encrypted IPFS file hashes, confidentiality is ensured.

Integrity

Maintaining data integrity on public blockchains comes with the inherent features of publicly deployed blockchains. Indeed, each peer node of a public blockchain network has its own replica of the whole blockchain content, and to commit new transactions on a blockchain, distributed consensus among the majority of nodes is needed. Therefore, integrity in a public blockchain is guaranteed by design. DamonChain maintains time-stamped records of sensor data in a private blockchain. This implementation strategy enables users to send blocks within specific time-frames to a distant node on the blockchain network. However, there is a valid argument that, since a private blockchain is totally centralized, it cannot alone guarantee nor immutability nor integrity of its records [157]. To guarantee verifiable data integrity in the private blockchains tier, DamonChain uses the immutable database characteristics of BigchainDB, so as to maintain auditable records of all the block headers of the private blockchain. Indeed, BigchainDB is a distributed database maintained by a publicly distributed network federation. The local IoT gateway does not need to be a participating node in the federation,

and can simply issue data transactions to BigchainDB. An earlier consideration in the design of DamonChain was to simply have the gateway upload all the generated IoT data and events to the BigchainDB back-end. However, this simplistic approach would require much more frequent uploads, also incurring heavy network overheads. Hence, to minimize the volume and frequency of data uploads, the local IoT gateway simply uploads the hash of each block that gets added to the private blockchain, while each block represents multiple data generation events bundled together. When an IoT user engages with a distant node on the public blockchain network, the IoT user grants the distant node read-only privileges to the hashes of the blocks uploaded to BigchainDB. Therefore, while users have the choice to send their data to the extent they want and when they want, distant nodes have a way to verify that the data stored in the private blockchain has not been tampered with.

Availability

An attack against availability of a system is called Denial of Service (DoS). Briefly, vast amounts of fake requests are sent to a target, in order to overwhelm it and make it unavailable to legitimate users. Such an attack is often performed through a network of infected devices (i.e. a *botnet*) in a distributed manner, hence the name Distributed DoS (DDoS) [158]. The public blockchain used by DamonChain is a distributed system which has monetary cost for signing transactions (i.e. transaction fees). It follows that, from an economic perspective, to perform a (D)DoS attack on this framework is very expensive. Moreover, generally speaking, the collective computing power available in blockchain makes it extremely hard to launch successful DoS attacks. Indeed, in order to overwhelm the entire blockchain system, multiple nodes across a significant number of various institutions must be attacked. At the time of writing, the average daily hash-rate of the main Ethereum public blockchain is 275 TH/s. This means that, to successfully perform a 51% attack, either to alter the blockchain contents, or to hinder the network availability, adversaries would have to amass more than 137.5 TH/s of hashing power[1].

5.3.2 Threat Model

Scenario 1: Data modification in edge-tier blockchains

While data integrity and immutability is an inherent feature of public blockchains, privately deployed edge-tier blockchains do not have built-in guarantees for data integrity. IoT data is collected and pushed into the edge-tier blockchains, which can be selectively expressed to third parties for monetary services. In this scenario, a malicious edge-tier owner entity modifies edge-tier blocks ϱ^j, since it is computationally feasible within private blockchains.

[1] https://etherscan.io/chart/hashrate

In order to circumvent this threat, the IBGW of the edge-tier blockchain automatically keeps uploading the hashes of ϱ^j to a BigChainDB repository, which is available to third-party requesters. Since each block is linked to its predecessor, modifying a single block will alter all subsequent block hashes, and the hashes of the actual blocks sent to the requester, $e_{k_p^r}(h_t)$, will not match up with the block hashes stored in the BigChainDB repository.

Scenario 2: Sniffing data sent from an edge-tier blockchain

Keeping the vehicle insurance use-case in mind, let's assume an insurance company y'_r, acting as an adversary, is interested in getting access to the data of the customer c_i where $y_r \in S^i$ while $y'_r \notin S^i$. y'_r needs h_t in order to get access to the c_i's data. h_t is encrypted with k_p^r, it can be only decrypted with k_s^r and the insurance companies do not share their private keys, y'_r is not able to get access to the customers of other companies.

Scenario 3: Launching a Denial-of-Service (DoS) attack on the core-tier blockchain

Briefly, vast amounts of fake requests are sent to a target, in order to overwhelm it and make it unavailable to legitimate users. Such an attack is often performed through a network of infected devices (i.e., a *botnet*) in a distributed manner, hence the name Distributed DoS (DDoS) [158]. The public blockchain used by the proposed framework is a distributed system which has monetary cost for signing transactions (i.e., transaction fees). It follows that, from an economic perspective, to perform a (D)DoS attack on this framework is very expensive. Moreover, generally speaking, the collective computing power available in blockchain makes it extremely hard to launch successful DoS attacks. Indeed, in order to overwhelm the entire blockchain system, multiple nodes across a significant number of various institutions must be attacked. At the time of writing, the average daily hash-rate of the main Ethereum public blockchain is 275 TH/s. This means that, to successfully perform a 51% attack, either to alter the blockchain contents, or to hinder the network availability, adversaries would have to amass more than 137.5 TH/s of hashing power.

Scenario 4: Launching a Denial-of-Service (DoS) attack on an edge-tier blockchain through a corrupted device:

In various IoT use-cases, there is the possibility of an edge-tier device being compromised by an adversary. While each IoT device can only issue transactions to the edge-tier blockchain, a DoS attack can be fatal for private organization or federation edge-tier blockchains. To mitigate this threat, we have used a tokenized approach at the edge-tier level. At the edge-tier, we are using the Tendermint consensus engine with a custom token. While these tokens do not carry any real monetary value, implementing transaction fees ensures that there is a limit to how many transactions a compromised device can issue. Each device is given a limited required amount of tokens T to be

TABLE 5.5: Parameters for evaluating the performance of the proposed framework in two use-cases.

Parameter	Value
Core-tier Testnet	Ethereum Rinkeby
Edge-tier Testnet	Hyperledger Burrow
Transactions on public blockchain	300 per minute
Transactions on private blockchain	1000 per minute
Gas price	0.000000001 ETH

able to issue transactions over a period of time. If the transaction fees for a single transaction are F_t, the maximum number of transactions a compromised device will be able to make is T/F_t.

5.4 Performance Analysis

This section details the performance analysis based on the experiments we conducted to validate the proposed framework. The proposed framework was developed and deployed on a Proof-of-Authority (PoA) based blockchain. To calculate the real-world transaction processing fees incurred by the smart contracts, we have used values equivalent to Ethereum, where each state altering transaction requires an amount of "gas" depending upon the complexity of the transaction. The price for each unit of gas at the time of writing this paper was noted to be 1 Gwei (equivalent to 0.000000001 ETH). The results were obtained with a Raspberry pi 3 board serving the function of an IBGW within the architecture of the proposed framework. Table 5.5 shows the experimental parameters used for our evaluation.

For the performance analysis, we assessed the feasibility of the blockchain-IoT framework using the following metrics:

- The computational and networking overheads of the IBGW for participating in the core-tier blockchain,
- The latency involved in signing transactions involving IoT data and cryptocurrency,
- The transaction fees incurred while participating in the blockchain-IoT framework.

The functions written in both smart contract implementations were kept at $\mathcal{O}(1)$ algorithmic complexity to minimize computational overheads and associated transaction fees.

The core-tier of the framework was developed and deployed on the Ethereum Rinkeby[2] testnet. This is a public test network operating with a proof-of-authority consensus protocol called "Clique", instead of the classical proof-of-work based consensus used by the Ethereum mainchain. Since the tokens in Ethereum test networks hold no real-world monetary value, issuing transactions does not have a real-world "cost". Therefore, a proof-of-authority consensus algorithm serves better security measures to prevent signing false blocks and performing DoS attacks.

In a proof-of-authority-based consensus protocol, a number of chosen "sealers", or pre-approved nodes are made responsible for mining and adding new blocks to the blockchain. Any node that attempts to join the Clique has to be approved by the pool of existing authority nodes. The approved authority nodes are only allowed a limited number of signatures when signing new blocks, so as to prevent forks from going on for too long or even overtaking the canonical chain.

The Ethereum test networks are useful to test decentralized applications in public blockchain environments, using cryptocurrency with no real-world monetary value. With the added security of the Clique consensus, we chose the Rinkeby testnet for deploying and testing our proof of concept implementation. Fig. 5.5 shows the dashboard of the Ethereum Rinkeby network, with real-time statistics and miner information. As illustrated on the dashboard, the miner clique is spread all over the world, and with international participation by full nodes and light nodes, Rinkeby provides a rich and robust test bed for gauging real-world performance.

[2] https://www.rinkeby.io

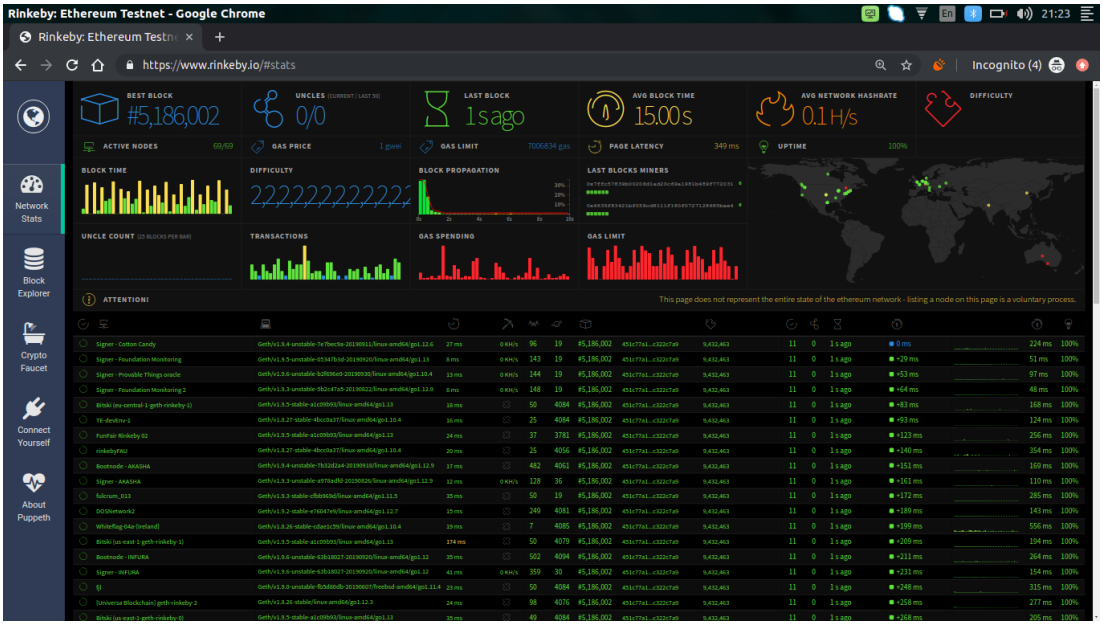


FIGURE 5.5: The Ethereum Rinkeby dashboard.

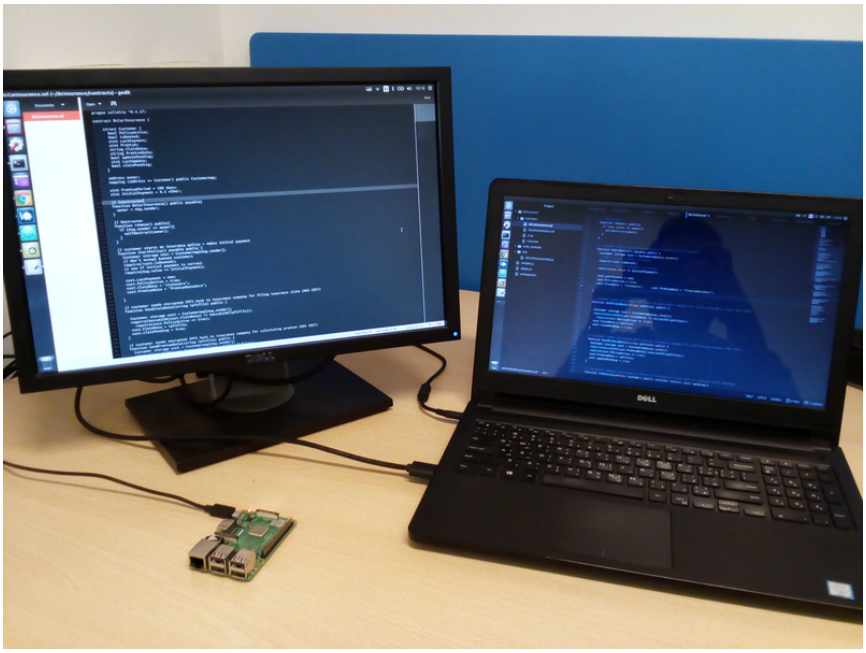


FIGURE 5.6: Testbed used for conducting the performance analysis.

5.4.1 Hardware and Software Used

We considered the case where one laptop (Dell Inspiron core-i7) represents the distant node and a single-board computer (Raspberry Pi 3 Model B+) represents the IoT user's local gateway, as shown in Fig. 5.6. The laptop has an 8GB memory with 1TB storage space, while the single-board computer has a 1GB SDRAM and 32GB of storage.

We used the Remix Ethereum IDE[3] for writing, compiling, debugging and locally testing the smart contract. Remix Ethereum is an in-browser IDE for Solidity, the language for writing smart contracts in Ethereum. After locally testing the smart contract functionality, we deployed it to the Rinkeby Ethereum test network.

We used Metamask[4] and the Ethereum CLI client Geth[5] to connect to the Rinkeby Ethereum test network and manage the accounts used for our experiments. The Remix Ethereum IDE served as a front-end for interacting with the deployed contract on both the nodes involved in the implementation.

Details of the publicly deployed contracts and the transactions we carried out can be accessed by monitoring the following addresses on the Rinkeby test network:

`0xb45d0cc4883ae0c18469904181d68775436b2eb7`
`0x82A4958337eCE6E358851F073AF123662b8Ce194`

[3] http://remix.ethereum.org/
[4] https://metamask.io/
[5] https://www.ethereum.org/cli

For the performance analysis, we considered the computational and networking overheads of the local IoT gateway for participating in the public Ethereum testnet, as well as the latency involved in signing transactions involving IoT data and cryptocurrency. The functions written in both smart contract implementations were kept at $\mathcal{O}(1)$ algorithmic complexity to minimize computational overheads and associated transaction fees.

5.4.2 Computational Overhead

We used the web browser plugin Metamask[6] and the light-client Geth[7] implementation for our decentralized applications. The light-client implementation allows IBGW to issue transactions without needing to store a full copy of the core-tier blockchain. Additionally, since our nodes do not mine and validate new blocks, the computational overhead on the IBGW is negligible.

5.4.3 Transaction Processing Speed

Over the multilayered blockchain architecture, our observations are that the deciding factor of the throughput is the transaction finality time. Our implementation in Ethereum appends new blocks over the core-tier blockchain each 14 seconds. We compared the transaction finality time of single monolithic blockchain implementation of Bitcoin, versus our layered implementation on an Ethereum test network. In our experiments, we verified that the transactions we made were finalised at the rate of 32 seconds on average, and 90 seconds at the most.

Fig. 5.7 shows the transaction processing times observed in our experiments for three separate operations. In the case of IoT data marketplace, to carry out one single sale of data, it would involve two transactions. Therefore, the average processing time is 64 seconds. In the case of connected vehicle insurance (not taking into account the time needed by the insurance company to analyze the received data) the time taken in transactions for making premium payments is approximately 96 seconds (since this process involves 3 transactions) and 64 seconds for processing insurance claims (involving this process 2 transactions). It is worth noting that core-tier blockchain implementations with lower block publishing times, the throughput will further increase.

5.4.4 Network Overhead

The tiered network architecture of the proposed framework significantly cuts down the network overhead which would have been generated in the case where all IoT data generation events were logged on a public blockchain. The added benefit of using Metamask to push transactions is that there is no incoming traffic from the core-tier network for updating a local copy of the

[6] https://metamask.io/
[7] https://www.ethereum.org/cli

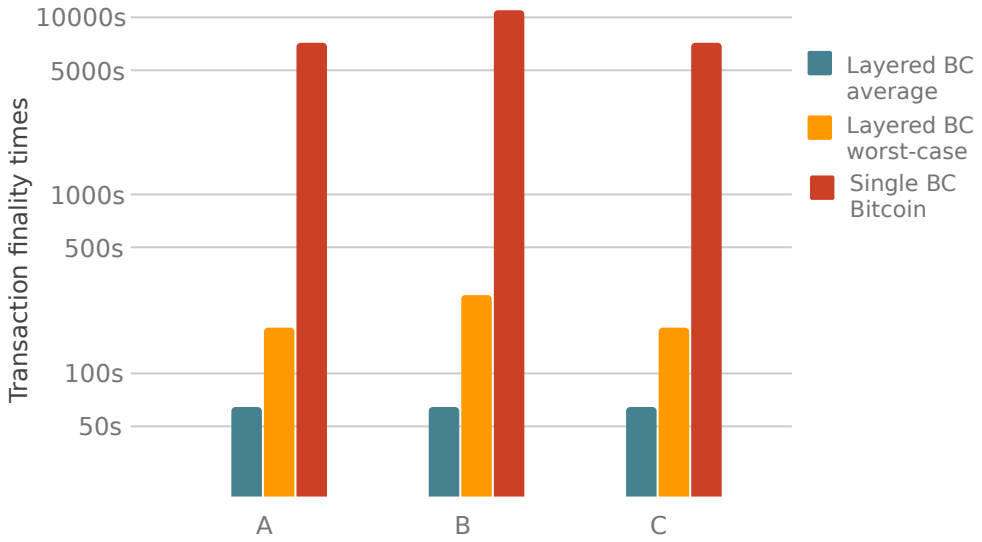


FIGURE 5.7: Transaction finality times in (a) making a sale in the data marketplace use-case, (b) updating insurance premium, and (c) processing an insurance claim in the vehicle insurance use-case.

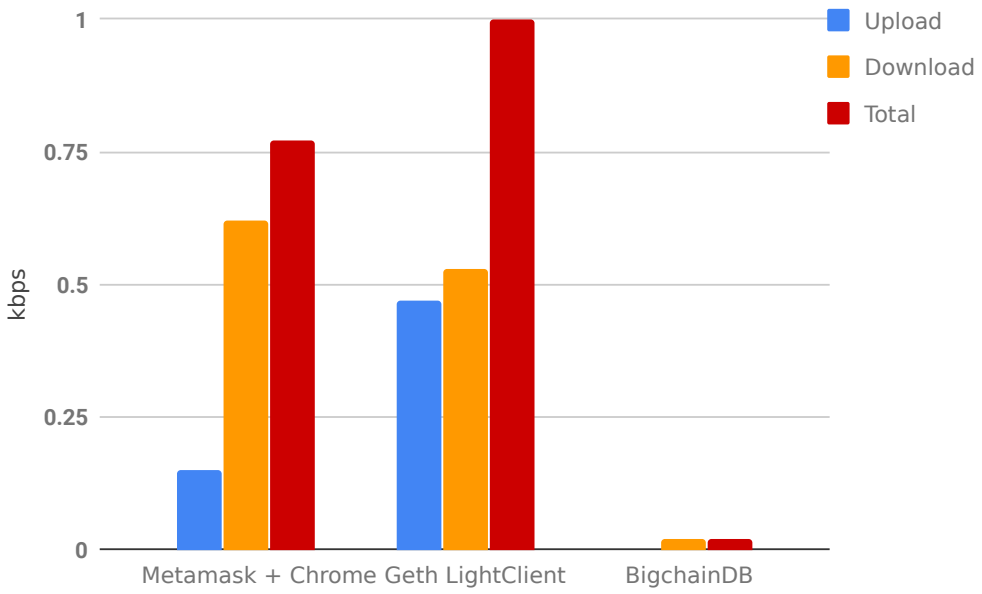


FIGURE 5.8: Network traffic overhead generated by using Metamask on Chrome, Geth, and with private blockchain hash storage in BigchainDB.

core-tier blockchain. Therefore, the overall network overhead experienced by the IBGW includes the periodic upload of each edge-tier block hash to BigchainDB through the HTTP POST API, along with the transactions that need to be made over the core-tier blockchain.

To analyze the network overhead of our application, we compared the traffic generated when using Metamask on a Google Chrome front-end, and the traffic generated in running an Ethereum light client on Geth. The memory and storage requirements of running a full Ethereum node are far and

beyond the hardware limitations of a RaspberryPi IoT gateway, so the light client is a more suitable option. By storing edge-tier block hashes onto the BigchainDB testnet at a speed of 1 transaction per minute, we maintained a 17 bps traffic for maintaining data integrity in the private blockchain. For interacting with the public blockchain, we considered the traffic generated in running Metamask as well as the Ethereum light client on Geth, while executing 1 read function per second.

Fig. 5.8 illustrates the overall traffic rates generated over a 24 hour period. We observed that since we do not maintain a full node on the core-tier public blockchain, the cost of decentralized authentication and authorization is minimal to the overall network traffic. Additionally, while the overall network traffic overhead from light-client Geth is very small, simply running Metamask on Chrome further decreases the network overhead. This is because the light-client implementation consistently maintains a record of the block headers in the core-tier blockchain.

5.4.5 Transaction fees

Each Ethereum transaction requires *"gas"* for execution, which has a fixed price in ETH. Therefore, the number of computational steps involved in any given transaction dictates how much gas is required to execute it. Our aim for the smart contract function design was to reduce as much sa possible the algorithmic complexity and minimize the transaction costs that the insurer and customer would have to pay when interacting with the smart contract.

For our experiments on a publicly deployed Ethereum cryptocurrency network, we deployed our smart contract on the Rinkeby test network and made observations on the transaction fees. At the time we obtained these results, the conversion rate from USD to ETH was recorded at 380 USD per ETH. The standard gas price in the Rinkeby testnet is 1 Gwei (corresponding to 0.000000001 ETH) per unit of gas. Once again, it is worth to recall that reading functions in the smart contract that simply return stored values on the blockchain are free of cost and instantaneous. These functions allow customers to view their updated insurance premiums and monitor the time of their last payment, and they allow the insurance company to check if customers have made timely premium payments and monitor customers' payment information. All functions that either transfer funds from one side to another, or alter the stored state of a smart contract's object variable have associated transaction fees, in order to incentivize miners and secure the network.

IoT Data Marketplace Table 5.6 shows the transaction fees incurred when executing the smart contract functions which update the stored state of the smart contract's object attributes. We can observe that for making a single sale of IoT data, a registered buyer would have to pay 1 cent as transaction fee, and the cost to the seller would be 3 cents.

TABLE 5.6: Gas usage and transaction fees for executing functions in the data marketplace smart contract.

Functions	Invoked By	When it is Invoked	Gas Usage	USD
DEPLOYCONTRACT()	Local IoT GW	One-time	746108	0.2835
REQUESTDATA()	Requester	First data request (registration)	63251	0.024
REQUESTDATA()	Requester	All subsequent data requests	27978	0.011
SENDDATA()	Local IoT GW	In the event of a data request	79136	0.03
BANREQUESTER()	Local IoT GW	One-time	28601	0.01

TABLE 5.7: Gas usage and transaction fees for executing functions in the vehicle insurance smart contract.

Functions	Invoked By	When it is Invoked	Gas Usage	USD
DEPLOYCONTRACT()	Insurer	One-time	1258610	0.478
STARTCOVERAGE()	Customer	One-time	103605	0.04
SENDCLAIMDATA()	Customer	In case of an accident	92832	0.035
PROCESSCLAIM()	Insurer	In case of an accident	22587	0.0086
SENDPREMIUMDATA()	Customer	Once in a decided time period	92832	0.035
MAKEPAYMENT()	Customer	Once in a decided time period	23045	0.0087
UPDATEPREMIUM()	Insurer	Once in a decided time period	38805	0.015
CHECKPAYMENT()	Insurer	Once in a decided time period	23933	0.009

Connected Vehicle Insurance Table 5.7 shows the transaction fees incurred when interacting with functions in the smart contract. Note that the CHECKPAYMENT() function only incurs a function if a customer has not made timely premium payments and the company decides to terminate their insurance policy. Read-only functions in the connected vehicle insurance contract that do not require transaction fees are not included in this table. We can infer the overall transaction cost for making a premium payment, both for the customer and the insurer. For making an insurance premium, a customer sends data for obtaining an insurance premium update and pays the premium after the insurer updates it. Therefore, the cost to the customer is 4.37 cents and, to the insurer the cost is 1.5 cents. Similarly, for making and processing an insurance claim, the transaction cost to the customer is 0.87 cents and 0.86 cents to the insurer.

5.5 Summary

In this chapter, we discussed the technical details about the access control mechanisms within our private-by-design, blockchain-based framework. The use-cases discussed in this chapter specifically entailed exchanging IoT data

in return for monetary services without involving trusted third party intermediaries. We demonstrated the capabilities of the proposed solution through a full-fledged implementation and made important observations by conducting a performance analysis. Results showed that, in terms of computational and networking overhead, the cost of achieving decentralized IoT data monetization can be very low. Moreover, the performance analysis included detailed observations on the transaction processing fees and transaction throughput. To conclude, by designing and implementing a two-tiered blockchain architecture, we have taken steps towards achieving privacy and scalability in developing a decentralized IoT platform, and by implementing two real use cases, we have demonstrated the malleability of the proposed framework. This chapter discusses use-cases where batches of IoT data are either monetized or exchanged in return for insurance services. The following chapter delves into decentralized access control and accountability mechanisms for sharing streams of data within a health monitoring context.

Chapter 6

Secure and Privacy-Preserving End-to-End IoT Communications

With the growing research interest in integrating blockchains into the IoT, healthcare is one of the most significant industries where a number of suitable use-cases have been identified. These include applications in medical record keeping, pharmaceutical supply chains, health insurance, health research analytics and remote health monitoring [159]. Remote health monitoring (RHM) enables doctors to monitor patients' physiological conditions from their homes, while freeing up expensive healthcare facilities and hospitals to those in urgent need. For the patients, RHM provides a more comfortable and cost-effective alternative to on-site monitoring. Non-invasive wearable sensor technologies are proven to be viable diagnostic tools for monitoring patients' physiological signs and activities remotely, in real time [160]. Naturally, wearable sensors and the data generated by them to facilitate RHM have been the subject of research and development in RHM systems.

Following the file transfer and monetization capabilities of the proposed blockchain based IoT framework, we adapt the framework and tailor it to a scalable and private-by-design remote health monitoring use-case. Leveraging blockchain's capabilities in decentralized accountability, authorization and authentication, our proposed solution is designed to securely share patients' sensitive data with doctors, without having the data pass through third party services.

In this solution, we use the core-tier as we did before: a medium for negotiation and accountability. The core-tier blockchain is represented within this chapter as a remote healthcare blockchain. The remote healthcare blockchain can either be deployed in a permissionless or permissioned fashion, however, for this implementation, we have opted to use the PoA based Ethereum blockchain, similar to the implementation in Chapter 5. At the edge-tier, instead of using separate blockchains, we utilized the decentralized Tor network for streaming data from patients to doctors in a peer-to-peer fashion. Here, the patients will have their own IoT gateways, that do not maintain an edge-tier blockchain. Since the data is to be sent to the doctor in real-time,

therefore the patient's local IoT gateway and the doctor's node have a Tor-based data delivery mechanism in between them. Local storage with data hashes stored in BigchainDB is optional, in the cases where batches of data are to be sent to healthcare providers, however in this chapter, to demonstrate the secure health data streaming capabilities, we rely exclusively on Tor-based hidden services for streaming health data.

To demonstrate and analyze the proposed solutions's application in real-world RHM scenarios, we developed a real-world implementation, and explored three RHM use cases:

- Cardiac patient monitoring; patients that require monitoring on a long-term basis,
- Sleep apnoea testing; patients that require monitoring over a short period of time,
- EEG monitoring for epilepsy; patients that require monitoring on an emergency basis in the event of a seizure.

Our implementation is built on Ethereum's Rinkeby test network, which provided an adequate platform for testing our solution's applicability in a blockchain network spread worldwide. For delivering patients' health monitoring data, we have implemented Tor hidden services. We chose Tor instead of other peer-to-peer data delivery solutions due to the fact that through Tor, we are able to transfer the data without any unnecessary metadata which is not relevant to the application.

6.0.1 Tor Network

Tor is a decentralized, privacy enhancing system designed to prevent traffic analysis attacks [161]. Tor provides a layer of privacy protection on top of TCP, while maintaining high throughput and low latency, which makes it ideal for data transfer applications. Since its initial release, researchers have conducted analyses on Tor's performance [162] and security [163].

Tor provides a layer of privacy protection for TCP through a three-hop routing path, using a layered encryption strategy as seen in onion routing mechanisms. Onion routing involves enested layers of encryption within the application layer, like the layers of its namesake. Tor encrypts the data to be sent, as well as the destination identifier, multiple times. The encrypted data is then sent through a virtual circuit which consists of randomly selected Tor relay nodes. Three hops are the default configuration within onion routing, however, this can be increased or decreased. Each relay decrypts and unravels one layer of encryption, reveals the address of the next relay, and passes the remaining data to it. The final relay decrypts the last layer of encryption and sends the data to its intended destination, without knowing the IP address of the source.

Additional considerations for end-to-end encryption are required since the entrance Tor router can directly observe the originator of a particular request

through the Tor network, and the exit node can examine the unencrypted payload, and the destination server. No single node can observe the sender and receiver. To achieve low latency, Tor does not re-order packets within the Tor network.

Ricochet[1] is an anonymous peer-to-peer instant messaging system that operates on Tor hidden services, which is used to relay messages without relying on centralized messaging servers. Being run on Tor hidden services, Tor follows rendezvous specifications [164] with self-authenticating hostnames. When establishing a Tor hidden service, a 1,02-bit RSA key pair is generated, and a SHA-1 digest of the public key is calculated. The .onion address is then the base32-encoded first half of the SHA-1 digest. Users would be able to access this hidden service through the .onion address, for example: *2qxw7mf2xnfh4mqr.onion*. In Ricochet, the contact ID follows the following format: *ricochet : 2qxw7mf2xnfh4mqr*.

6.1 Architecture of the RHM System

While blockchain addressing does provide pseudonymity, inferences can be made about accounts that generate specific kinds of data. In the IoT, prevalent privacy issues pertain to third-party entities monitoring user behaviour and preferences, and logging all IoT data generation events on a public blockchain will lead to privacy breaches [10]. Furthermore, for smart healthcare applications, maintaining healthcare records and transferring monitoring data over a single publicly deployed blockchain severely limits scalability. Considering the high frequency and volume of data generated in health monitoring applications on a nationwide scale, the added latency of the consensus algorithm will be a bottleneck to record every data transfer event. Other than this, it is worth recalling that blockchains are linearly growing data structures. Therefore, to log every single data transfer event on a public blockchain will cause an explosion in the storage requirements of blockchain full-nodes.

To address these limitations, we have implemented an architecture consisting of a public blockchain at the core of the health monitoring system, with Tor hidden services connecting patients to their doctors, as illustrated in Fig. 6.1. This architecture allows patients to register themselves with a particular healthcare provider, and engage in agreements about the nature of the data shared directly with the healthcare professional. The healthcare blockchain stands to provide accountability, identity management and healthcare information management for patients. None of the patients' actual health monitoring data is stored on the blockchain, and the agreements on the nature of the data being shared is kept encrypted. In the event of an investigation or an insurance claim, the healthcare provider can present details of the agreements between the patient and the doctor, while the patient's monitoring data remains private. In our implementation of the remote health

[1] https://ricochet.im/

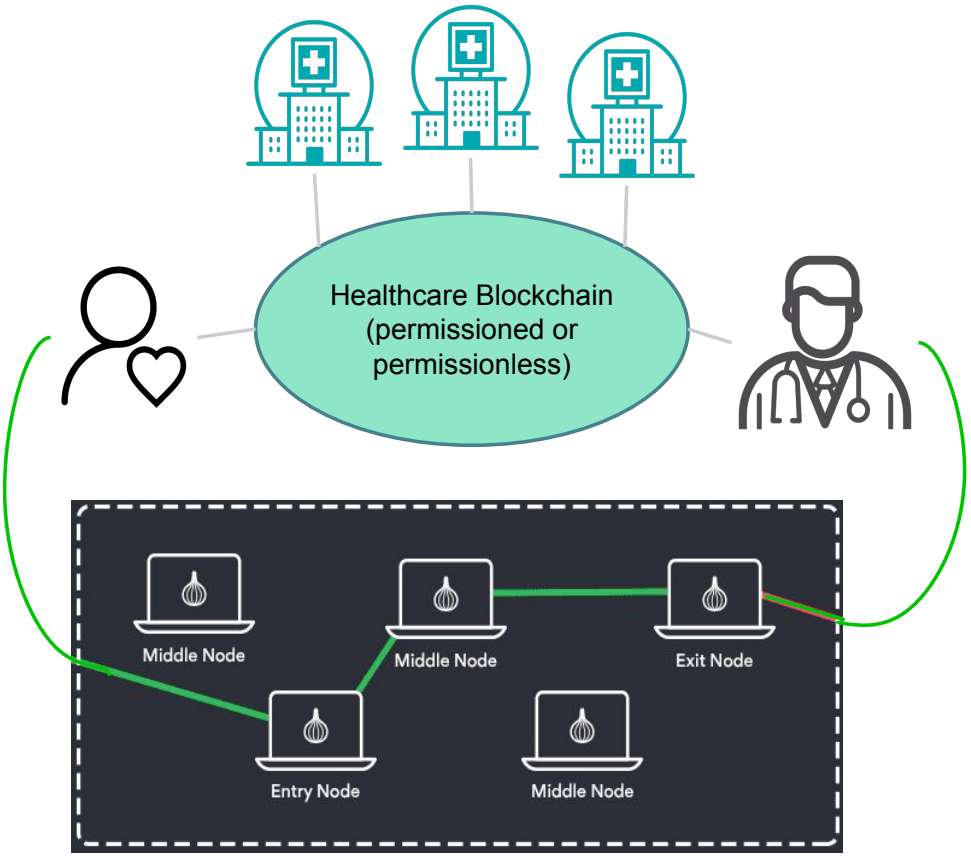


FIGURE 6.1: The proposed blockchain-based RHM architecture.

monitoring blockchain solution, we have used the Proof-of-Authority (PoA) variant of the PoW consensus algorithm, to prevent unauthorized entities to mine on the network. Limiting the mining pool in this case is advantageous since patients and healthcare providers must have a limited number of tokens within the RHM blockchain. If unauthorized entities were to mine blocks on the blockchain, they could launch a denial of service attack with all the tokens gained in their mining rewards.

The public healthcare blockchain consists of hospitals, and doctors who are registered with specific hospitals. While registering with a specific doctor, the patient and doctor both exchange their .onion addresses, which are kept encrypted in the blockchain's transaction payloads. The patient transmits their health monitoring data to the doctor via Ricochet, and logs the starting and stopping timestamp onto the public blockchain, along with the overall size of data transmitted during that time. These logs provide accountable proof of the patient having transmitted their healthcare data in accordance to the doctor's request. The doctor's request for data and subsequent recommendations, all encrypted, provide evidence of the medical advice given to the patient.

Privacy and scalability benefits

Using off-chain data transfer mechanisms, and using blockchains simply for negotiations and record-keeping in remote health greatly reduces the load on the blockchain system. With the overall number of interactions with the blockchain greatly reduced on a per-patient basis, the health monitoring blockchain system can scale up to a nationwide level. Furthermore, with advancements in "*sharding*" **[EthSharding]** on the Ethereum platform, it will eventually become possible to scale up the health-monitoring blockchain nationwide, with different shards being accountable for different regions. Using hidden Tor services and the Ricochet protocol greatly adds to the privacy that is built into the proposed solution. None of the patients' health data is accessible to anyone in the health-monitoring system except the doctor it is intended to be shared with. With end-to-end encryption, the patients' health data is not accessible to anyone over the Tor network. The solution does not involve third party servers in relaying patient data, and none of the data is stored onto the blockchain. By using Tor hidden services, we ensure that no metadata is transmitted to the other side which is not primary to the application itself.

6.2 Remote Health Monitoring Use-Cases

The proposed blockchain-based remote health-monitoring solution provides a medium for negotiation and accountability, identity management and a private and secure means to transmit health monitoring data. In this section, we will outline the decentralized accountability mechanism involved, as well as the working principles of the patient-doctor relationship under the proposed solution. We have looked at three main use-case scenarios where the solution can be applicable, out of many more. Firstly, the cardiac patients who require longer term monitoring, secondly, sleep apnoea studies that require short term monitoring over specific periods of time, and thirdly epileptic patients who may need to transmit EEG data in the event of a seizure, on an emergency basis.

Firstly, each hospital has a smart contract where doctors "register" themselves with verifiable registration information. "Registration" entails doctors enlisting themselves in a mapping data structure within the hospital's smart contract, which can be looked up by patients to find the doctor they are to register with. This mapping will include the doctor's public-facing information, as well as an address to the doctor's own smart contract, where the patients can register themselves. Fig. 6.2 is an illustration of the step-by-step sequence of events that take place when a patient registers with a doctor and avails remote health monitoring services.

Here, let $H = \{h_0, h_1, \ldots, h_n\}$ be the set of hospitals, each being member participants on the remote healthcare blockchain. Also, let $Y = \{y_0, y_1, \ldots, y_n\}$ be the set of doctors, and $X = \{x_0, x_1, \ldots, x_m\}$ be the set of patients, such that:

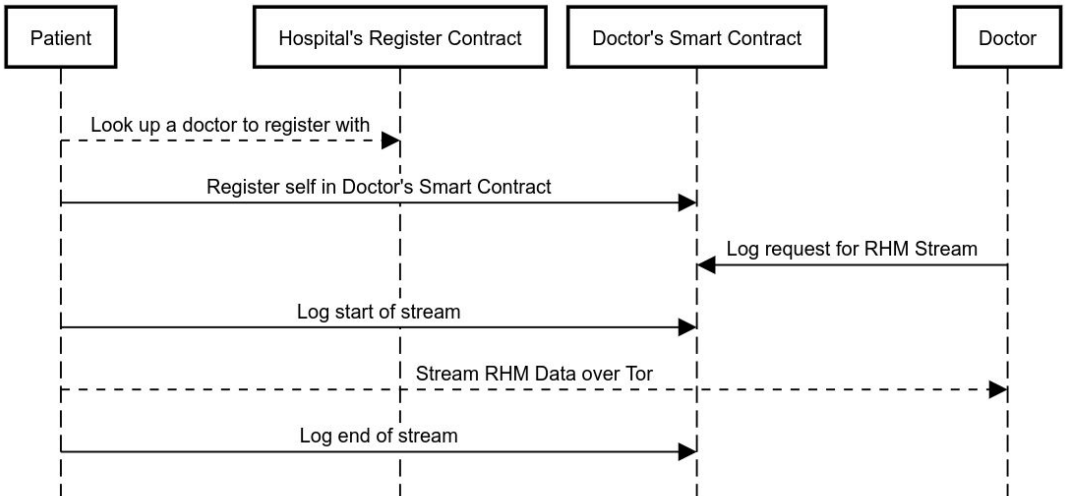


FIGURE 6.2: Sequence diagram of a remote health-monitoring instance, where a patient registers with the doctor and avails remote health-monitoring services. All interactions with the smart contract represent transactions. Dashed arrows represent off-chain interactions over Tor hidden services.

$$\forall h_i \exists Y^i \subset Y : (h_i \ni Y^i) \tag{6.1}$$

$$\forall y_i \exists X^i \subset X : (y_i \ni X^i) \tag{6.2}$$

where $\ni$ signifies registration. While the doctors and patients each have a public-private keypair $\{k_p, k_s\}$ for issuing transactions on the blockchain, for each doctor-patient pair, there will exist a pair of .onion addresses. When a patient x_j invokes the REGISTER() method in the smart contract of a doctor y_r, the patient encrypts their own .onion address o_j with the doctor's public blockchain key, and sends it to the doctor via the REGISTER() method. Similarly, in return, the doctor's blockchain node sends over a specific .onion address for the patient.

$$\forall y_r \exists \langle k_p^r, k_s^r \rangle : d_{k_s^r}(e_{k_p^r}(o_j)) = o_j \tag{6.3}$$

$$\forall x_j \exists \langle k_p^j, k_s^j \rangle : d_{k_s^j}(e_{k_p^j}(o_r)) = o_r \tag{6.4}$$

The reason for having two .onion addresses for each doctor-patient relationship is to enable the doctor's node to end transmission on a specific .onion address if someone transmits data for a longer period of time than what is decided.

With knowledge of the patient's condition and following off-chain correspondence, a doctor issues a request for data, with parameters based on the urgency or timespan of health-monitoring required. The time period for data transmission is set, the limits for the volume of data to be sent are set, and

this time-stamped request logged onto the blockchain serves as an accountable proof of the request.

Following the doctor's request for data, patients transmit their health monitoring data over Tor hidden services addressed in Ricochet to the .onion address provided to them by their doctor. The starting time of transmission is logged on to the blockchain, which serves as proof of authentication and time log of the start of data transfer. At the end of the data transfer, the patients log an end of transfer onto the blockchain, which includes a hash of the bulk of data which is sent to the doctor, for verifiability. The time logs will indicate how well a patient has fulfilled the doctor's request and the immutable contents of the blockchain will provide a hashed proof of the data sent. Following the remote health monitoring, a doctor can provide recommendations, all without having the patients' data go through a third party server or data management system.

Cardiac Patients *Long-term RHM:* For cardiac patients, the doctor's will issue a request for health monitoring data for a longer period of time. This longer period may include further data transfer parameters like specific times of the day, a specific volume of data during the day, whichever suits the patients' case best. In this case, the doctor will have to have a specific node listening over the hidden service for a longer period of time. Within the node, at the doctor's end, there will be safeguards to ensure that no malicious flooding of data is taking place, and only a specified amount of data is being received. In this particular case, real-time analytics can be employed at the patient's end, with gentle reminders to the patient for being careful during periods of exertion or stress. Predictive models at the doctor's end can help the doctor modify the course of healthcare being provided to the patient, in a peer-to-peer fashion.

Sleep Apnoea Patients *Short-term RHM:* For sleep apnoea patients, relevant research efforts have shown that home testing may lead to fewer false negatives, since patients are sleeping within the comfort of their own beds [165]. For transferring short-term sleep apnoea test data, doctors can request data over a smaller period of time. Here, the specific .onion address on the doctor's end will cease listening at the end of the specified time period. Such an application of peer-to-peer remote health-care monitoring will allow patients access to doctors beyond their geographical region, and will further the cause of medical globalisation.

Epileptic Patients *Sporadic RHM:* In the event of an epileptic seizure, patients are rushed to the hospital for a prompt EEG, so doctors can gain insight into the epileptic event. With the improving robustness of home EEG equipment [166], it is possible for doctors to receive data as early as possible following an epileptic seizure. In this case, typically doctors monitor epileptic events that take place over a specified amount of time, and recommend treatment options. The doctor will have a .onion address listening over a period of time, with specified limits of data to be transferred per day. Preventing excessive

amounts of data being transferred helps mitigate the network being flooded, and serves the purpose of all genuine patients. Under the proposed solution, patients can transfer near real-time data taken from home EEG equipment, to provide doctors with relevant data in time, without rushing or discomforting the patient.

6.3 Security Analysis

In this section, we outline the security analysis we performed on our proposed blockchain-based remote health monitoring solution based on multiple attack scenarios.

Scenario 1: Compromised .onion Addresses

For each doctor-patient relationship, the .onion addresses and blockchain keys both serve as multi-factor authentication for engaging in remote health monitoring. The .onion address provided to patients by their doctors comes via a blockchain transaction, which has the doctor's blockchain signature. On the other end, if a patient's .onion address is compromised, any data transferred to the doctor will either not have a start of transfer log on the blockchain, or the data hash stored on the blockchain at the end of the transfer will not match the corrupt data that was actually sent.

Scenario 2: Launching a Denial-of-Service (DoS) Attack on the Remote Healthcare Blockchain

The public blockchain used by the proposed solution is a distributed system which has a tokenized cost for signing transactions (i.e., transaction fees). All entities involved have only a limited supply of the blockchain's token, which mitigates the chance of performing a (D)DoS attack on the solution. The remote healthcare blockchain uses PoA consensus, where only hospitals and healthcare companies are entitled to mine on the blockchain. The hospitals and healthcare companies who maintain the consortium transfer tokens to all entities involved within a limit depending upon multiple factors, for example, what insurance a patient has, or whether the entity is a patient or a high ranking oncologist. Moreover, generally speaking, the collective computing power available in the consortium will make it extremely hard to launch successful DoS attacks.

Scenario 3: Sniffing Data Sent from a Tor Exit Relay

While issuing requests and delivering data through Tor hides any metadata regarding the location of the source or destination, an issue arises with the last relay node. While the final relay node may not know where the healthcare monitoring data has come from, it can observe the destination, and can

make inferences on the sensitive health data. To combat this, we have incorporated Ricochet into our solution to provide end-to-end encryption based on the Ricochet address of the destination entity. This way, none of the data is exposed to any off-chain entity, as well as any entity on the remote healthcare blockchain.

Scenario 4: Sniffing Data Transfer Parameters from the Doctor's Smart Contract

Let's assume an entity on the remote healthcare blockchain y'_r, acting as an adversary, is interested in getting access to the information of the patient x_j stored in the smart contract of a doctor y_r. This information can include the patient's identity, the volume of data sent to the doctor, as well as the nature of data which was sent. This information, h_j is encrypted with k^r_p, thus it can only be decrypted with k^r_s, and with the doctor not having had their account compromised, y'_r will not be able to get access to the patient's information.

Scenario 5: Impersonators and Malpractitioners in the Hospital's Smart Contract

The hospital maintains a register of doctors within its smart contract, with verifiable information regarding their medical licenses and identification. Any new entries made to the hospital's register go through a verification stage, where the members of the consortium all verify the medical license of the doctor, and based on a vote, the entry is allowed to be appended to the hospital's mapping data structure. In the event of malpractice, the hospital simply drops the entry of the doctor in question from its smart contract. The smart contract function used for removing these entries requires digital signatures from the hospital, therefore no unauthorized member from the blockchain can access this function.

6.4 Performance Analysis

This section details the performance analysis based on the experiments we conducted to validate the proposed solution. The proposed framework was developed and deployed on a Proof-of-Authority (PoA) based blockchain. The solution was implemented on the Ethereum Rinkeby platform, which provides an accurate insight into how well the solution will perform in real-life deployments. As discussed in Sec. 6.3, the involved entities all get a limited amount of tokens to engage with the system, which hold no real-world monetary value. We used Rinkeby's native tokens to visualize the transaction fees incurred when interacting with smart contract functions.

In Sec. 5.4, we discussed the feasibility of the blockchain-IoT framework with transaction finality times, computational overhead, and transaction fees. In this chapter, the aim is to maintain a near real-time data delivery from the

patients to their doctors. For the performance analysis in this use-case, we have considered the following performance metrics:

- The computational overheads on the patient and doctor nodes for participating in the remote healthcare blockchain,
- The propagation delay involved in delivering health monitoring data over Tor.

The most notable aspect of the performance analysis was to compare the performance of our data delivery mechanism against centralized solutions and existing blockchain-based solutions. We used Telegram's API as a centralized service to compare the data delivery performances. For blockchain-based solutions, we used Ethereum's Whisper[2] protocol.

Ethereum's whisper is a communications protocol for distributed applications within the Ethereum blockchain application stack. Whisper claims to achieve 'darkness' in that it delivers messages while preventing any metadata to go through. The aim of whisper was anonymity, and while Whisper does indeed function very well in centralized clusters such as the one used by Status[3], it does not succeed in delivering messages over public PoA blockchains. The reason in this is the fact that Ethereum blockchain nodes are not economically incentivized to run the whisper protocol. So in a network such as Rinkeby, you may not have an end-to-end route where all peers in between have whisper enabled. Therefore, for whisper, we have used theoretical values as a comparison.

Messages in whisper are broadcasted over a message bus without traditional "routing". Within a whisper message's envelope, some of the parameters required are the TTL, the *"PowTime"* and *"PowTarget"*, as seen in the snippet below:

```
message:= whisperv6.NewMessage{
    Payload: []byte("123"),
    PublicKey: publicKey,
    TTL: 60,
    PowTime: 2,
    PowTarget: 2.5
    }
```

The last two parameters signify the amount of time given to perform proof-of-work on each message. This is whisper's way of preventing flooding attacks, and within public networks at scale, this is what would cause a greater delay. For this analysis, we have only considered a PowTime of 2 seconds on whisper, to be compared to the actual propagation times seen though the Telegram API and our solution based on Tor hidden services.

[2] https://github.com/ethereum/wiki/wiki/Whisper
[3] https://status.im/

6.4.1 Computational Overhead

As was the case in Sec. 5.4, we used the web browser plugin Metamask[4] and the light-client Geth[5] implementation for our decentralized application. The light-client implementation allows patients and doctors to issue transactions without needing to store a full copy of the remote healthcare blockchain. Additionally, since our nodes do not mine and validate new blocks, the computational overhead on the Metamask implementations is negligible.

6.4.2 Helthcare Data Propagation Time

To compare message delivery times, we set up virtual machines on the Google Compute Engine service. We connected these VMs to the Ethereum Rinkeby network and Tor. While delivering messages, we observed the time taken through a centralized solution (Telegram) and our proposed solution. As already mentioned, we have only considered 2 seconds of *PowTime* for Whisper (worth bearing in mind that these 2 seconds do not include any actual propagation time).

Fig. 6.3 shows the message delivery times between two nodes, one in Sydney and another in Frankfurt. Not only does this highlight the potential in globalizing healthcare, it shows that the solution proposed in this work performs comparatively to the centralized solution at a near real-time level. This experiment confirms the hypothesis of a slightly higher delivery time than the centralized solution, but a marked improvement over whisper on a public Ethereum network.

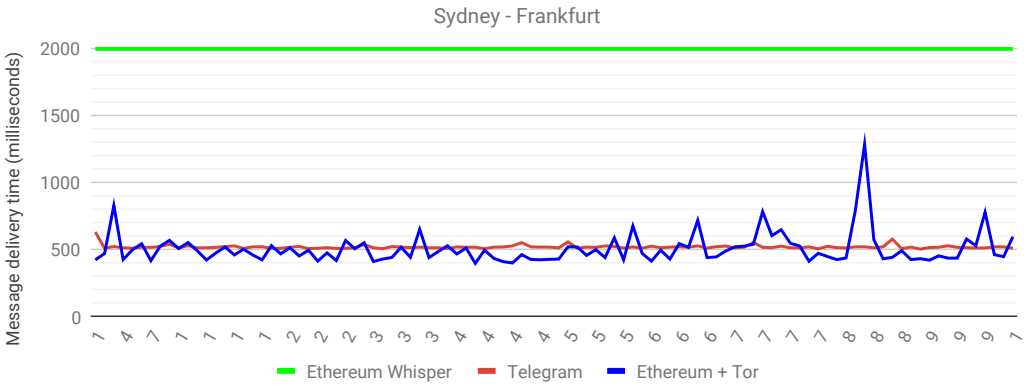


FIGURE 6.3: Message delivery times when delivering remote health monitoring data from Sydney to Frankfurt.

A more interesting outcome presents itself in Fig. 6.4, where say, a patient from Sydney wants to transfer health monitoring data to a doctor in Sydney. In this case, the centralized solution clearly has a double delivery time in milliseconds, owing to the fact that the centralized service does not have

[4] https://metamask.io/
[5] https://www.ethereum.org/cli

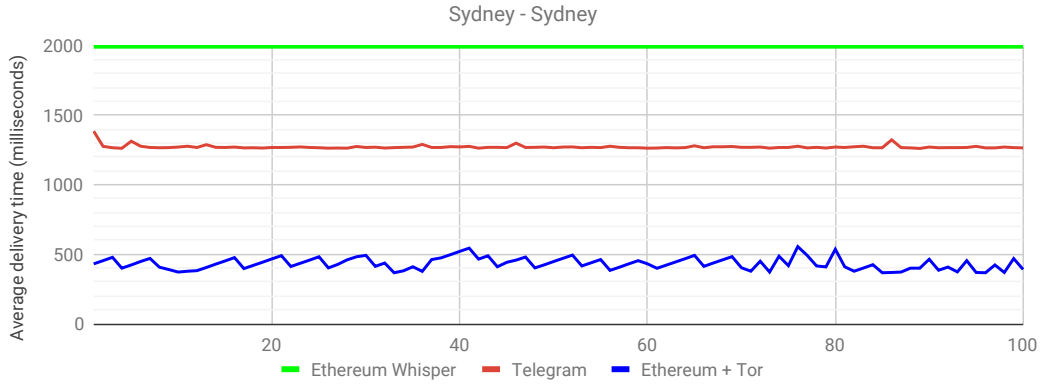


FIGURE 6.4: Message delivery times when delivering remote health monitoring data from Sydney to Frankfurt.

servers in Australia, and our proposed solution is making use of a decentralized infrastructure already in place. This experiment has helped highlight not only the capabilities of our proposed solution, but also the benefits of a decentralized architecture as a whole.

6.5 Summary

Today, a blockchain-IoT convergence is seen as a potential step forwards towards democratic and decentralized healthcare systems. However, the research area of blockchain-IoT convergence still remains nascent, though it is expected to yield several interesting outcomes for the future. In this chapter, the private-by-design, blockchain-based framework is adapted for secure end-to-end streaming in a remote health monitoring use-case. The remote health monitoring solution was built on the Ethereum blockchain platform, and Tor hidden services provided off-chain data delivery capabilities.

We demonstrated the proposed solution through a full-fledged implementation and made important observations by conducting a performance analysis. Our results showed that, in terms of message delivery times, the solution performed comparably to centralized solutions when the centralized servers were halfway in between the patient and the doctor. Moreover, our analysis showed that in scenarios where the centralized servers were distant, our solution had an improved performance due to its non-reliance on centralized servers. To conclude, by designing and implementing an on-chain and off-chain, we have taken steps towards achieving a private-by-design solution for remotely monitoring patients' health.

Part III

Conclusions and Future Work

Chapter 7

Conclusions

After achieving a truly democratic and decentralized fabric for conducting monetary transactions, blockchains are being hailed as the conduit to decentralizing the IoT. In traditional IoT architectures, centralized third parties provide essential services including authentication, authorization, access control and data management. Decentralized services with blockchains have the potential to fundamentally alter IoT service provision without reliance on centralized intermediaries. This thesis presents a blockchain-based framework for the IoT, which aims to provide secure decentralized IoT data transactioning, and mitigate the scalability and privacy issues in single blockchain solutions.

The blockchain-IoT framework presented in this thesis involves a control-plane/data-plane split within a tiered blockchain architectue, for scaling up blockchain-based security and securing the IoT edge's large attack surface. The framework is agnostic of blockchain platform used, as long as there is a permissionless blockchain at the core-tier and permissioned blockchains at the edge-tier. Thus within this multi-tiered blockchain architecture, multiple blockchains can interact with each other on the application-level to secure multiple IoT industry verticals.

The proposed architecture, when applied to an agricultural use-case demonstrates its application in supply chain traceability in general, along with sustainability in groundwater pumping through cooperation of multiple farms. The traceability system was implemented in Ethereum and Hyperledger Sawtooth, to highlight the framework's agnostic nature to the underlying blockchain platform. For the purpose of an edge-tier supply chain traceability solution, we found a permissioned blockchain implementation better suited. On the core-tier, connecting multiple farms together, a permissionless Ethereum blockchain provides a cooperational medium for farmers to reap higher long-term economic rewards, while maintaining sustainable groundwater irrigation.

The smart contracts deployed over the multiple tiers of the blockchain-based framework allow it to be tailored to multiple e-business models involving the exchange of batches of IoT data. To demonstrate this, we implemented the framework over two distinct use-cases: a decentralized IoT data marketplace

where users can monetize their data, and a connected vehicle insurance system where insurance services are delivered based on sensor data. Our implementations over the Ethereum Rinkeby test network and Hyperledger Burrow enabled us to observe how the framework would perform in real-world scenarios. Our results highlighted the feasibility for users to engage with the framework, over an existing decentralized core-tier blockchain. In terms of networking overheads, computational overheads and transaction processing fees, the cost of achieving decentralized IoT data monetization can be very low.

We demonstrated the applicability of the framework in securing end-to-end communications through a remote health monitoring use-case. This data streaming functionality does not make use of edge-tier blockchains, however, periodic hashes of the data being streamed can be stored within BigchainDB as proof of integrity. In this use-case the control plane/data plane split is maintained, with the control plane being at the core-tier blockchain, and the data plane being Tor hidden services. We made important observations in our real-life implementation by conducting a performance analysis. Our results showed that, in terms of message delivery times, the solution performed on par with centralized solutions when the centralized servers were halfway in between the patient and the doctor. Moreover, our analysis showed that in scenarios where the centralized servers were distant, our solution had an improved performance due to its non-reliance on centralized servers.

To conclude, the blockchain-based IoT framework presented in this thesis takes steps towards building a scalable and secure fabric for IoT data communications without the need for centralized trusted authorities. The framework relies on combinations of on-chain/off-chain solutions as well as tiered blockchain architectures for scalability. The framework is applicable in multiple IoT use-case scenarios, and can be tailored to fit newer, innovative e-business models. The vision of a decentralized IoT is one with device democracy, data ownership, and non-reliance on centralized authorities. Blockchains continue to inspire research progress towards realizing this vision, by offering a potential, fundamental paradigm shift on how the Internet of Things will be orchestrated in the future.

Chapter 8

Future Work

With research being an ongoing stream of collaborative consciousness, it is vital to identify where the flow can take us next. Within this work, we have presented contributions in designing decentralized architectures for the IoT using blockchains in combination with other decentralized mechanisms.

The architecture remains agnostic of the underlying consensus mechanisms used, as long as the core-tier blockchain uses a public consensus mechanism, and the edge-tier blockchain uses a private consensus mechanism. Ongoing work on blockchain consensus at the protocol level, in conjunction with our proposed framework can improve the adaptability and scalability of a blockchain-based IoT.

For improved scalability, it would be an interesting direction to have another hierarchical tier for the multi-tier blockchain architecture. A consortium of core-tier blockchains would improve the scalability by a considerable margin. The key challenge, and a great feat, would be to implement inter-blockchain routing at the higher-tier consortium, without resorting to centralized routing hubs.

Within an Ethereum implementation, after the "sharding" mechanism reaches adequate robustness, it would be interesting to see how the core-tier blockchain's scalability would improve when deployed in shards over a global test network.

In the remote health monitoring use-case, a further improvement would be to have a message buffering mechanism for offline delivery, since the Tor hidden service Ricochet requires both patient and doctor to be online during the data stream. The obvious trade-off would be in absolute decentralization versus using an autonomous broker for offline message delivery services.

Appendix A

Core-Tier Smart Contracts

A.1 Blockchain-Based Connected Vehicle Insurance

```
pragma solidity ^0.4.17;

contract BcCarInsurance {

    struct Customer {
      bool PolicyActive;
      bool isBanned;
      uint LastPayment;
      uint Premium;
      string ClaimData;
      string PremiumData;
      bool updatePending;
      uint LastUpdate;
      bool claimPending;
    }

    address owner;
    mapping (address => Customer) public Customermap;

    uint PremiumPeriod = 180 days;
    uint InitialPayment = 0.1 ether;

    // Constructor
    function BcCarInsurance() public payable{
      owner = msg.sender;
    }

    // Destructor
    function remove() public{
      if (msg.sender == owner){
        selfdestruct(owner);
      }
    }
```

```
// customer starts an insurance policy + makes initial payment
function StartPolicy() payable public {
  Customer storage cust = Customermap[msg.sender];
  // don't accept banned customers
  require(!cust.isBanned);
  // see if initial payment is correct
  require(msg.value == InitialPayment);

  cust.LastPayment = now;
  cust.PolicyActive = true;
  cust.ClaimData = "claimzero";
  cust.PremiumData = "PremiumDataZero";
}

// customer sends encrypted IPFS hash to insurance company for
    filing insurance claim (HAS COST)
function SendClaimData(string ipfsfile) public {
  Customer storage cust = Customermap[msg.sender];
  require(keccak256(cust.ClaimData) != keccak256(ipfsfile));
    require(cust.PolicyActive == true);
  cust.ClaimData = ipfsfile;
  cust.claimPending = true;
}

// customer sends encrypted IPFS hash to insurance company for
    calculating premium (HAS COST)
function SendPremiumData(string ipfsfile) public {
  Customer storage cust = Customermap[msg.sender];
  require(keccak256(cust.PremiumData)!=keccak256(ipfsfile));
  require(cust.PolicyActive == true);
  cust.PremiumData = ipfsfile;
  cust.updatePending = true;
}

// insurance company periodically checks if premium or claim data
    is updated (COSTS NOTHING)
function isPending(address customer) public constant returns (bool
    pending){
  require(msg.sender == owner);
  Customer storage cust = Customermap[customer];
  return cust.updatePending
      || cust.claimPending;
}

// checks if a customer is currently insured (COSTS NOTHING)
function isInsured(address customer) public constant returns (bool
    insured) {
  require(msg.sender == owner);
  Customer storage cust = Customermap[customer];
  return cust.PolicyActive &&
```

```
            !cust.isBanned &&
            cust.LastPayment + PremiumPeriod >= now;
    }

    // insurance company periodically performs checks if customers
       have made payment within required interval (MAY HAVE COST)
    function CheckPayment(address customer) public {
      require(msg.sender == owner);
      Customer storage cust = Customermap[customer];
      if (cust.PolicyActive && cust.LastPayment + PremiumPeriod < now) {
        cust.PolicyActive = false;
        cust.isBanned = true;
      }
    }

    // insurance company updates customer's insurance premium (HAS
       COST - include cost in premium)
    function updatePremium(address customer, uint newPremium) public {
      require(msg.sender == owner);
      Customer storage cust = Customermap[customer];
      cust.Premium = newPremium;
      cust.updatePending = false;
      cust.LastUpdate = now;
    }

    // customer pays insurance premium
    function MakePayment() public payable{
      Customer storage cust = Customermap[msg.sender];
      // only accept correct amount
      require(msg.value == cust.Premium);
      // only let payment through if premium is updated
      require(cust.LastUpdate + PremiumPeriod > now);
      // check if payment is overdue
      CheckPayment(msg.sender);
      // only accept payment if customer is insured
      require(isInsured(msg.sender));

      cust.LastPayment = now;
    }

    // insurance company processes claims based on customer's data
    function ProcessClaim(address customer) public payable{
      require(msg.sender == owner);
      require(isInsured(customer));
      Customer storage cust = Customermap[customer];
      customer.transfer(msg.value);
      cust.claimPending = false;
    }
}
```

A.2 Decentralized IoT Data Marketplace - Snippet

```
pragma solidity ^0.4.17;

contract DataMarketplace {

    struct Customer {
      bool membershipActive;
      bool isBanned;
      string fileHash;
      bool pending;
    }

    address owner;
    mapping (address => Customer) public Customermap;

    uint pricePerHour = 0.01 ether;

    // Constructor
    function DataMarketplace() public payable{
      owner = msg.sender;
    }

    // Destructor
    function remove() public{
      if (msg.sender == owner){
        selfdestruct(owner);
      }
    }

    // check price of a piece of requested data
    function CheckPrice(uint hourNum) public constant returns (uint
        price){
      return hourNum*pricePerHour;
    }

    // customer issues request for data
    function RequestData(uint hourNum) payable public {
      Customer storage cust = Customermap[msg.sender];
      // don't accept banned customers
      require(!cust.isBanned);
      // see if initial payment is correct
      require(msg.value == CheckPrice(hourNum));
      require(cust.pending=false);
      if (!cust.membershipActive){
        cust.membershipActive = true;
      }
      cust.pending = true;
    }
```

```
    // response to data requests
    function SendData(string ipfsfile, address customer) public {
      require(msg.sender == owner);
      Customer storage cust = Customermap[customer];
      require(keccak256(cust.fileHash) != keccak256(ipfsfile));
      require(cust.membershipActive == true);
      cust.fileHash = ipfsfile;
      cust.pending = false;
    }

    // periodic checks for pending requests
    function isPending(address customer) public constant returns (bool
        pending){
      require(msg.sender == owner);
      Customer storage cust = Customermap[customer];
      return cust.pending
    }

    // ban a customer from making further requests
    function banHammer(address customer) public {
      require(msg.sender == owner);
      Customer storage cust = Customermap[customer];
      cust.isBanned = true;
    }

}
```

Bibliography

[1] R. Minerva, A. Biru, and D. Rotondi. Towards a Definition of the Internet of Things (IoT). Technical report, IEEE IoT Initiative, 2015. Rev. 1.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: a survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[3] F. A. Alaba, M. Othman, I. A. Targio Hashem, and F. Alotaibi. Internet of things security: a survey. *Journal of Network and Computer Applications*, 88:10–28, 2017.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. Ddos in the iot: mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[5] E. Bertino and N. Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[6] P. Cardullo. Hacking multitude and big data: some insights from the turkish 'digital coup. *Big Data & Society*, 2(1), 2015.

[7] T. Borgohain, U. Kumar, and S. Sanyal. Survey of security and privacy issues of Internet of Things. Technical report, Harvard University, 2015.

[8] G. Greenwald and E. MacAskill. Nsa prism program taps in to user data of apple, google and others. *The Guardian*, 7(6):1–43, 2013.

[9] S. Nakamoto. Bitcoin: a peer-to-peer electronic cash system.

[10] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani. Applications of blockchains in the internet of things: a comprehensive survey. *IEEE Communications Surveys & Tutorials*, 21(2):1676–1717, 2018.

[11] P. Brody, V. Pureswaran, S. Panikkar, and S. Nair. Empowering the edge practical insights on a decentralized internet of things. *IBM Institute for Business Value. Technical Report*, 2015.

[12] N. Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[13] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[14] M. Pilkington. Blockchain technology: principles and applications. *Research Handbook on Digital Transformations*, 2015.

[15] R. Beck, J. S. Czepluch, N. Lollike, and S. Malone. Blockchain-the gateway to trust-free cryptographic transactions. In *Twenty-Fourth European Conference on Information Systems (ECIS)*, Istanbul, Turkey, 2016.

[16] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO '87*, pages 369–378, 1987.

[17] G. Wood. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[18] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI matters*, 1(2):19–21, 2014.

[19] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016.

[20] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: a technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016. ISSN: 1553-877X.

[21] V. Dhillon, D. Metcalf, and M. Hooper. The hyperledger project. In *Blockchain Enabled Applications*, pages 139–149. 2017.

[22] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Proceedings of the 6th International Conference on Principles of Security and Trust*, pages 164–186, 2017.

[23] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858, 2016.

[24] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[25] D. Dolev, C. Dwork, and L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM (JACM)*, 34(1):77–97, 1987.

[26] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*:203–213, 1998.

[27] L. Law, S. Sabett, and J. Solinas. How to make a mint: the cryptography of anonymous electronic cash. *National Security Agency Office of Information Security Research and Technology, Cryptology Division*, 1996.

[28] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[29] M. Correia, G. S. Veronese, N. F. Neves, and P. Verissimo. Byzantine consensus in asynchronous message-passing systems: a survey. *International Journal of Critical Computer-Based Systems (IJCCBS)*, 2(2):141–161, 2011. ISSN: 1757-8779.

[30] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[31] M. Liu, R. Yu, Y. Teng, V. Leung, and M. Song. Performance optimization for blockchain-enabled industrial internet of things (iiot) systems:

a deep reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, 2019.

[32] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. 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.

[33] D. Bradbury. In blocks [security bitcoin]. *Engineering Technology*, 10(2):68–71, 2015. ISSN: 1750-9637.

[34] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 280–285, 2014.

[35] N. Houy. It will cost you nothing to'kill'a proof-of-stake cryptocurrency. *Available at SSRN 2393940*, 2014.

[36] A. Jain, S. Arora, Y. Shukla, T. Patil, and S. Sawant-Patil. Proof of stake with casper the friendly finality gadget protocol for fair validation consensus in ethereum. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(3):291–298, 2018.

[37] L. S. Sankar, M. Sindhu, and M. Sethumadhavan. Survey of consensus protocols on blockchain applications. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5. IEEE, 2017.

[38] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: extending bitcoin's proof of work via proof of stake. *SIGMETRICS Performance Evaluation Review*, 42(3):34–37, Dec. 2014. ISSN: 0163-5999.

[39] V. Costan and S. Devadas. Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016:86, 2016.

[40] M. Walport. Distributed ledger technology: beyond blockchain. *UK Government Office for Science*, 2016.

[41] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[42] E. Androulaki, C. Cachin, A. De Caro, A. Kind, and M. Osborne. Cryptography and protocols in hyperledger fabric. In *Real-World Cryptography Conference*, 2017.

[43] J. Kwon. Tendermint: consensus without mining. *Draft v. 0.6, fall*, 1:11, 2014.

[44] D. Schwartz, N. Youngs, and A. Britto. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5, 2014.

[45] D. Mazieres. The stellar consensus protocol: a federated model for internet-level consensus. *Stellar Development Foundation*, 2015.

[46] M. 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.

[47] L. D. Xu, E. L. Xu, and L. Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.

[48] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac. Internet of things: vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[49] A. Barki, A. Bouabdallah, S. Gharout, and J. Traore. M2m security: challenges and solutions. *IEEE Communications Surveys & Tutorials*, 18(2):1241–1254, 2016.

[50] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos. Security and privacy for cloud-based iot: challenges. *IEEE Communications Magazine*, 55(1):26–33, 2017.

[51] M. Mohammadi and A. Al-Fuqaha. Enabling cognitive smart cities using big data and machine learning: approaches and challenges. *IEEE Communications Magazine*, 56(2):94–101, 2018.

[52] V. Gazis. A survey of standards for machine-to-machine and the internet of things. *IEEE Communications Surveys & Tutorials*, 19(1):482–511, 2017.

[53] D. S. Nunes, P. Zhang, and J. Sa Silva. A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys & Tutorials*, 17(2):944–965, 2015.

[54] M. Ammar, G. Russello, and B. Crispo. Internet of things: a survey on the security of iot frameworks. *Journal of Information Security and Applications*, 38:8–27, 2018.

[55] K. Zhao and L. Ge. A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on Computational Intelligence and Security*, pages 663–667. IEEE, 2013.

[56] J. S. Kumar and D. R. Patel. A survey on internet of things: security and privacy issues. *International Journal of Computer Applications*, 90(11), 2014.

[57] Z. Yan, P. Zhang, and A. V. Vasilakos. A survey on trust management for internet of things. *Journal of network and computer applications*, 42:120–134, 2014.

[58] N. Z. Aitzhan and D. Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 2016.

[59] A. Laszka, A. Dubey, M. Walker, and D. Schmidt. Providing privacy, safety, and security in iot-based transactive energy systems using distributed ledgers. In *Proceedings of the Seventh International Conference on the Internet of Things*, page 13. ACM, 2017.

[60] F. Knirsch, A. Unterweger, G. Eibl, and D. Engel. Privacy-preserving smart grid tariff decisions with blockchain-based smart contracts. In *Sustainable Cloud and Energy Services*, pages 85–116. Springer, 2018.

[61] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. *IEEE Transactions on Industrial Informatics*, 13(6):3154–3164, 2017. ISSN: 1551-3203.

[62] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3690–3700, 2018. ISSN: 1551-3203.

[63] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: vision and challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5):37–42, Sept. 2015. ISSN: 0146-4833.

[64] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. Blockchain for iot security and privacy: the case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.

[65] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh. A blockchain connected gateway for ble-based devices in the internet of things. *IEEE Access*, 2018.

[66] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 2018.

[67] K. Wüst and A. Gervais. Do you need a blockchain? *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*:45–54, 2018.

[68] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[69] H. Tschofenig and J. Arkko. Report from the Smart Object Workshop. Apr. 2012. URL: RFC6574. last accessed: February 8, 2020.

[70] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: the road ahead. *Computer networks*, 76:146–164, 2015.

[71] J. A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[72] O. Vermesan and P. Friess. *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[73] R. Roman, P. Najera, and J. Lopez. Securing the internet of things. *Computer*, 44(9):51–58, 2011.

[74] R. H. Weber. Internet of things: privacy issues revisited. *Computer Law & Security Review*, 31(5):618–627, 2015.

[75] G. V. Lioudakis, E. A. Koutsoloukas, N. Dellas, S. Kapellaki, G. N. Prezerakos, D. I. Kaklamani, and I. S. Venieris. A proxy for privacy: the discreet box. In *EUROCON, 2007. The International Conference on" Computer as a Tool"*, pages 966–973. IEEE, 2007.

[76] D. Chaum and E. Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[77] F. Li, Z. Zheng, and C. Jin. Secure and efficient data transmission in the internet of things. *Telecommunication Systems*, 62(1):111–122, 2016.

[78] K. El Defrawy and G. Tsudik. Prism: privacy-friendly routing in suspicious manets (and vanets). In *2008 IEEE International Conference on Network Protocols*, pages 258–267. IEEE, 2008.

[79] A. Wasef and X. Shen. Efficient group signature scheme supporting batch verification for securing vehicular networks. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010.

[80] X. Zhu, S. Jiang, L. Wang, and H. Li. Efficient privacy-preserving authentication for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 63(2):907–919, 2014.

[81] L. Malina, A. Vives-Guasch, J. Castellà-Roca, A. Viejo, and J. Hajny. Efficient group signatures for privacy-preserving vehicular networks. *Telecommunication Systems*, 58(4):293–311, 2015.

[82] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10 -(05):557–570, 2002.

[83] J. Domingo-Ferrer and V. Torra. A critique of k-anonymity and some of its enhancements. In *2008 Third International Conference on Availability, Reliability and Security*, pages 990–993. IEEE, 2008.

[84] J. Su, D. Cao, B. Zhao, X. Wang, and I. You. Epass: an expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things. *Future Generation Computer Systems*, 33:11–18, 2014.

[85] A. Alcaide, E. Palomar, J. Montero-Castillo, and A. Ribagorda. Anonymous authentication for privacy preserving iot target-driven applications. *Computers & Security*, 37:111–123, 2013.

[86] X.-J. Lin, L. Sun, and H. Qu. Insecurity of an anonymous authentication for privacy-preserving iot target-driven applications. *Computers & Security*, 48:142–149, 2015.

[87] Y. Zhang and J. Wen. The iot electric business model: using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications*, 10(4):983–994, 2017.

[88] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 523–533. Springer, 2017.

[89] S. Huckle, R. Bhattacharya, M. White, and N. Beloff. Internet of things, blockchain and shared economy applications. *Procedia Computer Science*, 98:461–466, 2016.

[90] T. Hardjono and N. Smith. Cloud-based commissioning of constrained devices using permissioned blockchains. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 29–36. ACM, 2016.

[91] M. A. Walker, A. Dubey, A. Laszka, and D. C. Schmidt. Platibart: a platform for transactive iot blockchain applications with repeatable testing. In *Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*, pages 17–22. ACM, 2017.

[92] Y. Rahulamathavan, R. C.-W. Phan, M. Rajarajan, S. Misra, and A. Kondoz. Privacy-preserving blockchain based iot ecosystem using attribute based encryption. In *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2017.

[93] N. Foukia, D. Billard, and E. Solana. Pisces: a framework for privacy by design in iot. In *Privacy, Security and Trust (PST), 2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 706–713. IEEE, 2016.

[94] P. K. Sharma, M.-Y. Chen, and J. H. Park. A software defined fog node based distributed blockchain cloud architecture for iot. *IEEE Access*, 2017.

[95] G. Zyskind, O. Nathan, et al. Decentralizing privacy: using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.

[96] M. Conoscenti, A. Vetro, and J. C. De Martin. Peer to peer for privacy and decentralization in the internet of things. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 288–290. IEEE, 2017.

[97] M. S. Ali, K. Dolui, and F. Antonelli. Iot data privacy via blockchains and ipfs. In *Proceedings of the Seventh International Conference on the Internet of Things*, page 14. ACM, 2017.

[98] J. Granjal, E. Monteiro, and J. S. Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[99] H. Suo, J. Wan, C. Zou, and J. Liu. Security in the internet of things: a review. In *2012 international conference on computer science and electronics engineering*, volume 3, pages 648–651. IEEE, 2012.

[100] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): a vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[101] S. Sicari, A. Rizzardi, C. Cappiello, D. Miorandi, and A. Coen-Porisini. Toward data governance in the internet of things. In *New advances in the internet of things*, pages 59–74. Springer, 2018.

[102] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar. A critical analysis on the security concerns of internet of things (iot). *International Journal of Computer Applications*, 111(7), 2015.

[103] H. Halpin and M. Piekarska. Introduction to security and privacy on the blockchain. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 1–3. IEEE, 2017.

[104] L. M. Axon and M. Goldsmith. PB-PKI: a privacy-aware blockchain-based PKI. Technical report, University of Oxford, 2016.

[105] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964, 2016.

[106] O. Novo. Blockchain meets iot: an architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 2018.

[107] H. Shrobe, D. L. Shrier, and A. Pentland. *Chapter 15 enigma: decentralized computation platform with guaranteed privacy*. In *New Solutions for Cybersecurity*. MITP, 2018, pages 425–454.

[108] T. Le and M. W. Mutka. Capchain: a privacy preserving access control framework based on blockchain for pervasive environments. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 57–64. IEEE, 2018.

[109] H. Es-Samaali, A. Outchakoucht, and J. P. Leroy. A blockchain-based access control for big data. *International Journal of Computer Networks and Communications Security*, 5(7):137, 2017.

[110] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy. Towards blockchain-based auditable storage and sharing of iot data. In *Proceedings of the 2017 on Cloud Computing Security Workshop*, pages 45–50. ACM, 2017.

[111] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. Lsb: a lightweight scalable blockchain for iot security and anonymity. *Journal of Parallel and Distributed Computing*, 2019.

[112] J. H. Park and J. H. Park. Blockchain security in cloud computing: use cases, challenges, and solutions. *Symmetry*, 9(8):164, 2017.

[113] K. Biswas and V. Muthukkumarasamy. Securing smart cities using blockchain technology. In *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems*, pages 1392–1393, 2016.

[114] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu. Blockchain based data integrity service framework for iot data. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 468–475. IEEE, 2017.

[115] B. Lee and J.-H. Lee. Blockchain-based secure firmware update for embedded devices in an internet of things environment. *The Journal of Supercomputing*, 73(3):1152–1167, 2017.

[116] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey. Towards better availability and accountability for iot updates by means of a blockchain. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 50–58, 2017.

[117] M. Steger, A. Dorri, S. S. Kanhere, K. Römer, R. Jurdak, and M. Karner. Secure wireless automotive software updates using blockchains: a proof of concept. In *Advanced Microsystems for Automotive Applications 2017*, pages 137–149. Springer, 2018.

[118] O. Alphand, M. Amoretti, T. Claeys, S. Dall'Asta, A. Duda, G. Ferrari, F. Rousseau, B. Tourancheau, L. Veltri, and F. Zanichelli. Iotchain: a blockchain security architecture for the internet of things. In *Wireless Communications and Networking Conference (WCNC), 2018 IEEE*, pages 1–6. IEEE, 2018.

[119] M. Vučinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti. Oscar: object security architecture for the internet of things. *Ad Hoc Networks*, 32:3–16, 2015.

[120] R. B. Chakraborty, M. Pandey, and S. S. Rautaray. Managing computation load on a blockchain–based multi–layered internet–of–things network. *Procedia Computer Science*, 132:469–476, 2018.

[121] A. Bahga and V. K. Madisetti. Blockchain platform for industrial internet of things. *Journal of Software Engineering and Applications*, 9(10):533, 2016.

[122] J. Hughes and E. Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*:29–38, 2005.

[123] D. Recordon and D. Reed. OpenID 2.0: a platform for user-centric identity management. In *Proc. of the Second ACM workshop on Digital identity management*, pages 11–16, 2006.

[124] D. Hardt. The oauth 2.0 authorization framework, 2012.

[125] F. Imbault, M. Swiatek, R. de Beaufort, and R. Plana. The green blockchain: managing decentralized energy production and consumption. In *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–5. IEEE, 2017.

[126] S. Kikitamara, M. van Eekelen, and D. I. J.-P. Doomernik. *Digital Identity Management on Blockchain for Open Model Energy System*. Master's thesis, Radboud Universiteit, 2017.

[127] M. Samaniego and R. Deters. Hosting virtual iot resources on edge-hosts with blockchain. In *Computer and Information Technology (CIT), 2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 116–119. IEEE, 2016.

[128] D. W. Kravitz and J. Cooper. Securing user identity and transactions symbiotically: iot meets blockchain. In *Global Internet of Things Summit*, pages 1–6, 2017.

[129] S. Huh, S. Cho, and S. Kim. Managing iot devices using blockchain platform. In *19th International Conference on Advanced Communication Technology (ICACT)*, pages 464–467, 2017.

[130] J.-H. Lee. Bidaas: blockchain based id as a service. *IEEE Access*, 6:2274–2278, 2018.

[131] P. Urien. Towards secure elements for trusted transactions in blockchain and blochchain IoT (BIoT) Platforms. In *Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, pages 1–5, 2018.

[132] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*, 2015.

[133] D. Shrier, W. Wu, and A. Pentland. Blockchain & infrastructure (identity, data security). *MIT Connection Science*:1–18, 2016.

[134] F. Wang, S. Liu, P. Liu, and Y. Bai. Bridging physical and virtual worlds: complex event processing for rfid data streams. In *International Conference on Extending Database Technology*, pages 588–607. Springer, 2006.

[135] M. Ma, P. Wang, and C.-H. Chu. Data management for internet of things: challenges, approaches and opportunities. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things*

(iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pages 1144–1151. IEEE, 2013.

[136] X. Hao, P. Jin, and L. Yue. Efficient storage of multi-sensor object-tracking data. *IEEE Transactions on Parallel and Distributed Systems*, 27(10):2881–2894, 2016.

[137] T. Lu, J. Fang, and C. Liu. A unified storage and query optimization framework for sensor data. In *Web Information System and Application Conference (WISA), 2015 12th*, pages 229–234. IEEE, 2015.

[138] I. P. Zarko, K. Pripuzic, M. Serrano, and M. Hauswirth. Iot data management methods and optimisation algorithms for mobile publish/-subscribe services in cloud environments. In *Networks and Communications (EuCNC), 2014 European Conference on networks and communications (EuCNC)*, pages 1–5. IEEE, 2014.

[139] X. Liang, J. Zhao, S. Shetty, and D. Li. Towards data assurance and resilience in iot using blockchain. *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*:261–266, 2017.

[140] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. Medrec: using blockchain for medical data access and permission management. In *Open and Big Data (OBD), International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE, 2016.

[141] E. S. Hamza, A. OUTCHAKOUCHT, and J. P. LEROY. A blockchain-based access control for big data. *International Journal of Computer Networks and Communications Security*, 5(7):137–147, 2017.

[142] T. Nugent, D. Upton, and M. Cimpoesu. Improving data transparency in clinical trials using blockchain smart contracts. *F1000Research*, 5, 2016.

[143] N. Ansari and X. Sun. Mobile edge computing empowers internet of things. *IEICE Transactions on Communications*, 101(3):604–619, 2018.

[144] K. Dolui and C. Kiraly. Towards multi-container deployment on iot gateways. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[145] M. S. Ali, M. Vecchio, and F. Antonelli. Enabling a blockchain-based iot edge. *IEEE Internet of Things Magazine*, 1(2):24–29, 2018.

[146] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto. BigchainDB: a Scalable Blockchain Database. Technical report, BigChainDB, 2016.

[147] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang. Blockchain challenges and opportunities: a survey. *Work Pap.–2016*, 2016.

[148] E. Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. Master's thesis, The University of Guelph, 2016.

[149] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda. Blockchain-based traceability in agri-food supply chain management: a practical implementation. In *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*, pages 1–4. IEEE, 2018.

[150] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis. Iot in agriculture: designing a europe-wide large-scale pilot. *IEEE Communications Magazine*, 55(9):26–33, 2017.

[151] A. Ursitti, G. Giannoccaro, M. Prosperi, E. De Meo, and B. de Gennaro. The magnitude and cost of groundwater metering and control in agriculture. *Water*, 10(3):344, 2018.

[152] A. D. Roy and T. Shah. Socio-ecology of groundwater irrigation in india. *Intensive use of groundwater challenges and opportunities*:307–335, 2002.

[153] H. A. Loáiciga. Analytic game—theoretic approach to ground-water extraction. *Journal of Hydrology*, 297(1-4):22–33, 2004.

[154] G. Hardin. The tragedy of the commons. *science*, 162(3859):1243–1248, 1968.

[155] S. Husnjak, D. Peraković, I. Forenbacher, and M. Mumdziev. Telematics system in usage based motor insurance. *Procedia Eng.*, 100:816–825, 2015.

[156] S. Derikx, M. de Reuver, and M. Kroesen. Can privacy concerns for insurance of connected cars be compensated? *Electronic markets*, 26(1):73–81, 2016.

[157] V. Gramoli. On the danger of private blockchains. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL)*, 2016.

[158] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[159] C. Agbo, Q. Mahmoud, and M. Eklund. Blockchain technology in healthcare: a systematic review. In *Healthcare*, volume 7 of number 2, page 56. Multidisciplinary Digital Publishing Institute, 2019.

[160] M. J. Deen. Information and communications technologies for elderly ubiquitous healthcare in a smart home. *Personal and Ubiquitous Computing*, 19(3-4):573–599, 2015.

[161] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[162] R. Wendolsky, D. Herrmann, and H. Federrath. Performance comparison of low-latency anonymisation services from a user perspective. In *International Workshop on Privacy Enhancing Technologies*, pages 233–253. Springer, 2007.

[163] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

[164] A. Biryukov, I. Pustogarov, and R.-P. Weinmann. Trawling for tor hidden services: detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy*, pages 80–94. IEEE, 2013.

[165] M. T. Saletu, S. T. Kotzian, A. Schwarzinger, S. Haider, J. Spatt, and B. Saletu. Home sleep apnea testing is a feasible and accurate method to diagnose obstructive sleep apnea in stroke patients during in-hospital rehabilitation. *Journal of Clinical Sleep Medicine*, 14(09):1495–1501, 2018.

[166] E. Ratti, S. Waninger, C. Berka, G. Ruffini, and A. Verma. Comparison of medical and consumer wireless eeg systems for use in clinical trials. *Frontiers in human neuroscience*, 11:398, 2017.