CONTRASTING IOT DATA SECURITY SOLUTIONS

by

Bishal Thapa, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
August 2023

Committee Members:

Stan McClellan, Chair

Damian Valles

Semih Aslan

Andres Carvallo

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my advisor and committee chair, Dr. McClellan, for his continuous support throughout the last 2 years. I am truly appreciative of his efforts in providing various research opportunities in his lab. His supervision has been key to my academic growth.

I would also like to thank Dr. Valles for his extremely valuable support throughout my graduate career. His guidance and advice have helped me become a better research student. I am also grateful to Dr. Aslan for assisting me in participating in research conferences that helped me to engage with prominent figures within my research field. I also wish to thank Prof. Carvallo for helping me get involved in different interesting research projects in the Star labs. Finally, I want to thank Dr. Droopad and Dr. Viswanathan for their unwavering support and continuous guidance over the last 2 years.

I also want to acknowledge “Sertainty Corporation” for providing us with the platform to experiment with their proprietary technology. This truly has been a wonderful research experience.

I am immensely grateful to my parents who taught me to always be kind, and my brother and sister who are not only my biggest critics but also my biggest supporter and source of inspiration. Their unconditional love and support throughout my academic ventures has made me who I am. I also deeply appreciate my muma, buba, and my 13 months old nephew, who have been my biggest sources of inspiration.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES viii

LIST OF ABBREVIATIONS x

ABSTRACT xii

CHAPTER

1. INTRODUCTION 1

    1.1. IoT and IIoT 1
        1.1.1. Applications and Importance 1
        1.1.2. Challenges 2
    1.2. Thesis Problem Statement 4
        1.2.1. Background and Motivation 4
        1.2.2. Associated Issues 6
    1.3. Thesis Structure 7

2. LITERATURE REVIEW 8

    2.1. IoT Architecture 10
    2.2. IoT Data Security. 12
        2.2.1. Encryption 14
            2.2.1.1. Symmetric Key Encryptions 16
            2.2.1.2. Asymmetric Key Encryptions 17
        2.2.2. Physical Unclonable Function (PUF) 20
        2.2.3. Zero-Trust Architecture for IoT 21
    2.3. Blockchain 22
        2.3.1. Implementations in IoT 23
        2.3.2. Issues involving blockchain adoption for IoT. 27
            2.3.2.1.Scalability 28
            2.3.2.2.Complexity and Processing constraints 29
            2.3.2.3.Security, Transparency, and Privacy Issues 29
    2.4. Blockchain architecture used for experiments. 30
    2.5. Intelligent Cipher Transfer Object (ICTO) 31
        2.5.1. Unbreakable Exchange Protocol (UXP) 35
        2.5.2. Possible implementations in IoT 36
        2.5.3. Drawbacks 38
    2.6. Blockchain and ICTO as possible solutions 39

3. EXPERIMENTAL SETUP 41

3.1. Technology Description 41
3.2. Setup 42

4. RESULTS AND DISCUSSIONS 46

4.1. Comparison of Blockchain and UXP relative to constrained devices' storage space 46
4.1.1. Theoretical Analysis 47
4.1.2. Experimental Analysis 47
4.2. UXP Overhead 52
4.3. Effect of Blockchain and ICTO on IoT Device Memory Storage 55
4.4. Execution Time 57
4.5. CPU Clock Cycles 60
4.6. Resource measurement 65
4.6.1. RSS Memory 66
4.6.2. VSZ Memory 69
4.7. Network Analysis 72
4.8. Memory Footprint 76

5. CONCLUSIONS AND FUTURE WORK 79

5.1. Summary 80
5.2. UXP technology for IoT 87
5.3. Future Works 88
5.4. Open-Source Implementation 88
5.5. Recommendation for ICTO development 89

REFERENCES 90

LIST OF TABLES

Page

Table 1. Comparison between RSA and ECC [57, 61]..... 19

Table 2. Summary of output size information 48

Table 3. Information about difference in block size and UXP Size. 51

Table 4. UXP Overhead incurred for encapsulating compressible and incompressible payload. 53

Table 5. Comparison of blockchain and UXP encapsulating compressible payload in terms of memory space consumption..... 56

Table 6. Comparison of blockchain and UXP encapsulating incompressible payload in terms of memory space consumption..... 56

Table 7. Execution time measurements for compressible payload..... 58

Table 8. Execution time measurements for incompressible payloads. 59

Table 9. CPU clock cycle measurements for compressible payloads..... 61

Table 10. CPU clock cycle measurements for incompressible payloads..... 61

Table 11. CPU clock cycle measurements of incompressible payloads in Raspberry Pi. 64

Table 12. Summary of RSS memory measurements of blockchain and ICTO experiment for compressible payloads 67

Table 13. Summary of RSS memory measurements of blockchain and ICTO experiment for incompressible payloads 67

Table 14. Raw data of VSZ memory measurements of blockchain, ledger, and ICTO experiment for compressible payloads..... 70

Table 15. Raw data of VSZ memory measurements of blockchain, ledger, and ICTO experiment for incompressible payload 70

Table 16. Size comparison..... 73

Table 17. TCP Header length measurements..... 73

Table 18. TCP latency measurements..... 75

Table 19. Memory footprint comparison for ASCON, ECC, and UXP 76

Table 20. Comparisons of blockchain, ledger, and UXP..... 87

LIST OF FIGURES

Page

Figure 1. IoT connected devices with forecasts for future years [7] 2

Figure 2. Three-Layer IoT Architecture [38] 11

Figure 3. IoT Security [45] 13

Figure 4. Lightweight cryptographic diagram [50] 16

Figure 5. Challenges of Integrating Blockchain in IoT Applications. 28

Figure 6. Self-Protecting Intelligence Module 32

Figure 7. Portable Dynamic Rule Set (PDRS) 36

Figure 8. Flowcharts depicting ICTO creation and method to access the ICTO. 34

Figure 9. General overview of UXP Object creation [75] 36

Figure 10. Blockchain Setup 43

Figure 11. UXP Setup 43

Figure 12. Fitted line for blockchain and UXP storing compressible payload with training and testing datasets depicting Equation 2 and Equation 3. 49

Figure 13. Fitted line for blockchain and UXP storing incompressible payload with training and testing datasets depicting Equation 4 and Equation 5. 50

Figure 14. Difference in UXP Size vs Block Size 52

Figure 15. Bar graph showing UXP overhead for different incompressible payload sizes. 54

Figure 16. A bar chart showing the number of blocks before blockchain takes more storage than UXP 57

Figure 17. Blockchain vs UXP vs ledger in terms of time required to store payload in general purpose machine. 59

Figure 18. Blockchain vs ledger in terms of time required to store payload in Raspberry Pi. 60

Figure 19. Comparison of CPU clock cycles for blockchain, ICTO, and ledger in general purpose machine. 62

Figure 20. Comparison of CPU clock cycles for blockchain, and ledger in Raspberry Pi. 63

Figure 21. CPU clock cycles requirement for different difficulty levels in Raspberry Pi. 64

Figure 22. RSS memory comparison for blockchain, ledger, and ICTO on general purpose machine. 68

Figure 23. RSS memory requirement of blockchain and ledger on virtual box. 69

Figure 24. VSZ memory comparison for blockchain, ledger, and ICTO on general purpose machine. 71

Figure 25. VSZ memory requirement of UXP, blockchain, and ledger on ubuntu virtual box. ... 72

Figure 26. Cumulative Header Size vs Payload Size 74

Figure 27. Network Latency vs Payload Size 75

Figure 28. Memory footprint comparison for compressible payloads 77

Figure 29. Memory footprint comparison for incompressible payloads 77

LIST OF ABBREVIATIONS

Abbreviation	Description
AAA	Authentication, Authorization, and Accounting
ABE	Attribute-Based Encryption
AES	Advanced Encryption Standard
AMQP	Advanced Message Queuing Protocol
CoAP	Constrained Application Protocol
DHT	Distributed Hash Tables
DLT	Distributed Ledger Technology
DPoS	Delegated Proof of Stake
ECC	Elliptical Curve Cryptography
ECDHE	Elliptic curve Diffie-Hellman Ephemeral
FPGA	Field-Programmable Gate Array
GSM	Global System for Mobile Communications
ICTO	Intelligent Cipher Transfer Object
IIoT	Industrial Internet of Things
IoT	Internet of Things
LoRaWAN	Long Range Wide Area Network
LTE	Long Tern Evolution
LZ	Lempel and Ziv
MITM	Man in the Middle
MQTT	Message Queuing Telemetry Transport
NIST	National Institute of Standards and Technology

OPC UA	Open Platform Communications United Architecture
OS	Operating system
PBFT	Practical Byzantine Fault Tolerance
PDRS	Portable Dynamic Rule Set
PKI	Public Key Infrastructure
PoS	Proof of Stake
PoW	Proof of Work
PUF	Physical Unclonable Function
RAM	Random Access Memory
REST	Representational State Transfer
RSA	Rivest Shamir Adleman
RSS	Resident Set Size
SDK	Software Development Kit
SELCOM	Selective Compression Algorithm
TCEQ	Texas Commission on Environmental Quality
UXP	Unbreakable Exchange Protocol
VD	Value Density
VSZ	Virtual Memory Size
XMPP	Extensible Messaging and Presence Protocol
ZTAA	Zero Trust Application Access
ZTDA	Zero Trust Data Access
ZTNA	Zero Trust Network Access

ABSTRACT

The rapid proliferation of IoT applications has raised concerns regarding the privacy and security of the generated data. These concerns pose a significant challenge, especially in resource constrained IoT endpoints that face limitations in implementing robust security mechanisms due to their limited resources. To tackle these challenges, various data security solutions integrating edge computing and cloud computing have emerged. This research focuses on exploring technologies that can be deployed at the edge of the IoT network, aiming to enhance data security in a limited resource environment.

The research explores the Intelligent Cipher Transfer Object (ICTO) technology and its viability in resource constrained IoT applications. The paper primarily focuses on contrasting the ICTO technology with the core blockchain technology. Additionally, these technologies are also compared with a blockchain with proof of work implementation as a consensus mechanism. The paper aims to uncover the trade-offs, benefits, and limitations associated with each solution, considering factors such as computational overhead, memory utilization, complexity, and network overhead. Moreover, the research explores the potential application of these technologies within the resource constrained IoT domain. The analysis considers the unique challenges and constraints posed by resource constrained IoT devices. This research aims to contribute to the understanding of viability and potential of aforementioned technologies in resource constrained environment.

Keywords: ICTO, blockchain, ledger, constrained environment

1. INTRODUCTION

1.1. IoT and IIoT

The “Internet of Things” (IoT) is a network of everyday objects and machines that are embedded with sensors and digital devices which can communicate with each other and collect data. The concept of sensors to objects was first implemented in the 1980s [1], and the phrase “ the Internet of Things” was coined in 1999 by Kevin Aston [2]. Today IoT is highly prevalent and practical as a result of recent developments in several technologies [3]. The IoT in its most basic form can be considered a network of interconnected devices that can exchange information using the Internet or Internet-capable protocols [4] . Forbes defines IoT as “a giant network of interconnected things, and the relationship can be between people-people, people-things, and things-things” [5]. The “Industrial Internet of Things” (IIoT) leverages the technologies of IoT and adapts them to the relatively more complex requirements of industrial environments. As such, IIoT can be considered as a subset of IoT which primarily focuses on industrial applications.

1.1.1. Applications and Importance

IoT is one of the fastest-growing technological fields. New innovative solutions are being developed at an accelerating rate to address contemporary issues like smart cities, smart agriculture, smart homes, and manufacturing. IoT solutions proliferate rapidly in various domains, often because of application-specific advancements in technology. Areas most applicable to IoT implementations include home automation, healthcare, smart agriculture, smart city, smart grid, wearables, finance, manufacturing, supply care and logistics, hospitality, and transportation. [6]

Developments in wireless networks and next generation cellular networks are accelerating the adoption of IoT and IoT-driven applications. As of 2022, 13.14 billion IoT devices were connected worldwide in 2022, and this number is expected to be more than 29 billion by 2030 as depicted in Figure 1 [7]. The possibilities with IoT are endless and nearly all our virtual and physical objects in the environment can be interconnected with proper IoT architecture.

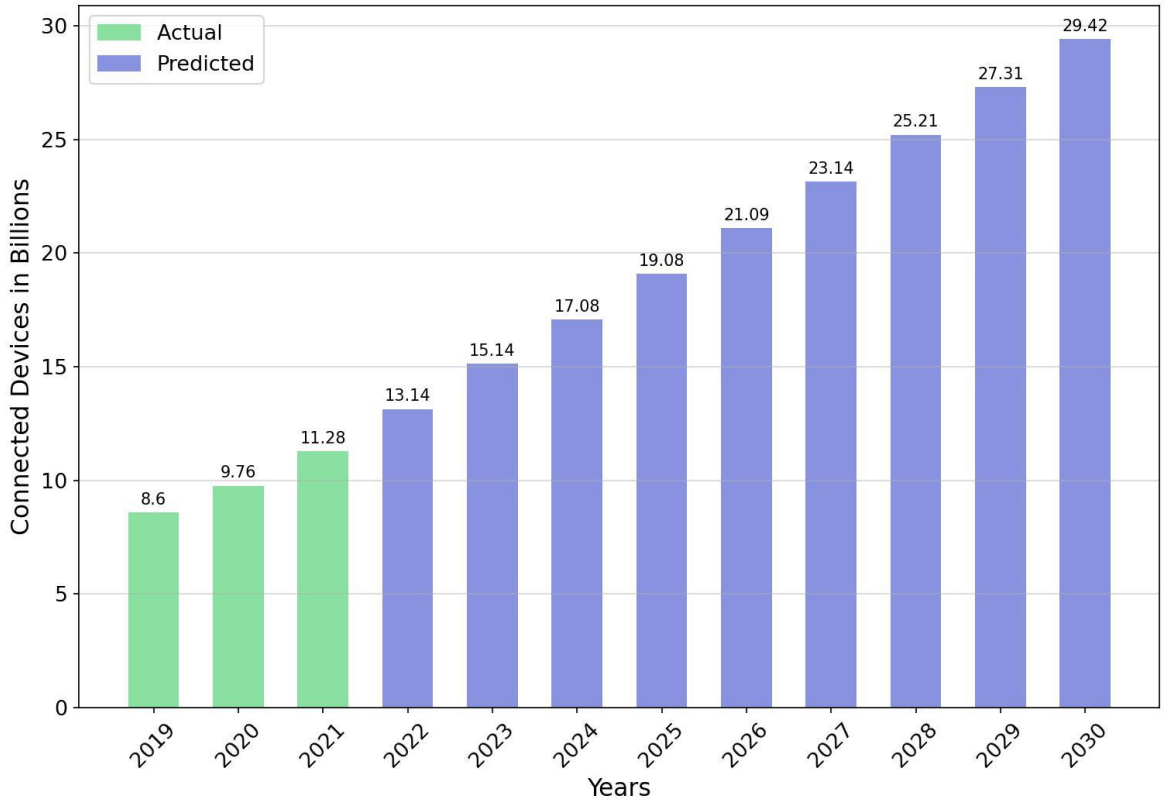


Figure 1. IoT connected devices with forecasts for future years [7]

1.1.2. Challenges

IoT is a booming industry, but various challenges and constraints need to be dealt with while implementing IoT architectures. Areas including Big Data, networking, scalability, heterogeneity, interoperability, security and privacy are amongst the most prevalent IoT system concerns [8]. Among these concerns, security and privacy issues are quite significant [9, 10].

Data present in IoT deployments that are not properly secured and protected are at risk of being exploited. The severity of such outcomes may outweigh the benefits of IoT as a viable technology platform [9].

IoT deployments handle large amounts of data. The data may be collected from the physical world simple sensors embedded on devices and machines gathering information like motion, humidity, querying location, temperature, etc. [10]. The data from such sensors may not be trustworthy for a variety of reasons. For example, IoT devices are often constrained by energy and computational capability and these devices deployed in remote locations can easily be tampered with as they are mostly equipped with few physical security mechanisms [11]. As the number of such high-risk devices connected to any network increases, the chances for exploitation in security also increase. Because of the low cost of such devices, the barrier to entry is quite low. As a result, the IoT space natively exposes a low cost of entry coupled with a potentially high cost of compromise [11]. This is a dangerous combination.

IoT networks constantly deal with several security issues, including authentication, authorization, information leakage, privacy, verification, tampering, jamming, and eavesdropping [11]. Issues such as tampering, denial of service, and sensors as security threats have been studied and discussed in the literature [12]. The various types of attacks in application layer, network layer, and sensing layer of IoT network is described thoroughly in [13, 14]. The alteration of IoT data, physical modification of the device by an attacker, jamming of the radio signal used by IoT devices to communicate, and distributed denial of service are some of the constant security concerns [15]. Moreover, the IoT network layer is vulnerable to several persistent threats. Targeted attacks on networking resources, purposefully created collision, spoofing on network traffic, selective forwarding, sinkhole attacks, wormhole attacks, sybil

attacks, flooding, and node replication [12] are some of the network level threats IoT deployments will certainly encounter. While it may seem easier to prioritize the highest level of security for any IoT application, it's important to consider that high security often comes at the expense of expensive hardware, cost, and overall efficiency, particularly for lightweight IoT devices. Because of the numerous potential attacks in the IoT system, selecting an ideal technology that emphasizes security as required for application without compromising efficiency can be quite challenging. Thus, it becomes crucial to find a balance between security and efficiency, by understanding the trade-offs involved.

1.2. Thesis Problem Statement

1.2.1. Background and Motivation

IIoT is rapidly transforming different industries by enabling more automation and efficiency for industrial processes, and by connecting a myriad of sensors, actuators, and machines in manufacturing plants, healthcare, oil and gas production, and other sectors [13]. Many existing industrial processes are structured around manual data collection. In such scenarios, a worker may take a sample or collect a specimen from an ongoing process and record parameters related to the specimen to make sure the process is functioning correctly, the outcomes of the process are within tolerance, or other aspects of the process are permissible according to regulatory requirements. In such scenarios, the accuracy, reliability, and provenance of the data and/or sequence of data samples is critical.

One such practical scenario is the "Ingram Project" [14] which automates the collection of certain environmental data. This project focuses on the accuracy, reliability, and provenance of data from processes associated with the production of cement. Cement is a binding agent used

in construction that holds the materials together. Cements are dry powders that act as glue [16]. Cement is one of the oldest building materials and modern cement is considered an important material in construction business because of its reliability and versatility [17]. Cement is a major ingredient of concrete, which is a vital material for different engineering structures like buildings, roads, dams, and bridges and used for decorative purposes as well like patios, staircases, sculptures, and tables [18]. Because of its immense use in construction, the cement industry is ever increasing and implementation of modern IoT has made monitoring different variables easier and more efficient. Quality control processes associated with the production of cement are largely manual and present an opportunity for industry-specific implementation of an IoT-based solution. However, the security of such data and potential penalties associated with corrupt or errant data can be significant. Such cement plants contain essential data that must be accurate and trustworthy because the information from critical data plays a vital role in the success of the plants.

When it rains, stormwater can wash away loose soil as well as various chemicals present in and around a cement plant. This storm water mixed with such debris will eventually reach the sewer systems or water sources like rivers and lakes, polluting the water sources which can affect large populations. Organizations like the US Environmental Protection Agency (EPA) control such activities by mandating a Stormwater Pollution Prevention Plan (SWPPP) for the management and regulatory oversight of such processes [19]. Hence, regular visual as well as quantitative inspections are carried out regarding the pollutant sources. For such inspections, a specific stormwater outflow must be sampled, examined, and maintained in an indelible record for attestation of regulatory compliance. The stormwater pollutant data must be maintained by the plant for inspection. In Texas, this data must be presented when demanded by the Texas

Commission on Environmental Quality (TCEQ) [20]. TCEQ has different regulations that involve monitoring and controlling the stormwater outflow in such plants and could fine manufacturing plants as high as $25,000 per day if such regulations are not met [21]. Automating such processes with reliable, verifiable, security-conscious implementations can be an important touchstone in the validation of IoT deployments and constituent technologies. These issues first came to light as the result of a Sr. Design Project involving Ingram Ready Mix Cement plants [14]. As a result, we refer to this type of deployment and technology proof-of-concept as the "Ingram Project."

1.2.2. Associated Issues

There are various issues that need to be addressed when implementing an IIoT architecture in projects such as the Ingram Project. The sampling of critical data such as stormwater runoff in cement or concrete plants is performed manually to ensure data reliability. An employee physically has to go to the site to collect the sample after rainfall within 30 minutes to comply with the TCEQ regulations [20]. The collected data is secured and returned to a test facility for analysis and validation. In addition to being cumbersome, this process is prone to human error, difficult to scale, and complicated to secure. A preferable strategy would be to automate the data collection using a network of digital sensors and a scalable, secure data repository. However, such data needs to be highly reliable and accurate from the point of sampling through collection and after being situated in a repository. As a result, a question of integrity arises when relying on data from such digital sensors deployed in an IoT environment.

Devices like sensors, and actuators need to be properly authenticated, protected, and maintained properly. These devices are the source of information generation and these large number of smart devices need to operate for a long period of time and continuously in some

scenarios. So, optimization of the power in such devices is a major topic of interest. Hence, the IoT architecture for the "Ingram Project" and similar is bound by severe resource constraints including energy consumption [22].

Multiple approaches regarding data security and provenance have been discussed for activities such as the Ingram Project. For example, a "Distributed Ledger Technology" (DLT) [23] may provide sufficient reliability to address the issue. However, such technologies present other issues related to efficiency and scalability. A data-centric solution which performs highly scalable encryption, cloaking, and archival in the context of a validation and authentication architecture may provide an acceptable alternative. This research performs implementation, testing, and comparison of these technologies in the context of a canonical application such as data logging and verification for a cement plant outflow monitoring solution, or the "Ingram Project."

1.3. Thesis Structure

This thesis paper is organized into five chapters. Chapter 1: 'Introduction' presented the background of IoT, and provided the description of the problem that this thesis is trying to focus on. Chapter 2: 'Literature Review' provides an extensive overview of the technologies currently used in the IoT applications. The setup of the experiments and data processing techniques used for the experiments are described in Chapter 3 'Experimental Setup'. The results of these experiments are described thoroughly in Chapter 4 'Results and Discussions'. The final chapter is Chapter 5 'Conclusions and Future Work' which describes the conclusion of the research and suggests the future work that could enhance the project for potential use in IoT applications.

2. LITERATURE REVIEW

IoT is referred to as the next generation worldwide network as recent advances in technology have made the cost of connecting IoT devices cheaper, faster, and convenient. Everyday more and more devices are produced having Wi-Fi capabilities and sensors built into them. However, there still lie certain limitations in implementing the technology that are the focus of numerous ongoing studies to ensure the IoT technology can be implemented in a variety of use cases. An important issue is that these deployed IoT endpoints are usually designed to be operated for many years, even decades [24]. So, it's extremely important to limit energy consumption as replacing batteries can be impractical as well as expensive. The limited resources in IoT endpoint don't support complex calculations and storage as they are designed for simple tasks. So, most of the major issues in IoT arise due to the limited resources in the IoT devices. Furthermore, the growing number of IoT devices and applications like big data analysis and real-time processing are also extremely resource intensive IoT [25].

One of the ways to solve this issue is to use cloud computing, where the resource intensive tasks of the IoT applications are performed in cloud-based servers where there are no resource constraints. This is referred to as task offloading, and in some cases may reduce the resource consumption and enhance the performance of the IoT application [26]. The major benefit of cloud computing is that cloud-based servers are not limited by computational and storage resources. Cloud computing has an added benefit of being able to buy/rent cloud services from 3rd parties, alleviating concerns related to real-estate, personnel, infrastructure, processors, power, and storage. However, cloud servers are often located away from the data source which means that data is not processed at the source of generation. This presents two distinct issues. Firstly, transporting the data to cloud will often result in an increase in data transmission

overhead, which further causes network congestion and increase latency [27, 28]. Secondly, the data could be vulnerable during transit and data could get lost or stolen.

An alternative to task offloading may be edge computing. Edge computing enables remote devices to process data at the network's "edge" either via a localized service or capabilities embedded within the device itself [27]. Edge computing puts IoT applications closer to the IoT endpoint data source. It would be more efficient if the tremendous amount of data that is generated at the network's edge is processed at the source [29]. The peak traffic flows will be reduced as the edge computing nodes are closer to the data source [28]. Furthermore, edge computing reduces bandwidth consumption, network latency and responds to demands quicker as compared to cloud computing for IoT [28].

There are various technologies that can be integrated with edge or cloud computing to fit the IoT application that address the core IoT issues. One approach may be a version of Blockchain technology which may provide sufficient reliability to address the data security issue. However, blockchain may invite new issues regarding efficiency and scalability. Another approach may be a data-centric solution which performs highly scalable encryption, cloaking, and archival in the context of a validation and authentication architecture. However, such an approach may also invite new issues of memory and complexity.

Blockchain technology, commonly understood as a distributed ledger [23], is a contemporary choice for storing sensitive data. A blockchain is a continuously growing chain of blocks [30], each of which includes a cryptographic hash of the block before it, a time stamp, and some application data [31] plus other information depending on use case. Blockchain has been proposed as a promising candidate for IoT networks mainly because of its enhanced interoperability, privacy, and security features [32]. However, it also possesses challenges as IoT

networks are extremely resource constrained and in order to guarantee consistency each node must keep the same copy of the blockchain [33], which might prove inefficient or impossible. Hence, the blockchain as a distributed ledger needs to be modified before implementation in IoT projects.

The self-protecting, self-controlling 'Intelligent Cipher Transfer Object (ICTO)' is a different alternative for storing sensitive IoT data. ICTO is an advanced data encapsulation technology that provides a data assurance solution which is self-contained and self-governing [34]. Usually, network solutions for issues in IoT revolve around securing network endpoints and infrastructure, but not the data itself [35]. Network-based security solutions are problematic, as new instance of data leakage and tampering regularly appear as evidenced by 57 million IoT malware attacks in the first half of 2022, a 77% increase from 2021 [36]. A different approach is to protect the data itself and treat it as an endpoint. An ideal solution would be to secure the data from its source of origin so that only those users with authority can access the information. To achieve this for IoT, the data must contain an ability to decide access rights for its use, as opposed to conventional approaches such as network-based authentication, authorization, and accounting (AAA) which provide access rights to infrastructure [35]. Data must also be independent from conventional infrastructure-based access controls so that no matter where data is, it will always retain its rules. Data must also understand itself and appreciate its value such that data can govern when and how it might be accessed [35]. This is the fundamental basis on which ICTO has been defined and hence could prove to be a solution in IoT deployments.

2.1. IoT Architecture

IoT architecture refers to the different components that make up the IoT system. There are different IoT architectural layouts and the most widely considered architecture for IoT are

three-layer architecture and a five-layer architecture [37]. The three-layer architecture is described in this section. The three layer architecture consists of the physical layer (sensing layer), network layer, and the application layer (perception layer) [15].

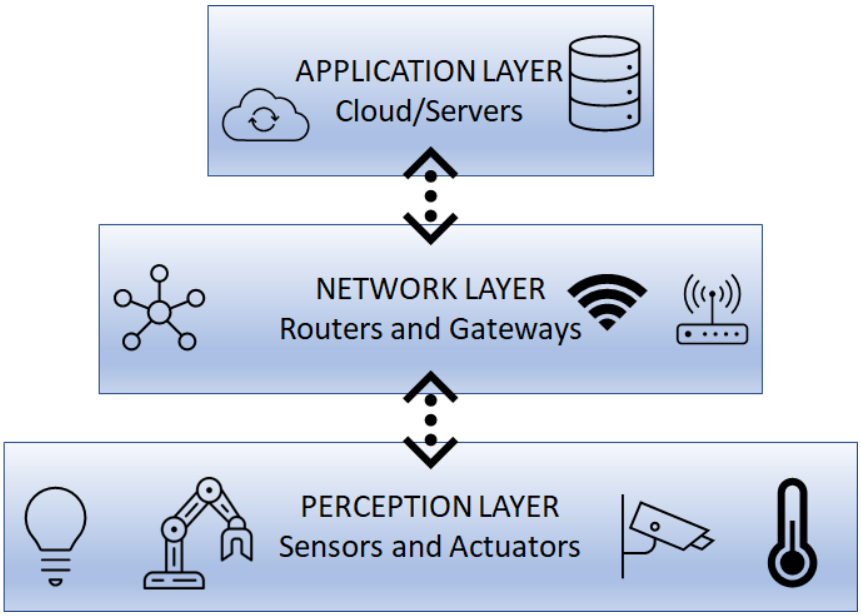


Figure 2. Three-Layer IoT Architecture [38]

The perception layer, or sensing layer represents the physical sensors, actuators, and IoT devices that perform the task of generating the IoT data. They act as a bridge between the physical world and digital world. The first layer of any IoT system involves the IoT devices that sense the information of the IoT objects and perform functionalities like sensing humidity, water level, acceleration, etc. [10]. This layer consists of two sections: the sensing part and the sensing network. The sensing part basically generates the information, and the sensing network is responsible for sending control signals to controller or transporting the data to gateway so that it is ready to be transported through the network layer [15].

The network layer follows the sensing layer, and this layer is responsible for transporting the data gathered from the sensing layer IoT devices using wired or wireless network. This layer describes how the data gathered will be transported throughout the application. All the

functionalities related to data transport like routing, switching, and operation of gateway devices is carried out at this layer [15]. The network layer consists of two parts: communication technology for IoT and communication protocol for IoT [38]. Some of the common technologies used to transport the data are Long Term Evolution (LTE), Global System for Mobile Communications (GSM), Wireless Fidelity (Wi-Fi), Bluetooth low energy, Long Range Wide Area Network (LoRaWAN), ZigBee, etc. [12, 14]. Communication protocols often used for IoT applications are Message Queuing Telemetry Transport (MQTT) [39], Advanced Message Queuing Protocol (AMQP) [40], Constrained Application Protocol (CoAP) [41], Open Platform Communications United Architecture (OPC UA) [42], Representational State Transfer (REST), and Extensible Messaging and Presence Protocol (XMPP) [43].

The application layer, or perception layer, is the interface providing the user interaction that provides specific IoT service solution to the user. The data received from the network layer is processed and analyzed at this layer to provide specific services to user [37]. This layer also deals with data processing, recognizing malicious data, computational functionalities, and producing results [15].

2.2. IoT Data Security.

The IoT ecosystem is a vulnerable space where there is a constant threat of cyberattacks. The attacks have become more unpredictable and the IoT systems need to be properly protected as they do not have well defined security measures, are highly dynamic, and continuously change due to mobility [44]. The IoT systems are vulnerable as they are usually connected to a big network and could be a victim of different attacks. Furthermore, these systems also need to be protected physically or be under supervision if controlled by different parties. The most common vulnerabilities of IoT systems arise due to lack of adoption of security techniques, encryptions,

authentication measures, and access control. This lack of adoption of security measures could be caused by difficulty in deploying existing security tools and techniques on the IoT devices which are limited in resources. These measures could also turn out to be highly expensive in terms of complexity, computability, memory, and cost especially when running on a large number of resource constrained devices [44].

Security is a major aspect of IoT as IoT systems are at constant security risk. Security is a combination of authentication, authorization, integrity, availability, non-repudiation, and confidentiality [45]. The data from the IoT endpoints could be vulnerable during rest or during transit. Security is essential at every level from data generation to data processing. It is necessary to secure data immediately after generation at the IoT endpoint when the data is at rest. Network security is concerned with securing data during transit which involves the communication between IoT endpoints, and often cloud-based storage. Protection of data is also essential during storage, and processing of the data, no matter the environment.

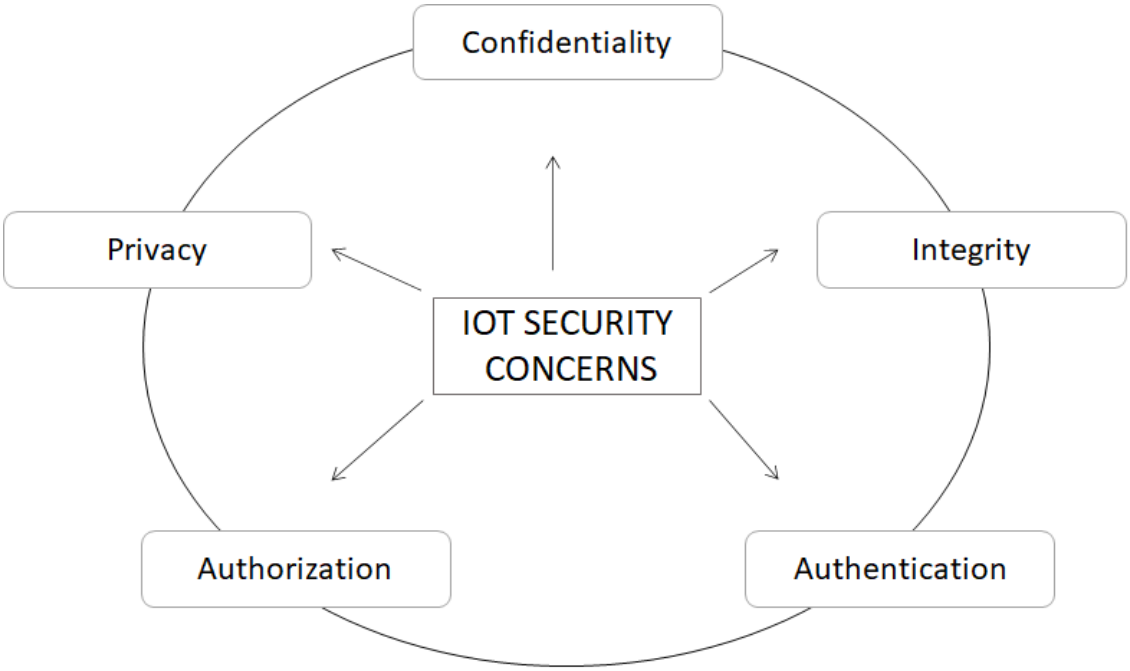


Figure 3. IoT Security [45]

Authentication is the process of verifying the identity of endpoints in the IoT ecosystem. Authentication is the first step of verifying endpoints identity and only after being authenticated, the endpoints can take part in the communication in the network [46]. Without secure authentication, it is impossible to guarantee the reliability of the data from the endpoints. Authentication goes hand-in-hand with validation as critical parts of IoT architecture. Data validation is the mechanism of ensuring the quality of the data from the IoT endpoints. The main purpose of data validation is to ensure the accuracy of the data received. Data from the IoT sensors and smart gadgets needs to be validated to create trust in the IoT system and increase the reliability of the system. The endpoints need to be authenticated and the data received from such endpoints needs to be validated before data processing. Another core requirement of a secure IoT environment is the integrity of the IoT data. Integrity means the protection of the data against illicit modification of data. Illegitimate use of data or its modification translates to loss in privacy as well as the authenticity of the original data. So, it is critical to maintain the integrity of the data. There must be the utmost confidence in the data supplied by the end devices.

Regarding the security of the data in IoT applications, there are different techniques that are frequently used to encrypt and secure the data at rest or transit. Some of the most common data security techniques used in IoT are discussed in Section 2.2.1.

2.2.1. Encryption

While there are many different encryption algorithms that could be considered for data security, algorithms applicable for resource constrained IoT endpoints must be lightweight, cost efficient, and practical. For example, if block cipher encryption techniques are used, then they should have smaller block sizes. Smaller key size will also help to consume less power from those devices. This will help preserve the battery life of the IoT devices as well as save storage

space. As these devices are also unable to perform complex calculations, encryption algorithms with simpler rounds and simpler key schedules are ideal for them. These encryptions techniques used for limited resources devices fall under the category of “lightweight” encryption algorithms. While considering the right encryption technique for a particular use case, both the security aspect and the resource consumption aspect must be considered. A tradeoff between security, performance, and cost must be made while designing and choosing the right encryption technique, especially when application is for resource constrained devices.

The National Institute of Standards and Technology (NIST) has been actively involved in standardization of different lightweight cryptographic algorithms to address the issues faced by resource constrained IoT devices. The NIST lightweight cryptography primarily focuses on developing cryptographic algorithms that are designed to provide a balance between security, resource efficiency, and performance [47].

NIST selected the Authenticated encryption for Small Constrained devices (ASCON) family for lightweight cryptography applications on February 7, 2023, as it met the criteria for lightweight cryptography [48]. ASCON is a family of authenticated encryption and hashing algorithms designed especially for lightweight usage and easy implementation. Ascon is designed to be lightweight, fast in hardware and software, scalable, minimal overhead, and easy to implement in both software and hardware [49]. Currently, NIST is working with the Ascon team to draft a new lightweight cryptography standard. In addition to this latest cryptographic algorithm, NIST has standardized various block cipher, digital signatures, and hash functions as a part of their cryptographic standards and guidelines, as described in Figure 4.

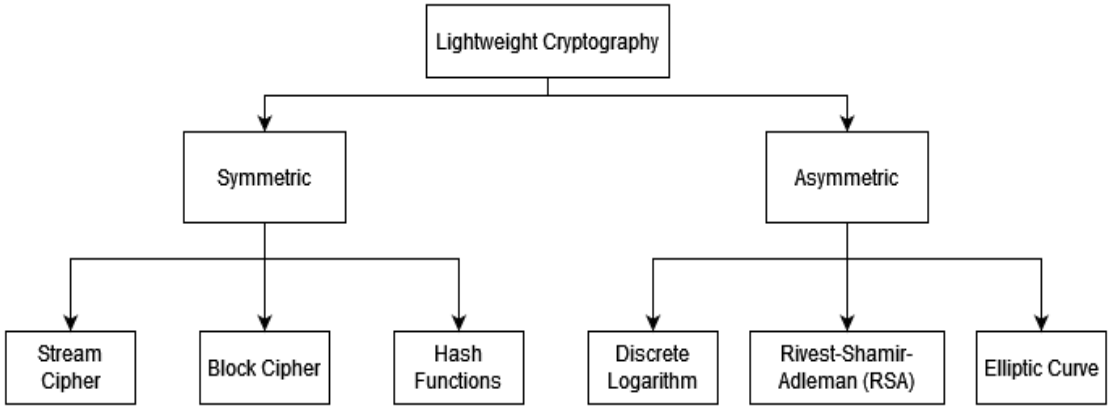


Figure 4. Lightweight cryptographic diagram [50]

There are generally two types of key encryption techniques used in IoT: symmetric key encryption, and asymmetric key encryption. The lightweight cryptographic (LWC) diagram [50] depicting the common lightweight cryptographic algorithms is shown in Figure 4. Some of the common symmetric and asymmetric key encryptions are discussed in Section 2.2.1.1 and Section 2.2.1.2.

2.2.1.1. Symmetric Key Encryptions

A Symmetric Key Algorithm is a common type of encryption which uses the same key for encryption and decryption. This technique has the benefit of speed and efficiency which makes it extremely potent for IoT devices with constrained resources. However, using the same key for data encrypting and decoding means that the key must be securely placed by all parties involved. This means that the key should be shared between the IoT device and the cloud through a secure channel, which is one of its major disadvantages. AES is one of the most common symmetric key encryption techniques that is widely used in data encryption, wireless networks, secure messaging, and network communications. It can have three different key lengths of 128-bit, 192-bit, and 256-bit. AES is extensively used for encrypting IoT data because it consumes very low power and memory [51], which is suitable for IoT. Advanced Encryption

Standard (AES) is one of a lightweight symmetric key encryption technique that can be used in IoT constrained devices. The AES use case on IoT as hardware and software implementation has been examined and outcomes of these evaluations concluded that the hardware implementation of AES is 23 times faster than the software implementation of AES [52]. However, the key expansion function which is used to generate the key schedule is 1.88 times slower in hardware implementation than in software implementation [52]. This is especially important resource constraint IoT devices where power is limited.

2.2.1.2. Asymmetric Key Encryptions

In contrast, asymmetric key encryption also known as Public Key Infrastructure (PKI) performs encryption using two different keys, a “public” key and a “private” key. PKI can also provide unique identity for end devices and provides security through asymmetric encryption. The most common form of PKI involves a pair of keys for a device, which are private key and public key. The public key is derived from the private key. The private key is secret to the device whereas the public key can be shared with anyone. The public key of the intended receiver can be used to encrypt data from an IoT endpoint, to protect it when sent through an insecure channel. Only the corresponding private key, which is held by the receiver, can then be used to decrypt that message. Authentication employed with public key cryptography is a popular data authentication mechanism. This authentication mechanism can resolve problems of authentication like identifying devices, or users in IoT [46]. Additionally, it can be useful in restricting unauthorized users from access to devices, securing communication among end devices, and protecting data from tampering and stealing.

Even though PKI seems rigid and trustworthy, it is capable of being compromised. The notable concern is that the private keys are like passwords and if an attacker obtains a private

key, then the attacker can perform all actions of the owner of the private key without detection. Furthermore, the risk of "man in the middle" (MITM) persists even in the PKI scenario. If an attacker intercepts a public key and replaces it with a different public key, then the MITM will be able to decrypt any messages intended for the original owner. The attacker can change messages and re-encrypt them with the original public key in the form of "silent failure". In this way, the receiver won't know that the original messages have been intercepted or compromised.

The most popular of the cryptographic data security cipher suites are Rivest Shamir Adleman (RSA) algorithm and Elliptic curve Diffie-Hellman Ephemeral (ECDHE) algorithm which is based on Elliptical Curve Cryptography (ECC). RSA is a type of asymmetric key encryption that could provide security in IoT while also functioning properly in resource constrained IoT endpoints. RSA is used for digital signatures as well as encryption. The private key of the RSA algorithm is generated from two large random prime numbers[53]. One of the major advantages of RSA is that the private key is not transmitted for the purpose of encryption, and the private key is impossible to be known by the hacker using the same channel [54]. Encryption is performed using the public key and decryption is performed using the private key, as in all asymmetric approaches. RSA requires two large prime numbers and performs modulo operation on them to generate the key pair. Because of the large key size of the RSA, it is not considered to be lightweight cryptography [55].

Elliptic Curve Cryptography (ECC) is a strong cryptography suite that could provide security in IoT. ECC is a type of public-key cryptographic algorithm that can be used for encryption, digital signatures, and authentication, and is based on the Elliptic Curve Discrete Logarithm Problem [56]. The generation of encryption keys in ECC is based on the properties of the elliptic curve. One of the major advantages of ECC over algorithms like RSA is that ECC

keys are much shorter which makes ECC extremely useful for the resource constrained IoT use case. ECC also performs better than RSA in terms of energy consumption and data throughput values [57]. The major benefit of using ECC is the small key length which results in quick energy efficient encoding [53]. ECC keys are often $1/6^{th}$ the size of RSA keys [58]. ECC is a better alternative than RSA for securing resource constrained IoT devices [57, 59, 60]. A comparison of RSA and ECC is provided in Table 1 based on the size of key required to gain the same level of security. For instance, 512 bits of RSA key is equivalent to 106 bits of ECC key. It signifies that length of RSA key is 20.7% of ECC key whereas length of ECC key is 483.02% of RSA key.

Table 1. Comparison between RSA and ECC [57, 61]

RSA (bits)	ECC (bits)	Percentage size difference (ECC vs RSA)	Percentage size difference (RSA vs ECC)
512	106	20.70	483.02
768	132	17.19	581.82
1024	160	15.63	640.00
2048	224	10.94	914.29
3072	512	16.67	600.00

An Attribute-Based Encryption algorithm (ABE) is also an option for constrained IoT devices. ABE uses public key encryption techniques based on flexible access policies [61]. The data can be encrypted by the source using a set of attributes and the access level of every information is regulated by those attributes. ABE encryption has been shown to provide good encryption but can cause significant impact on execution time and energy cost, particularly on resource-constrained devices [61]. The negative outcomes of ABE can be reduced by considering a smaller number of attributes, thus weakening the overall approach.

2.2.2. Physical Unclonable Function (PUF)

A Physical Unclonable Function (PUF) is a unique device authentication scheme for IoT. It provides a digital fingerprint for a device which is based on the uniqueness of the physical variations that occur naturally on the chips during manufacture. It depends on the random variations in the physical microstructure of the semiconductor. It is not possible to clone such microstructures as these variations are randomly introduced during manufacture by factors that are uncontrollable and unpredictable. PUF technology allows every device to have unique characteristics, distinguishing it from other identical devices that are manufactured using the same process.

PUFs provide hardware-unique mapping, which means that the same input applied to identical PUF circuits will result in distinct outputs [59]. This property can then be used to generate a cryptographic key dynamically and unique to the device. PUFs implements challenge-response authentication to identify the hardware device. A stimulus or "challenge" is applied to the hardware and the hardware-unique mapping property is used to regenerate a key which in turn is used to generate the output of the challenge or "response". A specific challenge and its corresponding output form the challenge-response pair (CRP). The authentication of a device is done by providing a challenge for which the response is already known to the device, and then it verifies that the response is similar to the expected response. PUF can be used to generate different types of keys for authentication. Each device uses its PUF to regenerate a private key, and a server can use the public key to verify encrypted data. In this way, the PUF can be used to authenticate devices using encryption techniques such as a PUF-based Elliptic Curve algorithm and PUF-based digital signatures [59].

One of the major benefits of PUF is that the key is not stored in digital format and is

produced when needed using the physical characteristics of the circuit elements. This means that the key is not present in the nonvolatile memory of the device, and hence resistant to being stolen or copied by an attacker. Also, the challenge-response pair (CRP) can reveal neither the microstructure of the device nor the key. So, PUFs are resistant to spoofing attacks. However, there are some limitations to this technology. Albeit the PUF is unclonable, it doesn't mean that the generated key is similarly unclonable. Once the PUF key is generated, that key can be cloned. Additionally, since each challenge-response combination can only be used once for authentication, a significant number of challenge-response pairs must be collected [59].

2.2.3. Zero-Trust Architecture for IoT

Zero trust architecture focuses on the principle of "trust no one, verify everything "in response to access of systems and the resources [60]. This architecture assumes that no device or user is trusted, regardless of external or internal network. Zero-trust architecture can play a significant role in enhancing the security in IoT applications. It can ensure strong authentication mechanisms for devices and provide access to only authorized entities to interact with devices and provide access to information. The architecture promotes network segmentations, which allows for isolation of resource constrained IoT devices which will reduce risk in case of security breach. Zero-trust management could also help to ensure the validity of the data before its use. Thus, it's a strong argument to implement zero trust architecture encompassing Zero Trust Application Access (ZTAA), Zero Trust Network Access (ZTNA), and Zero Trust Data Access (ZTDA) in IoT. These frameworks can be employed in the IoT environment to enhance security and control access for devices. These concepts can be used to implement strong authentication and monitoring to ensure that access to applications and data are not granted solely based on network or location.

However, implementation of a zero-trust management within IoT networks may not be feasible and cost-effective. This is especially true for edge IoT devices, as they may be unable to support the zero architecture because of their limited power and computational resource [60].

2.3. Blockchain

Blockchain is an immutable ledger that stores transactions in blocks of data which are linked together like a chain; hence the name blockchain. IBM [62] defines blockchain as a decentralized, shared, immutable database that makes it easier to track assets and record transactions in a network. Blockchain technology is an advanced database system which enables transparent information sharing within a network. The blocks in the blockchain are linked together by a hash function. Every block has a timestamp, transaction, and the hash of the previous block. The first block in the blockchain is referred to as a "Genesis block", and it indicates the start of the blockchain. The function of the hash function is to validate the integrity and non-repudiation of the data that is stored inside the block [8]. Every participating node in the blockchain network gets updated regularly with the copy of the original.

On a blockchain network, practically anything of value may be recorded and traded, lowering the risks involved for all parties [62]. Since decentralized blockchain is a distributed ledger technology which records data that will have immutability, no one can alter the ledger or tamper with the data after it has been recorded in the ledger. Traditional blockchain technology has been modified and changed to address the different types of problems in different domains. Two of these modifications are being adopted in the presented research to address the issues that IoT applications face.

2.3.1. Implementations in IoT

Often, the term “blockchain” is understood in casual usage to mean “cryptocurrency” due to the recent popularity of digital currency exchanges. Blockchain is the core technology behind cryptocurrency, but the application of blockchain is not limited to cryptocurrency. The distributed ledger is the major technology that introduced the concept of a Peer-to-Peer Electronic Cash System called Bitcoin in 2009 [63]. As a result, the distributed ledger is the foundation of digital currency known as cryptocurrency that has changed the digital payment system. Distributed ledgers do not rely on a central governing system like a banking system and exist virtually or digitally. This legitimate use of distributed ledgers, and the cryptography for securing transactions has made it a reliable alternative to traditional banking systems. The crypto architecture leverages computational power to solve complex mathematical puzzles, a process commonly known as “mining” [32]. Mining yields cryptocurrency units such as Bitcoin or Ethereum. Besides finding applications in cryptocurrency, the impact of the core blockchain technology cannot be underestimated in other fields like IoT. While the potential influence of blockchain on IoT may seem obvious, a debate of “where to host the blockchain” remains [64]. There are various implementations of blockchain depending on where it is hosted. Several approaches to implement blockchain technology to suit IoT applications.

Distributed ledger technology is one of the blockchain implementations that addresses the “single point of failure” issue. The ledger technology uses consensus mechanisms like Proof of Work (PoW), Proof of Authority (PoA), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT) to achieve distributed nature [65]. The most common consensus mechanism is the PoW which relies on computational work to validate the new blocks and add them to the chain. IoT systems implementing a centralized architecture

where central server carries out all the operations in the network [8] are vulnerable to this issue because only the central server stores all of the information. Another issue relates to the security as all the sensitive data is kept on single server making it a prime target for various types of attacks [8]. The risk of tampering and counterfeiting is addressed in blockchain by the distributed nature of the ledger. Multiple sources confirm the transactions before being recorded in a blockchain. This provides high levels of reliability and security. As a result, the use of distributed ledger technologies like blockchain has become widespread in IoT. However, blockchains in a distributed environment might not be suitable for IoT use cases because of the various resource constraints.

IoT applications deal with a tremendous volume of data, and it isn't practical to store all the data in the blockchain itself. So, decentralized storage or off-chain storage can be used to address this issue. Blockchain can provide a practical framework for distributed data storage as well as protection [66]. Blockchain can store metadata while the data itself is stored in Distributed Hash Tables (DHTs) [67]. The blockchain can then determine if data access is allowed when some entity requests data from that DHT [66]. This is one of the examples of integrating blockchain technology with edge computing and it's an effective way to address the low computational power of the IoT devices [66]. Edge computing along with blockchain technology can be used to create a structure for IoT data storage. The data from the IoT sensors is forwarded to the edge device which places metadata into a blockchain whereas the data itself is transported to DHT for storage. The metadata contains the information about which sensor generated that specific dataset. When an entity needs the data, the entity posts a new transaction to the blockchain. The blockchain validates the entity requesting the data and sends the data if the entity is authenticated by the blockchain [26]. This implementation of the blockchain in IoT

has many advantages as discussed earlier, but also introduces different critical issues. The blockchain needs to know which IoT sensor node is generating the data. There could be hundreds of sensor nodes feeding data to the blockchain. The most approach to disambiguating the sensors is to use a traditional PKI which the blockchain can use to identify the IoT device. The public key of the IoT sensor device can be used by the blockchain to verify where the data is coming from, but it generates an issue of reliability of the data. Even though PKI enables strong encryption, attackers can easily access and modify the data if they know the private key of the IoT sensor device. Another issue is that such blockchain IoT architecture needs miners to run the blockchain, which might not always be possible for IoT applications.

Another implementation of blockchain technology with IoT involves using gateway devices. Via gateway device, a blockchain can be hosted with different configurations. One such configuration uses a gateway as a full blockchain node. In this configuration, the gateway device communicates with the IoT endpoints and the gateway devices hosts the blockchain as a node by verifying and routing data [68]. Another similar implementation is using gateway as a thin client rather than a full node. In this implementation, gateway communicates with the IoT endpoints and only stores relevant data fragments [68] which could potentially decrease the overall cost.

Most of the versions of blockchain that incorporate IoT try to deal with issues like scalability and complexity. The most common issue to integrate these two technologies is the storage problem. A '*Ladder'* blockchain model which aims to reduce the storage by dividing the different blocks of blockchain into three different distinctions, Hot-Blocks, Warm-Block, and Cold-Blocks had been used to address the scalability issue [69]. These three distinctions are based on the necessity of the data inside the block for upcoming blocks. Hot-Blocks include blocks that are likely to be frequently retrieved and hence are stored on the node itself. Warm-

Blocks are less frequently needed to be retrieved, and they are stored in caches of different nodes so that they can be retrieved quickly if needed. And Cold-Blocks are highly unlikely to be required and hence are stored in remote databases. But they can be retrieved if needed from the database at the expense of time and energy. The distinguishing factor for these blocks is done by calculating the importance of the blocks using concept of blockchain *value-density* (VD) [69]. VD of a block relates to the "block retrieve number" and depends on how important the block was in the past, and how frequently it might be used in the coming stages [69]. This version of a blockchain tackles the issue of scalability of the blockchain in a unique fashion, which is very helpful for cases like IoT where storage space is limited. The experiment on this blockchain architecture showed that the space overhead has been reduced. The cost of lowering the storage space is the increase in time to retrieve the blocks if needed and increase in complexity of the blockchain to introduce parameter VD and separate the blocks as per the VD.

Finding approaches to solve the storage issue had been key research field in IoT. The use of data compression to reduce the blockchain size is not uncommon. A selective compression algorithm (SELCOM) addresses the storage issue [70]. The blocks of the distributed blockchain are compressed to create a checkpoint which is defined by the blockchain owner. The group of checkpoints are then used to create a second blockchain called a checkpoint chain. Using these two blockchain, certain compressed blocks can be deleted or maintained by the node. This architecture allows each node in the network to store selective blocks as per the need [70]. This version of blockchain reduces the storage overhead tackling the issue of the storage issues in cases like IoT with limited resources, but it introduces complexity because checkpoints must be selected by each block, and a second blockchain is required to store the checkpoint.

Challenges and issues of integrating cloud and blockchain technologies with IoT have

been examined and proposed [71]. The paper presents a summary of issues that persist in cloud based IoT and blockchain based IoT in terms of security, privacy, losses and risks, scalability, latency, energy-consumption, cost, payment, and flexibility. Cloud based IoT may excel in terms of memory storage, performing complex tasks, and scaling the network whereas blockchain based IoT has potential for better protection against losses and risks, as well as preserving privacy, and security. A hybrid-IoT architecture where blockchain meets edge computing may provide some advantage[71].

A neat comparison of the blockchain hosted on edge device (a local cluster) and blockchain hosted on cloud (IBM Bluemix) was performed in [64], which concluded that network latency is a dominant factor in performance analysis and it indicated that the blockchain in local edge cluster outperforms blockchain hosted in cloud. This is one of the reasons the version of blockchain we used in our blockchain experiment is hosted on the edge device.

2.3.2. Issues involving blockchain adoption for IoT.

While there are ways to integrate blockchain with IoT, the blockchain architecture fails to address the issue of severe energy and processing constraints that exist at the end devices of IoT. The implementation of the IoT concept faces significant hurdles in memory and power management, computing performance, and security mechanisms [46]. A large number of battery powered IoT end devices have constrained memory and processing power [46]. Because of these limitations, these devices cannot perform various complex encryption-decryption algorithms [46]. This limited resource presents challenges for blockchain technology due to its complexity, delay, and high computing costs [68]. Maintaining a blockchain network is not a resource efficient task and lack of computational resources, and memory storage, and need to reserve power in IoT devices is problematic [64]. So, it is impractical for end devices to directly host a

blockchain network due to the lack of adequate computing and communication capability [8]. The different challenges of integrating blockchain with IoT are depicted in Figure 5.

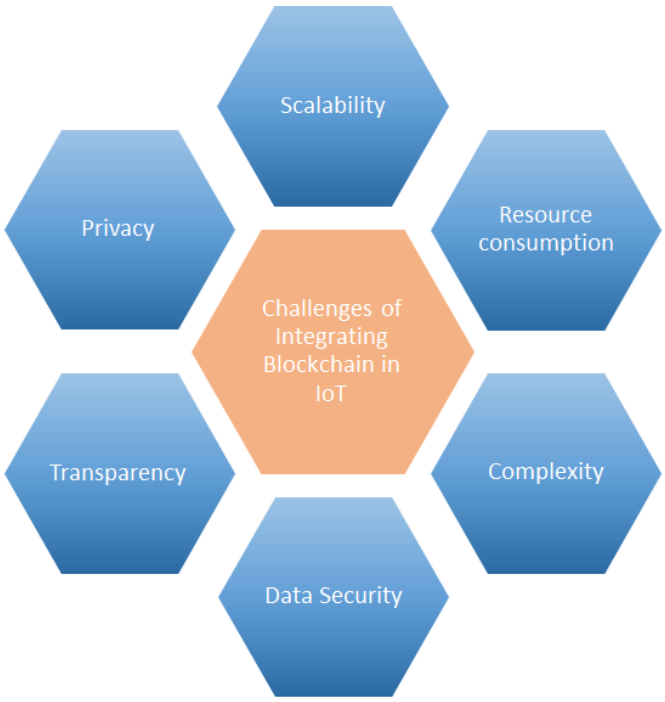


Figure 5. Challenges of Integrating Blockchain in IoT Applications.

2.3.2.1. Scalability

While traditional decentralized blockchain does eliminate the central server requirement, decentralization in blockchain means that the same copy of the ledger must be stored in all the nodes. Even if the blockchain is not decentralized, blockchain still must store a large amount of data from the IoT endpoints. Scalability issues will occur when the size of the blockchain starts to grow. This is because all nodes need to store the added block in their version of the blockchain. And as the data size increases, the storage and power requirement also increase which hinders the scalability of the blockchain [72]. This also holds true for centralized blockchain where the size of the ledger will continue to increase as time passes, rendering it impractical to store blockchain information on IoT devices like sensors. These end devices don't have the storage space required to store the vast amount of data. Hence, managing IoT data with

blockchain creates a burden on the IoT devices' storage space [73].

2.3.2.2. Complexity and Processing constraints

The end devices in the IoT ecosystem have extremely low computing capability. They are not guaranteed to run the high computing requirement of the blockchain nodes. The blockchain nodes perform tasks like Proof of Work (POW), and encryptions and require high computing power. These complex tasks in blockchain reduce throughput and increase delays [72]. The types of computations that the end devices can perform are limited. With complex computing comes high power consumption. Power constraint is one of the major issues while implementing blockchain in IoT. The end devices in IoT can be a lot of cheap sensors that need to run continuously to gather data from the environment. These devices usually run for months or even years on a single battery life. Hence, reducing the energy consumption from IoT end devices is a major challenge and to save power, devices have mechanisms like hibernation when not needed and minimizing use of high-power processing tasks. Blockchain nodes must perform high computing tasks that have higher power requirement than most IoT end devices can afford.

2.3.2.3. Security, Transparency, and Privacy Issues

Another significant issue with blockchain is the fact that the technology is only as secure as its cryptographic algorithms it implements [8]. The public key infrastructure that the blockchain uses can be compromised and it could affect the reliability of the data. If an intruder gets hold of the public key of a node in blockchain implemented in IoT, then intruder can store false data on the blockchain. Transparency and privacy are also important issues that aren't always fully addressed by Blockchain. While blockchain guarantees transparency of transactions for applications like finance, the same isn't desired for IoT use-cases like storing medical information, data from industrial plants, and so on. This poses a challenge in creating an

effective access control in blockchain to maintain balanced degree of transparency as well as complete protection of data privacy [73].

2.4. Blockchain architecture used for experiments.

All the different versions of blockchain introduced in earlier chapter suggest that every one of them has their own benefits and demerits depending upon their application. Every version of blockchain tries to solve certain issues that are critical to integrate blockchain with IoT. Different versions of blockchain are ideal for different use cases. Because of all the power and processing constraints described earlier, it is not possible to make the IoT end devices as nodes if a traditional distributed ledger is to be implemented. It is possible to add hardware to end devices that have enough memory, processing power, and battery supply so that it can act as nodes but that will increase the costs and complexity as discussed earlier. Furthermore, we would need such hardware for every one of the hundreds of end devices deployed. This will be highly cost inefficient and impractical. So, a compromise must be made to employ blockchain technology in such IoT use cases. And keeping our Ingram project and taking in consideration the issues associated with blockchain architecture described in Section 2.3.2, the current research uses a centralized storage architecture that communicates with IoT end devices through a protected channel. This kind of implementation will not be a distributed ledger but rather a blockchain ledger which resides in an IoT endpoint during block creation. The data generated from the various sensors are stored as a local version of blockchain in IoT end device. The IoT endpoint will then communicate with the distant server (cloud) to transfer the blocks once the blockchain takes up all the usable memory resource in the IoT device. This requires that the data that we input in the blockchain be already validated by some algorithm. There are different data validation techniques like Public Key Infrastructure (PKI) and Physical Unclonable Functions

(PUF). Apart from the latest block in the blockchain, every previous block is deleted in the IoT endpoint to free up the space to generate new blocks containing new data. These won't alter the blockchain because the complete blockchain is always present in the remote server. Not employing the distributed nature in our version of blockchain does take away the various benefits of decentralized ledger but it also deals with the critical issues for IoT use cases like complexity, limited storage space, limited resources, scalability, and data protection.

Since distributed ledger is also a common technology used in IoT applications, a general implementation of a distributed ledger was incorporated as a part of the experiment. Proof of work (PoW) was introduced as the consensus mechanism to simulate a version of blockchain that is commonly implemented for distributed ledger. This architecture was chosen to facilitate a meaningful comparison with the core blockchain technology.

2.5. Intelligent Cipher Transfer Object (ICTO)

ICTO is a data protection technology comprising of a self-controlling and a self-protecting object [34]. ICTO is a solution for data assurance which is different from traditional data protection mechanisms in a sense that it is less dependent on reversible encryption patterns and on keys and passwords for authentication. It is an up-and-coming technology that can protect any type of data using self-encryption and self-governance. ICTO continues to operate effectively both inside and outside of a secure environment [34].

The foundation of the ICTO is built upon the necessity of a secure and dynamic data protection scheme. ICTO treats the data as "Intelligent" data. Generally, the standard for "intelligent" data refers to encryption and authentication mechanisms, however ICTO translates the static data into a self-protecting intelligent module and controls security by implementing privacy policies. This standard for intelligent data incorporates the control engine to recognize,

verify, and regulate the action of the user on the specific data. Figure 6 presents a depiction of a self-protecting intelligence module.

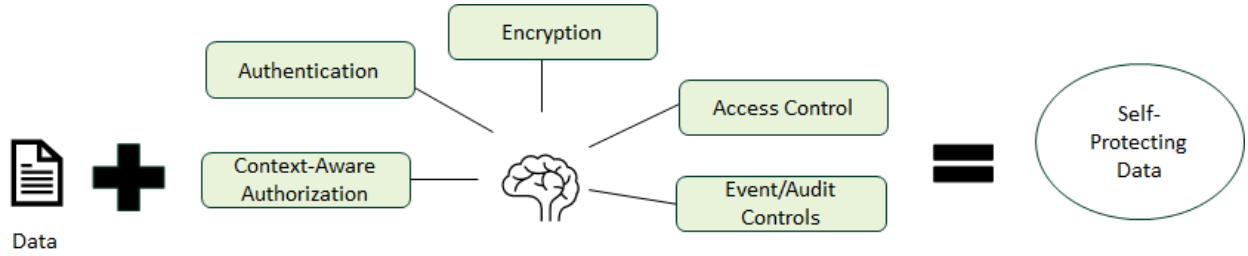


Figure 6. Self-Protecting Intelligence Module

An ICTO is a self-controlling technology that can contain a Portable Dynamic Rule Set (PDRS) which includes a set of participants. The contents of the PDRS are depicted in Figure 7. PDRS has the ability to verify that the user trying to access is part of the set of participants, and then decide whether to provide access to the participant. Hence, PDRS can read the access request from certain agents and verify the agent as well. After successful verification, the agent can only access those portions of data that are authorized to that agent, while the remaining portion is inaccessible. The ICTO at first is cloaked by cloaking patterns applied through the dynamic participant controller. This is temporary and is replaced when the owner of the data creates their unique rule set. Each ICTO is cloaked using cloaking patterns which are applied randomly to all or some part of the participant [34].

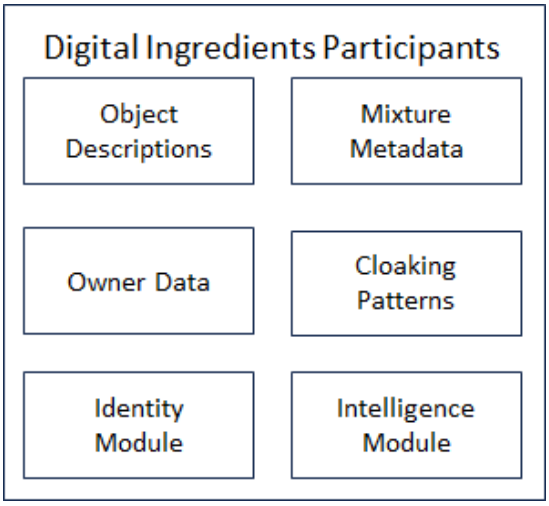


Figure 7. Portable Dynamic Rule Set (PDRS)

The cloaking patterns and PDRS set can establish multiple layers of encryption in a nested fashion. For instance, applying the first encryption can encrypt the already encrypted digital mixture, and subsequently applying second encryption can encrypt the doubly encrypted mixture. This results in an extremely secure pattern. In case of decryption, the second decryption is executed to generate the first decryption. Finally, the first decryption is applied to decrypt the final portion to generate the digital mixture. The set of cloaked participants is added to the digital mixture to generate the ICTO. Furthermore, additional security may be introduced to the digital mixture by applying digital signatures, and shuffling data. The "ICTO Creation" portion of Figure 8 depicts the flowchart of creation of an ICTO. The "ICTO Access" portion of Figure 8 illustrates the method of accessing data protected by ICTO. Verification and validation process begins as soon as ICTO is activated. Access attempts, authorized users, and present environment is verified as per the rule set specified in the PDRS. The verification criteria could include digital signatures, challenge/response pairs, biometrics, or similar methos to authenticate the identity of the user. PDRS is then executed to determine the access levels of the user to the digital mixture. Once the user is confirmed, only access to the specific portion of the data defined in the ruleset is authorized. This doesn't affect the rest of the portion of ICTO that is not authorized for that user.

This selective access and verification of ICTO to the user pertains to the notion of Zero-Trust architecture.

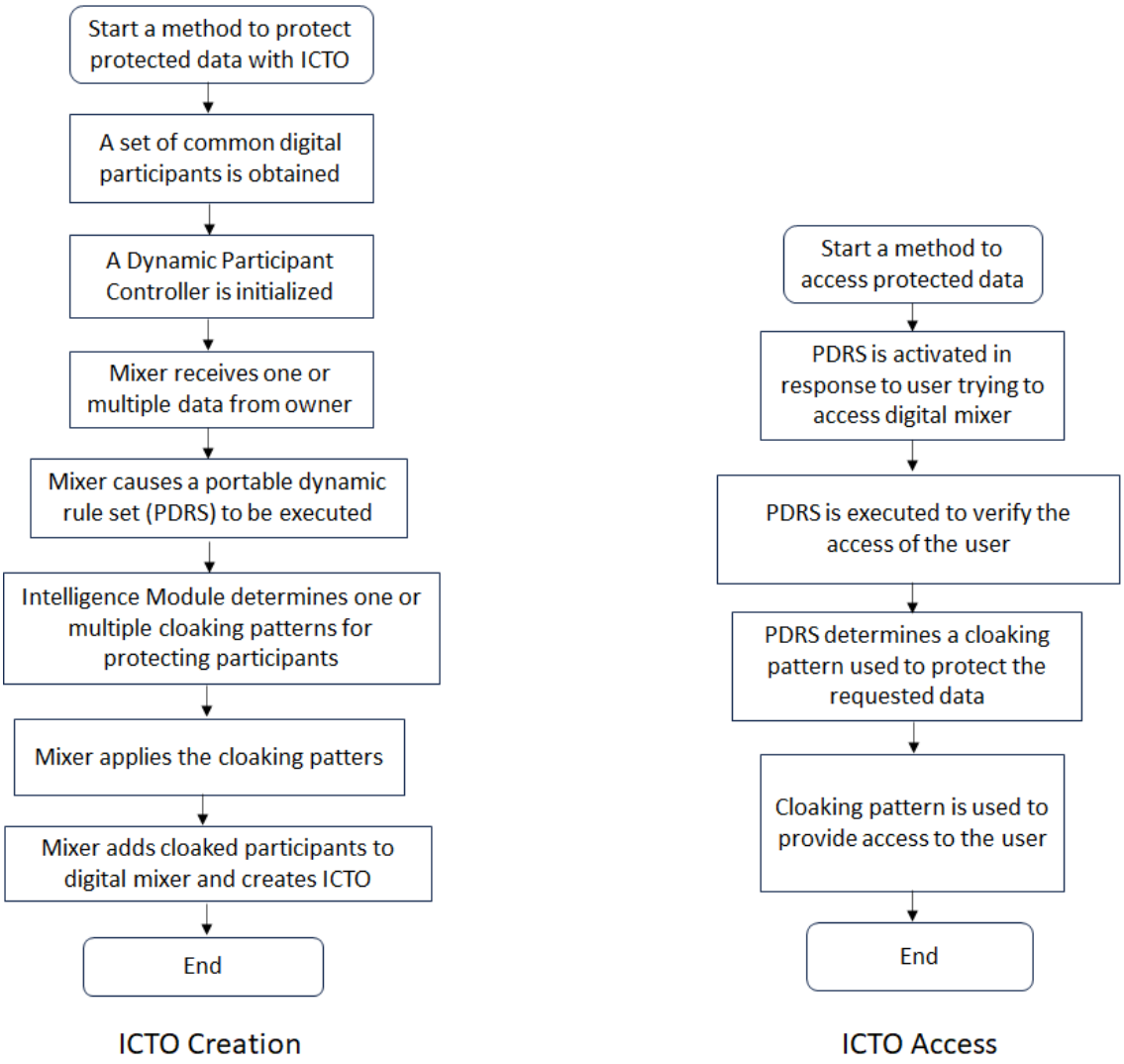


Figure 8. Flowcharts depicting ICTO creation and method to access the ICTO.

Zero-Trust architectures are built upon the foundation of self-protecting and self-aware data, often referred to as “intelligent data”. Dependencies on application of just traditional encryptions have been replaced to address to requirement of cyber security challenges. So, an intelligent data employing zero trust architecture could be a solution. Such intelligent data can control the access of data regardless of how, when, where, and who accesses that data. And ICTO is a prime embodiment of Zero-Trust Architecture [74] with Zero Trust Data Access

(ZTDA) that controls the data access based on the user trying to gain the access regardless of the network or location. Every attempt to access specific portions of the data is considered untrusted and requires user verification and authorization to access the data.

2.5.1. Unbreakable Exchange Protocol (UXP)

ICTO has different embodiments and applications. One such embodiment of the ICTO is the Unbreakable Exchange Protocol (UXP) technology [75]. UXP provides a method to secure and control access to data. The final protection entity of the UXP technology is identified as a "UXP Object". UXP Object has the ability to self-govern its access and self-protect. The access to the data is determined by the pre-defined ruleset by the owner of the data. Access policies as well as mitigating policies can be employed through the ruleset.

The creation of the UXP object starts with an "ID Definition XML" which is the source code for the UXP Identity (UXP ID). UXP Identity is a digital identity which is a core part of the UXP technology. It appears as a binary file with a ". iic" extension. A UXP Identity file without the UXP technology libraries appears to a computer operating system as a binary file with random contents. The UXP Identity doesn't contain any information about the data but consists of a virtual filesystem that manages a proprietary program called KCL Code and headers. The KCL code adds intelligence to the UXP Object and plays a crucial role in encryption key generation for the UXP Object. Standard AES-256 GCM encryption is used as a protection mechanism in UXP. Additionally, proprietary algorithms are also employed in conjunction with AES to establish a secure protection management system. This could make the protection scheme much secure than traditional encryptions. The encryption keys are never shared. Furthermore, these keys are cloaked and securely concealed making them inaccessible. The contents of the UXP Identity cannot be modified. The main function of the UXP Identity is to

implement the control access and rules defined by the data owner [75].

After the creation of the UXP Identity, one or more data files of different size and type can be included to create a UXP object using the UXP Software Development Kit (SDK). The final UXP Object appears as the binary file with a ".uxp" extension[75]. The general overview of creating a UXP object along with the components of ID Definition XML and UXP Identity is depicted in Figure 9.

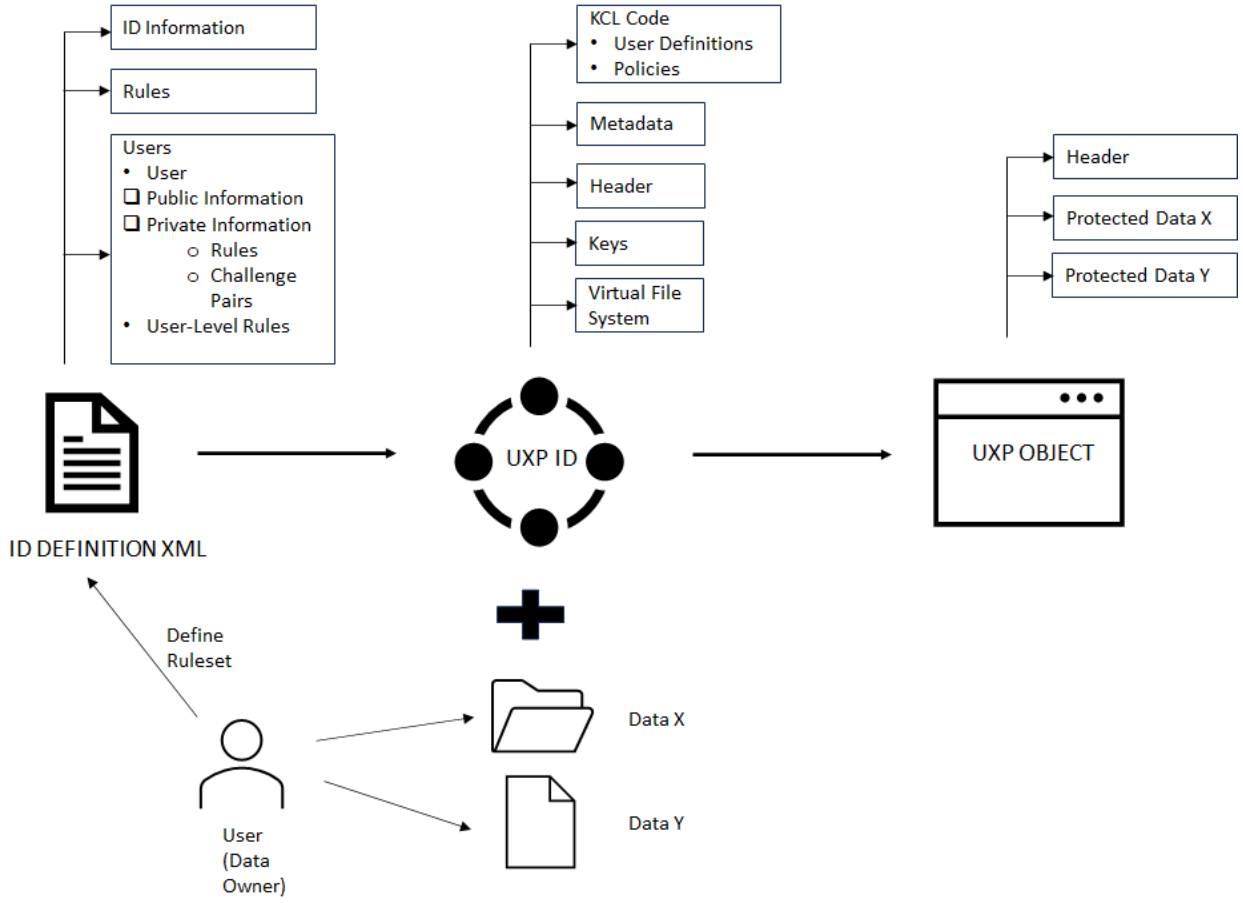


Figure 9. General overview of UXP Object creation [75]

2.5.2. Possible implementations in IoT

The deployment of any security solution for IoT is not an easy task due to the scaling factor, possibly unprotected environment, resource constraints in devices, cost factor, and dynamic environment [44]. However, among different preexisting solutions, ICTO could be a

viable candidate because ICTO is a digital mixture of a variety of different technologies that is currently being used in IoT applications. The IoT applications require the data to be protected with strong encryption and mechanisms to maintain data integrity, data protection, authentication, and privacy. The encryption currently used in lightweight IoT is commonly dependent on symmetric keys or Public Key Infrastructure (PKI). However, ICTO is not based on pre-shared key or PKI encryption. The keys in fact are never exposed (e.g., the public key in PKI) and the keys are generated using a NIST approved algorithm. Neither external process nor personnel have control of the keys. This restricted access and encapsulation of encryption keys makes a strong case for the encryption mechanism used in ICTO. Another major benefit of ICTO technology is that it uses "Selective Decryption" to ensure the privacy and security of the data. The Intelligent object can be protected in such a way that it can authenticate users (machine or human user) and provide limited access depending upon rules set by the owner of the data. Not everyone has the same level of authentication on the information contained in the data, so only those parts of the data that the user is authenticated to view by the owner are decrypted [76]. This preserves the integrity of the data and provides an easy way to share data without increasing risk. ICTO uses the concept of "data as an endpoint" and has the ability to protect the data at the source. Protecting the data at the site of generation i.e., IoT endpoint ensures the integrity of the data. Traditionally, data is taken as a form of passive entity that is encrypted, transformed, and protected by external mechanisms through predefined ruleset [35]. In contrast, ICTO uses the concept where the owner of the dataset can employ different policies to govern the data, thus creating a dynamic protection mechanism with owner-defined rules. The issue of privacy in IoT data is addressed by this property of ICTO as the data owner creates policies governing how and by whom the data can be accessed. This also addresses the issue of authorization as the owner

can restrict and control data access.

ICTO works efficiently both in-transit and at-rest [34] which ensures that the encapsulated data is safe. The ICTO may be transported using any channel, with or without additional protection. The question that the lack of channel protection could lead to the ICTO being vulnerable to attackers is valid, however even if attackers observe the ICTO, they cannot tamper with or observe information the ICTO holds because they are not able to decode the ICTO. The ICTO paradigm converts data to a smart entity that can track itself, keep a tamper-resistant log of where the data have been, and be independent of the application. It can record if someone unauthorized tries to access the ICTO which also addresses the issue of data confidentiality that is vital for data security.

One of the major issues in IoT is that as soon as the data leaves the private network, it is completely vulnerable. It is difficult to track who has access to the data or if the data is duplicated and transferred. However, ICTO has the advantage of being able to detect anomalies and activate the defenses no matter where the intelligent object resides. The file can deny access, and alert the owner of authorized access attempt [76]. All these properties make ICTO an attractive solution for the IoT use case.

2.5.3. Drawbacks

While ICTO has many benefits and enormous potential in IoT market, there are some drawbacks that need to be addressed. Currently the Software Development Kit (SDK) [77] is not applicable to the broader market of available IoT end devices, which are typically embedded computing platforms. As described earlier, the end devices are highly constrained in power, memory, and processing. The SDK requires relatively more memory than may be practical in IoT end devices. If the size of the SDK is reduced and adapted to particular embedded target

platforms, then the data from the end devices could be protected and validated immediately upon origination.

Furthermore, the size of the encapsulated data may be prohibitive. ICTO encapsulates the data with different protection schemes, encryptions, and clocking patterns. This means complex computations are necessary, which require power and could be a hinderance to battery powered IoT devices. The encapsulation also amounts to a substantial transmission overhead. With more overhead, the cost of the system increases. This is because of the network cost and the data storage cost. Network cost refers to the cost of the network infrastructure that is needed to transmit the data generated by the IoT devices. The additional overhead could also affect the real time data availability, as IoT systems relying on low data rate channels could have issues transporting large overhead for small data payloads. However, more experiments need to be performed to confirm the actual overhead and the impact on overall cost.

2.6. Blockchain and ICTO as possible solutions

All the data security techniques and algorithms described in Section 2.2 have some limitations. The IoT data encryption algorithms depend on keys and passwords for encryption and decryption that can be attacked by hackers to decrypt the data. This might seem trustworthy but attackers that are familiar with the encryption algorithm can potentially reverse the decryption process and the data will be vulnerable. They depend on application-based rules, and they can be overwritten by attackers having knowledge of the application. Fully homomorphic encryption is also an option as it removes the trust aspect that is otherwise essential and difficult to gain between the parties that produce the data and transfer or store the data. A fully homomorphic technique allows special computations on ciphertexts and generate encrypted results [47], but it is too cumbersome to apply in the IoT space as it requires substantial

computational power.

ICTO was chosen for the present research rather than other IoT data protection solutions because the sensor data produced in the Ingram Project context can be protected right from the edge with various data protection mechanisms and could be easier to implement at the endpoints. The control for data protection, management, and administration becomes part of the data itself and controls the data access. This approach is less reliant on predictable reversible encryption sequences for security, keys and passwords for authentication, and external apps for execution. Additionally, a set of participants can be defined which will be included with a portable dynamic rule set that can verify if the user is authorized to access the data. The authorization of data access is protected using several cloaking patterns. Thus, an ICTO approach will keep the data safe and be secured, effective, and functional regardless of the environment [34]. These characteristics are particularly valuable in an IoT deployment, and in the context of sensitive environments such as the Ingram Project.

After an extensive literature review on the use cases of blockchain as a viable technology in IoT, we chose a centralized blockchain as a data storage IoT application and a blockchain with Proof of Work (PoW) implementation to compare with the core blockchain. The core blockchain was mainly chosen to address the resource constraint of IoT endpoints. While there are different blockchain architectures that solve this issue, each has different limitations. The blockchain architecture used in this research is an approach to ledger storage blockchain that could possibly work for constrained IoT use cases.

3. EXPERIMENTAL SETUP

In this chapter, the experimental setup used for blockchain, ledger, and ICTO are described. This section also describes the methodology used to collect data, the methods used to process data, and the steps used to differentiate the impacts of blockchain, ledger, and ICTO. Section 3.1 describes the experimental configuration used for the technologies. Section 3.2 describes the different kinds of analysis performed to compare the technologies.

3.1. Technology Description

As outlined in Section 2.3, there are different versions of blockchain that could be employed in IoT, and each version tries to solve some critical issue. In this context, this research focuses on a distinct version of blockchain, diverging from the conventional distributed ledger. It uses a centralized cloud server architecture along with computing on the IoT endpoint. This unique implementation instantly counteracts the issues related to traditional distributed ledger, particularly the complexity and scalability that persists in implementing distributed ledger in IoT especially when edge computing is implemented. However, this model doesn't address the "single point of failure" that is solved by distributed ledger.

For experimentation purposes, an alternative version of blockchain was also designed, as an implementation of a distributed ledger. This implementation incorporated the PoW consensus mechanism, which involved four difficulty levels. PoW is a popular blockchain consensus mechanism widely used in distributed ledger technology. Throughout this research, the blockchain architecture incorporating the proof of work consensus mechanism will be referred to as the ledger.

Similarly, the experimental configuration for ICTO involves deploying computing on resource constrained IoT endpoint. An implementation of ICTO technology called an

Unbreakable Exchange Protocol (UXP) was used for experimental purposes [78]. The software development kit (SDK) for UXP was executed on the resource constrained device, generating the UXP object encapsulating sensor data. Both the blockchain and UXP SDK are assumed to operate on the network endpoint so that the comparison of these technologies could provide some meaningful results.

3.2. Setup

Figure 10 shows a general operational setup of the blockchain where a bundle of blocks of blockchain is assumed to be transported to the server. Figure 11 shows the operational setup of ICTO, where each block can be transmitted to the server as soon as the UXP object is created.

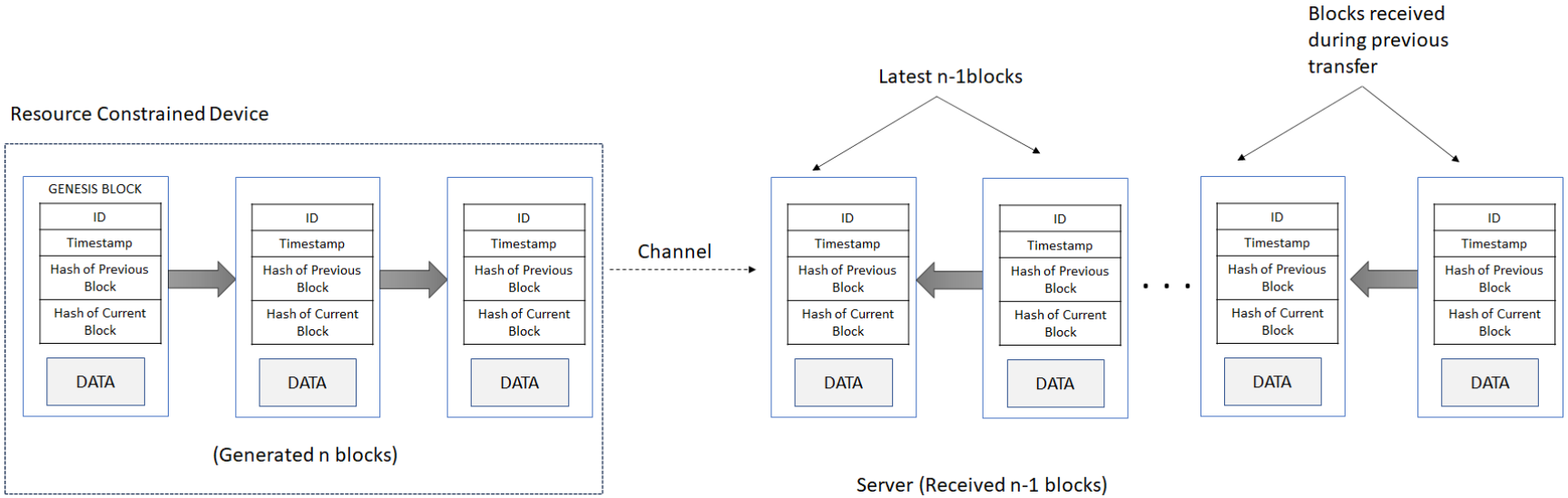


Figure 10. Blockchain Setup

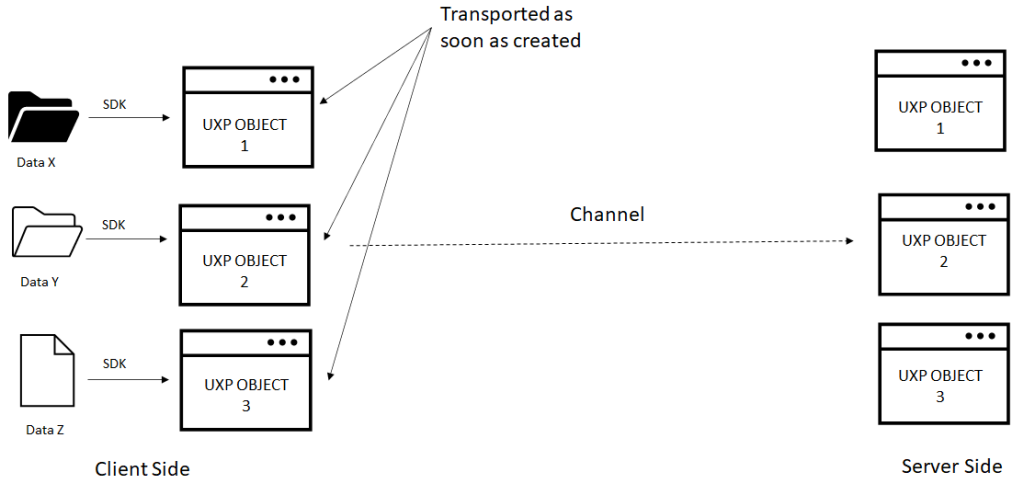


Figure 11. UXP Setup

Two distinct configurations were evaluated. The initial setup involved running the blockchain, ledger, and ICTO SDK on a computer with Linux operating system. This system featured 8 GB RAM and 4 CPUs operating at 2400MHz. This machine will be referred to as "General Purpose Machine" in this paper. Additionally, to simulate the experiments on a low-resource IoT endpoint, the blockchain and ledger were also executed on a Raspberry Pi 400 [79]. Raspberry Pi 400 was equipped with a 4 GB RAM and a CPU operating at 1.5 GHz. To validate the experiments from Raspberry Pi and simulate a low-resource device, a virtual machine was implemented with 2GB of RAM and 2GB of memory.

The local version of the blockchain to be run on the IoT endpoint was simulated on these machines. Data payloads of specific sizes were generated via a Python script, and the same payload dataset was used for experiments on three technologies. Two different kinds of payload were generated using the Python script. The first type of payload referred to as incompressible payload, was generated using the "os.urandom()" function which is generally suitable for cryptographic purposes, so the ICTO technology couldn't compress it. The second type of payload referred to as compressible payload, was generated using the random() function to get random strings and numbers which can be compressed using lossless compression algorithms. These two different versions of the same payload size were chosen to address the compression factor. Compressing the original data without losing any information has major advantages, especially in lightweight IoT because it means that a smaller number of bytes have to be transported through channel and it helps to relieve the memory and processing constraints.

The UXP object is inherently compressed and encrypted during creation. So, to make a valid comparison of blockchain, and ledger with UXP, the payloads for blockchain and ledger experiments were compressed (for compressible payloads) using a lossless LZ77 compression

with Huffman coding. Both the compressible and incompressible payloads were also encrypted using ECC hybrid encryption scheme with AES. ECC was chosen as the encryption mechanism as ECC is a popular encryption scheme for resource constrained IoT as described in Section 2.2.1.2. Python's "tinyec" library was used to generate an ECC key pair [80].

A Bash script [81] was used to automate the Python script for executing the blockchain, the ledger, and the UXP creation. The experiments were performed for at least 150 iterations to address various sources of randomness. The 95% confidence interval (CI) was generated using the t-distribution as the population standard deviation is unknown. Python's "scipy" library was used to perform operations to calculate the 95% CI [82].

4. RESULTS AND DISCUSSIONS

The results of experiments and data gathering are primarily divided into two sections: experiments performed on a general-purpose computer running the Linux operating system and experiments performed on Raspberry Pi 400. A secondary experiment on a virtual machine to address memory issues is also presented. The comparison based on the memory storage of the device was done only using Linux, since the memory space consumed is identical for any hardware. Furthermore, the experiments were further categorized into two parts: compressible data, which encompasses experiments conducted with lossless compression techniques on payloads, and incompressible data, which comprises of payloads that could not be compressed using lossless compression techniques. Each section concludes with a discussion of the experimental results, emphasizing the significance of the experiments and their respective summary.

4.1. Comparison of Blockchain and UXP relative to constrained devices' storage space

Memory storage is a significant limitation in resource constrained devices. A thorough analysis was done to address the memory storage requirements imposed by blockchain and ICTO. In this section, a general comparison between blockchain technology and ICTO is presented, considering that both versions of the blockchain i.e., core blockchain and ledger have the same memory requirements for the experiments. A theoretical analysis was carried out to estimate the amount of data to be expected from a typical IoT system. An experiment was conducted to determine the precise amount of memory storage needed to store blocks of blockchain and UXP object encapsulating payloads of varying sizes.

4.1.1. Theoretical Analysis

Let us consider a constrained IoT device that generates data of size ‘D’ every second. If ‘N’ such devices produce similar data during ‘T’ seconds per day, we can approximate the total data ‘TD’ generated from a particular IoT node as:

$$TD = D * N * T \quad (1)$$

If a single sensor node generates 10 kB of data every 10 min, runs 24 hours a day, and there are 10 such sensors, then via Equation 1, we get the total data generated by Node A to be 14.4 MB per day. Clearly, a massive volume of data is generated by a single sensor node generating just 10kB of data every 10 minutes. By employing conventional blockchain technology for IoT, the IoT endpoint has to store all the blockchain data locally. Storing such a massive volume of data is not possible in an IoT endpoint (memory constraint) without adding extra storage, which increases cost and energy consumption. Furthermore, this volume of data is generated by only one endpoint. There could be many endpoints operating simultaneously, each generating a similar data volume. This will vastly increase the amount of data that needs to be stored by each resource constrained endpoint.

4.1.2. Experimental Analysis

Table 2 presents the data obtained from experimental analysis regarding the memory storage used by blockchain and UXP for compressible and incompressible payloads. The experimental dataset was used to perform linear curve fitting and generate equations that serve as an approximation for determining the block size and UXP size for various payloads. The resulting linear equations are shown in Figure 10 and Figure 11.

Assume that a lightweight constrained memory device generates a payload represented by ‘x’. Through curve fitting, the size of the block ‘bc’ storing payload of size ‘x’ can be

approximated using Equation 2 for compressible payloads and Equation 4 for incompressible payloads. Similarly, UXP of size ‘uxp’ that encapsulates compressible data and incompressible data of payload ‘x’ can be approximated using Equation 3 and Equation 5 respectively.

$$bc = 0.75 * x + 0.30 \quad (2)$$

$$uxp = 0.75 * x + 25.30 \quad (3)$$

$$bc = x + 0.27 \quad (4)$$

$$uxp = x + 25.34 \quad (5)$$

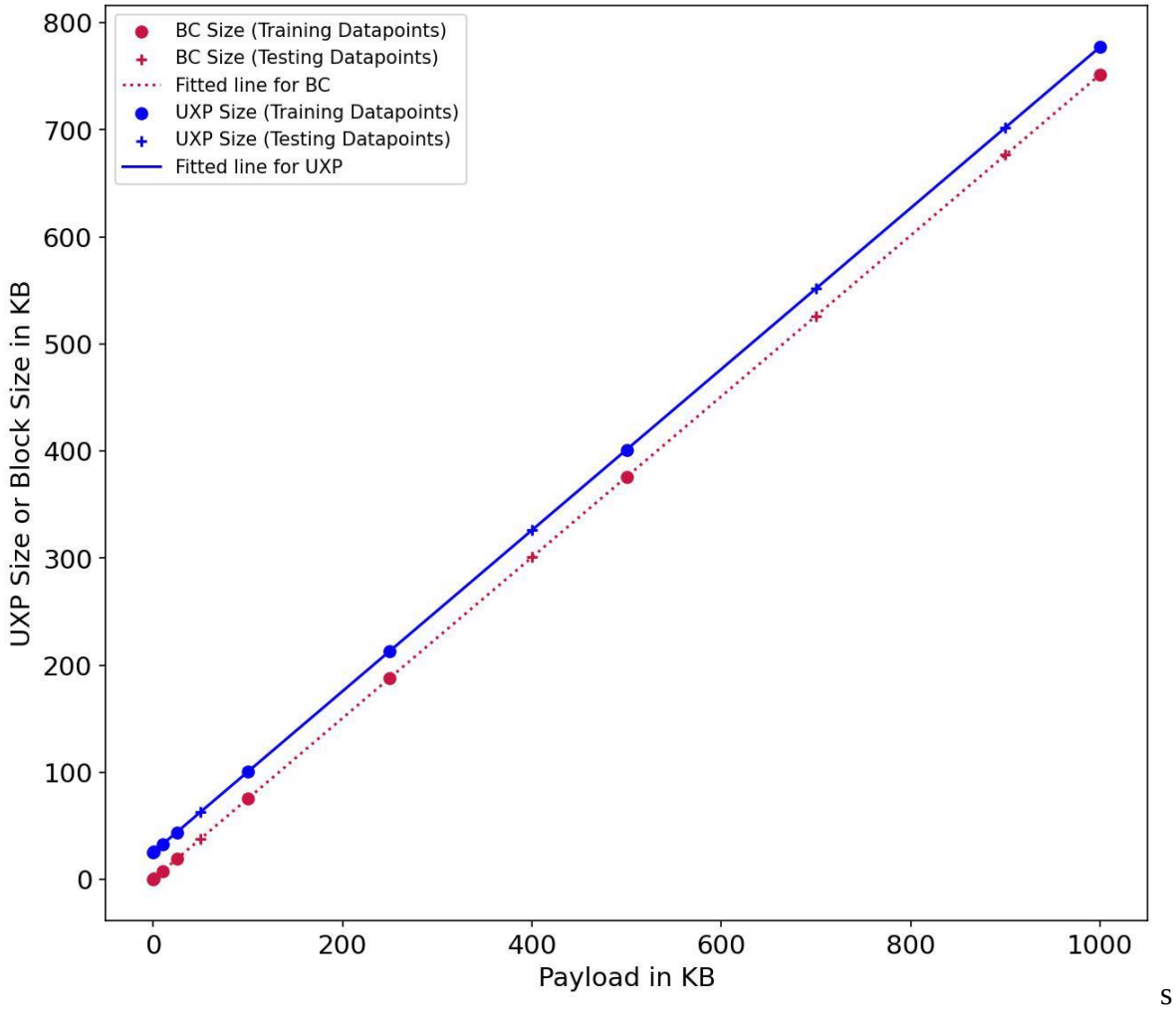
This size approximation of the block and UXP object encapsulating compressible and incompressible payloads of specific size can be used to determine the memory requirement of blockchain and UXP.

Table 2. Summary of output size information

Payload	Compressible Payloads			Incompressible Payloads		
	Blockchain	UXP		Blockchain	UXP	
	Mean	Mean	± CI * 10^{-3}	Mean	Mean	± CI * 10^{-3}
1Byte	0.29	25.30	1.12	0.26	25.33	1.09
1kB	1.06	26.07	1.14	1.26	26.34	0.88
10kB	7.81	32.82	0.87	10.26	35.38	0.97
25kB	19.07	44.10	0.83	25.26	50.34	1.07
100kB	75.42	100.49	1.00	100.27	125.37	0.77
250kB	188.13	213.30	0.97	250.27	275.42	0.10
500kB	375.94	401.32	0.98	500.27	525.49	1.14
1MB	751.6	777.33	1.10	1000.27	1025.65	1.01

Payload sizes within the specific range of 1Byte to 1MB was used as the training dataset for fitting the linear equations. To assess the accuracy and performance of the equations, a separate testing dataset was prepared. The testing dataset included payload sizes of 25kB, 400kB, 700kB and 900kB for both experiments. It is evident from Figure 12 and Figure 13 that the testing datapoints, represented by red dots for blockchain and blue dots for UXP, align closely with the straight line approximated by the training dataset. The confidence interval depicted in

Table 2 had a significantly low range, which made it unfeasible to illustrate in Figure 12.



s

Figure 12. Fitted line for blockchain and UXP storing compressible payload with training and testing datasets depicting Equation 2 and Equation 3.

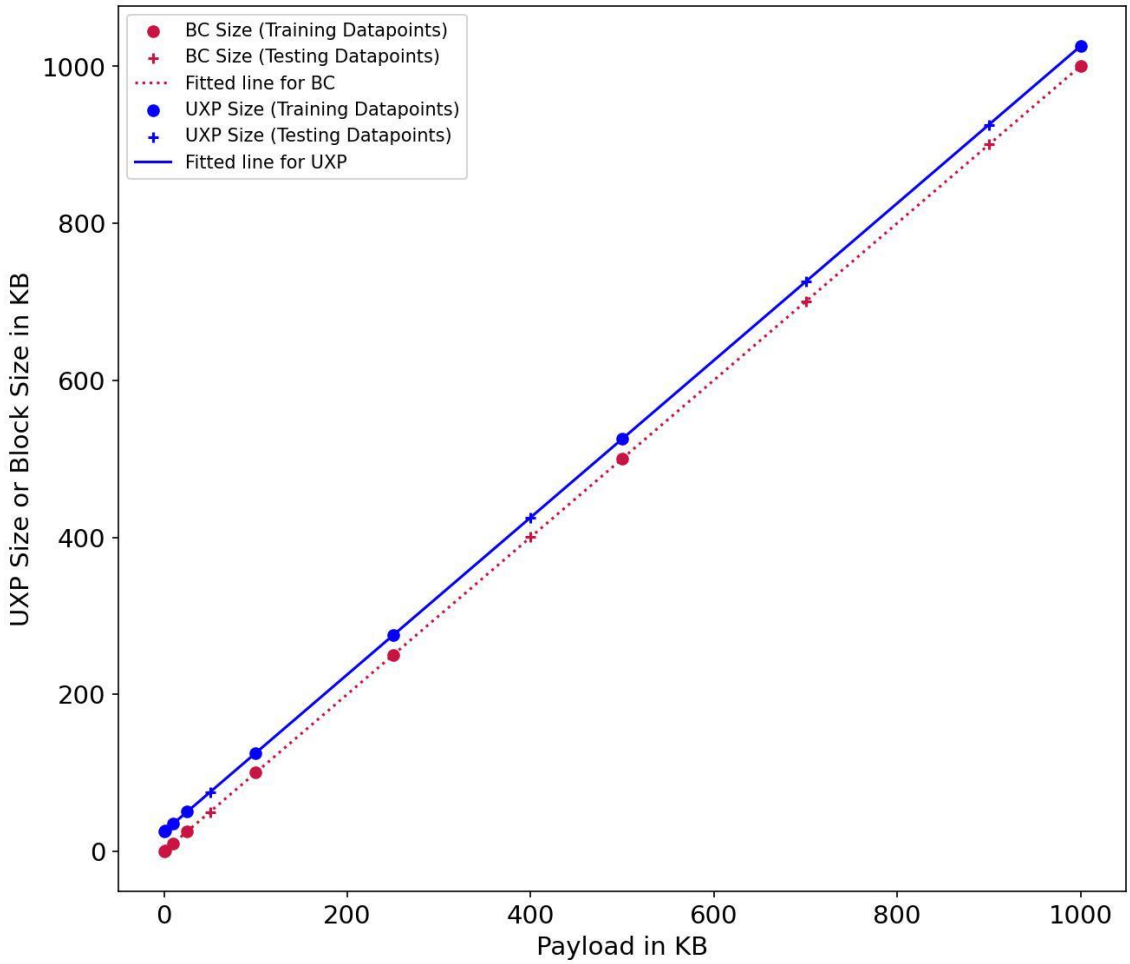


Figure 13. Fitted line for blockchain and UXP storing incompressible payload with training and testing datasets depicting Equation 4 and Equation 5.

Using Equation 1, it is possible to estimate the size of data generated from an IoT end node. And using Equation 2 and Equation 3, the size of the block necessary to store the payload or size of UXP to encapsulate the compressible payload can be approximated. For instance, the block containing a payload of 50 kB will be 37.8 kB and UXP encapsulating 50 kB payload will be 62.8 kB. Similarly, Equation 4 and Equation 5 can be used to approximate the block size for UXP size for incompressible payloads. For instance, the block containing a payload of 50 kB will be 50.27 kB and UXP encapsulating 50kB payload will be 75.34 kB. To summarize, these equations offer a practical approach to approximate the size of a block or UXP required to store or encapsulate payload of specific size.

Table 3. Information about difference in block size and UXP Size.

Payload Size (x)	Mean size for compressible payload (uxp - bc)	Mean size for Incompressible payload (uxp - bc)
1Byte	25.00	25.07
1kB	25.00	25.08
10kB	25.03	25.07
100kB	25.07	25.10
250kB	25.17	25.15
500kB	25.38	25.22
1MB	25.73	25.37
Mean Value	25.20	25.15

Table 3 presents the experimental results containing the mean difference in sizes between blockchain blocks and UXP objects. Figure 14 visually represents the disparity in size between a block and UXP encapsulating compressible and incompressible payloads of the same size. It shows that the difference in size of the UXP object and blockchain storing same sized payload has a mean of 25.20 kB for compressible payload and 25.15 kB for incompressible payload.

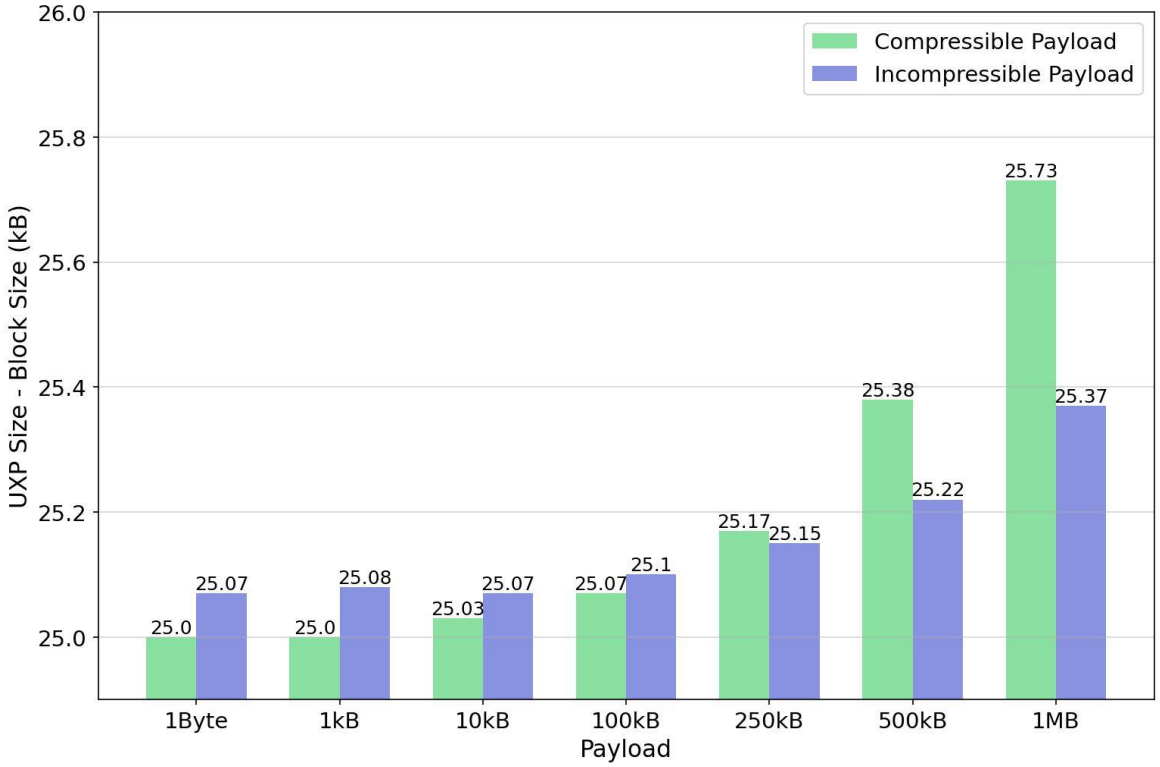


Figure 14. Difference in UXP Size vs Block Size

4.2. UXP Overhead

As discussed in Section 2.5, UXP is a digital mixture of different technologies and UXP encapsulates the data with several protection mechanisms to secure it. This comes at the cost of a considerable amount of overhead. An experiment was designed to determine the exact overhead incurred to encapsulate the data within the range of interest. Data payload of range 1 Byte to 1 MB was encapsulated using ICTO and the resulting size of the UXP object was noted. The overhead size was calculated by subtracting the size of UXP object from the size of the payload. The information of the experiment is summarized in Table 4 where the payload size is represented by ‘x’ and size of UXP object by ‘uxp’. So, the size of the overhead is the difference between the size of UXP object and payload size. Revising Equation 3 for compressible payload and Equation 5 for incompressible payload from Section 4.1.2.

$$uxp = 0.75 * x + 25.30 \quad (3)$$

$$uxp = x + 25.34 \quad (5)$$

Table 4. UXP Overhead incurred for encapsulating compressible and incompressible payload.

Payload size (x)	Mean overhead of compressible payload (kB) (uxp-x)	Mean overhead of incompressible payload (kB) (uxp-x)
1Byte	25.27	25.33
1kB	25.27	25.34
10kB	25.27	25.34
100kB	25.34	25.37
250kB	25.45	25.42
500kB	25.67	25.49
1MB	26.00	25.65
Mean Value	25.47	25.41

Table 4 provides a comprehensive analysis of the UXP encapsulation overhead for incompressible and compressible payload sizes, respectively, ranging from 1 Byte to 1 MB. It is evident from Table 4 that the payload overhead for different payload sizes is remarkably consistent. Table 4 clearly shows that even when the UXP size increases, the UXP overhead remains constant with a mean value of 25.41 kB for incompressible payload and 25.47 kB for compressible payload.

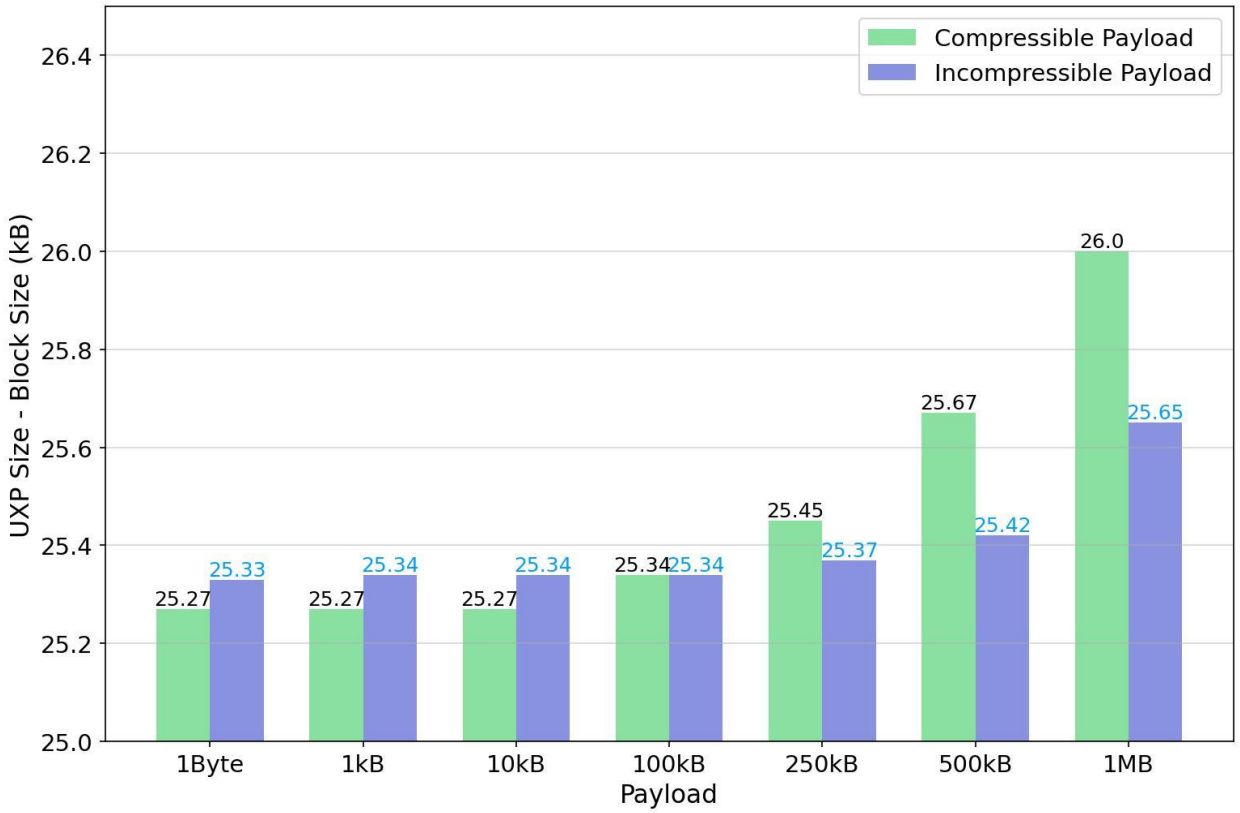


Figure 15. Bar graph showing UXP overhead for different incompressible payload sizes.

The consistent overhead for different payload sizes is also evident from Figure 15. This consistent overhead, which is unaffected by the size of the incompressible or compressible payload, represents a significant benefit of ICTO. The header length of the UXP doesn't grow exponentially with the increase in payload size but remains constant within a mean value of 25.47 kB or 25.41 kB. This suggests that UXP becomes more efficient as the size of the payload increases. This constant increase in the difference in overhead and payload size, with an increase in payload size, is one of the major benefits of UXP and presents a compelling argument for the viability of ICTO for lightweight IoT.

4.3. Effect of Blockchain and ICTO on IoT Device Memory Storage

Assume that a lightweight constrained memory IoT device generates data of size ‘x’, and the total usable memory storage of the device is ‘z’. Let ‘b’ be the size of block required to store the payload ‘x’. Then the maximum number of blocks ‘y’ that can be stored in the IoT device can be computed as:

$$y = \frac{z}{\mathrm{b}} \tag{6}$$

Let us suppose for an IoT device, usable memory storage ‘z’ be 1 MB. Equation 6 can be modified as:

$$\mathrm{y} = \frac{1\mathrm{MB}}{\mathrm{b}}$$

UXP objects can be transported as soon as they are created. However, for blockchain it is efficient to transport large number of blocks at once. This is because transporting data through any channel requires communication overhead which should be minimized. As described in Section 2.4, the blocks must be encrypted before transport, which also increases overhead. UXP object is already a protected object with no additional protection necessary during transit or rest. For blockchain data security during transport, one approach is to encrypt several blocks at once. As a result, as many blocks must be temporarily stored in the IoT device.

Table 5 and Table 6 illustrate the threshold after which the memory space consumed by blockchain exceeds that of single UXP object, and this value is represented by random variable ‘m’. The results indicate that as the size of the dataset grows, UXP outperforms blockchain in terms of memory space consumption. If the payload is compressible and its size is 1 kB (assuming 1 kB data is stored in every block or UXP object), the size of blockchain will surpass that of a single UXP object after 24 blocks. As a result, beyond the 24-block threshold, the size of blockchain exceeds the size of single UXP object.

Table 5. Comparison of blockchain and UXP encapsulating compressible payload in terms of memory space consumption.

Payload (x)	Block Size (b)	Maximum blocks (y)	UXP size (kB)	No. of Blocks after which blockchain takes more space than UXP (m)
1Byte	0.29	3448	25.30	87
1kB	1.06	943	26.07	24
10kB	7.81	128	32.82	4
25kB	19.07	52	44.12	2
100kB	75.42	13	100.48	1
250kB	188.125	5	213.29	1
500kB	375.94	2	401.32	1

Table 6. Comparison of blockchain and UXP encapsulating incompressible payload in terms of memory space consumption.

Payload (x)	Block Size (b)	Maximum blocks (y)	UXP size (kB)	No. of Blocks after which blockchain takes more space than UXP (m)
1Byte	0.26	3846	25.33	97
1kB	1.26	793	26.34	20
10kB	10.26	97	35.38	3
25kB	25.26	39	50.34	1
100kB	100.27	9	125.37	1
250kB	250.27	3	275.42	1
500kB	500.27	1	525.49	1

Table 5 and Table 6 provide a comparison between unencrypted blocks of blockchain and UXP in terms of memory space consumption for compressible and incompressible payloads, respectively. Thus, a maximum of 793 blocks with 1 kB of incompressible payload can be stored per 1 MB. In contrast, a maximum of 943 blocks with 1 kB of compressible payload can be stored per 1 MB. As the payload size increases, the maximum number of blocks that can be stored in the IoT device is reduced. Additionally, the number of blocks after which a blockchain requires more space than a single UXP object also decreases rapidly for both compressible and incompressible payload. Figure 16 presents a bar chart showing the number of blocks after which blockchain takes up more space in the IoT device in comparison to one UXP object

encapsulating the same payload. This value is represented by the variable ‘m’ in Table 5 and Table 6. This reduction in storage capacity as payload increases creates memory issues in resource constrained IoT devices.

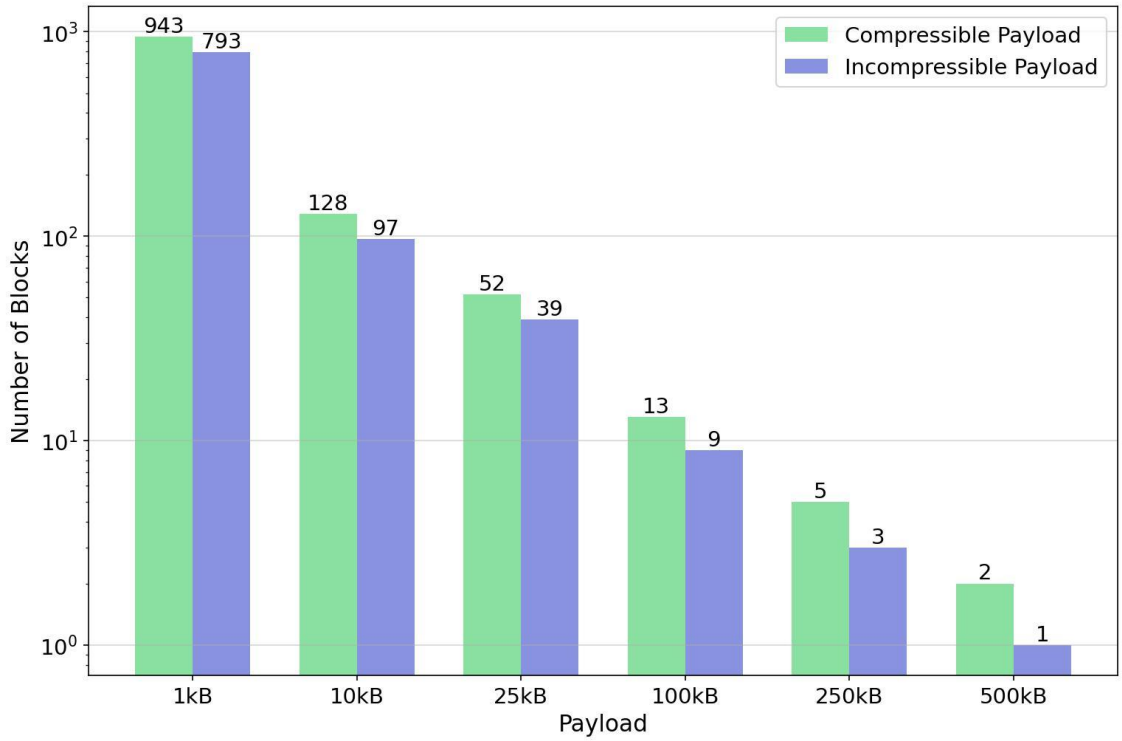


Figure 16. A bar chart showing the number of blocks before blockchain takes more storage than UXP.

4.4. Execution Time

Measuring the time required to store or encapsulate data is crucial for understanding performance of the resource constrained IoT device and battery requirement. Regarding this context, an experiment was conducted to compare the time required to store payload in blockchain, time required to store data in the ledger, and the time required to encapsulate payload using UXP technology. Table 7 and Table 8 present data depicting the time required to store incompressible and compressible payloads in blockchain, encapsulated via UXP, and stored in the ledger respectively while Figure 17 and Figure 18 illustrate the data in a plot.

As noted from an experiment in general purpose machine, the time to encapsulate data

using UXP is more than 7 times greater than the time it takes to store data in the blockchain. Similarly, the time taken by UXP technology is more than 3 times greater than the time it takes to store data in the ledger. From experiments in Raspberry Pi, it was found that the time to store payload in ledger is more than 1.4 times greater than the time required to store data in blockchain for compressible payloads and incompressible payloads. This disparity in execution time between the blockchain and ledger can be attributed to the computational requirements of PoW, which requires calculating a hash with specific difficulty level. This task poses a challenge for IoT devices with limited resources and computing power. Interestingly, the time difference between storing 1 Byte of payload and 1 MB of payload in the blockchain in general purpose machine, on average, was just 7.15ms and 13.25ms. This indicates that the payload size is independent of the time required to store data in the blockchain.

Figure 17 and Figure 18 also highlight the narrowness of confidence intervals for time measurements of blockchain in both cases, indicating extremely low standard deviation and consistent time requirements regardless of payload size. Conversely, time measurements for UXP experiments have larger standard deviation, reflecting greater variability in the dataset. This could possibly stem from the different encryptions, and cloaking mechanisms used by UXP. To depict this variability and randomness in the experiments, confidence intervals are computed.

Table 7. Execution time measurements for compressible payload.

Payload	General Purpose Machine						Raspberry Pi					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI
1Byte	254.5	5.6	488.2	56.4	1824.4	84.1	1196.0	25.4	1735.7	118.7	?	?
1kB	249.6	2.1	502.9	66.6	1895.5	128.6	1188.7	18.8	1971.9	159.9	?	?
100kB	246.4	0.9	500.4	66.7	2004.7	153.8	1199.2	33.0	2015.6	178.6	?	?
500kB	251.8	1.1	525.5	69.3	2015.5	160.4	1226.9	34.9	2232.9	232.3	?	?
1MB	267.7	8.2	577.4	73.8	2354.9	297.2	1244.3	8.9	2376.5	203.6	?	?

Table 8. Execution time measurements for incompressible payloads.

Payload	General Purpose Machine						Raspberry Pi					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI
1Byte	255.3	6.2	502.7	73.8	2263.8	153.1	1335.8	60.1	2158.7	246.6	?	?
1kB	246.9	1.0	499.2	63.2	2062.9	159.1	1395.7	139.1	2052.1	186.1	?	?
100kB	246.4	1.1	493.6	56.9	2114.6	238.0	1393.9	117.6	2204.3	343.4	?	?
500kB	255.7	2.1	537.3	73.9	2140.9	178.2	1455.1	112.9	2492.5	235.4	?	?
1MB	262.4	0.9	556.9	58.3	2474.6	132.8	1539.1	53.9	2592.1	360.1	?	?

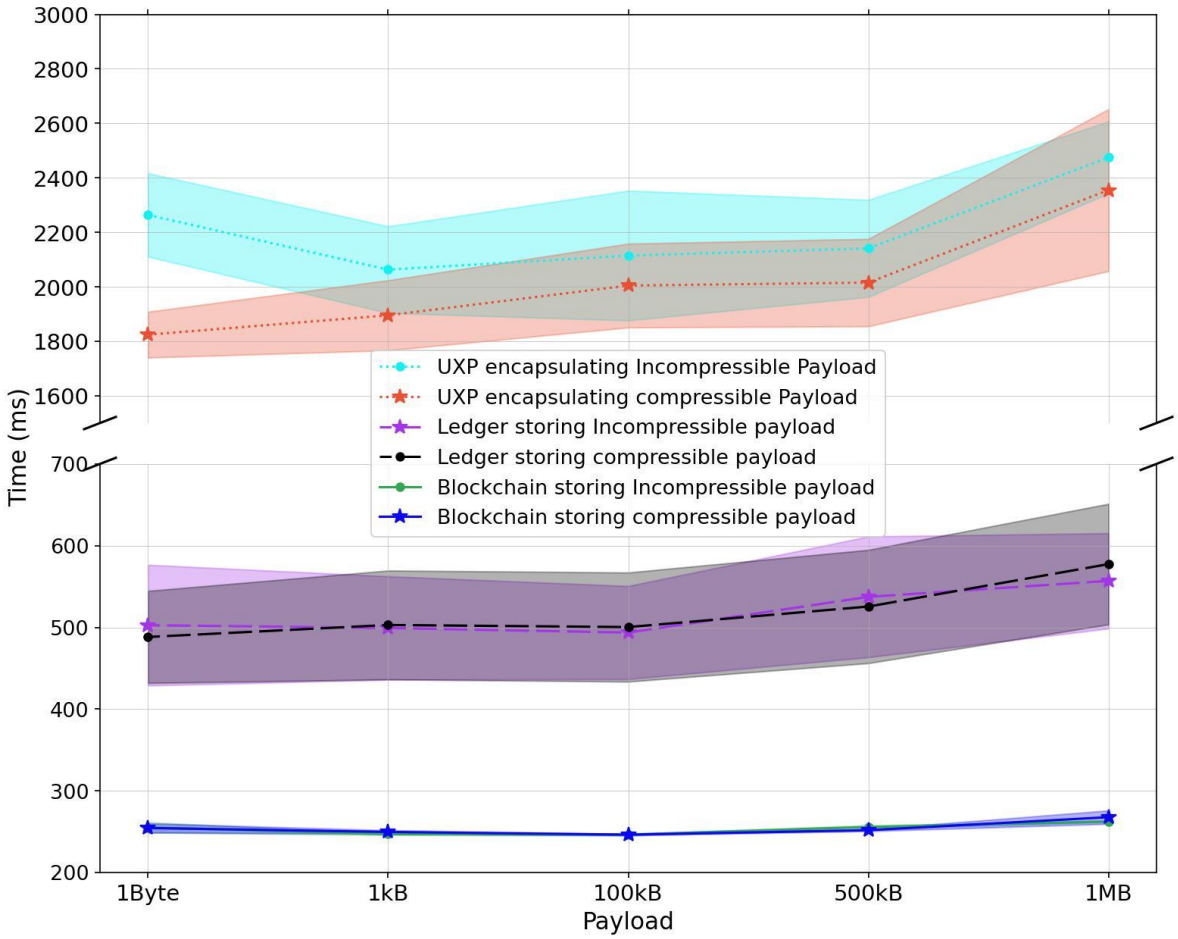


Figure 17. Blockchain vs UXP vs ledger in terms of time required to store payload in general purpose machine.

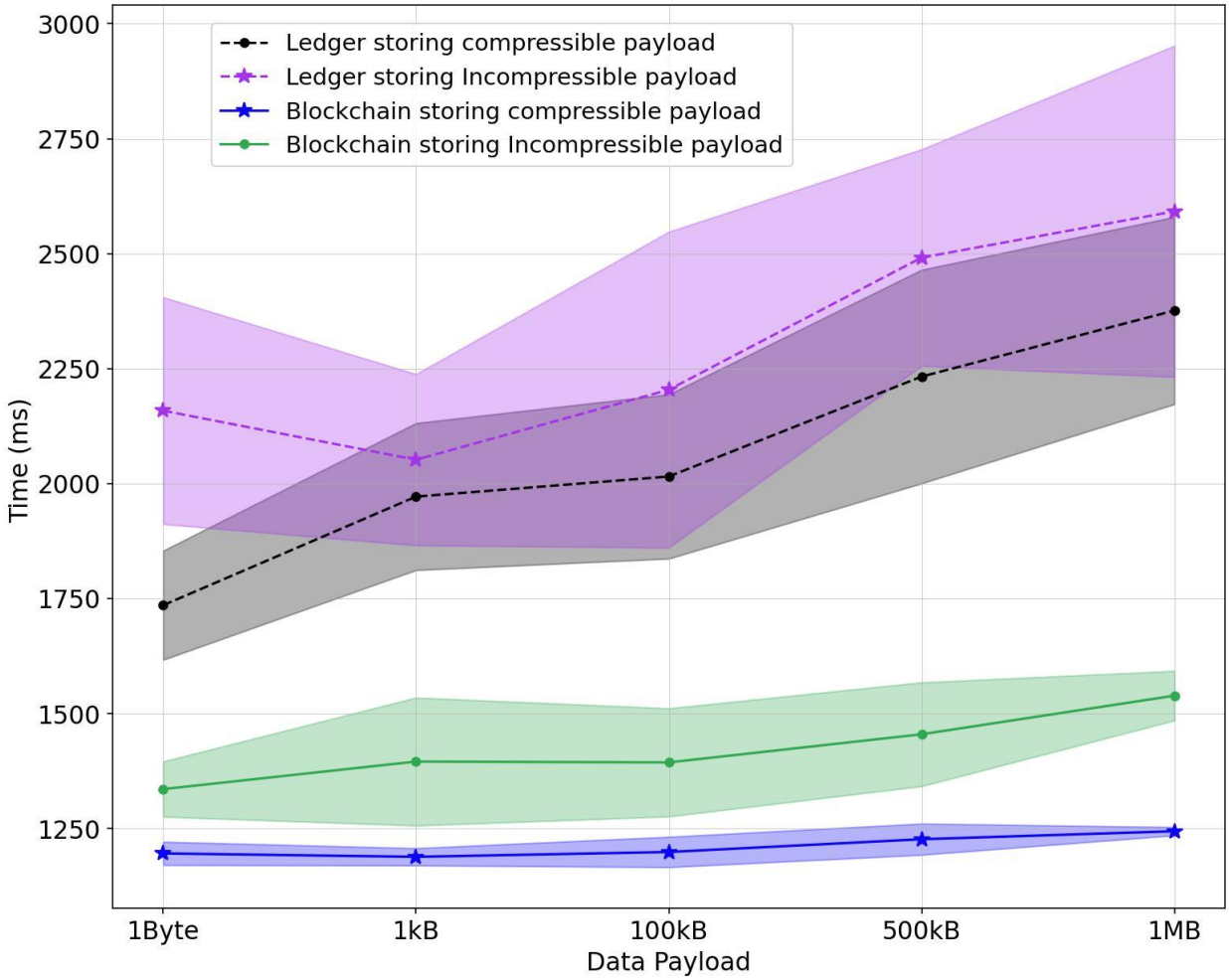


Figure 18. Blockchain vs ledger in terms of time required to store payload in Raspberry Pi.

4.5. CPU Clock Cycles

The measurement of CPU clock cycles is an important parameter for resource constrained IoT devices, as it provides insights into performance evaluation. CPU clock cycles directly influence the power consumption, heat dissipation, and resource allocation, making it a crucial parameter to understand. So, to compare the blockchain and ICTO for resource constrained IoT device, an experiment was performed to understand the number of clock cycles used by these technologies for different payloads. Table 9 and Table 10 presents data from measurements of CPU clock cycles for blockchain, UXP, and ledger technologies storing compressible and incompressible payload of various sizes, respectively. This is illustrated in Figure 19 and Figure

20.

The results from general purpose machine demonstrate that the CPU clock cycles required for encapsulating payload by UXP were only more than 2 times higher than storing data in blockchain. Interestingly, the CPU clock cycle requirement for UXP was nearly equal to the time requirement for ledger. This was the case for both compressible as well as incompressible payloads. Similarly, the results from Raspberry Pi demonstrate that the CPU clock cycles required for storing payload by ledger was more than 1.8 times higher than storing data in blockchain for compressible payload and more than 2 times higher for incompressible payload.

Table 9. CPU clock cycle measurements for compressible payloads.

Payload	General Purpose Machine						Raspberry Pi					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean	± CI
1Byte	5.07	0.14	10.59	1.34	11.74	0.04	11.53	0.1	21.88	3.08	?	?
1kB	4.95	0.05	10.67	1.63	11.68	0.04	11.53	0.08	24.33	4.16	?	?
100kB	4.89	0.02	11.84	1.99	12.05	0.04	11.45	0.03	25.54	3.19	?	?
500kB	5.02	0.02	11.48	1.66	13.39	0.03	11.88	0.03	28.76	4.1	?	?
1MB	5.34	0.2	12.58	1.75	16.44	0.21	12.25	0.03	31.26	3.5	?	?

Table 10. CPU clock cycle measurements for incompressible payloads.

Payload	General Purpose Machine						Raspberry Pi					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean * 10^8	± CI	Mean	± CI
1Byte	5.09	0.15	10.91	1.77	11.65	0.03	12.57	0.25	25.21	3.94	?	?
1kB	4.90	0.01	10.84	1.51	11.75	0.03	12.80	0.04	25.74	3.29	?	?
100kB	4.88	0.02	11.34	1.85	12.05	0.05	12.88	0.03	26.48	4.78	?	?
500kB	5.00	0.02	11.71	1.76	11.95	0.21	14.35	0.03	32.87	4.32	?	?
1MB	5.16	0.02	12.94	1.39	15.86	0.2	16.04	0.04	34.09	3.72	?	?

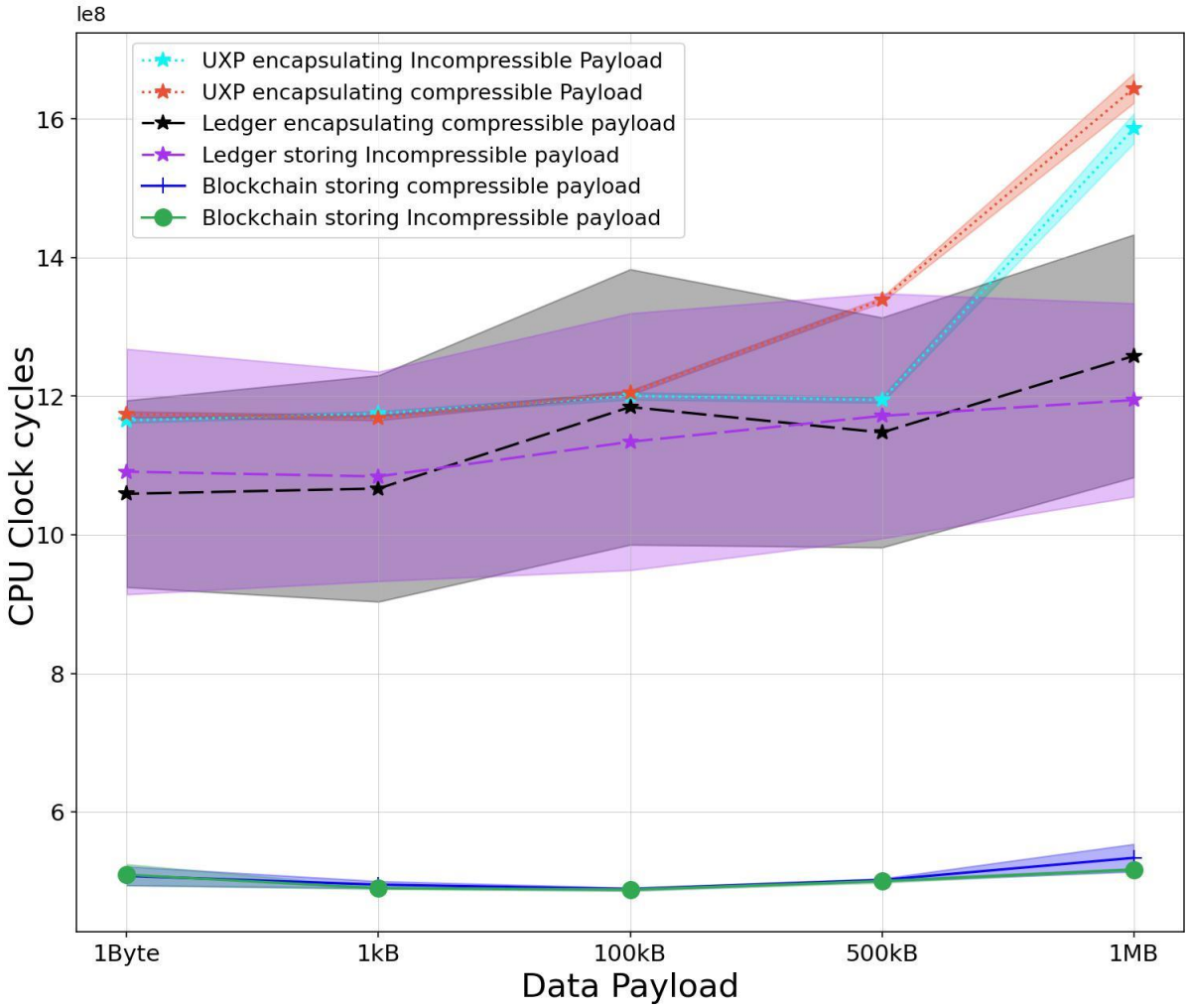


Figure 19. Comparison of CPU clock cycles for blockchain, ICTO, and ledger in general purpose machine.

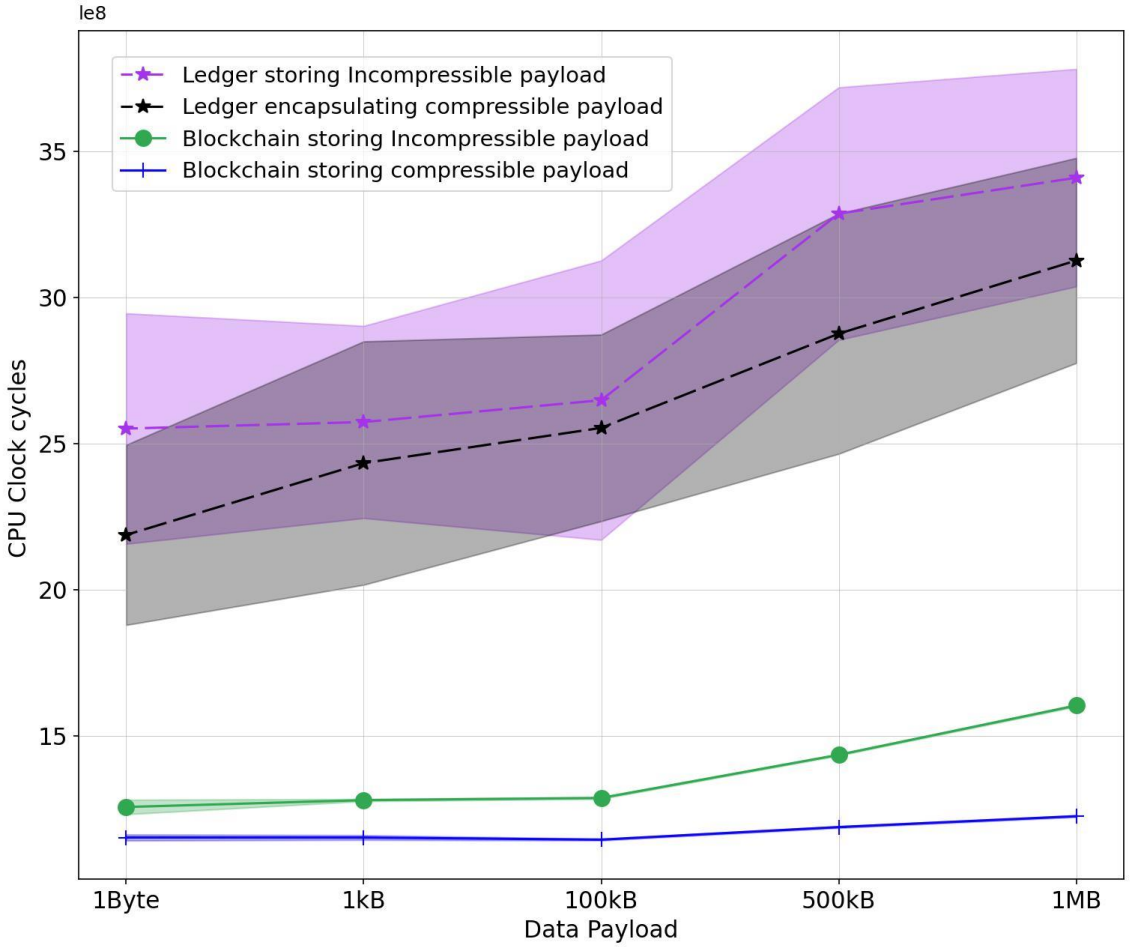


Figure 20. Comparison of CPU clock cycles for blockchain, and ledger in Raspberry Pi.

The ledger implements PoW as the consensus mechanism for blockchain which is a complex task of finding the right hash. To understand this complexity, multiple experiments were done with varying difficulty levels in Raspberry Pi. Table 11 presents data from measurements of CPU clock cycles for ledger technologies with varying difficulty levels storing incompressible payload of various sizes. The results demonstrate that the CPU clock cycles required for storing payload by ledger with higher difficulty is extremely unpredictable as depicted by large confidence interval in Figure 21. It was found that the CPU clock cycle requirements for ledger with difficulty level set to 6 is more than a staggering 75 times higher than that for ledger with difficulty level set to 4. Furthermore, it was found that increasing

difficulty level from 4 to 5 increases the CPU clock cycle by more than 7 times. This shows the extreme complexity of finding the right hash in a resource constrained device like Raspberry Pi.

Table 11. CPU clock cycle measurements of incompressible payloads in Raspberry Pi.

Payload	Difficulty 4		Difficulty 5		Difficulty 6	
	Mean * 10^9	± CI * 10^{-1}	Mean * 10^9	± CI	Mean * 10^9	± CI
1Byte	2.55	3.94	26.21	8.79	407.08	136.18
1kB	2.57	3.29	31.20	6.66	269.56	93.05
100kB	2.65	4.78	20.73	8.93	349.12	104.36
500kB	3.29	4.32	27.13	6.90	249.12	59.24
1MB	3.41	3.72	26.34	6.01	505.75	203.49

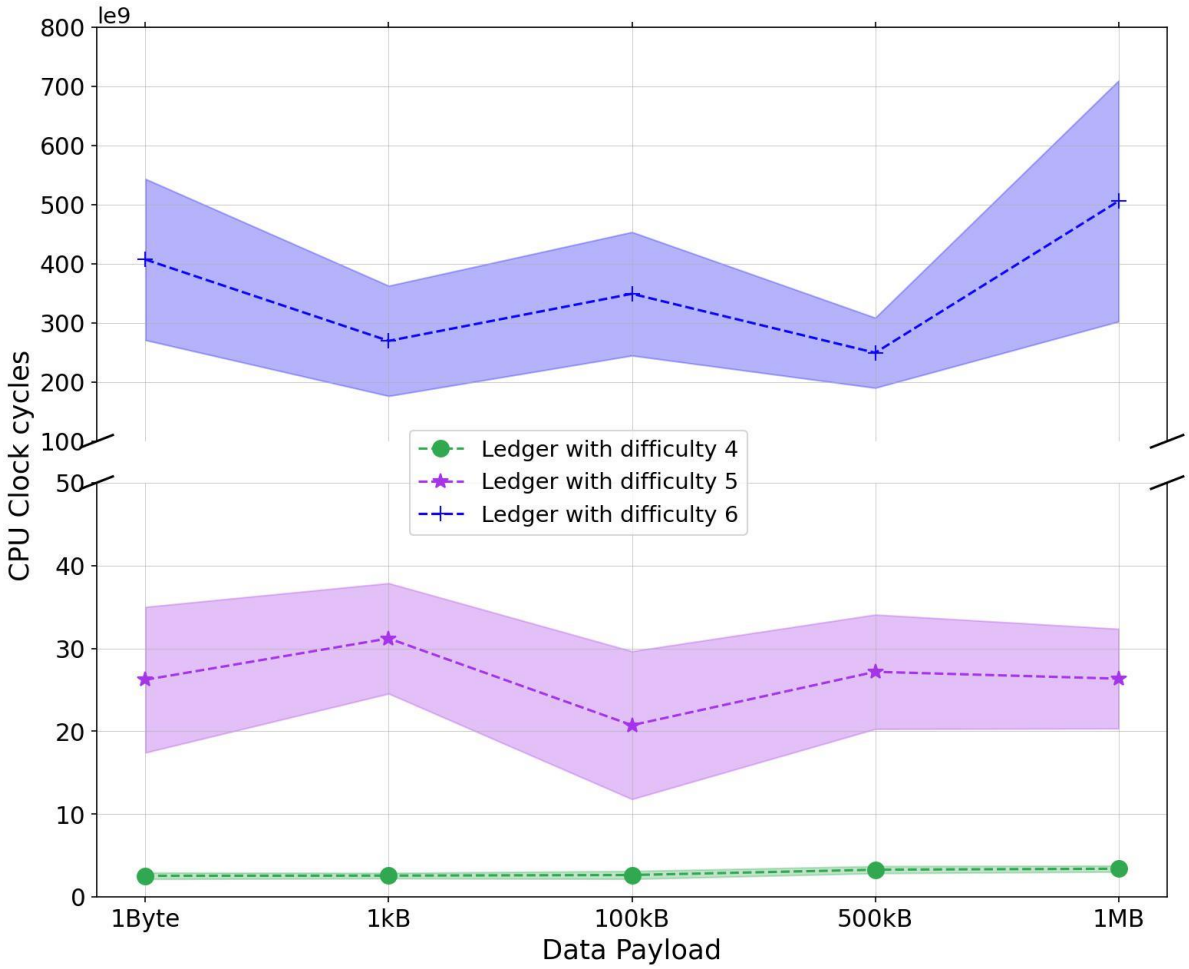


Figure 21. CPU clock cycles requirement for different difficulty levels in Raspberry Pi.

4.6. Resource measurement

When choosing the ideal technology for IoT applications, it is important to balance security, resource efficiency, and performance. Keeping this in mind, an experiment was designed to compare the memory consumption by blockchain, the ledger, and ICTO on a local machine. In terms of memory consumption, it is important to understand the Resident Set Size (RSS) and Virtual Memory Size (VSZ) to address the memory constraint of running blockchain or ICTO on the local machine. Resident set size is a measure of how much memory is consumed by a particular process in the physical RAM of the machine [83]. Virtual memory size is a measure of how much memory a particular process can access including the swapped memory, allocated memory that's not used, and memory from external libraries. Both memory types are an important measure of how much constraint a particular program is applying on the CPU. The actual physical memory or RAM used by a program is reflected by the resident set size, whereas total memory space allocated for the program is reflected by the virtual memory size. This is why both the parameters are measured to get a clearer understanding of memory efficiency of the technologies. Especially in the case of resource constrained applications, it is important to understand the holistic view of memory management. Monitoring the RSS provides insights into the memory footprint of a program, helping ensure it remains within available physical memory limits. Additionally, monitoring the VSZ can help understand the memory requirements of a program and potential memory expansion that can occur through techniques like swapping or paging.

Due to the inherent unpredictability of the operating system, there was a significant level of randomness involved during the experimentation process. The randomness of the experiment arises from the unpredictable and dynamic nature of OS. There could be several processes

running at any given time and the CPU is optimized for efficiency purposes not predictability. This makes any experiment on CPU prone to randomness. To address this randomness, a confidence interval was created. Experiments performed on Raspberry Pi were inconclusive leading to experiments on a virtual box with ubuntu OS with 2GB RAM and 2GB memory.

4.6.1. RSS Memory

Table 12 and Table 13 present the average resident set size memory values obtained from the experiments for compressible and incompressible payloads respectively, while Figure 22 and Figure 23 display this data on a graph, including the confidence interval. Figure 22 and Figure 23 clearly illustrate that the RSS memory allocated for storing data in blockchain and ledger is significantly higher than the RSS memory size allocated when encapsulating the same payload using ICTO technology. The results indicated that, for the payload size without our range of interest, RSS size for storing payload on blockchain was more than 1.4 times greater than RSS size for encapsulating payload using ICTO. Also, RSS size for storing payload on ledger was more than 1.6 times greater than RSS size for encapsulating payload using ICTO. Similarly, experiments on ubuntu virtual box also showed that the RSS memory consumed by UXP was the lowest as depicted in Figure 26. RSS size for storing payload on ledger and blockchain was more than 2.1 times and 1.8 times greater than RSS size for encapsulating payload using ICTO respectively.

Additionally, no significant difference in the RSS size was observed between the blockchain and ledger, with maximum difference between RSS memory of 3.68 MB and 5.87 MB. Both experiments yielded comparable results in terms of RSS memory usage. Furthermore, no distinct separation could be made about the RSS memory difference between the compressed and incompressible payloads.

Table 12. Summary of RSS memory measurements of blockchain and ICTO experiment for compressible payloads

Payload	General Purpose Machine						Virtual Box					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI
1Byte	17.26	0.01	19.16	0.01	10.99	0.44	20.58	0.08	25.10	0.49	11.8	7.93
1kB	17.30	0.01	19.15	0.02	11.26	0.47	20.50	0.11	25.20	0.55	11.7	9.24
100kB	17.27	0.01	19.39	0.08	11.72	0.62	21.12	0.18	27.00	0.62	11.7	9.24
500kB	17.93	0.06	20.70	0.33	12.92	0.76	23.13	0.86	27.20	0.87	11.9	5.72
1MB	18.15	0.09	21.83	0.56	12.18	0.83	24.11	0.65	28.12	0.70	12.0	8.06

Table 13. Summary of RSS memory measurements of blockchain and ICTO experiment for incompressible payloads

Payload	General Purpose Machine						Virtual Box					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI $* 10^{-1}$	Mean (MB)	± CI $* 10^{-1}$	Mean (MB)	± CI $* 10^{-2}$
1Byte	17.26	0.02	19.12	0.03	10.93	0.37	25.04	4.61	26.85	5.33	12.00	0.31
1kB	17.27	0.01	19.12	0.02	11.00	0.57	25.47	4.65	27.27	5.66	12.14	0.23
100kB	17.41	0.05	19.50	0.11	13.16	0.69	25.64	3.98	27.23	5.50	12.49	0.17
500kB	17.64	0.08	21.09	0.40	15.06	1.16	26.13	5.98	27.64	6.79	12.93	0.18
1MB	17.97	0.13	22.69	0.87	15.18	1.47	26.94	5.13	28.09	7.79	13.49	0.78

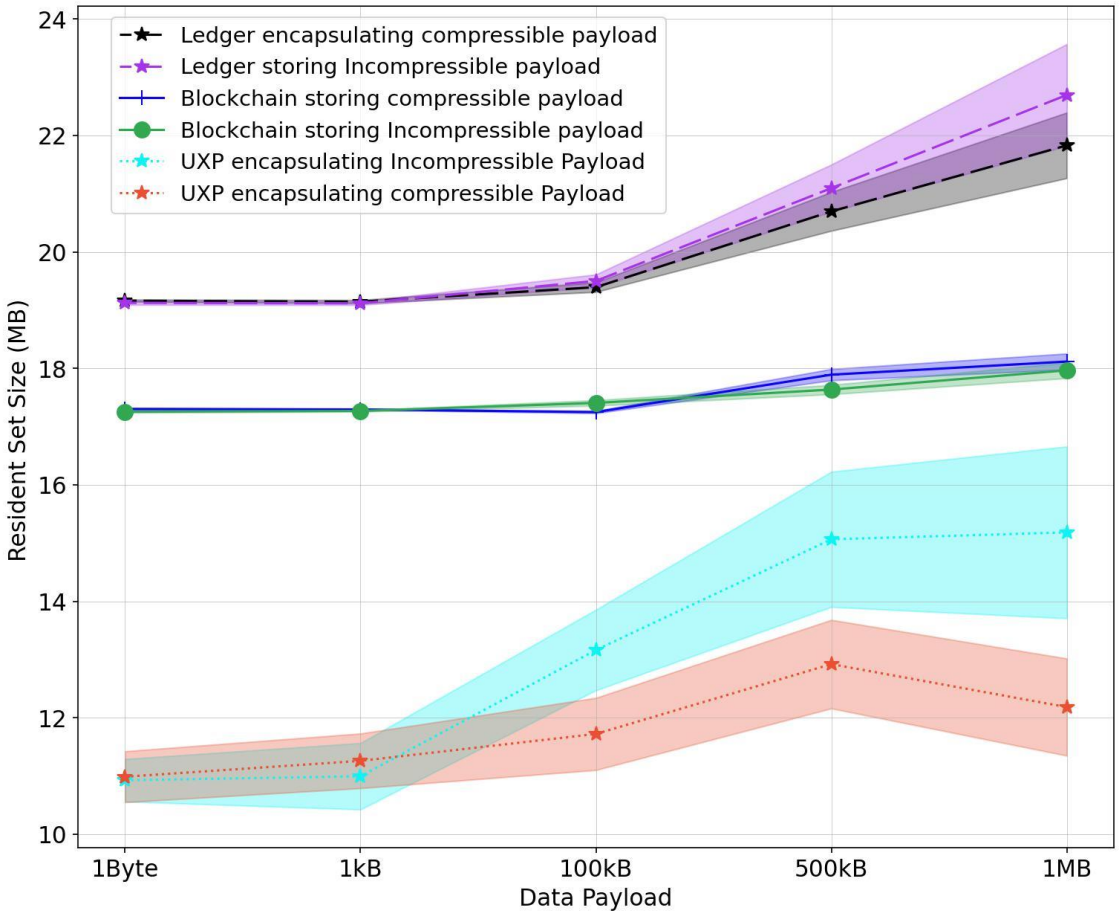


Figure 22. RSS memory comparison for blockchain, ledger, and ICTO on general purpose machine.

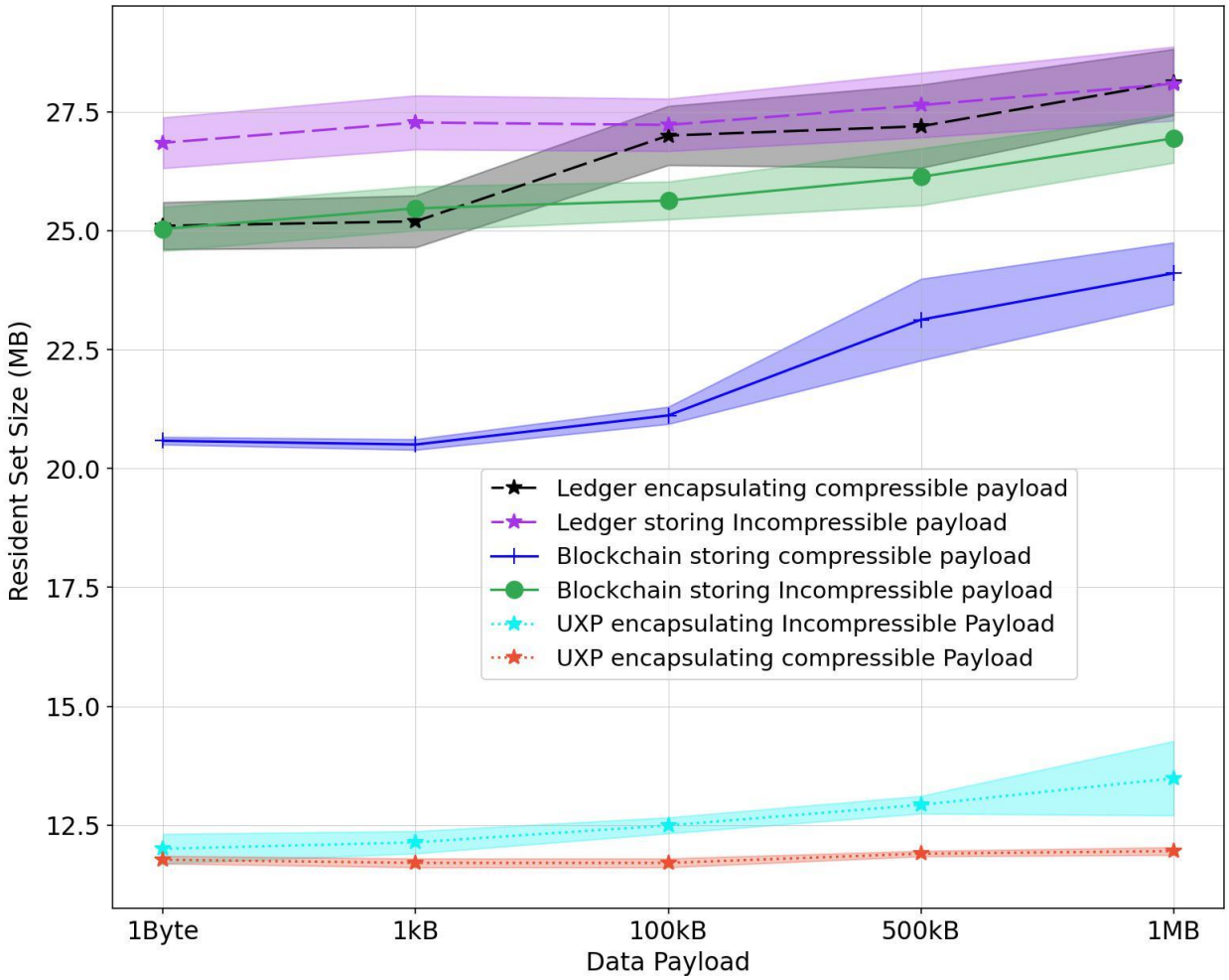


Figure 23. RSS memory requirement of blockchain and ledger on virtual box.

4.6.2. VSZ Memory

Table 14 and Table 15 depict the average virtual memory size values acquired from the experiments for compressible and incompressible payloads, while Figure 24 and Figure 25 graphically represent this data. In contrast to RSS memory, the VSZ memory allocated for storing data in blockchain was found to be substantially smaller than the VSZ memory size allocated when encapsulating the same payload using ICTO technology. From experiments in general purpose machine, VSZ memory for ICTO was found to be more than 3 times greater than VSZ memory required for blockchain and ledger. Experiments from ubuntu virtual machine

confirmed these results, as it was found that VSZ memory for ICTO was found to be more than 1.2 times greater than blockchain and 1.7 times greater than ledger for compressible payload. While measuring the VSZ memory, high memory spikes were observed in irregular fashion. These spikes exhibited an unpredictable nature and reached very high values. Such occurrences could potentially be associated with memory leaks that may occur during program operation.

Like RSS memory, no discernible difference in the VSZ memory was observed between the blockchain and ledger implementations. However, it is important to point out that the VSZ memory for UXP for higher sized payloads was significantly higher than for compressible payloads. This is because of the size difference of the compressible and incompressible payloads, which increases with increase in payload sizes.

Table 14. Raw data of VSZ memory measurements of blockchain, ledger, and ICTO experiment for compressible payloads

Payload	General Purpose Machine						Virtual Machine					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (MB)	± CI * 10^{-2}	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI
1Byte	28.70	0	30.56	0	90.21	4.21	29.43	0.18	37.45	0.59	73.21	9.70
1kB	28.70	0	30.56	0	88.71	5.08	29.82	0.18	38.86	0.75	75.01	9.39
100kB	28.84	0.82	30.84	0.05	90.27	4.18	29.59	0.23	45.09	0.56	79.38	8.56
500kB	29.01	2.65	31.96	0.32	90.20	4.21	29.16	0.48	43.54	1.03	77.93	8.87
1MB	29.34	4.93	33.12	0.58	91.68	3.01	29.66	0.41	41.72	0.87	81.45	8.06

Table 15. Raw data of VSZ memory measurements of blockchain, ledger, and ICTO experiment for incompressible payload

Payload	General Purpose Machine						Virtual Machine					
	Blockchain		Ledger		UXP		Blockchain		Ledger		UXP	
	Mean (MB)	± CI *10^{-2}	Mean (MB)	± CI	Mean (MB)	± CI	Mean (MB)	± CI *10^{-1}	Mean (MB)	± CI *10^{-1}	Mean (MB)	± CI
1Byte	28.70	0	30.56	0	86.16	7.99	39.46	4.46	41.58	6.90	75.73	9.47
1kB	28.70	0	30.56	0	88.54	6.59	40.00	4.67	41.76	6.23	75.88	9.40
100kB	28.78	1.40	30.96	0.07	88.46	6.69	39.73	4.12	41.82	4.61	77.44	9.09
500kB	28.93	4.89	32.29	0.40	91.26	4.73	40.02	3.77	41.82	4.87	77.29	9.17
1MB	29.25	9.70	33.95	0.87	91.17	4.18	41.29	4.70	42.39	5.02	82.01	8.05

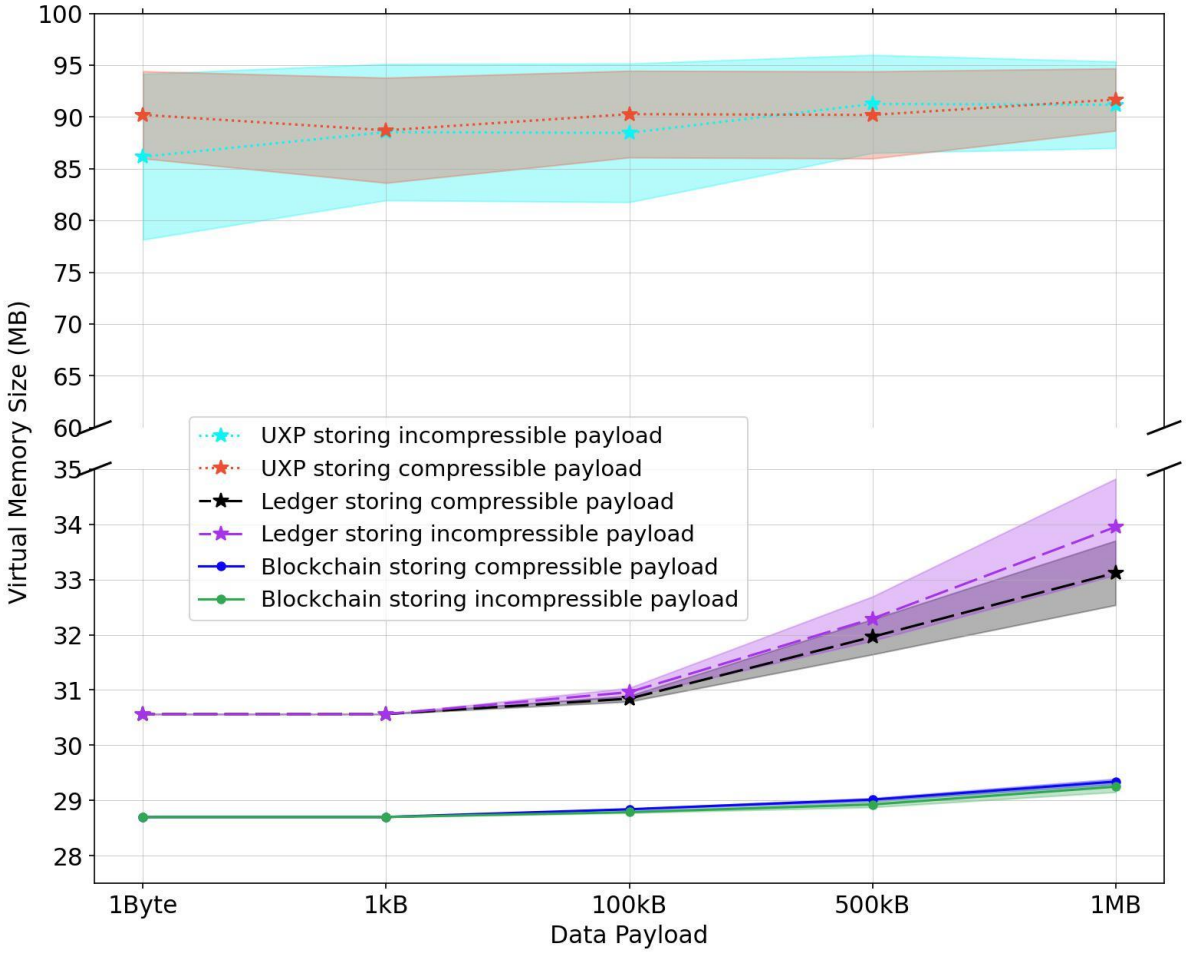


Figure 24. VSZ memory comparison for blockchain, ledger, and ICTO on general purpose machine.

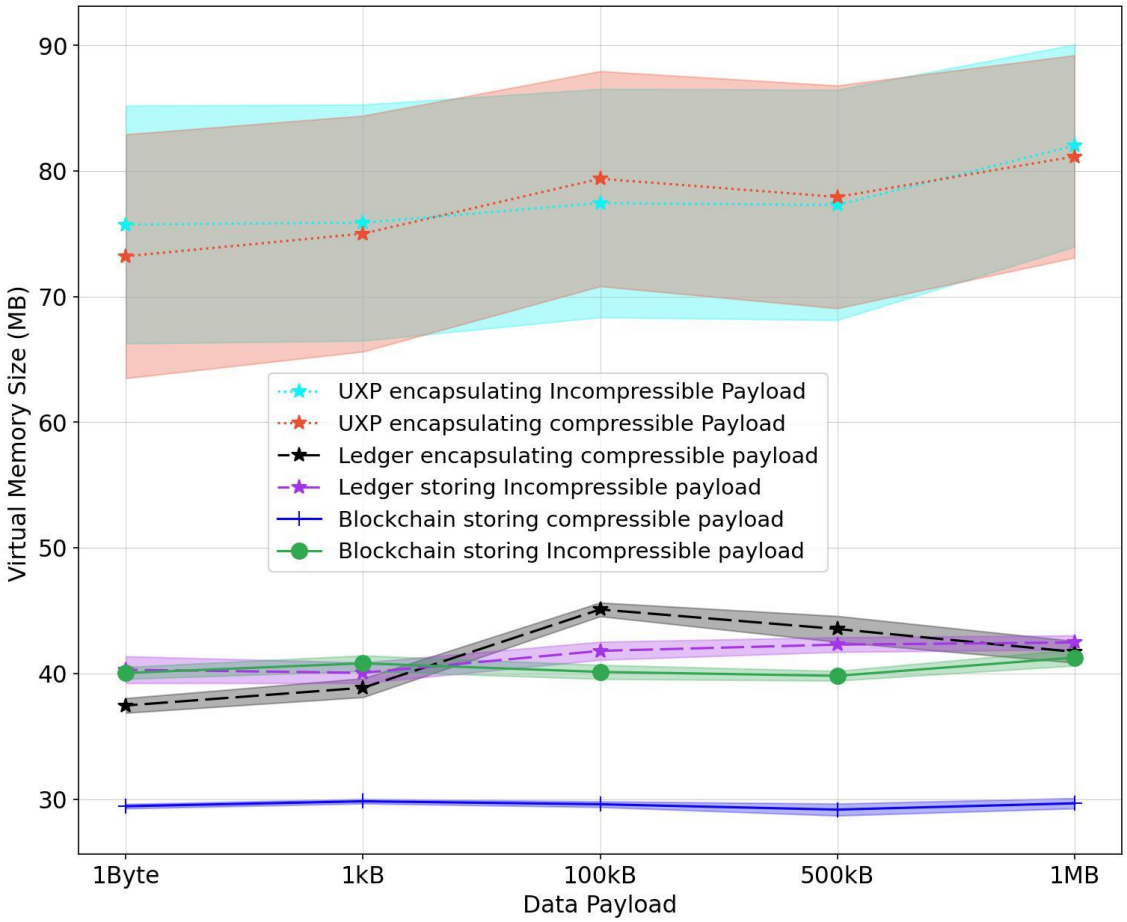


Figure 25. VSZ memory requirement of UXP, blockchain, and ledger on ubuntu virtual box.

4.7. Network Analysis

Network overhead is a crucial parameter to consider, particularly in IoT applications, as every transferred byte could contribute to the overall cost. This is especially significant for resource constrained devices, where battery power is limited and it's important to efficiently use the battery power. Furthermore, latency is another important factor, particularly for real time IoT applications where timely delivery of information is of utmost priority.

One of the major advantages of ICTO over other technologies including blockchain is that the payload encapsulated by the ICTO technology is inherently protected using multiple encryptions that are not dependent just on encryption keys. An experiment was performed to identify the performance of blockchain and ICTO over conventional Transmission Control

Protocol (TCP) sockets. The general-purpose Linux machine was used as TCP client whereas another identical machine acted as the server. The data was gathered on the client side. The payload on blockchain was compressed with LZ77 (Lempel-Ziv) compression, encrypted with ECC encryption, and transported through TCP socket. The UXP object was created using compressible payload and transported through TCP socket. For a baseline comparison, naked payload was compressed with LZ77 compression and transported though socket. Table 16 depicts the size of the naked payload after compression, blocks of blockchain, and UXP object.

Table 16. Size comparison.

Payload size (x)	Naked compressed size (kB)	Block size (kB)	UXP object size (kB)
1Byte	0.03	0.79	25.30
1kB	0.80	2.34	26.07
100kB	75.17	151.07	100.49
500kB	375.67	752.10	401.32
1MB	751.33	1503.42	777.33

Table 17 depicts the TCP overhead size incurred during transport of payload from client to server. Overhead for Naked payload and UXP was found to be similar for higher payloads. The overhead for blockchain was found to be significantly higher than for naked payload and UXP for higher payloads. This can be attributed to the higher size of the blocks than naked payload and UXP Object. But the inherent encryption overhead size of the block in the blockchain increases with increase in payload size. However, the inherent overhead of UXP object is constant around mean value of 25.20 kB.

Table 17. TCP Header length measurements.

Payload	Naked Payload		Blockchain		UXP	
	Mean (kB)	± CI	Mean (kB)	± CI	Mean (kB)	± CI
1Byte	0.280	0.004	0.325	0.013	1.049	0.012
1kB	0.284	0.005	0.389	0.015	1.085	0.009
100kB	2.490	0.026	5.199	0.059	3.351	0.043

500kB	11.302	0.179	23.295	0.310	12.750	0.181
1MB	22.071	0.297	45.055	0.547	22.504	0.227

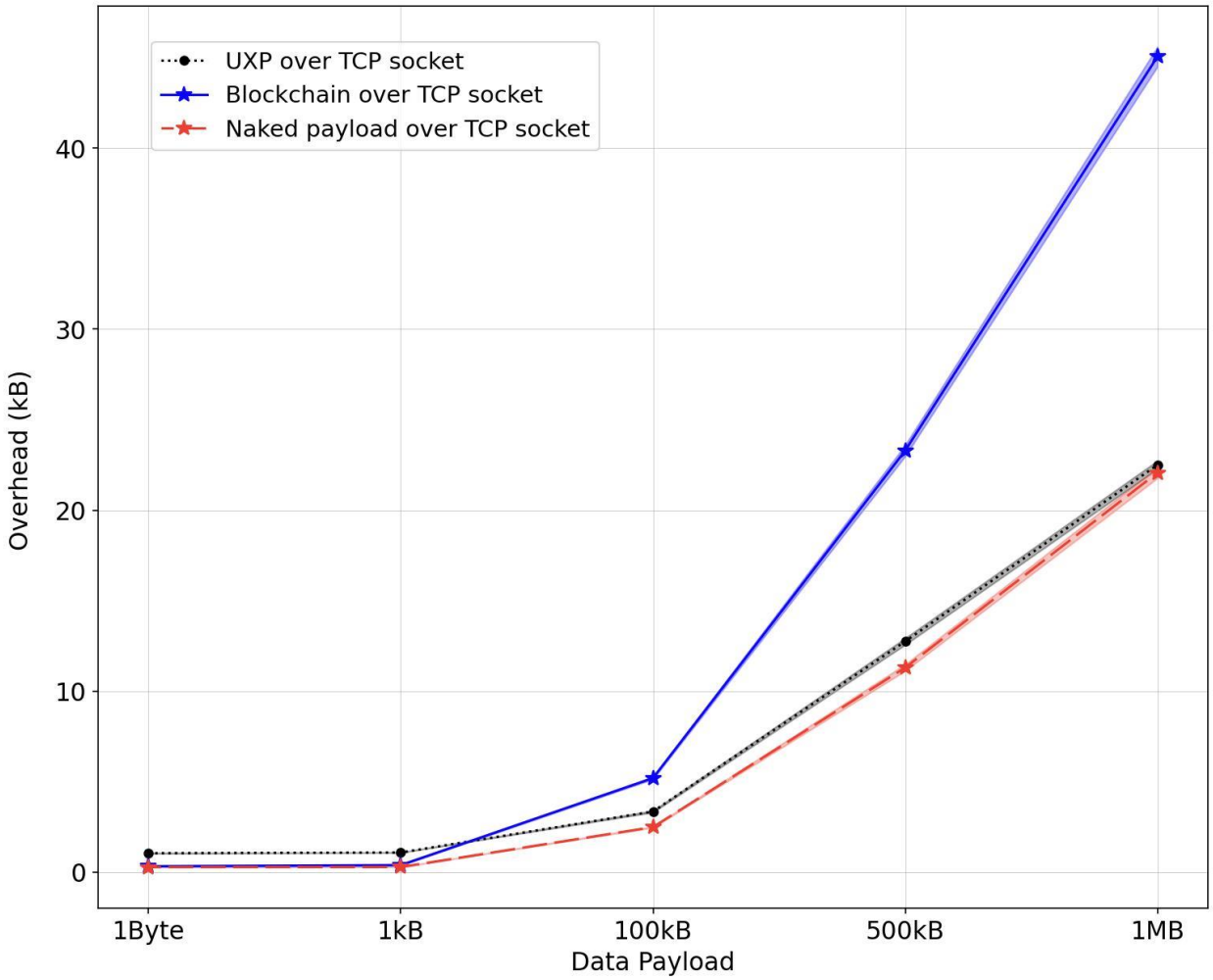


Figure 26. Cumulative Header Size vs Payload Size

Table 18 depicts the total time required to transport the data from client to server. Figure 27 illustrates this in a plot and clearly shows that the total time required to transport UXP was found to be the most efficient. The time required for transporting blocks was always found to be higher than UXP object with different margins for different payload. It was found that as the size of the payload increases, the time required for transporting blocks was as much as five times higher than time required for transporting UXP. The time required for transporting payload

compressed with just LZ77 without any security mechanism was found to be significantly similar to time required to transport UXP.

Table 18. TCP latency measurements.

Payload	Naked Payload		Blockchain		UXP	
	Mean (ms)	± CI	Mean (ms)	± CI	Mean (ms)	± CI
1Byte	0.76	0.43	0.92	0.08	0.53	0.05
1kB	0.58	0.06	0.12	0.21	0.58	0.05
100kB	1.03	0.10	2.03	0.27	0.81	0.07
500kB	1.35	0.13	4.10	0.35	1.29	0.10
1MB	1.60	1.45	7.51	0.84	1.38	0.11

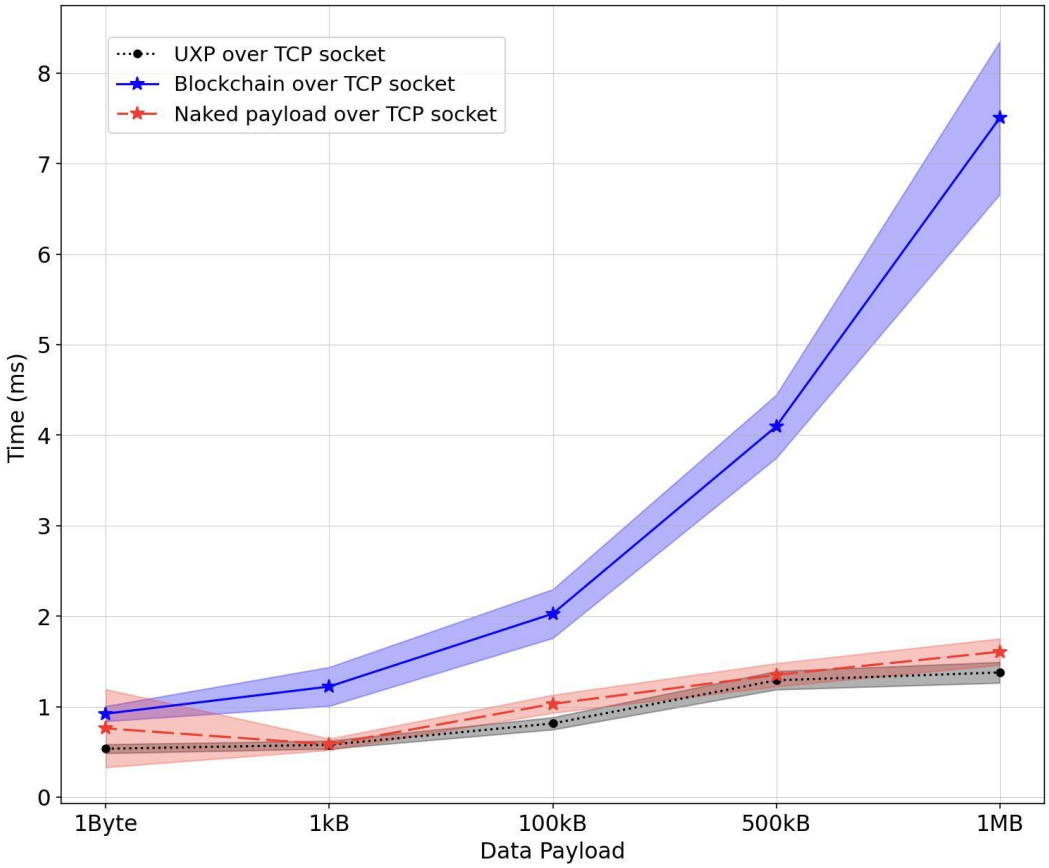


Figure 27. Network Latency vs Payload Size

4.8. Memory Footprint

An experiment was performed to determine the memory footprint of encapsulating payload with UXP and encrypting blocks of blockchain with Ascon. As described in Section 2.2.1, Ascon is an algorithm designed for lightweight usage and easy implementation with minimal overhead [49]. Ascon was implemented to encrypt the individual blocks of blockchain. The resulting block was compared with the UXP object based on its memory footprint. The memory requirement for blockchain and ledger is identical, so only one iteration of experiment was done. “Ascon-128” variant was used for the experiment with key size of 16 bytes [49]. The ASCON implementation was performed in python programming language. C language was considered for ASCON implementation due to the general notion that C is more efficient than python. But execution time of C implementation of ASCON was found to be more than python implementation of ASCON. This could be explained by the optimal python implementation of various mathematical operations which could be more efficient than reference implementation of ASCON in C. So, python implementation of ASCON was used for the comparison [84].

Table 19. Memory footprint comparison for ASCON, ECC, and UXP

Payload	Compressible Payload			Incompressible Payload		
	Ascon encrypted block (kB)	ECC encrypted block (kB)	UXP object (kB)	Ascon encrypted block (kB)	ECC encrypted block (kB)	UXP object (kB)
1Byte	0.32	0.79	25.30	0.29	0.73	25.33
1kB	1.10	2.34	26.07	1.29	2.74	26.34
100kB	75.45	151.07	100.49	100.29	200.77	125.37
500kB	375.97	752.10	401.32	500.29	1000.77	525.49
1MB	752.07	1503.42	777.33	1000.29	2000.77	1025.65

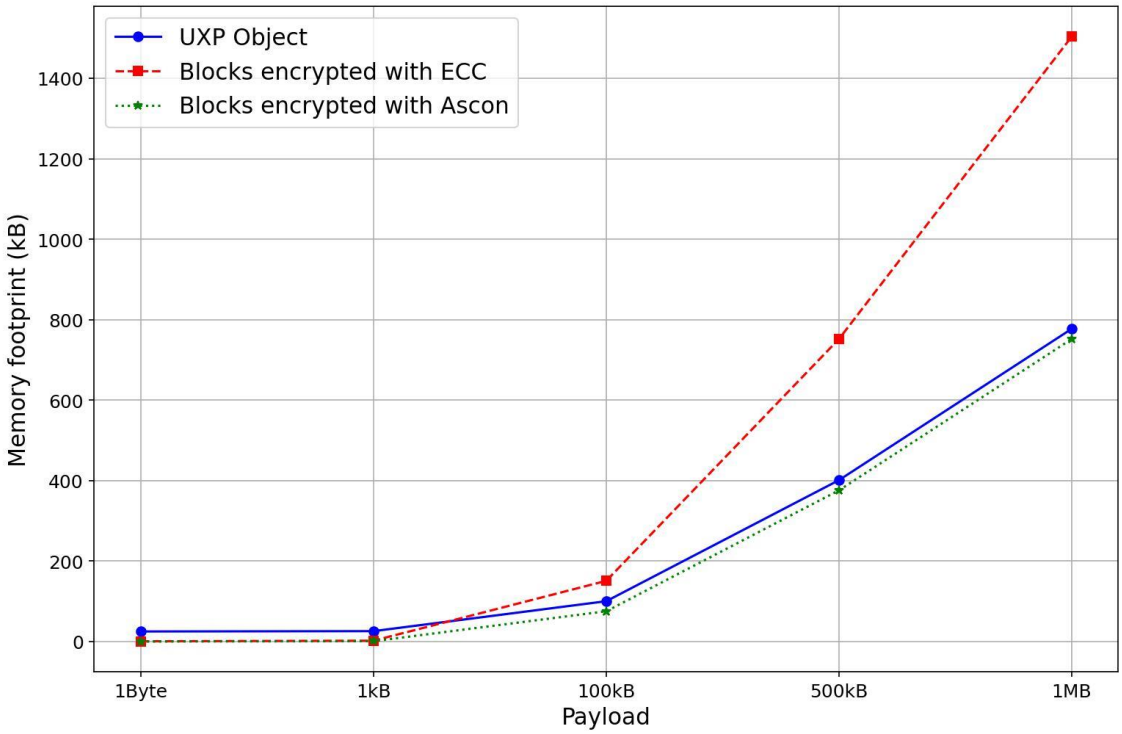


Figure 28. Memory footprint comparison for compressible payloads

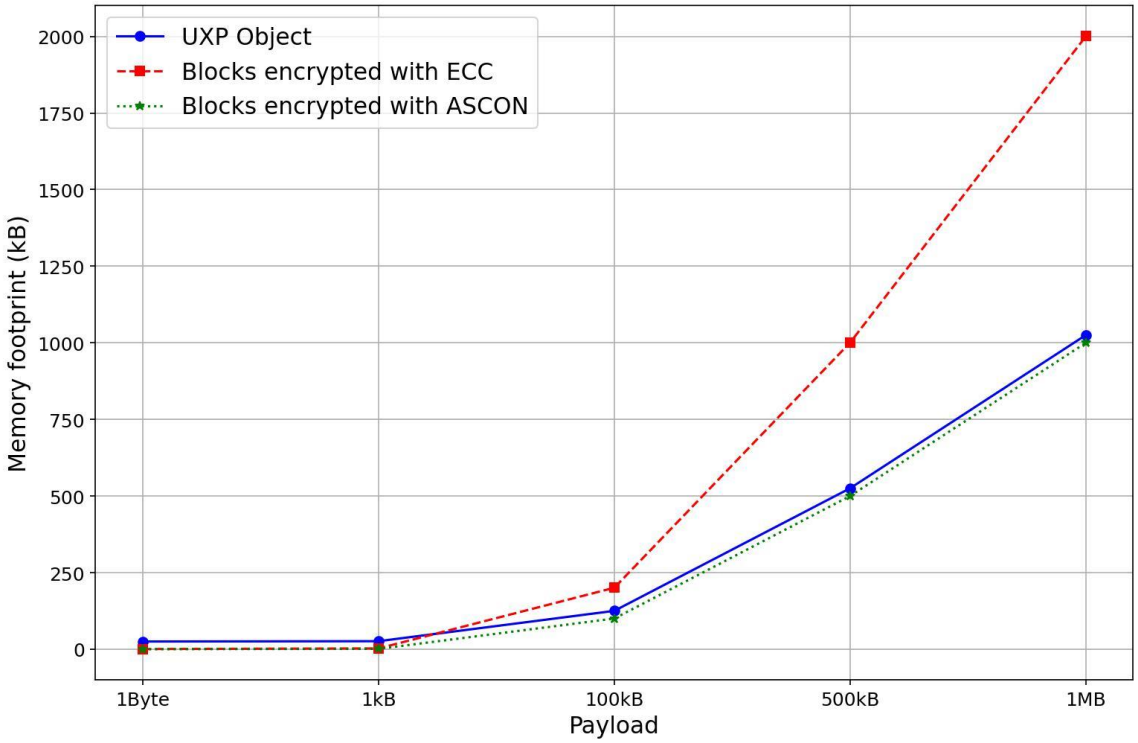


Figure 29. Memory footprint comparison for incompressible payloads

Figure 28 and Figure 29 illustrate the memory size comparison of UXP encapsulation, and blocks of blockchain or ledger encrypted by ECC and Ascon algorithms for compressible and incompressible payloads respectively. It was found that the memory footprint of UXP and Ascon encrypted blocks were almost identical for larger payloads. However, the memory footprint of blocks encrypted by ECC encryption increased with increase in payload size as depicted in Figure 28 and Figure 29. Ascon is known for its low memory footprint, and for UXP to be in par with Ascon encryption is very impressive, provided that UXP performs multiple encryptions as well as treats data as an intelligent entity.

5. CONCLUSIONS AND FUTURE WORK

This research study was designed to explore specific questions targeting technologies for IoT applications. Three different technologies: blockchain technology with central server, blockchain with proof of work implementation, and an up-and-coming ICTO technology with inherent data protection mechanisms were explored for IoT applications. Multiple experiments were designed to compare these technologies based on their viability for IoT applications. The experiments were performed on two different datasets; the first dataset contained payload which was incompressible by traditional lossless compression and the second dataset contained payload which was compressed using lossless compression for both versions of blockchain and was inherently compressed by ICTO technology.

One of the implementations of ICTO technology is a UXP object, which is a complete package incorporating built in security and privacy features with user-defined rules. On the other hand, blockchain technology lacks inherent built-in security features and relies on external encryption mechanisms for data protection. This characteristic of blockchain technology can serve as both a benefit and a weakness. The benefit lies in the flexibility offered by blockchain technology, as it can be tailored to align with specific requirements of different applications. For instance, IoT applications that demand maximum protection could utilize blockchain technology with strong and robust encryptions, while applications that focus on real-time data could opt for blockchain technology with encryptions with less overhead and faster encryption times. However, the lack of inherent security in blockchain technology implies that the technology itself is not sufficient for secure operation in IoT applications. External encryption mechanisms need to be implemented alongside blockchain to guarantee the desired level of data security.

5.1. Summary

The experiments were designed to compare the capability of core blockchain technology, blockchain technology with proof of work as a consensus mechanism to represent technology used for distributed ledger, and ICTO technology for IoT applications. UXP, which is an implementation of ICTO technology was used for the experiments. The results showed the benefits and the disadvantages of all technologies when viewed from the aspect of different IoT application requirements.

The first experiment in Section 4.1 involved comparison of blockchain and UXP relative to constrained device's storage space point of view. This experiment provides an approximation for size of the block and UXP object encapsulating compressible and incompressible payloads of specific size which can be used to determine the memory requirement of blockchain or UXP when storing payloads of different sizes. The experiment further demonstrates a consistent difference in size between blocks and UXP objects for both compressible and incompressible payloads, with an average value of 25.2 kB for compressible and 25.15 kB for incompressible payload. The compressible payloads experiment also illustrated nearly the same difference in block size and payload size as with incompressible payloads. This finding confirms that the compression technique used in UXP is also a simple lossless LZ77 compression. An important point to note is that the payload in the blockchain for this experiment was not encrypted. Hence, the overhead for the blockchain is the base overhead value. However, it is important to consider that various encryption methods can be used, and additional overhead could be introduced by the chosen encryption mechanism. This overhead for the specific payload size could be added to the base overhead value to determine the total overhead. This shows the flexibility of the blockchain over ICTO. The encryption in UXP is pre-determined whereas for blockchain there are different

encryption mechanisms available that can be tailored to suit specific application requirements.

Experiment 2 in Section 4.2 showed that the ICTO technology offers a notable advantage of maintaining a constant overhead for incompressible payloads. This characteristic of ICTO holds a significant value for resource constrained IoT devices generating substantial amounts of data. All the payloads could be bundled together and encapsulated by ICTO to generate a UXP object with a constant limited overhead. This property of UXP object highlights the scalability potential of UXP which is an important issue in IoT applications.

An experiment was performed to address the effect of blockchain and ICTO on device's memory assuming that the device has a '1MB' storage which is common for IoT endpoint devices. The number of blocks that can be stored in an IoT endpoint is hindered by the memory capacity of the IoT endpoint. This limitation arises from the cumulative nature of blockchain, as each block is linked to the previous blocks. This is a benefit of blockchain, but it also compounds storage issues. In contrast, UXP objects aren't linked with each other and do not have a cumulative nature like blockchain. So, the UXP object can be transported as soon as they are created freeing up the memory in resource constrained IoT endpoints. These results indicate that it is more efficient to store blocks of blockchain in a memory constrained device than UXP object. This is primarily because of the overhead of UXP object needed for data protection. However, it is worth noting that when transporting the data stored in blockchain, additional data protection mechanisms must be employed before transport. In contrast, UXP objects are already protected during their creation, eliminating the need for external protection mechanisms during transport. Furthermore, the UXP object can be transported via an unprotected channel as soon as created. However, at least the latest block in a chain must be retained in the IoT device to maintain the integrity of the blocks. To store a greater number of blocks in the IoT endpoint,

there are two possible alternatives. The first option is to increase the memory capacity of the IoT device. creating additional costs and power consumption. The second alternative is to reset the blockchain frequently, which requires transporting the previous blocks to a separate storage facility. Therefore, the choice of the technology depends on factors such as the usable memory of the IoT device, the nature of the channel (protected or unprotected), and the required data security (inherent security with user defined rules through UXP or different flexible external encryptions for blockchain blocks).

Comparison of Execution Time for the technologies was carried out as described in Section 4.4. The experiments conducted indicate that the core blockchain technology exhibits superior time efficiency compared to UXP technology. The payload encapsulation time required for UXP is more than seven times higher than storing payload in blockchain and more than three times higher than storing payload in the ledger. However, although the blockchain architectures may execute faster, they do not provide the same level of data security as suggested by UXP/ICTO approach. As discussed in Section 2.5, ICTO provides multilevel encryption via AES 256 as well as proprietary encryption in a nested fashion. However, a singular ECC encryption was used to encrypt the blocks of blockchain and ledger. The substantial time difference highlights the need for ICTO technology to address the time requirements associated with creating UXP objects as prolonged operational time in IoT devices translates to increased battery consumption, which is a concern for resource constrained devices running on limited battery power. Similar experiments conducted on a Raspberry Pi 400 platform also suggest that the core blockchain technology exhibits less execution time than ledger. This disparity in execution time between the blockchain and ledger can be attributed to the computational requirements of PoW, which requires calculating a hash with specific difficulty level. This task poses a challenge for

IoT devices with limited resources and computing power.

The measurement of CPU clock cycle during the operation of technologies showed that the core blockchain technology exhibits superior CPU efficiency compared to ledger and UXP technology. UXP utilizes more than two times the CPU clock cycles for encapsulating payloads compared to storing payload in blockchain. To address this issue, UXP should focus on developing a lightweight version of the technology, specifically tailored for IoT devices, aiming to reduce CPU usage and optimize overall efficiency. However, it was also found that the CPU clock cycles requirement for ledger and UXP was nearly identical as shown in Figure 19. This shows that the UXP and ledger require almost equal amount of time to execute clock cycle on the CPU. Both UXP and ledger implementing PoW have similar processing demands and will execute at comparable speed. Ledger technology is widely used in IoT and usually the difficulty level is not fixed as in the case of this research experiment. So, the clock cycle requirement for UXP being comparable to ledger shows a great potential for UXP in IoT applications. This experiment also shows the expensive and complex nature of the proof of work. This experiment is also crucial to indicate that UXP has a huge potential to replace the use of distributed ledger especially for IoT applications with limited resources. Increasing the difficulty level of the ledger and implementing multiple encryption mechanisms will only further add to the resource consumption. However, the UXP implementation of ICTO is a complete data protection scheme that doesn't rely on any additional encryption mechanisms as it is already equipped with multiple nested encryptions. So, this experiment shows immense potential of UXP implementation in resource constrained IoT applications.

Experiments on memory consumption as depicted in Section 4.6 showed an interesting observation regarding UXP technology. UXP technology needed minimum amount of RAM

memory but maximum allocation of virtual memory among the three technologies. This indicates that UXP has a higher overall virtual memory footprint but lower physical memory (RAM) consumption. The UXP technology utilizes less memory in RAM but requires a large amount of virtual memory for its operations. This shows that UXP should focus on effective memory management techniques to reduce the reliance on virtual memory. Furthermore, this also shows the potential of UXP in IoT devices with limited physical memory available given that the virtual memory footprint issue is solved. This outcome highlights the importance of comprehending the memory usage of various technologies when selecting the appropriate technology for resource constrained IoT devices.

Network experiments also showed promising results for UXP technology as depicted in Section 4.7. The experiments revealed that both the overhead value and network latency were nearly identical during the transport of UXP and naked payload with lossless compression. This finding demonstrates the highly network efficient mechanism of UXP and highlights the viability of UXP for resource constrained environments. However, both the overhead and latency for blockchain was found to be higher than UXP by a significant margin which increased with increase in payload size. Furthermore, the network performance for distributed ledgers has one major difference with blockchain. Ledger requires broadcasting of the blocks to every node in the network whereas blockchain only requires unicasting of the block to the central server. This implies that the devices are broadcasting and receiving multiple blocks of blockchain. This could significantly increase the burden on the IoT device. Additionally, ICTO doesn't need the channel to be protected and can be transported through even a basic TCP socket without any security compromise. However, blockchain blocks should not be transported through a bare TCP socket without additional protection mechanism. It is still a risk to transport payloads even after

protection using PKI and strong encryptions as these encryption techniques have their own issues as discussed in Section 2.2.1. So, usage of other transport protocols like Message Queuing Telemetry Transport (MQTT), and Constrained Application Protocol (CoAP) is common for lightweight IoT applications. However, these applications also come with severe overhead and delay. An interesting experiment was performed to understand the comparison of TCP and MQTT [82]. The results from the experiments indicate that MQTT has a transmission delay of at least a factor of 2 and an overhead inefficiency of an order of magnitude compared to a TCP socket [82]. The overhead increase and delay in MQTT come at the cost of message queuing ability, increased security over TCP socket, and different quality of service. However, based on this experiment and our thesis result, we can infer that even when comparing UXP and blockchain over MQTT instead of TCP, UXP still turns out to be more efficient in terms of overhead and latency.

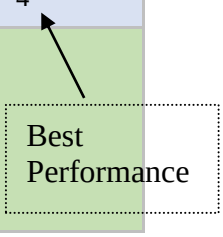
Similarly, comparison of size of UXP object and blockchain block encrypted by ASCON showed more promising results for UXP. ASCON is designed to be lightweight especially in terms of memory footprint. And the results showed that encapsulating the same payload by UXP and storing ASCON encrypted payload in blockchain is nearly identical.

In summary, core blockchain is ideal for resource constrained devices if it is integrated with encryption mechanisms that are designed for lightweight encryptions and provide smaller overhead. UXP on the other hand is a complete package that focuses on data security with enhanced protection schemes like user defined ruleset. It is a ready-to-use technology with inherent security. It is extremely useful in cases where security and scalability are vital. Ledger technology is useful to counteract the “single point of failure”, but the complexity of consensus mechanisms could hinder it from directly implementing on end devices. The results of the

experiment and the properties of the ICTO technology highlight the significant potential implementation of ICTO in various applications, extending beyond the realm of IoT. The results indicate the immense scalability potential of ICTO technology, because of its constant overhead regardless of the payload size. Scalability is one of the core issues in IoT applications and using technology like blockchain invites such scalability issues because of its cumulative nature. But ICTO creates an independent UXP object for a given payload and adds different layers of security in compromise of certain overhead. The results also show that UXP exhibits resource inefficiency compared to blockchain technology. However, this inefficiency could be attributed to the comprehensive data security scheme of UXP. The results also showed that the UXP technology is better in terms of RAM consumption and CPU clock cycle requirements than blockchain with PoW implementation. This suggests that UXP has the potential to replace the complex distributed ledger technologies employed in IoT applications. The overall conclusion from the results of the experiments and the literature survey is depicted in Table 20.

Table 20. Comparisons of blockchain, ledger, and UXP

CRITERIA	Blockchain	Distributed Ledger	UXP
Complexity	5[a]	2[b]	3[b]
Single Point of Failure	1[b]	5[a]	1[b]
Scalability	3[a]	3[b]	4[a]
RAM Efficiency	4[a]	2[b]	5[a]
Cost	4[a]	2[b]	2[b]
CPU Efficiency	5[a]	3[b]	3[b]
Validation	3[c]	4[d]	5[e]
Trust	2[c]	4[d]	5[e]
Transparency	2[c]	3[d]	5[e]
Data Security	2[f]	3[f]	5[e]
Vulnerability	2[c]	2[d]	5[e]
Average	3	3	4
Scale and Legend	1 - Very poor, 2- Poor, 3 - Fair, 4 - Good, 5- Very Good a = Simple, eliminates, efficient, easy, scalable, low cost b = Complex, cannot eliminate, inefficient, difficult, high cost c = Central authority d = Trustless, multiple nodes, consensus mechanism e = User access control, multiple nested encryptions, owner ruleset f = No inherent security, depends on external security mechanisms		



5.2. UXP technology for IoT

UXP technology transforms payload to a UXP object which is an intelligent and proactive entity. The payload is protected through user defined rules alongside inbuilt encryptions and ruleset. The keys used in the UXP protection scheme are cloaked, inaccessible, and never shared. This makes UXP a good candidate for IoT applications. It is especially attractive for IoT applications requiring an all-round data protection mechanism. However, the downside of the technology is it is resource intensive as per the experimental results. If the technology is compressed to fit for a resource constrained device, UXP technology could prove to be a promising technology for IoT applications even in resource-constrained applications.

5.3. Future Works

For this study, the comparison between the ICTO and blockchain technology was made to highlight their viability for IoT applications and to understand their benefits and shortcomings for use in IoT applications. To validate and strengthen the conclusions of this thesis, additional experiments are necessary especially by running a complete version of these technologies in a resource constrained IoT device which is not possible for ICTO at the moment because of its significant SDK size. The blockchain along with a distributed ledger comprising multiple nodes could be executed on IoT endpoints to measure the resource requirements and make comparisons. Additionally, an experiment to measure the real time power consumption could be conducted to truly understand the batter consumption by these technologies when operated in a resource constrained device. Additionally, Field-Programmable Gate Array (FPGA) implementation of the UXP could also prove to be extremely useful in understanding the complexity of the technology and its viability in lightweight applications. Furthermore, more optimized implementation of ASCON and ECC can be compared with UXP to truly understand the relative performance between these technologies. It could be extremely useful as ASCON is designed for lightweight applications and relative performance examination could show useful insights.

5.4. Open-Source Implementation

The proprietary ICTO technology which was designed for user defined rules and strong data protection scheme has undoubtedly shown valuable prospect in terms of safeguarding different types of data. Experimental analysis of the ICTO technology has suggested that it is worth considering the potential usefulness of adopting an open-source approach for ICTO technology. Open-source implementation of ICTO would no-doubt provide several notable

advantages. Firstly, it would foster a collaborative environment allowing for the boarder community to enhance this already promising technology. Inclusive development and expert contribution to technology will only make it better and increase its commercial usage for different applications not limited to simple data protection. Secondly, an open source promotes transparency and auditability, which will enable the researchers to find vulnerabilities and improve trust by overcoming those issues. Additionally, an open-source ICTO implementation will encourage interoperability and compatibility with different frameworks, increasing the usage of the technology. All these factors show the benefit of an open-source implementation of ICTO that could play a revolutionizing role in various markets that prioritize data security.

5.5. Recommendation for ICTO development

The innate properties of ICTO technology and the results of the experiments show the high potential of ICTO in the field of IoT. To reach this potential, ICTO could develop hardware implementation of the technology to better match it for IoT usage. The size of the SDK could also be significantly reduced to make it more practical for resource constrained devices. A lightweight version of the technology that focuses on efficiency without losing data security should be considered. Its innate data security with user defined rules makes it an exciting prospect for IoT applications and the potential is endless.

REFERENCES

[1] "The little-known story of the first IoT device," Industrious, Feb. 07, 2018. https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/ (accessed Feb. 23, 2023).

[2] "Kevin Ashton Invents the Term 'The Internet of Things' : History of Information." https://www.historyofinformation.com/detail.php?id=3411 (accessed Feb. 23, 2023).

[3] "What is the Internet of Things (IoT)?" https://www.oracle.com/internet-of-things/what-is-iot/ (accessed Feb. 07, 2023).

[4] 1 A Look at IIoT: The Perspective of IoT Technology Applied in the Industrial Field. Accessed: Feb. 07, 2023. [Online]. Available: https://learning.oreilly.com/library/view/industrial-internet-of/9781119768777/c01.xhtml

[5] J. Morgan, "A Simple Explanation Of 'The Internet Of Things,'" Forbes. https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/ (accessed Feb. 07, 2023).

[6] S. H. Shah and I. Yaqoob, "A survey: Internet of Things (IOT) technologies, applications and challenges," in 2016 IEEE Smart Energy Grid Engineering (SEGE), Aug. 2016, pp. 381–385. doi: 10.1109/SEGE.2016.7589556.

[7] "IoT connected devices worldwide 2019-2030," Statista. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed Feb. 08, 2023).

[8] S. Kim, G. Chandra Deka, and P. Zhang, Advances in Computers, First. [Online]. Available: https://learning.oreilly.com/library/view/role-of-blockchain/9780128171929/S0065245819300397.xhtml

[9] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," Computer, vol. 44, no. 9, pp. 51–58, Sep. 2011, doi: 10.1109/MC.2011.291.

[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Commun. Surv. Tutor., vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[11] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures," IEEE Commun. Surv. Tutor., vol. 22, no. 1, pp. 616–644, 2020, doi: 10.1109/COMST.2019.2953364.

[12] P. Varga, S. Plosz, G. Soos, and C. Hegedus, "Security threats and issues in automation IoT," in 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), May 2017, pp. 1–6. doi: 10.1109/WFCS.2017.7991968.

[13] “The Industrial Internet of Things - What’s the Difference Between IoT and IIoT?” https://www.leverege.com/blogpost/difference-between-iot-and-iiot (accessed Oct. 30, 2022).

[14] A. A. Qaysi, A. Oliva, and N. Holleman, “Ingram Project (EE Senior Design).”

[15] H. I. Ahmed, A. A. Nasr, S. Abdel-Mageid, and H. K. Aslan, “A survey of IoT security threats and defenses,” Int. J. Adv. Comput. Res., vol. 9, no. 45, pp. 325–350, Oct. 2019, doi: 10.19101/IJACR.2019.940088.

[16] “The story of cement manufacture,” Cembureau. https://lowcarboneconomy.cembureau.eu/the-story-of-cement-manufacture/ (accessed Feb. 24, 2023).

[17] Marcus, “Concrete – The Most Used Material in the World | Kilgore Companies | Construction Jobs,” Kilgore Companies | Enter Tagline Here, Nov. 09, 2021. https://www.kilgorecompanies.com/concrete-the-most-used-material-in-the-world/

[18] “Where is cement used?,” Cembureau. https://lowcarboneconomy.cembureau.eu/where-is-cement-used/ (accessed Feb. 24, 2023).

[19] O. US EPA, “Our Mission and What We Do,” Jan. 29, 2013. https://www.epa.gov/aboutepa/our-mission-and-what-we-do (accessed Apr. 27, 2023).

[20] “Homepage,” Texas Commission on Environmental Quality. https://www.tceq.texas.gov (accessed Apr. 27, 2023).

[21] “penaltypolicy2021.pdf.” Accessed: Apr. 27, 2023. [Online]. Available: https://www.tceq.texas.gov/assets/public/comm_exec/pubs/rg/rg253/penaltypolicy2021.pdf

[22] R. Ramakrishnan, D. K. Sasikala, D. A. Nagappan, and A. Profressor, “Challenges, Issuesof Energy Efficiency in IoT devices and an analysis of batterylife power consumption in IoT devices and Applications.,” vol. 7, no. 11, 2020.

[23] S. S. Panda, B. K. Mohanta, M. R. Dey, U. Satapathy, and D. Jena, “Distributed Ledger Technology for Securing IoT,” in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Jul. 2020, pp. 1–6. doi: 10.1109/ICCCNT49239.2020.9225333.

[24] T. Xu, J. B. Wendt, and M. Potkonjak, “Security of IoT systems: Design challenges and opportunities,” in 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Nov. 2014, pp. 417–423. doi: 10.1109/ICCAD.2014.7001385.

[25] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, “An Approach to QoS-based Task Distribution in Edge Computing Networks for IoT Applications,” in 2017 IEEE International Conference on Edge Computing (EDGE), Jun. 2017, pp. 32–39. doi: 10.1109/IEEE.EDGE.2017.50.

[26] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A Survey of Computation Offloading for Mobile Systems,” Mob. Netw. Appl., vol. 18, no. 1, pp. 129–140, Feb. 2013, doi: 10.1007/s11036-012-0368-0.

[27] “What Is Edge Computing? | Microsoft Azure.” https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-edge-computing/ (accessed Apr. 10, 2023).

[28] W. Yu et al., “A Survey on the Edge Computing for the Internet of Things,” IEEE Access, vol. 6, pp. 6900–6919, 2018, doi: 10.1109/ACCESS.2017.2778504.

[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[30] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, “Decentralized Applications: The Blockchain-Empowered Software System,” IEEE Access, vol. 6, pp. 53019–53033, 2018, doi: 10.1109/ACCESS.2018.2870644.

[31] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” Bus. Inf. Syst. Eng., vol. 59, Mar. 2017, doi: 10.1007/s12599-017-0467-3.

[32] H.-N. Dai, Z. Zheng, and Y. Zhang, “Blockchain for Internet of Things: A Survey,” IEEE Internet Things J., vol. 6, no. 5, pp. 8076–8094, Oct. 2019, doi: 10.1109/JIOT.2019.2920987.

[33] G. Wang, Z. Shi, M. Nixon, and S. Han, “ChainSplitter: Towards Blockchain-Based Industrial IoT Architecture for Supporting Hierarchical Storage,” in 2019 IEEE International Conference on Blockchain (Blockchain), Jul. 2019, pp. 166–175. doi: 10.1109/Blockchain.2019.00030.

[34] G. S. Smith, M. L. S. Weed, D. M. Fischer, and E. M. Ridenour, “System and methods for using cipher objects to protect data,” US11093623B2, Aug. 17, 2021 Accessed: Nov. 08, 2022. [Online]. Available: https://patents.google.com/patent/US11093623B2/en?q=ICTO&oq=ICTO

[35] “IDC Perspective: Is Data the New Endpoint? (IDC #AP43120017, October 2017).”

[36] “2022 SonicWall Cyber Threat Report.” [Online]. Available: https://www.sonicwall.com/medialibrary/en/white-paper/mid-year-2022-cyber-threat-report.pdf

[37] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, “IoT architecture challenges and issues: Lack of standardization,” in 2016 Future Technologies Conference (FTC), Dec. 2016, pp. 731–738. doi: 10.1109/FTC.2016.7821686.

[38] “8. Internet of Things — mPython board 2.2.2 documentation.” https://mpython.readthedocs.io/en/master/tutorials/advance/iot/index.html (accessed Apr. 29, 2023).

[39] “MQTT Version 5.0.” OASIS. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[40] “Home | AMQP.” https://www.amqp.org/ (accessed May 23, 2023).

[41] C. Bormann, A. P. Castellani, and Z. Shelby, “CoAP: An Application Protocol for Billions of Tiny Internet Nodes,” IEEE Internet Comput., vol. 16, no. 2, pp. 62–67, Mar. 2012, doi: 10.1109/MIC.2012.29.

[42] “Unified Architecture,” OPC Foundation. https://opcfoundation.org/about/opc-technologies/opc-ua/ (accessed May 23, 2023).

[43] “XMPP | An Overview of XMPP.” https://xmpp.org/about/technology-overview/ (accessed May 23, 2023).

[44] E. Bertino, “Data Security and Privacy in the IoT.” OpenProceedings.org, 2016. doi: 10.5441/002/EDBT.2016.02.

[45] L. B. BME CrySyS Lab, “9 Important Security Requirements to Consider for IoT Systems,” IoTAC, Feb. 25, 2021. https://iotac.eu/9-important-security-requirements-to-consider-for-iot-systems/ (accessed May 23, 2023).

[46] “IoT | Security and Privacy Paradigm | Souvik Pal, Vicente García Díaz,.” https://www-taylorfrancis-com.libproxy.txstate.edu/pdfviewer/

[47] I. T. L. Computer Security Division, “Lightweight Cryptography | CSRC | CSRC,” CSRC | NIST, Jan. 03, 2017. https://csrc.nist.gov/projects/lightweight-cryptography (accessed May 23, 2023).

[48] “NIST Selects ‘Lightweight Cryptography’ Algorithms to Protect Small Devices,” NIST, Feb. 2023, Accessed: May 23, 2023. [Online]. Available: https://www.nist.gov/news-events/news/2023/02/nist-selects-lightweight-cryptography-algorithms-protect-small-devices

[49] “ASCON.” Accessed: Apr. 25, 2023. [Online]. Available: https://ascon.iaik.tugraz.at/index.html

[50] A. Badr, Y. Zhang, and H. G. Umar, “Dual Authentication-Based Encryption with a Delegation System to Protect Medical Data in Cloud Computing,” Electronics, vol. 8, p. 171, Feb. 2019, doi: 10.3390/electronics8020171.

[51] Y. Chandu, K. S. R. Kumar, N. V. Prabhukhanolkar, A. N. Anish, and S. Rawal, “Design and implementation of hybrid encryption for security of IOT data,” in 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Aug. 2017, pp. 1228–1231. doi: 10.1109/SmartTechCon.2017.8358562.

[52] M. Mali, F. Novak, and A. Biasizzo, “HARDWARE IMPLEMENTATION OF AES ALGORITHM,” J. Electr. Eng., no. 9, 2005.

[53] O. G. Abood and S. K. Guirguis, “A Survey on Cryptography Algorithms,” Int. J. Sci. Res. Publ. IJSRP, vol. 8, no. 7, Jul. 2018, doi: 10.29322/IJSRP.8.7.2018.p7978.

[54] F. Mallouli, A. Hellal, N. Sharief Saeed, and F. Abdulraheem Alzahrani, “A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms,” in 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Jun. 2019, pp. 173–176. doi: 10.1109/CSCloud/EdgeCom.2019.00022.

[55] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, “Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions,” J. Ambient Intell. Humaniz. Comput., May 2017, doi: 10.1007/s12652-017-0494-4.

[56] D. Hankerson and A. Menezes, “Elliptic Curve Discrete Logarithm Problem,” in Encyclopedia of Cryptography and Security, H. C. A. van Tilborg and S. Jajodia, Eds., Boston, MA: Springer US, 2011, pp. 397–400. doi: 10.1007/978-1-4419-5906-5_246.

[57] M. Suárez-Albela, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, “A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices,” in 2018 Global Internet of Things Summit (GIoTS), Jun. 2018, pp. 1–6. doi: 10.1109/GIOTS.2018.8534575.

[58] A. V and S. R. Kalli, “A survey on security of mobile handheld devices through elliptic curve cryptography,” ACCENTS Trans. Inf. Secur., vol. 2, pp. 32–35, Dec. 2016, doi: 10.19101/TIS.2017.26001.

[59] J. R. Wallrabenstein, “Practical and Secure IoT Device Authentication Using Physical Unclonable Functions,” in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Aug. 2016, pp. 99–106. doi: 10.1109/FiCloud.2016.22.

[60] M. Samaniego and R. Deters, “Zero-Trust Hierarchical Management in IoT,” in 2018 IEEE International Congress on Internet of Things (ICIOT), Jul. 2018, pp. 88–95. doi: 10.1109/ICIOT.2018.00019.

[61] B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini, and G. Anastasi, “On the Feasibility of Attribute-Based Encryption on Constrained IoT Devices for Smart Systems,” in 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Jun. 2019, pp. 225–232. doi: 10.1109/SMARTCOMP.2019.00057.

[62] “What is Blockchain Technology - IBM Blockchain | IBM.” https://www.ibm.com/topics/what-is-blockchain (accessed Feb. 08, 2023).

[63] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”.

[64] M. Samaniego, U. Jamsrandorj, and R. Deters, “Blockchain as a Service for IoT,” in 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Dec. 2016, pp. 433–436. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102.

[65] P. Arul and S. Renuka, “Blockchain technology using consensus mechanism for IoT- based e-healthcare system,” IOP Conf. Series: Materials Science and Engineering, Feb. 2021. doi: 10.1088/1757-899X/1055/1/012106.

[66] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, “Blockchain for Large-Scale Internet of Things Data Storage and Protection,” IEEE Trans. Serv. Comput., vol. 12, no. 5, pp. 762–771, Sep. 2019, doi: 10.1109/TSC.2018.2853167.

[67] M. Kashoek and D. Karger, “Koorde: A simple degree-optimal distributed hash table”.

[68] K. R. Özyılmaz and A. Yurdakul, “Work-in-progress: integrating low-power IoT devices to a blockchain-based infrastructure,” in 2017 International Conference on Embedded Software (EMSOFT), Oct. 2017, pp. 1–2. doi: 10.1145/3125503.3125628.

[69] P. Chen, X. Zhang, H. Liu, L. Xiang, and P. Shi, “Ladder: A Blockchain Model of Low-Overhead Storage,” in Blockchain and Trustworthy Systems, H.-N. Dai, X. Liu, D. X. Luo, J. Xiao, and X. Chen, Eds., in Communications in Computer and Information Science. Singapore: Springer, 2021, pp. 115–129. doi: 10.1007/978-981-16-7993-3_9.

[70] T. Kim, S. Lee, Y. Kwon, J. Noh, S. Kim, and S. Cho, “SELCOM: Selective Compression Scheme for Lightweight Nodes in Blockchain System,” IEEE Access, vol. 8, pp. 225613–225626, 2020, doi: 10.1109/ACCESS.2020.3044991.

[71] R. A. Memon, J. P. Li, J. Ahmed, M. I. Nazeer, M. Ismail, and K. Ali, “Cloud-based vs. blockchain-based IoT: a comparative survey and way forward,” Front. Inf. Technol. Electron. Eng., vol. 21, no. 4, pp. 563–586, Apr. 2020, doi: 10.1631/FITEE.1800343.

[72] B. W. Nyamtiga, J. C. S. Sicato, S. Rathore, Y. Sung, and J. H. Park, “Blockchain-Based Secure Storage Management with Edge Computing for IoT,” Electronics, vol. 8, no. 8, Art. no. 8, Aug. 2019, doi: 10.3390/electronics8080828.

[73] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, “A survey on the adoption of blockchain in IoT: challenges and solutions,” Blockchain Res. Appl., vol. 2, no. 2, p. 100006, Jun. 2021, doi: 10.1016/j.bcra.2021.100006.

[74] Y. He, D. Huang, L. Chen, Y. Ni, and X. Ma, “A Survey on Zero Trust Architecture: Challenges and Future Trends,” Wirel. Commun. Mob. Comput., vol. 2022, p. e6476274, Jun. 2022, doi: 10.1155/2022/6476274.

[75] “1-UXP-Intro.pdf.” Accessed: Jun. 12, 2023. [Online]. Available: https://www.sertainty.com/wp-content/uploads/2021/04/1-UXP-Intro.pdf

[76] “Sertainty_Technology_Brief_v3.pdf.” Accessed: May 14, 2023. [Online]. Available: https://www.sertainty.com/wp-content/uploads/2021/05/Sertainty_Technology_Brief_v3.pdf

[77] “Sertainty-UXP-Technical-Overview.pdf.” Accessed: Jun. 10, 2023. [Online]. Available: https://www.sertainty.com/wp-content/uploads/2021/04/Sertainty-UXP-Technical-Overview.pdf

[78] “UXP Identity Construction Summary,” Sertainty. https://www.sertainty.com/developers/uxp-identity-summary/uxp-identity-construction-summary/ (accessed Jun. 10, 2023).

[79] “Raspberry Pi 400 Personal Computer Kit,” Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-400/ (accessed Jun. 10, 2023).

[80] alexmgr, “tinyec.” May 13, 2023. Accessed: May 20, 2023. [Online]. Available: https://github.com/alexmgr/tinyec

[81] “Bash - GNU Project - Free Software Foundation.” https://www.gnu.org/software/bash/ (accessed Jun. 10, 2023).

[82] “scipy.stats.t — SciPy v1.10.1 Manual.” https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.t.html (accessed Jun. 10, 2023).

[83] Embedded Linux for Developers. Accessed: May 23, 2023. [Online]. Available: https://learning.oreilly.com/library/view/embedded-linux-for/9781787124202/

[84] “Ascon – Implementations.” https://ascon.iaik.tugraz.at/implementations.html (accessed Jun. 24, 2023).