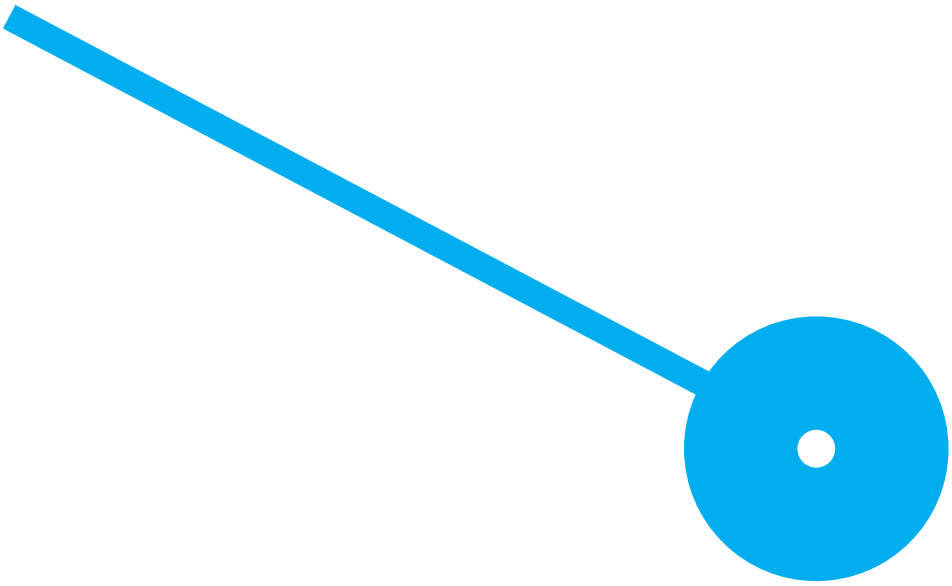
SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

MASTER
INFORMATICS ENGINEERING

Web3 Ecosystem for Device as a Service

Maria da Conceição Pereira Tavares

JULY/2025



SCHOOL OF
MANAGEMENT AND
TECNOLOGY
POLYTECHNIC OF
PORTO

M MASTER
INFORMATICS ENGINEERING

Web3 Ecosystem for Device as a Service

Maria da Conceição Pereira Tavares

8190709

Advisor(s)

PhD António Alberto dos Santos Pinto

Dissertation submitted in fulfilment of the requirements for the Master's degree in Informatics Engineering in the School of Management and Technology of the Polytechnic of Porto.

JULY/2025

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo.
(Success is born from the desire, determination, and persistence to reach a goal.)

José de Alencar

Le succès est une conséquence, et non un but.
(Success is a consequence, not an objective.)

Gustave Flaubert

Integrity Statement

I, **Maria da Conceição Pereira Tavares**, student no. **8190709**, of the Master's Degree in Informatics Engineering at the School of Management and Technology of the Polytechnic of Porto, declare that I have not plagiarised or self-plagiarised; therefore, the work titled "**Web3 Ecosystem for Device as a Service**" is original and of my own authorship, not having been used previously for any other purpose. I further declare that all sources used are cited in the text and in the final bibliography, according to the referencing rules adopted in the institution.

Acknowledgments

More than just the final stage of an academic journey, this dissertation is the culmination of a challenging, lonely, and emotionally transforming path. It evolved with determination, innumerable hours of work, persistent doubts, and the conviction that each effort leaves its mark.

I would like to sincerely thank my advisor, Professor António Alberto dos Santos Pinto, for his coaching and continuous encouragement during the development of this dissertation. His expertise and knowledge were important for the research process and for my academic and personal growth during this time.

Thank you to the School of Management and Tecnology - Polytechnic of Porto (ESTG), my academic home for the past few years, for embracing me and for providing an environment of support in which I could learn and prosper. I forever carry the imprint of this institution, that contributed to mould me into the professional I am today.

To Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), for admitting me as a research fellow and for providing an atmosphere in which this project could succeed, especially within the scope of BLOCKCHAIN.PT project. It was a special opportunity that helped me grow professionally and as a researcher.

I also offer my heartfelt appreciation to the scientific community. My paper, "Blockchain-Assisted Device as a Service (DaaS)" was published in the Proceedings of the 6th International Congress on Blockchain and Applications (Springer, 2025) and achieved the Best Paper Award. The recognition of my work by experts in the field, its formal presentation, and its inclusion in a scientific journal was a moment of honour and personal fulfilment.

A very special thanks goes to my dearest friend, David Marques, whose presence was invaluable. He provided me with intellectual support, constant encouragement, and practical help throughout this journey, especially in my times of doubt and stress. His kindness and trust in me frequently gave me the power to continue. I am truly grateful for all we have shared and will continue to share as individuals and as a team.

To friends and family who, even from far away or in small gestures, showed genuine support and interest in my journey, thank you.

Lastly, I am grateful to myself, the one who made this project a reality. This dissertation is the result of my dedication, hard work, and commitment to keep going even when things seemed hopeless. I accepted that challenge with strength and now I proudly appreciate its rewards.

Abstract

The management of distributed physical devices, especially in large-scale scenarios such as governmental digital inclusion initiatives, poses challenges related to security, control, and transparency. The management of the device lifecycle in unsupervised environments is exacerbated by concerns such as theft, loss, mistreatment, and lack of trustworthy processes for transparency and remote monitoring.

This dissertation presents a novel blockchain-based Device as a Service (DaaS) system that combines smart contracts, a web portal for legitimate users, and an embedded verification module at the United Extensible Firmware Interface (UEFI). The architecture ensures decentralised, transparent, and tamper-resistant device management, allowing remote locking and unlocking, real-time state verification, and immutable activity recording without the need for intermediaries.

Implemented on the Ethereum blockchain (Sepolia testnet), the system uses smart contracts written in Solidity for device registration and state management, Keyed-Hash Message Authentication Code (HMAC)-based cryptographic tokens to ensure data integrity, and Role Based Access Control (RBAC) to restrict actions to legitimate entities. A hardware-level security layer is supplied by a customised UEFI module, that validates device status updates before the Operating System (OS) boots.

The functionality, security, and scalability of the system are demonstrated by validation in realistic scenarios, such as device registration, access control processes, and audit logs. This evaluation proves the solution's ability to securely automate remote device management while providing transparency and resistance against attacks such as replay attacks.

The present study addresses the gaps presented in current remote management strategies by offering a comprehensive and novel system for decentralised device management that can be applied in business, government, and educational environments. The work received recognition with a Best Paper Award, highlighting its scientific value and practical application.

Keywords: Device as a Service (DaaS) · Smart Contracts · Blockchain · Remote Device Management

Resumo

A gestão de dispositivos físicos distribuídos, especialmente em cenários de larga escala como iniciativas governamentais de inclusão digital, apresenta desafios significativos ao nível da segurança, controlo e transparência. A complexidade da gestão do ciclo de vida dos dispositivos em ambientes não supervisionados é agravada por riscos como roubo, perda, uso indevido e ausência de processos fiáveis de monitorização remota e auditoria transparente.

Esta dissertação apresenta um sistema inovador de Dispositivos como Serviços (DaaS) baseado em blockchain, que combina contractos inteligentes (*smart contracts*), um portal web para utilizadores legítimos e um módulo de verificação embutido na Interface Unificada de Firmware Extensível (UEFI). A arquitetura proposta assegura uma gestão de dispositivos descentralizada, transparente e resistente a adulterações, permitindo operações de bloqueio e desbloqueio remoto, verificação do estado em tempo real e registo imutável das atividades, sem dependência de intermediários.

Implementado na blockchain Ethereum (rede de teste Sepolia), o sistema utiliza contratos inteligentes desenvolvidos em Solidity para registo e controlo de estado dos dispositivos, *tokens* criptográficos baseados em Código de Autenticação de Mensagem Baseado em Hash (HMAC) para garantir a integridade dos dados e Controlo de Acessos Baseado em Funções (RBAC) para restringir ações a entidades autorizadas. Um módulo personalizado na UEFI fornece uma camada de segurança a nível de hardware, validando o estado do dispositivo antes do arranque do sistema operativo.

A funcionalidade, segurança e escalabilidade do sistema foram demonstradas através de validação em cenários realistas, incluindo registo de dispositivos, controlo de acessos e auditoria de eventos. Esta avaliação comprova a capacidade da solução para automatizar de forma segura a gestão remota de dispositivos, assegurando simultaneamente transparência e resistência a ataques, como os ataques de repetição (*replay attacks*).

O presente trabalho responde às lacunas identificadas nas abordagens atuais de gestão remota, propondo um sistema abrangente e inovador de gestão descentralizada de dispositivos, com aplicação em ambientes empresariais, governamentais e educacionais. Esta proposta foi distinguida com um prémio de melhor artigo (*Best Paper Award*), sublinhando a sua relevância científica e aplicação prática.

Keywords: Device as a Service (DaaS) · Smart Contracts · Blockchain · Remote Device Management

Contents

List of Figures

List of Tables

List of Listings

Acronyms

ABE Attribute Based Encryption.

AES Advanced Encryption Standard.

AI Artificial Intelligence.

AMM Automated Market Maker.

API Application Programming Interface.

B2B Business to Business.

BIOS Basic Input/Output System.

blockDAG block-based DAGs.

BYOD Bring Your Own Device.

CBDC Central Bank Digital Currencies.

CeFi Centralised Finance.

CIA Confidentiality, Integrity, Availability.

CPU Central Processing Unit.

DaaS Device as a Service.

DAC Distributed Autonomous Corporation.

DAG Directed Acyclic Graph.

DAO Decentralised Autonomous Organisation.

DApp Decentralised Application.

DDL Decentralised Digital Ledger.

DeFi Decentralised Finance.

DEX Decentralised Exchange.

DID Decentralised Digital Identifier.

DNS Domain Name System.

DPoS Delegated Proof of Stake.

DSDM Dynamic Systems Development Method.

DT Digital Twin.

DXE Driver Execution Environment.

ECC Elliptic Curve Cryptography.

EFI Extensible Firmware Interface.

GPS Global Positioning System.

GPT GUID Partition Table.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HMAC Keyed-Hash Message Authentication Code.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IoMT Internet of Medical Things.

IoT Internet of Things.

IP Internet Protocol.

IPFS InterPlanetary File System.

LEA Lightweight Encryption Algorithm.

LPoS Leased Proof of Stake.

M2M Machine to Machine.

MBR Master Boot Record.

ML Machine Learning.

NEM New Economy Movement.

NFT Non-Fungible Token.

OS Operating System.

P2P Peer-to-Peer.

PBFT Practical Byzantine Fault Tolerance.

PEI Pre-EFI Initialisation.

PEP Policy Enforcement Point.

PKI Public Key Infrastructure.

POA Possession, Ownership, Availability.

PoA Proof of Activity.

PoB Proof of Burn.

POC Proof of Concept.

PoC Proof of Capacity.

PoET Proof of Elapsed Time.

PoI Proof of Importance.

PoS Proof of Stake.

PoSpace Proof of Space.

PoW Proof of Work.

PXE Preboot Execution Environment.

RBAC Role Based Access Control.

REA Resources, Events, Agents.

RFx Request for.

RPoW Reusable Proof of Work.

RSA Rivest Shamir Adleman.

TEE Trusted Execution Environment.

TLS Transport Layer Security.

TPM Trusted Platform Module.

TPS Transactions Per Second.

TradFi Traditional Finance.

TURS Tsinghua University User Reputation System.

txDAG transaction-based DAGs.

UEFI United Extensible Firmware Interface.

URL Uniform Resource Locator.

USP Unique Selling Point.

UUID Universally Unique Identifier.

XACML Extensible Access Control Markup Language.

zk-SNARK Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.

Chapter 1

Introduction

In an increasingly interconnected world, driven by scientific, technological and innovative advances, digitalisation has become a reality [14, 15]. Emerging technologies such as Blockchain, Artificial Intelligence (AI), and Internet of Things (IoT) have contributed to efficiency, connection, and innovation [15].

In the context of the COVID-19 pandemic, initiatives such as the programme established by the Portuguese Government in 2020 serve as examples of efforts to accelerate digital inclusion [16–18]. This initiative uses government-purchased equipment that is loaned to elementary and secondary school students so that they can use them and access the Internet. Besides the expected benefits of this initiative, the unsupervised domestic environment posed some barriers, notably the chance of theft, loss, or misuse of the devices.

The DaaS paradigm has emerged as a potential response to these challenges, offering a model in which devices are treated as services rather than consumable goods. This paradigm places a strong focus on device scalability and flexibility, which is consistent with the global trend of shifting from ownership to use [19, 20]. However, while widely studied in business environments such as the retail industry, the DaaS paradigm has not received enough attention, particularly when it comes to its use in scenarios that require higher levels of privacy, control, and immutability.

In light of this, the reference scenario adopted for the proposed DaaS model is depicted in Figure 1.1. Three primary actors participate in this model: the Users, who will be the end-user of the device; the Contractor, who will purchase, manage, and authorise device use; and the Supplier, who will provide the necessary devices to the Contractor. This scenario is flexible enough to be used in a variety of environments, including government initiatives for digital accessibility or business strategies for leasing devices.

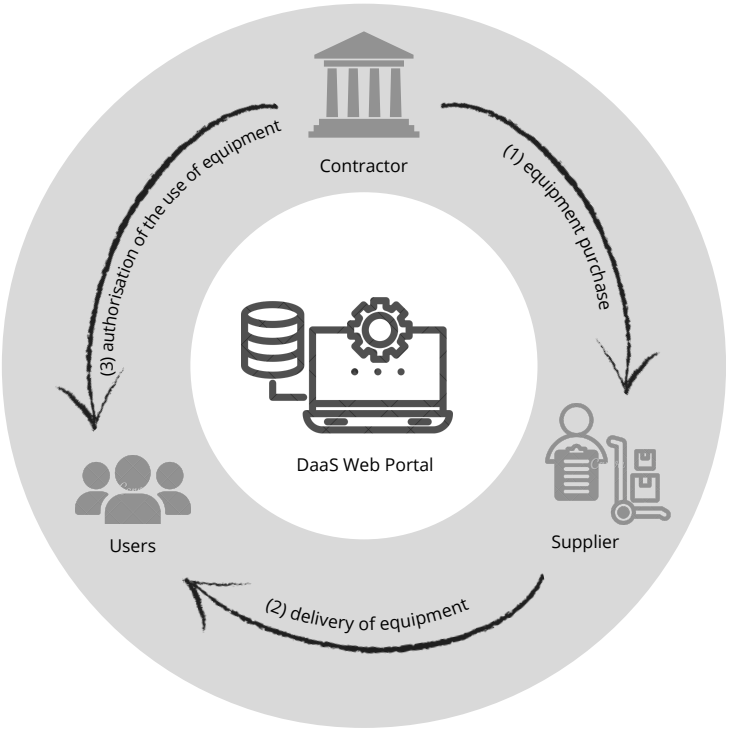


Figure 1.1: Reference scenario for the proposed DaaS model.

1.1 Novel Contributions

This work's main scientific contribution is the integration of modern technologies to address current problems in remote device management. This dissertation proposes the development of a blockchain-assisted DaaS system that combines an UEFI firmware-based control module with smart contracts. The main objective is to establish a decentralised, secure, and scalable system that can remotely manage devices while minimising challenges such as theft and mistreatment in distributed environments. The proposed solution enables immutability, transparency, and resilience, qualities that are crucial for globally deployed applications.

1.2 Publications

This research has led to the following academic achievements:

- **Publication:**
  "Blockchain-Assisted Device as a Service (DaaS)", Maria C. Tavares, Rui P. Mendonça, Diogo Meneses, André Santos, and António Pinto.
  In *Proceedings of the 6th International Congress on Blockchain and Applications*, Springer Nature Switzerland, Cham, pp. 110–119, 2025.

- **Award:**
  Best Paper Award for the work *"Blockchain-Assisted Device as a Service (DaaS)"* at the 6th International Congress on Blockchain and Applications.

1.3 Organisation

This dissertation is structured in chapters. Chapter 2 presents the principles of blockchain technology, highlighting its importance to distributed systems and the concepts that support its immutability, security, and transparency. Smart contracts and the UEFI firmware, two key components used as the framework for the proposed system, are covered in Chapter 3. There includes an investigation of their potential for safe device lifecycle management, as well as their boundaries. A detailed related work of the DaaS paradigm is presented in Chapter 4, with an emphasis on decentralised management approaches, firmware security, and device access control. It highlights the shortcomings of current solutions and research gaps. Chapter 5 describes the architecture and smart contract development of the proposed blockchain-assisted DaaS solution. The solution is evaluated according to the specified requirements in Chapter 6. It discusses how the system solves concerns such as auditability, tamper resistance, and remote access control that are raised in the literature. In Chapter 7, the solution findings are summarised and future research and improvement directions are suggested.

Chapter 2

Blockchain

At its core, blockchain is a special type of database sometimes referred to as a Decentralised Digital Ledger (DDL), and it is one of the most common types of distributed ledger technology systems [21–23]. Similarly to a spreadsheet, a blockchain holds data but restricted to certain limitations: each item of data storage (or block) must refer to the one before it, and data can be added but never changed or removed.

A blockchain's basic unit is a block, which is a collection of data with a particular structure. As a result, fresh data may only be added in the form of blocks, which together make up the blockchain database. Each block contains a unique hash identifier, derived from the hash of the preceding block. The fact that every block on the blockchain is built on top of the one before it, is what sets it distinct.

The technique known as hashing allows each individual block to reference the previous one by producing a fixed-size string (hash) from arbitrary input data, in this case the block itself. This is accomplished by using mathematical operations known as secure hash functions [24], accommodating various input types such as text, photos, and videos.

Blockchain has the significant advantage of not requiring trust on a central authority in order for the system to operate. It means that in the digital era, a blockchain functions as a decentralised source of verifiable truth. It enables a network of users to coordinate any new additions and reach consensus over what should be stored in the database.

Blockchain use a Peer-to-Peer (P2P) network architecture, giving users equal access to update privileges [12,25]. Each user in this network builds their own copy of the database by acquiring blocks from other users, eliminating the necessity of a central coordinator. On a blockchain, thousands of users can keep their copies, promoting redundancy and security. Since blockchain contributions are irreversible, the network ensures immutability, transparency, and trustlessness even in the face of many user disconnections.

Brief History of Blockchain

Stuart Haber and W. Scott Stornetta presented a novel method of securing time-stamped digital documents in the 1990's by employing a chain of cryptographically secured blocks [26]. This approach was ahead of its time. Merkle trees [27,28] were introduced in 1992 to increase efficiency. The system was inactive despite its potential, and its patent expired in 2004, four years before Bitcoin became popular.

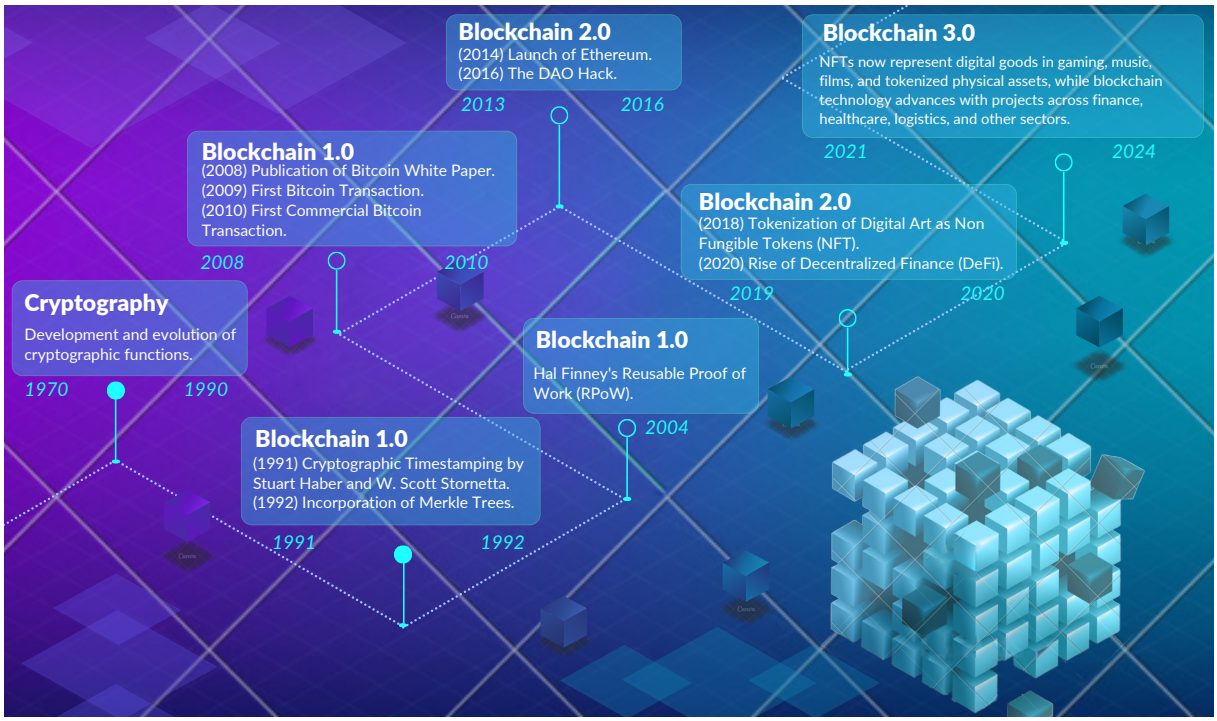


Figure 2.1: Blockchain Evolution Timeline: From Early Thoughts to the Present.

Reusable Proof of Work (RPoW) [29], developed by Harold Finney in 2004, was designed to solve the issue of double spending [30, 31] in digital payment systems. This technology was a critical turning point in the development of cryptocurrencies, as it relies on a reliable server to register token ownership globally. Fast forward to late 2008, "Bitcoin: A Peer-to-Peer Electronic Cash System" [32] was revealed in a white paper written by Satoshi Nakamoto (pseudonym). Through a decentralised network of nodes verifying transactions and mining new bitcoins using a Proof of Work (PoW) mechanism, Bitcoin offered a solution to the double spending problem [33].

The first Bitcoin block was mined by Satoshi Nakamoto on January 3^{rd}, 2009, marking the beginning of the cryptocurrency. In particular, on January 12^{th}, 2009, Harold Finney received ten bitcoins, making him the first recipient of Bitcoin in history. Then in 2013, Vitalik Buterin, co-founder of the Bitcoin Magazine, recognised the need for a scripting language in Bitcoin to enable Decentralised Applications (DApps). So, he started working on Ethereum [34], a blockchain that supports smart contracts. These contracts enable transactions based on established conditions and are implemented in the Ethereum chain. Ethereum is more versatile than just a payment platform; it has given rise to a number of applications, including social media, gaming applications, and Decentralised Finance (DeFi).

The evolution of blockchain technology has spurred innovations beyond Bitcoin and Ethereum, including alternative consensus mechanisms. Today, blockchain technology transcends its origins in cryptocurrencies, finding applications in various industries, including healthcare, supply chain management, gaming, digital identification, governance, and energy.

As the blockchain landscape continues to evolve, significant milestones mark its growth and expansion into various sectors. Figure 2.1 shows some of these milestones and the

trajectory of blockchain technology from its beginnings to the present.

Blockchain Functionality and Network Structure

The most fascinating feature of blockchain technology lays not necessarily in the database as itself, but rather in how it may be deployed by a network of unaffiliated entities. It is not an easy process without a central authority, and blockchain networks rely primarily on a combination of game theory[1] and cryptography to achieve this.

Fundamentally, a blockchain is simply an open database with transparent transaction records that anyone can verify for accuracy. However, ensuring the integrity of the transaction when network members lack confidence is a significant challenge. To address this challenge, blockchain networks are based on cryptographic techniques. By modifying data using a shared key, symmetric encryption [36, 37] secures communication; however, using a single shared key introduces risks.

For increased security, public key cryptography [38] is an option that uses two keys: a shared public key and a private key that is kept hidden. The private key is used to sign messages, while the corresponding public key is used to validate the signature (see Fig. 2.2). This cryptographic core provides blockchain systems with security and trust.

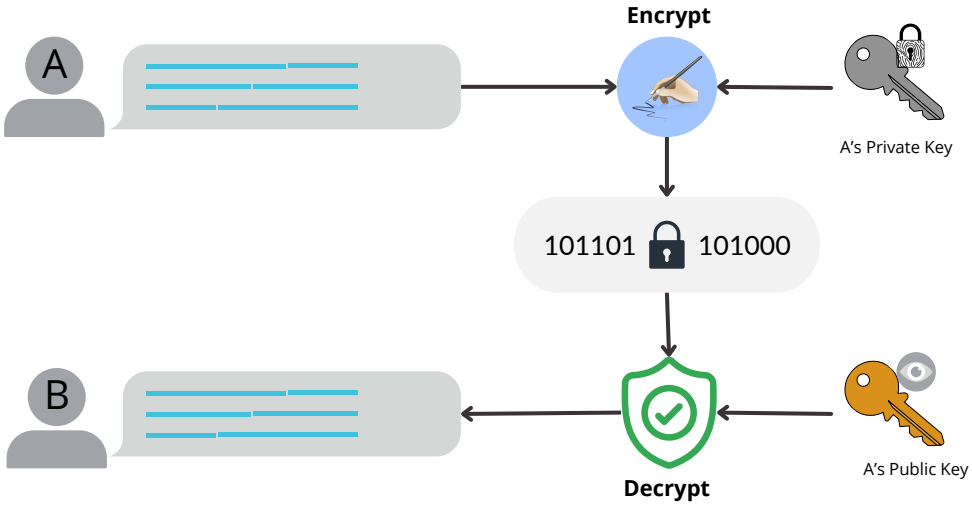


Figure 2.2: An illustration of Public Key Cryptography's Asymmetric Encryption.

Messages with digital signatures and transaction information are sent over a P2P network in the context of blockchain transactions. Each node confirms that the signatures correspond to the sender's public key to validate the legitimacy of transactions. Figure 2.3 provides a sample flowchart that outlines the fundamental steps involved in performing transactions through the blockchain network, from transaction initiation to final verification and block insertion.

There are two primary categories of blockchain technology: public and private blockchains. Anyone interested in transactions can use public blockchains, which are open and decentralised and provide unrestricted, permissionless access to data [3, 11, 12]. They are frequently employed in cryptocurrency mining and trading. Private blockchains, on the

[1] In the context of blockchain technology, game theory refers to the study of mathematical models that analyse strategic interactions among decentralised entities within the network [35].

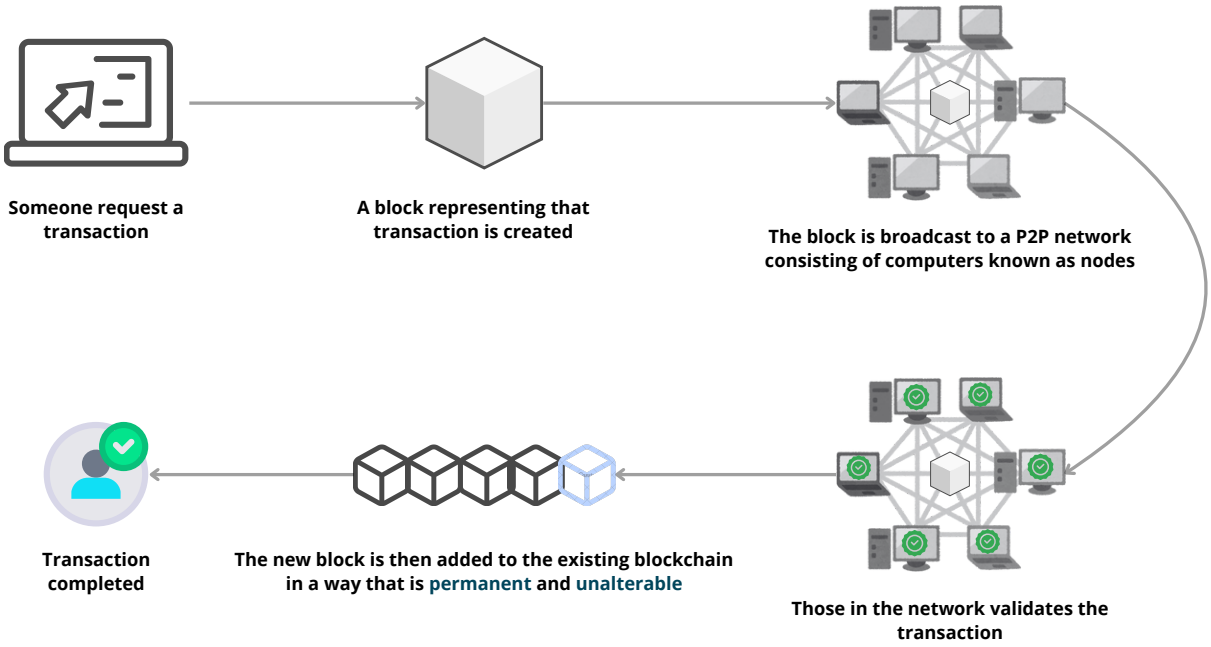


Figure 2.3: Overview of the blockchain transaction process. Adapted from [1–5].

other hand, are closed and restricted, requiring system administrators to approve transactions. These blockchains provide improved security and permission-based access in closed systems and networks, usually found in organisations [3, 11, 12].

Hybrid blockchains provide greater autonomy and flexibility by combining aspects of private and public blockchains. They satisfy particular user needs by offering flexibility in the management of both centralised and decentralised systems [3, 11, 12]. Consortium blockchains enable network management within a single entity or consortium. They are semi-decentralised in nature and are administered by organisations. They are frequently employed in industries such as finance, government, etc [3, 11, 12].

To provide an in-depth understanding of the distinctive characteristics and possible applications of each type of blockchain, the Table 2.1 offers an overview. It covers the benefits, drawbacks, common use cases, and underlying technologies related to blockchains. These insights can be used for selecting the appropriate blockchain technology according to specific requirements and conditions in various areas of the economy.

Table 2.1: Comparison of blockchain types: advantages, disadvantages, use cases and technologies. Adapted from [1, 3, 11, 12].

	Public	**Private**	**Hybrid**	**Consortium**
Advantages	Independence Transparency Trust	Access Control Performance	Access Control Performance Scalability	Access Control Scalability Security
Disvantages	Performance Scalability Security	Trust Auditability	Transparency Upgrading	Transparency

Tecnologies	Bitcoin Ethereum Litecoin	Multichain Hyperledger Fabric Corda Quorum	XinFin	Ripple
Use Cases	Cryptocurrency Document validation	Supply Chain Asset Ownership	Medical Records Governance	Banking Research Supply Chain

The network structure of blockchain systems consists of different kinds of nodes interacting to preserve the integrity of the distributed ledger.

Full nodes are essential for validating and spreading new blocks since they keep an entire copy of the blockchain. These nodes verify transactions and guarantee the historical consistency of the blockchain, but maintaining a whole node requires resources, especially when it comes to bandwidth and storage needs[2] [25, 41].

As a solution, light nodes rely on full nodes for information while performing minimum validation. These nodes just download the block headers, which are smaller in size, as opposed to downloading entire blocks. In this way more users can participate, even on devices with limited capabilities [25].

Mining nodes can be classified as both full or light nodes, but they are a specialised subset that carry out the PoW consensus mechanism. These nodes provide processing capacity to verify transactions and safeguard the network in cryptocurrencies such as Bitcoin. The competitive nature of mining means that participation requires specialised equipment with more capabilities than those of household devices, such as Graphics Processing Units (GPUs).

Since public blockchains are permissionless by nature and allow any node to engage in participation, it is ambiguous how modifications can be made without the need for central authority, and forks come in handy in this case. Hard forks cause the network to be split into two segments, one of which follows the original path while the other adopts the new protocols [12, 42]. This results in two incompatible networks that cannot communicate with one another. On the other hand, soft forks offer upgrades that are compatible with the current system and do not cause any disruption to it [12,42]. This makes it possible to introduce new features without causing problems for existing users. This opt-in feature of the upgrades guarantees that cross-communication is unaffected even though new features might not be visible to non-upgraded nodes.

Uses Cases and Limitations

Although sending money worldwide is just the transfer of digital data, there are often costs associated with the process due to intermediaries. Blockchain technology provides an answer by making affordable and fast money transfers available. Central Bank Digital Currencies (CBDC) have the potential to transform the global banking sector and the remittance industry, and many nations are investigating their issuance [43].

[2]By 16 Jan 2024, the bitcoin blockchain weighed in at over 530 gigabytes. At intervals of ten minutes, the nodes downloaded and verified the blocks [39, 40].

With the goal of constructing a more transparent, inclusive, and open financial system based on public blockchain technologies, DeFi has attracted a lot of attention. Through the removal of financial brokers, DeFi gives individuals direct control over their assets and facilitates P2P trading [44, 45]. Furthermore, blockchain technology enables numerous users to access the same database as a shared source of truth, guaranteeing trustworthy record-keeping with excellent data integrity.

Many organisations play a role in supply chains, and they frequently have to link their systems to make it easier for goods to make their way from suppliers to customers. And, for a healthcare system to operate effectively, digital health records must be shared and stored efficiently. In both cases, the blockchain technology improves efficiency and benefits the economy and customers by optimising Business to Business (B2B) interactions and having strong security standards, it can be used to safely store sensitive data. Blockchain technology is used in a number of industries, such as energy [46–49], gaming [50–52], and donations to charity [22, 53, 54]. Additionally, it offers organisations an other form of governance [55–58].

Although blockchain technology holds great promise, it also has drawbacks. A crucial component of blockchain technology, its immutability, presents challenges when adjustments are required. Asset access may be permanently lost in the event of private key loss. Furthermore, there has been concerns about the consumption of energy linked to blockchain networks, like Bitcoin [1, 59, 60]. The need for storage grows as the blockchain gets larger, which might make entrance obstacles higher. Moreover, a distributed system with fewer machines may have a network that is less resilient to attacks. Although a 50%+1 attack [61, 62] is a possibility, carrying it out would be extremely expensive and difficult.

2.1 Consensus Mechanisms

Reaching consensus among entities in a distributed system is difficult, especially when there is no central coordinator dictating tasks. Similarly to the Byzantine Fault Tolerance problem [63–65], which requires individuals to cooperate and come to a common truth to prevent fatal system failures, this situation calls for collaboration. In computer networks, this problem is especially relevant because some parties may not be trusted. By enabling each node, i.e. a user on the network, to independently verify the accuracy of data, blockchain solves this problem. Nodes must prove their commitment by working if they want to submit a message, in this case a block. This is the basis of Proof of Work (PoW) [32, 66, 67], a consensus mechanism that allows nodes to work together in distributed environments. Even in the event that some agents act maliciously, these techniques guarantee that the majority of agents within the system can agree on a single reality.

Hashing is performed in PoW to identify a particular hash that satisfies the blockchain network's difficulty level [32]. The resulting hash must start by a fixed number of leading zeros, in its hexadecimal format, that increases over time [68]. Since nodes have to generate multiple hash attempts, this constraint makes finding a valid hash arbitrary and expensive to compute [32]. The nodes broadcasts the block to the network after finding a valid hash. To preserve consistency, each block includes the previous block's hash. A crucial component of blockchain feasibility is that, despite the computational effort

involved in determining an accurate hash, verifying its validity is straightforward. Other nodes on the network can quickly confirm the legitimacy and authenticity of a block after a node gets the right hash. This dynamic exemplifies the concepts of game theory that underlie blockchain systems, since it is expensive to produce but cheap to validate. If a node produces an incorrect block, other nodes will reject it and the node could be out of pocket [32, 68].

PoW was the first consensus mechanism and is still important. Mining is the process of applying PoW to solve a block; those who engage in this process are called miners. Although PoW offers a high degree of security and is effective in preventing system abuses, it has drawbacks, especially when it comes to expensive equipment and significant energy consumption as the network grows [59, 60]. Alternative consensus mechanisms such as Proof of Stake (PoS) [69–71] were proposed to overcome these drawbacks.

To select validators from a group of nodes, PoS uses a pseudorandom election approach. To authenticate transactions, validators stake the blockchain's native currency, using the encrypted funds as proof. While honest validators receive new block production as compensation, malicious validators run the risk of losing their stake [69–71].

2.1.1 Proof of Work (PoW) *versus* Proof of Stake (PoS)

Proof of Work (PoW) was first presented by Cynthia Dwork & Moni Naor in 1993 as an answer to denial-of-service attacks and service abuses [66]. In 1999, Markus Jakobsson & Ari Juels established the term "Proof of Work" formally in [67]. This consensus mechanism's basic premise is that while solving an issue requires extensive computation, validating the answer is simple.

The release of Bitcoin in 2009 was a turning point for PoW, since it was adopted as a consensus mechanism to validate transactions and add new blocks to the blockchain [32]. This application of PoW attracted a lot of attention and was recognised as a consensus algorithm for cryptocurrencies, which is essential for Bitcoin in order to protect the network from different kinds of attacks.

As part of the PoW mechanism at Bitcoin, miners repeatedly search for a particular number within a specified range. Once this value is found, a block can be created. A block is composed of up of two primary components: a header and transactions (see Figure 2.4). The nodes broadcast transactions, which are tracked by the network, and the block header contains metadata about all the transactions in the block, such as the Merkle Tree root of all transactions, version number, creation time, current difficulty, and nonce.

Proof of Work has several benefits, most notably in terms of reliability and security. Robust security is provided by the process of solving computational puzzles, which guarantees that only legitimate blocks are uploaded to the blockchain, and manipulation is prevented because verifying a nonce is less computationally expensive than finding it. Furthermore, PoW creates a connection between network control and processing power, making it more difficult for adversaries to establish majority control. An opponent would have to own more than 50% of the network's processing power to execute a 50%+1 attack, which is very challenging for large and decentralised networks such as Bitcoin [13].

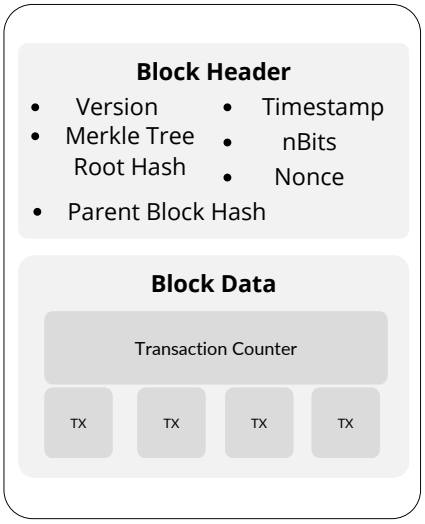


Figure 2.4: Overview of the block data. Adapted from [1–5].

Yet, PoW has some significant disadvantages. It appeals to environmental concerns because it demands a lot of computation and energy resources and, due to the fact that entities with more resources have a competitive advantage in mining, this desire for high processing power may cause network centralisation. This raises the possibility of organised attacks, such as a 50%+1 attack, in which the blockchain might be manipulated by a single miner or a group controlling more than half of the network's computational power [13, 72].

Also, due to its resource-intensive nature, PoW-based systems may not be as scalable as they may be for applications which require lightweight and efficient consensus mechanisms because to their high resource consumption and miner competition [72]. Despite its proven security and reliability, these drawbacks have led to the exploration of other consensus mechanisms.

Proof of Stake (PoS), introduced by Sunny King and Scott Nadal in 2012 [69], offers a more energy-efficient alternative. A selection process based on the amount of cryptocurrency, which validators hold and stake as a security deposit, replaces the competitive mining process in PoW. The probability of being chosen to validate transactions and create new blocks increases with the amount staked.

The most significant terminological and functional change in blockchain consensus mechanisms is the shift from miners to validators and from mining to forging blocks. Since they do not solve challenging computational puzzles, validators in PoS systems use a lot less energy than PoW miners; rather, the mechanism is conditional upon the quantity of cryptocurrency invested, hence minimising overall consumption of energy and promoting a more decentralised network [72].

To compete with other nodes and become validators in PoS, a node must submit a validator fee. Validators choose the individual with the most coins staked when an adequate number of transactions have been accumulated. This individual then produces and broadcasts a new block; this block is uploaded to the blockchain, validated by the network, and the transactions are performed. Then, the transaction fees are given to the chosen individual as a reward [13].

PoS became recognised after being adopted by some cryptocurrencies. Launched in 2012, Peercoin was the first cryptocurrency to employ Proof of Stake [69]. More recently, in September 2022, an event known as "Merge" allowed Ethereum, the second-largest cryptocurrency in the world, to switch from Proof of Work to Proof of Stake. This change demonstrated how PoS is becoming more and more preferred in order to solve the scalability and sustainability issues that PoW-based systems have to address.

PoS also offers some benefits. First, it is widely accessible because it relies on the legitimacy of transactions based on the amount of cryptocurrency staked rather than processing power. This encourages more decentralisation and lower barriers for entry for new individuals. Additionally, PoS is more scalable and able to manage a higher transaction flow than PoW, making it a better option for applications that require effective consensus mechanisms [13, 72].

On the other hand, PoS can be more susceptible to some types of attack, such as bribery attacks [73, 74], in which an attacker forks the blockchain and secretly builds a longer chain in order to reverse transactions. The initial transactions are reversed when the longer chain is acknowledged as legitimate after it is discovered. Stakeholders with large holdings have greater authority in PoS, which could eventually lead to centralisation. Also, they can implement changes without the approval of the larger community, which goes against the decentralised nature of blockchain technology [13].

To understand the relative strengths and weaknesses of both mechanisms, it is crucial to compare them across several criteria such as cost, resilience to 50%+1 attacks, decentralisation, energy efficiency, and security. The results are displayed in Table 2.2.

Table 2.2: Comparison of Proof of Work (PoW) and Proof of Stake (PoS) consensus mechanisms. Adapted from [13].

Criteria	**Proof of Work (PoW)**	**Proof of Stake (PoS)**
Cost	**High** due to the need for specialised hardware and significant electricity consumption.	**Lower** because validators do not require an enormous amount of processing power. The main fee for staking cryptocurrency is the initial deposit.
Resilience to 51% Attack	**Highly resistant**, especially for large networks like Bitcoin, it is not realistic to control 50%+1 of the network's processing power.	Economic disincentives would make it challenging for an attacker to get and hold a significant quantity of the coin, making it **economically impossible**.
Decentralisation	**Can lead to centralisation** in mining pools where a few large pools control a majority of the hashing power.	**Promotes decentralisation by lowering barriers to entry and allowing more involvement**. Large stakeholders, however, have a lot of influence.
Energy Efficiency	**Low**, due to high energy consumption for solving computational puzzles.	**High** since it uses significantly less electricity and doesn't require a lot of processing work.

Security	**Provides robust security through computational difficulty**, preventing easy manipulation.	**In general secure but potentially more vulnerable to low-cost attacks**. The monetary rewards for validators to behave respectfully are what ensure security.

In blockchain technology, Proof of Work and Proof of Stake are two of the most prominent and widely employed consensus algorithms, although they are not without drawbacks. PoW is frequently questioned for its high energy consumption and tendency to centralisation, while PoS is more energy efficient but still has drawbacks such as the "nothing at stake" problem and the risks of centralisation in networks where there are large socioeconomic differences among individuals.

Several alternative consensus procedures have been established to solve these constraints and handle certain use cases; these will be covered in more detail in the next section.

2.1.2 Alternative consensus mechanisms

To address the limitations of the Proof of Work (PoW) and Proof of Stake (PoS) consensus mechanisms and respond to specific use cases, several alternative consensus mechanisms have been proposed.

Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) is a consensus algorithm designed to solve the Byzantine Generals Problem, addressing the challenge of reaching consensus in a distributed network where some nodes may behave maliciously or fail unexpectedly. It was first proposed by Miguel Castro & Barbara Liskov in 1999 [75] and is now a key component in distributed systems failure tolerance. PBFT is frequently employed in permissioned blockchain networks, which have a small number of nodes under the authority of a reliable party.

PBFT operates through a series of phases: pre-preparation, preparation, commitment, and finalisation. During the pre-preparation stage, every replica node in the network receives a proposed block from the parent node. Each replica validates the block and broadcast a preparation message to the network during the preparation stage. Before broadcasting a commitment message during the commitment phase, nodes have to wait to receive preparation messages from a majority of replicas. Then, the node completes the block and updates its ledger when it receives the commitment messages. This mechanism allows the system to accept up to one-third of the failure or malicious nodes, as long as more than two-thirds of the nodes agree in order to reach consensus.

Practical Byzantine Fault Tolerance has a number of benefits, such as strong fault tolerance, fast productivity, and energy efficiency [72, 76]. It processes transactions faster and does not rely on energy-intensive mining approaches like Proof of Work. Furthermore, PBFT ensures strong fault tolerance without the need for validators to own any cryptocurrency on the network. However, large networks are not an appropriate match for PBFT due to their high communication cost, which limits their scalability [72, 76]. It

can also suffer from delayed consensus in big networks under unfavourable circumstances, and it is susceptible to Sybil attacks [76, 77].

Delegated Proof of Stake

Daniel Larimer introduced Delegated Proof of Stake (DPoS) in 2014, which is a significant improvement over more conventional consensus algorithms like Proof of Work and Proof of Stake. Delegated Proof of Stake uses a representative democracy to validate transactions and manage the blockchain, which should increase efficiency and scalability. In 2015, the BitShares platform launched the first iteration of this mechanism [78].

DPoS functions similar to a representative electoral system; the stakeholders use their cryptocurrency holdings to cast votes for a restricted number of delegates. These delegates update the blockchain with new blocks and validate transactions. Each stakeholder's voting power is based on their holdings, giving those with more assets greater influence [72, 79]. Rotating between them, elected delegates share power, produce blocks, and minimise the probability of centralisation.

Because DPoS restricts the number of nodes involved in block production, it offers faster transaction processing speeds while remaining efficient and scalable. Because it avoids the energy-intensive activities associated with PoW, this approach is more environmentally conscious. Furthermore, DPoS provides a flexible and adaptable governance approach that enables immediate elimination of ineffective delegates [72, 79].

Proof of Capacity

Proposed by Dziembowski et al. in 2013, Proof of Capacity (PoC), formerly called Proof of Space (PoSpace), is a consensus algorithm designed to offer a more energy efficient solution for the conventional Proof of Work mechanism. Instead of using continuous processing power, PoC uses the available storage capacity to validate transactions and protect the network [80]. The two main phases of PoC are mining and plotting.

Plotting is the stage in which miners precompute and store massive amounts of data on their hard drives called *nonces*, which has a major impact on their ability to solve the mining challenge and receive rewards. In the mining stage, miners calculate a *deadline* (the amount of time they have to wait before being allowed to create a new block) considering the nonces they have stored. Using the nonce data, the miners calculate the deadline and choose the smallest deadline among their nonces. The smallest deadline wins the privilege to add a new block to the blockchain and receives the reward associated with it if no other miner produces a block during this time [72, 81].

PoC is advantageous to conventional PoW mechanisms: firstly, it is a consensus mechanism that is far more ecologically friendly and energy efficient because it depends on hard drive space rather than powerful processing power; secondly, PoC uses regular hard drives instead of specialised mining hardware, which are more easily available and encourage decentralisation by decreasing the entry barrier for miners [72, 81].

Proof of Activity

The hybrid consensus algorithm known as Proof of Activity (PoA), first presented by Bentov et al. in 2014 [82], merges aspects of Proof of Work and Proof of Stake to improve

the security and effectiveness of blockchain technology. PoA was established to solve specific problems with Bitcoin, such as the "*tragedy of the commons*", which could cause the network to become unstable if miners only pay transaction fees after all bitcoins have been mined [83].

This mechanism begins with a PoW phase, where miners solve a difficult challenge to produce an empty block that has a header and the miner's reward address. Next, this block proceeds to the PoS phase, where it is signed by a random set of validators chosen by their cryptocurrency holdings. The chance of a validator being chosen is correlated with their investment, which means that individuals with larger stakes will have greater influence [82]. For the purpose of choosing validators, the "*follow-the-satoshi*" subroutine converts a pseudorandom value into a satoshi (smallest unit of the cryptocurrency) and then chooses validators based on this value. The block is broadcast to the network and added to the blockchain when the validators have verified it the required number of times [76, 82].

By employing both PoW and PoS security characteristics, PoA's dual-phase approach lowers the probability of a 50%+1 attack. Nevertheless, it has some drawbacks from both algorithms, like PoW's high energy consumption and PoS's potential currency accumulation, which could cause the network to become more centralised. Despite these challenges, PoA is more resistant to attacks that exclusively target currency holdings or processing power.

Proof of Importance

The New Economy Movement (NEM) project proposed the Proof of Importance (PoI) consensus algorithm with the aim of overcoming the drawbacks of the Proof of Stake algorithm. In opposition to PoS, in which the probability of generating new blocks and receiving rewards is directly correlated with the quantity of digital currency owned, PoI introduces more metrics to encourage network activity and improve decentralisation. Each account on the NEM blockchain is assigned a significance score that affects its probability of being chosen to add a new block — a procedure known as *harvesting* [84]. Three primary elements are taken into account when calculating the significance score: vesting, transaction partnerships, and the amount and magnitude of transactions [72].

Vesting, which has a significant effect on an account's significance score, is the quantity of vested coins an account holds. Long-term commitment to the network is rewarded with vested coins, which are those that have remained in the account for a predetermined period of days. Furthermore, the significance score of the account is positively affected by transactions larger than a minimal size, with larger and more frequent transactions having a greater impact [72]. By balancing the incentives for owning and actively consuming coins, this scoring system encourages volatility and network involvement.

PoI adopts a measure that takes into account variables such as vested balance, decayed and weighted outlinks, and summing to unity to determine significant scores, which guarantees decentralisation and security. These precautions help prevent potential manipulations and loop attacks [72, 84]. Through the integration of these elements, PoI offers an effective framework that promotes engagement, strengthens decentralisation, and preserves the blockchain's security and integrity.

Proof of Elapsed Time

Developed specifically for permissioned blockchains, Proof of Elapsed Time (PoET) was initially introduced by Intel in 2016 [20, 85] . Compared to Proof of Work, PoET requires less energy and computational waste since it takes advantage of a Trusted Execution Environment (TEE) to provide a secure and fair election process [72].

A random wait time is assigned to every node that participates in PoET. In a process similar to a lottery, the node with the shortest waiting period is chosen to mine the subsequent block, ensuring equality by offering every node a comparable opportunity of selection. After that, this node builds a new block and sends it out to the network for validation [72, 76].

The main benefits of PoET are its balanced leader election process and energy efficiency. This mechanism enhances the efficiency and transparency of the network by using the TEE to ensure that waiting periods are truly random and safe. However, its dependence on Intel hardware provides a centralised point of control that goes against the decentralised nature of blockchain technology. Despite this, PoET works effectively in permissioned blockchain environments where security and efficiency are crucial.

Leased Proof of Stake

An improved version of the classic Proof of Stake consensus process, known as Leased Proof of Stake (LPoS), is applied on the Waves Platform [86]. By allowing coin owners to lease their holdings to network nodes, LPoS improves the scalability and flexibility of the stake process.

With LPoS, users can lease their stake rights to full nodes, adding an additional layer to the PoS system, where nodes with a particular amount of money can add new blocks to the blockchain. Users can earn a portion of the rewards by lending a portion of their account balance to these nodes [72].

The main benefit of LPoS is that it makes staking more accessible to a larger audience. It encourages more decentralisation by allowing users with smaller holdings to participate in network security and earn rewards. Furthermore, by sharing stake power, LPoS improves network scalability by enabling more effective block validation and increased transaction performance. However, due to the system's reliance on node operators, network security can become compromised by potential threats like centralisation if many users lease their coins to a small number of nodes.

Proof of Burn

Developed by Iain Stewart, Proof of Burn (PoB) is a consensus technique that preserves decentralisation and network security while reducing the high energy consumption of Proof of Work [87]. A percentage of the bitcoin that miners produce is sent to an address, that can be validated to be unspendable and does not have a private key associated with it, in order to *burn* or destroy it. This process resembles to the energy and computational power used in PoW, but PoB uses the miner's current cryptocurrency holdings instead of physical resources [72, 76].

Compared to conventional Proof of Work, PoB has several benefits. First, it enhances decentralisation because miners autonomously choose how much cryptocurrency to burn

depending on their willingness to contribute to the network; there is no central authority in charge of the burning process. Second, because PoB burns cryptocurrency instead of doing advanced computations, it uses a lot less energy than PoW. Also, PoB allows miners to burn their own currencies in order to recover their initial investment through mining rewards, which encourages miners to be active on the network for an extended period of time.

Despite its benefits, PoB is not without challenges. One drawback is that it might cause a concentration of power among miners who can afford to burn more coins, similar to the problems seen in the PoS systems. In an effort to address these challenges and maintain a fair distribution of mining impact while promoting sustainable network growth, innovations such as adaptive burn rates have been developed [88]. These innovations modify the burning requirements dynamically based on network conditions.

Directed Acyclic Graph

Unlike usual blockchain networks, Directed Acyclic Graphs (DAGs) are essentially data structures, their successful use in cryptocurrencies such as NXT [89], IOTA [90], and IoT Chain [91] makes them worthy of consideration in the discussion of alternative consensus mechanisms. DAGs offer different advantages over blockchains storing transactions topologically in a graph.

A DAG ensures that the information only travels in the established directions; an acyclic graph excludes cycles, which means that information cannot loop back to the original node. Since DAGs have a blockless structure, transactions are added to the network instantly without the waiting periods associated with block delays, resulting in faster transaction verification [72, 92].

DAGs have many advantages. First, due to their blockless design, DAGs provide better speed and scalability, supporting thousands of transactions per second. Second, DAGs preserve energy because they do not require mining or specialised tools, which reduces power consumption and makes them more economical and ecologically friendly. Third, DAGs improve security and decentralisation; by limiting the number of previous transactions to validate transactions, they prevent forks and double-spending attacks. Last, DAGs networks are appropriate for microtransactions due to their economical transaction costs and instantaneous validation; this is particularly applicable in the context of Machine to Machine (M2M) and IoT interactions, where frequent low-value transactions are common [72, 92].

There are two categories of DAG-based systems: block-based DAGs (blockDAG) and transaction-based DAGs (txDAG). Each transaction in a txDAG establishes a direct reference to each subsequent transaction, creating a network where new transactions validate older ones. An example is Tangle [90, 93], where each transaction validates two preceding transactions, encouraging decentralisation and scalability. In contrast, blockDAG work similarly to conventional blockchains but link blocks in a graph as instead of a single chain, enabling the development and validation of several blocks at once, improving scalability and addressing issues such as orphan blocks[3] [76].

[3]Orphan blocks are valid blocks that are not part of the main blockchain due to temporary forks caused by simultaneous block mining. The network only accepts the longest valid chain, causing the shorter chain containing the orphan block to be discarded. Orphan blocks do not generate rewards for

For a complete comparison of all these mechanisms, see Table 2.3, which outlines their respective advantages and limitations, providing a clearer understanding of their operational dynamics and suitability for different blockchain applications.

Table 2.3: Comparison of consensus mechanisms: description, advantages and limitations.

Mechanism	Description	Advantages	Limitations	Reference
Proof of Work (PoW)	Introduced by Dwork & Naor (1993) to prevent denial-of-service attacks, PoW uses computationally intensive puzzle-solving to validate blocks, first applied in Bitcoin (2009) for transaction verification and block creation.	Strong security is provided by the complexity of solving puzzles Connects computing power and network control, requiring 50%+1 control for attacks	Highly energy-consuming and resource-intensive Risk of centralisation due to costly expenses	[32, 66, 67]
Proof of Stake (PoS)	King & Nadal (2012) developed PoS as an energy-efficient substitute for PoW. Select validators who generate blocks without mining based on stake size.	Efficiency of the energy, promoting decentralisation Supports higher scalability and transaction flow	Susceptible to bribery attacks Large stakeholders have more influence, risking centralisation	[13, 69, 72]
Practical Byzantine Fault Tolerance (PBFT)	Proposed by Castro & Liskov (1999) to address the Byzantine Generals Problem, PBFT allows consensus in distributed networks with possibly suspicious or failure nodes. Commonly used in permissioned blockchains.	Strong fault tolerance Fast processing Energy efficient Does not require validators to own cryptocurrency	Scalability in large networks is restricted by high communication costs Susceptible to Sybil attacks and delays in large networks	[75]

the miner, as they are not included in the main blockchain [94, 95].

Delegated Proof of Stake (DPoS)	Larimer (2014) introduced DPoS, a consensus mechanism that uses representative democracy to manage the blockchain and confirm transactions. It was first used on the BitShares platform in 2015.	Increased efficiency and scalability Environmentally friendly, avoids energy intensive mining	Influence weighted by holdings may lead to centralisation risks Dependence on elected delegates	[72, 78, 79]
Proof of Capacity (PoC)	As an energy-efficient substitute for PoW, PoC was suggested by Dziembowski et al. (2013). Validates transactions based on available storage capacity instead of continuous processing power.	Energy efficient and environmentally friendly Encourages decentralisation by reducing entry barrier	Requires large storage capacity Less established than PoW in common applications	[72, 80, 81]
Proof of Activity (PoA)	To improve security and efficiency, Bentov et al. (2014) proposed PoA, which mixes features from PoW and PoS. Miners first solve a PoW challenge, and after a PoS phase, the validators selected based on their stakes sign the block.	Reduces risk of 50%+1 attack Leverages benefits of both mechanisms	High energy consumption from PoW Risk of centralisation due to bigger stakes	[76, 82]

Proof of Importance (PoI)	PoI, which was proposed by the NEM project (2015), enhances on PoS by allocating significance scores which influence a user's probability of being selected to add additional blocks. These ratings are based on vesting, transaction activity, and alliances.	Promotes network activity and decentralisation Enhances security by discouraging inactive holdings	Complex calculation of significance score Require substantial vested coins to participate	[72, 84]
Proof of Elapsed Time (PoET)	PoET, developed by Intel (2016), lowers energy and computational waste by using a TEE to randomly assign a wait time to each node. The block that has the shortest wait time is produced.	Energy efficient and fair leader selection Secure with use of TEE	Dependent on Intel hardware, centralising control Less suited for public networks	[20, 72, 76, 85]
Leased Proof of Stake (LPoS)	An enhanced version of PoS used on the Waves Platform, LPoS lets users lease their stake to full nodes, boosting scalability and participation.	Allows smaller stakeholders to participate Enhances decentralisation and scalability	Risk of centralisation if stakes concentrate on few nodes Relies on trusted node operators	[72, 86]
Proof of Burn (PoB)	A consensus mechanism where miners burn cryptocurrency by sending it to an unspendable address, simulating resource expenditure and reducing energy use.	Reduces energy consumption Enhances decentralisation through voluntary burn	Risk of concentration among high-stake miners Adaptive burn rates needed to balance equity	[72, 76, 87]

Directed Acyclic Graph (DAG)	A blockless data structure that stores transactions in a topological graph, used in cryptocurrencies like IOTA and IoT Chain to achieve faster verification.	Highly scalable Energy efficient and eco-friendly Suitable for microtransactions	Potential complexity in implementation Vulnerable to certain attacks if not properly designed	[72, 76, 92]

2.2 Decentralised Finance (DeFi)

One of the most prominent and revolutionary applications of blockchain technology is Decentralised Finance, or DeFi for short. In essence, DeFi is based on the decentralisation concept, which calls for the delegation of operational and administrative authority from centralised organisations to distributed networks. DeFi is an ecosystem of financial products that operate on blockchain infrastructures, especially those that are controlled by smart contracts, allowing trustworthy, transparent and autonomous transactions [44, 45].

Bitcoin, as the first P2P digital currency built on a blockchain, marked an initial step in DeFi. However, the launch of Ethereum introduced key innovations, including the ERC-20 token standard, executable smart contracts, and a more versatile programming framework. These factors encouraged the development of more elaborate DApps, such as digital asset loans and lending platforms. Moreover, stablecoins, Decentralised Exchanges (DEXs), and lending and liquidity protocols are some of the DeFi uses cases [33, 44, 96].

DeFi has several benefits for consumers, the main one being the simplicity of access to a wide range of financial services [33, 44]. DeFi distinguishes itself from traditional finance[4] through its inclusivity, allowing anyone with Internet access to create a digital wallet and engage in financial activities without institutional approval, minimum balance requirements, or identity verification [21, 33, 97].

DeFi is positioned as an instrument for financial empowerment due to this characteristic; however, it is critical to distinguish between DeFi and Centralised Finance (CeFi). CeFi describes financial services, even in the cryptocurrency world, that are provided by centralised entities; for example, staking tokens or trading assets on centralised exchanges usually requires users to give up control of their assets and trust transaction management to a third party [21, 33, 96]. On the other hand, DeFi systems use self-executing smart contracts to work without the need for intermediaries; customers have more control and privacy because they negotiate directly and have custody of their own wallets [21, 33, 96, 97].

However, CeFi and DeFi both have their own pros and cons. In general, CeFi is easier to use, provide customer service, fiat on-ramps, and technical friendliness [33]. In contrast, DeFi offers greater confidentiality, autonomy, and resistance to censorship, but it also requires a higher level of technical expertise [21]. Since the user is exclusively responsible

[4]Traditional Finance (TradFi) is the term used to describe the traditional financial system, which consists of banks, stock exchanges, credit organisations, and financial regulators. This system depends on intermediaries such as banks and brokers to administer assets and carry out transactions.

for preserving funds, there are potential risks associated with software vulnerabilities, malicious contracts, and human error [33].

Furthermore, there are still numerous threats related to the DeFi ecosystem, including as counterparty risk, regulatory ambiguity, low-liquidity assets, and a lack of cross-chain interoperability [21,33]. It requires a lot of knowledge and keen awareness when exploring the area. Despite such challenges, DeFi continues to evolve and novel concepts are always being explored. It has the potential to improve economic accessibility and revolutionise the global financial system, especially in areas where traditional financial services are limited or nonexistent.

2.2.1 Cryptocurrency

At the heart of the DeFi ecosystem lies the concept of cryptocurrencies. Unlike fiat currency[5], cryptocurrencies are digital assets based on blockchain technology that use encryption to verify transactions and maintain records without the need for centralised authority [2]. These assets serve not only as currencies for P2P exchange, but also as collateral, governance tokens, and liquidity instruments within DeFi protocols.

A key feature of cryptocurrencies is their decentralisation, which fosters a distributed system with universal standards that are resilient to arbitrary modifications by any authority [96]. Cryptocurrencies also enhance privacy and security. Important characteristics like immutability and resistance to censorship guarantee that every unit is distinct and protected from duplication or falsification [96]. These characteristics position cryptocurrencies as a viable medium for international trade and represent a safe, transparent and decentralised digital economy.

With the launch of Bitcoin in 2009, the world of cryptocurrencies has grown and cryptocurrencies can be broadly classified into two categories: Bitcoin and Altcoins[6]. Altcoins, often known as "alternative coins," are developed to address specific use cases and introduce functionalities beyond those of Bitcoin. They provide a variety of functionalities. Well-known examples are Ethereum [34], Litecoin [99], and Ripple [100], each of which was created with distinct characteristics and innovative technologies.

It is important to distinguish between cryptocurrencies, CBDC and digital tokens. As previously stated, cryptocurrencies are decentralised assets built on blockchains, whereas CBDC are issued and controlled by governments, sometimes without the use of decentralised technologies like Blockchain [21]. Another distinction in the world of cryptocurrencies is between coins, which are native assets of the specific blockchain (such as Bitcoin on the Bitcoin, Ether on Ethereum, among others), and tokens, which are issued on top of pre-existing blockchains to represent assets or enable features.

A particular subset of tokens is Non-Fungible Tokens (NFTs), which is characterised by their unique and unchangeable nature. Although less vital to DeFi applications, NFTs might appear in supplementary use cases such as digital identification, collateralisation,

[5]Fiat money is defined as currency that is manufactured by the government and isn't backed by any physical commodity, but rather by the confidence of the population in the issuer, which is usually a central bank [98].

[6]Any cryptocurrency that isn't Bitcoin is referred to as a "Altcoin". The prefix *"alt"* comes from the word alternative, indicating an alternative to Bitcoin, and *"coin"* stands for currency.

and the representation of unique assets in decentralised marketplaces such as digital art, games, collections and other unique goods.

To engage in DeFi applications, users must store and manage their cryptocurrencies in crypto wallets, which allow direct interactions with the blockchain. These wallets produce and keep public and private key pairs that ensure exclusive control over funds and enable the production of digital signatures. The security of the private key is essential since losing it will permanently restrict access to the assets.

Cryptocurrency wallets can be classified based on their storage method and Internet connectivity. Hot wallets, being online, provide direct access to financial assets, increasing usability. In contrast, cold wallets remain offline, offering more protection against cyber threats. Another classification considers their composition. Hardware wallets, which store private keys offline, require technical proficiency but provide enhanced security, and, software wallets, available in desktop, mobile, and web formats, offer varying levels of accessibility.

Within the DeFi ecosystem, hot wallets are preferred for seamless interaction with DApps. Nonetheless, cold storage solutions, such as hardware wallets, are often chosen by experienced users and those managing substantial financial holdings [2].

2.2.2 Core Components and Uses Cases

DeFi covers a broad spectrum of financial services that are often provided by centralised organisations. Depending on the legislative environment in their area, users can participate in token swaps, lending, borrowing, payments, and stakes with just a cryptocurrency wallet and an Internet connection [33, 44]. With a couple of clicks, these services can be accessed via DApps, eliminating the need for complex paperwork, lengthy processing periods, and physical bank offices.

By 2025, staking mechanisms, liquidity pools, and lending and borrowing platforms continue to account for the majority of DeFi activity [101]. DeFi's capabilities are naturally extended to more intricate services like insurance and mortgages, although they are still in their early stages.

Stablecoins

A stablecoin is a kind of cryptocurrency that relies on a reference asset, such as gold or the euro, and was designed to keep its value constant. Usually, algorithmic changes, collateralisation techniques, or a mix of the two are used to ensure the stability of this value [21]. One of the drawbacks of all cryptocurrencies is price volatility, which stablecoins address [21, 96]. For instance, if a user loans funds using Bitcoin as collateral and the cryptocurrency's market value drops significantly, they might have to deposit additional assets or risk liquidation. This problem is mitigated by stablecoins, which offer consistent value and are consequently suitable for transactions, savings, and lending platforms [21, 96].

Decentralised Exchange (DEX)

DEXs operate without the need for trusted intermediaries. Smart contracts enable transactions on DEXs, which happen directly between the user's wallets. Some exchanges use

Automated Market Maker (AMM) to ensure continuous liquidity. Transaction fees are paid to liquidity providers in proportion to their pool membership. Protocols like Balancer, SushiSwap, and Uniswap are a few examples of decentralised exchanges that have contributed to the expansion of DeFi.

Lending and Borrowing

Among the most popular DeFi services are lending and borrowing for cryptocurrency. Users often deposit their crypto assets into a protocol, which makes them available to borrowers in exchange for earning interest [21, 96]. To secure the loan, the borrower must provide security, sometimes in the form of more cryptocurrency holdings [21, 96]. This process is regulated by smart contracts, which ensure the automatic application of terms and conditions, including loan-to-value ratios and liquidation triggers [21]. Aave, Compound, and MakerDAO are a few instances of lending platforms.

Asset Management

DeFi asset management protocols allow users to invest and diversify their portfolios without the need for intermediaries [21, 96]. These systems automate fund allocation across many protocols, like yield farming, staking, and lending, by applying smart contracts. Strategies for asset tokenisation and rebalancing aid in dynamic risk management and return optimisation [21, 96].

Oracles

Oracles serve as a middleware that connect smart contracts with real-world data sources. Since off-chain data is by nature inaccessible to blockchains, oracles fetch external information allowing contracts to execute based on real-time events [96]. Oracles serve as an infrastructural layer that increases the efficiency of the core network, not only a conduit between blockchain and the external world [96]. Decentralised oracle networks such as Chainlink as well as others enhance DeFi functionality by ensuring accurate and resistant data flows across platforms.

Insurance

DeFi insurance protocols protect individuals against hazards such as smart contract failures, exchange hacks, and stablecoin depegging [21]. These systems, in contrast of traditional insurers, automate capital pools for payments and claim processes using smart contracts [21]. DeFi insurance plays a pivotal role in protecting user assets against potential financial losses caused by system vulnerabilities [21]. Platforms like Etherisc and InsurAce offer community-funded coverage, which improves transparency, lower costs, and increases confidence in the DeFi ecosystem.

The development of DeFi applications continues to be mostly done on Ethereum, however other blockchains, such Binance Smart Chain (BSC) and Solana, are growing due to to their minimised transaction fees and increased throughput per second. Nonetheless, it is important to carefully evaluate each application's legitimacy.

Studies has shown that smart contracts contain vulnerabilities as well as that security audits are vital to mitigate risks. This prerequisite for programmable confidence highlights the value of smart contracts, which are at the centre of all DeFi activities.

Chapter 3

Smart Contracts and UEFI

This chapter delves into two important elements that are significant to the proposed solution: smart contracts and UEFI. Although smart contracts are the main component for automating and executing distributed transactions on the blockchain, UEFI offers the interface between the hardware and the OS, enabling low-level remote device management actions.

Section 3.1 goes over smart contracts, covering their background, operational logic, and implementation in DApps, and Section 3.2 discusses the function of UEFI in allowing secure boot-time activities and how it may be integrated into the remote management architecture.

3.1 Smart Contracts

Nick Szabo originally proposed the concept of smart contracts in the 1990s. His proposal aimed to expand the functionality of contractual agreements in the digital environment by taking advantage of computer protocols [102]. Szabo envisioned automated contracts that could encourage, validate, or regulate contract negotiations or execution without human interaction [102]. Although initially theoretical, this concept served as a basis for later implementations.

In 2013, with the introduction of the Ethereum blockchain, smart contracts gained popularity [103]. Ethereum allows users to publish executable programs, known as smart contracts, on the blockchain, ensuring consistent execution across all participating nodes [34].

A smart contract is a deterministic and autonomous computer program stored on a blockchain that is triggered by specific predefined conditions [1,23]. The primary function of a smart contract is to automate and self-regulate agreement fulfilment [104, 105].

In a traditional contract, participants rely on intermediaries such as lawyers or notaries to make sure the terms are fulfilled; in contrast, a smart contract has its criteria encoded directly into the blockchain, and executes autonomously when the established conditions are met. Its decentralised execution reduces costs and improves the efficiency of transactions while minimising the likelihood of tampering [105].

This concept, along with technological advancements, distinguishes smart contracts with some qualities that ensure their efficacy and integrity in distributed networks. Table 3.1

outlines the main characteristics of smart contracts and their applicability to the blockchain ecosystem.

Table 3.1: Characteristics of smart contracts in blockchain environments.

Characteristic	Description
Deterministic	Ensures that, for the same input and initially state, the execution outcome is similar on all network nodes, ensuring predictability and consistency.
Immutable	Once published on the blockchain, the smart contract code cannot be modified.
Transparent	The contract's source code is accessible to everyone, allowing any interested party to inspect it and increasing trust in the behaviour of the system.
Autonomous	Once deployed, the smart contract logic runs automatically in compliance with the specified conditions, without the need for third-party intervention.
Trustless	Third parties are not required to validate the integrity of the process or ensure that all requirements are met since the execution model relies on distributed consensus approaches.

The programming principle behind smart contracts is comparable to the *if/when* statement in software systems. When a particular condition is met, the smart contract executes the applicable terms. This structure allows it to perform tasks such as digital asset transfer, identity validation, and resource access management. Once deployed, the smart contract is permanently stored on the blockchain and any contract-related transactions have to be signed and broadcast by the network participants.

3.1.1 Smart Contract Lifecycle

Understanding the lifecycle of a smart contract is essential to understanding its creation, operation, and evolution in distributed environments. The lifecycle of a smart contract can be split into some stages which are comparable in some ways to the lifecycle of standard contracts but have their own automation, transparency, and security features.

Figure 3.1 illustrates a typical lifecycle of smart contracts. This lifecycle has six primary stages: negotiation, modelling and design, business logic coding, blockchain deployment, execution and evaluation, and network updates. The details on each of these stages are provided next.

Negotiation

In negotiation, each involved participant identifies the potential for collaboration and establishes the goals to be achieved [6, 8, 106]. These discussions can cover asset swaps, business processes, or other types of interaction [7]. When it comes to smart contracts specifically, these discussions need to be clear terms that can be coded. When negotiations are completed, the draft contract must articulate what is expected and outline the corrective actions to be taken if a participant does not meet the promises made under the terms of the agreement [6, 106].

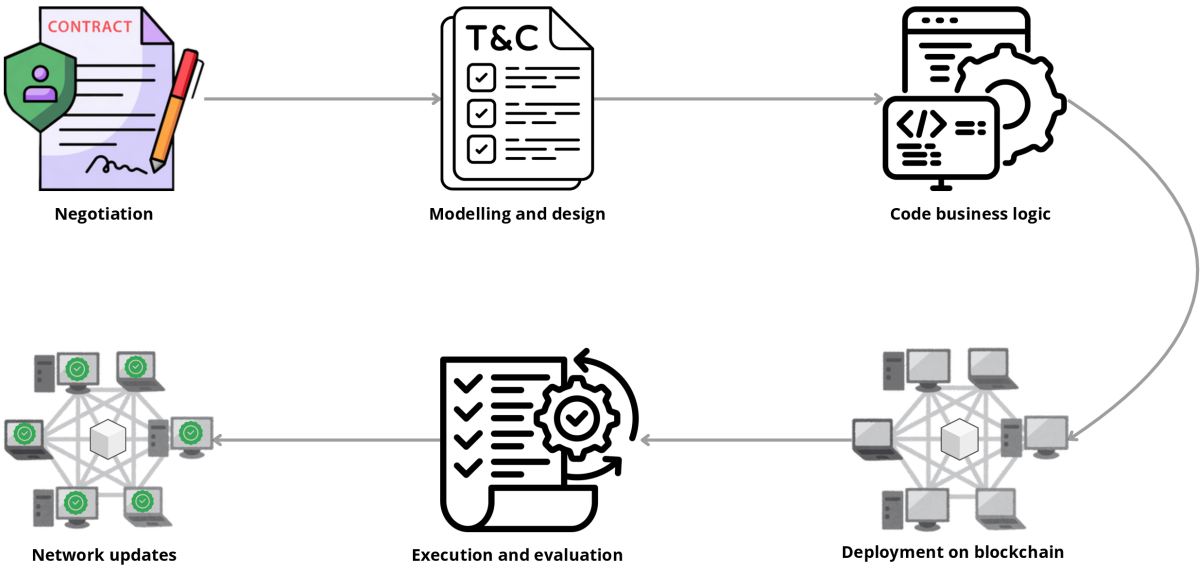


Figure 3.1: Lifecycle of a smart contract. Adapted from: [6–8].

Modelling and design

Smart contract development includes modelling and design, implementation, and validation, similar to software development [8].

Whereas standard contracts often begin with a formal document such as a proposal or an *Request for (RFx)*, smart contracts define trigger conditions at this stage. These triggers are conditional clauses programmed to be released when particular requirements are met, such as fluctuations in financial market indices or location-based events using Global Positioning System (GPS) data [7]. Automating the triggering mechanism in this way reduces the need for manual intervention and enables seamless execution.

Code business logic

Once the contract terms and conditions are clear and defined, the implementation and validation process begins. In standard contracts, business logic is defined by legally clauses that all parties can review and negotiate [106]. In smart contracts, this reasoning is translated into computer code that is written down [7, 8, 103]. The complete specification of the business logic in the code is essential to ensure that the smart contract is executed autonomously.

Ensuring the reliability of the contract at this stage is crucial. As smart contracts are immutable, coding errors can lead to costly issues such as loss of funds or incorrect contract fulfilment. Furthermore, before being published, smart contracts are typically tested on test networks. These tests replicate contract execution, enabling developers to ensure that it performs as envisioned under different conditions without affecting real assets or the main network.

Deployment on Blockchain

Following successful validation, the smart contract is deployed on a blockchain platform [8]. In standard contracts, intermediaries, such as solicitors and notaries, provide

security and authenticity [106]. In smart contracts, the authenticity and communication transfer between participants depend on encryption. Alongside cryptographic algorithms, blockchain technology ensures privacy, integrity, and non-repudiation of transactions [7]. To accomplish this, digital signatures and cryptographic keys are used, which ensure that the parties to the smart contract are correctly identified and that transactions cannot be modified.

The deployment makes the smart contract accessible to all users on the network [8]. Modifications cannot be made post-deployment; any change requires the creation and deployment of a new contract [7]. Furthermore, access to digital assets can be restricted or locked within the logic of the smart contract, depending on its design [8]. For example, some smart contracts are designed to prevent unauthorised inflows or outflows of digital assets. Authentication is facilitated through the digital wallets of the participants, each containing cryptographic keys.

Execution and evaluation

In standard contracts, execution occurs once all parties have signed the agreement [106]. However, smart contracts are executed automatically once the predefined conditions are satisfied and validated by the blockchain network [8, 103].

The distributed nodes on the blockchain evaluate the logic of the smart contract and ensure that the outcomes align with the agreed terms. Upon execution, the transaction is immutably recorded on the blockchain, serving as evidence of completion [7].

Network updates

Following execution, the smart contract initiates automatic updates throughout the blockchain network. This contrasts with standard contracts where the parties require tangible proof to confirm that the terms have been fulfilled [106]. The blockchain is updated with the new smart contract state, ensuring the immutability of data [7, 8].

In summary, smart contracts significantly automate and decentralise some stages of the contract lifecycle compared to traditional contracts. They automatically execute predefined terms, eliminating the need for human intervention in tasks such as legal review, approvals, and execution through intermediaries [106].

Moreover, a decentralised network of nodes provides validation and storage, eliminating the need for a central authority and lowering the likelihood of data loss or tampering. The inherent immutability of blockchain technology enhances transparency and traceability by ensuring that contract records remain tamper-proof after execution.

Lastly, the paradigm of trust evolves from relying on third parties to applying consensus mechanisms and cryptographic assurance. In this sense, smart contracts improve the efficiency, security, and legal integrity of digital transactions, in addition to automating contractual processes.

3.1.2 Benefits and Drawbacks

Given their benefits over traditional contracts, smart contracts are gaining attention, especially in the context of distributed systems. One of its strengths lies in their ability

to perform automated processes [104, 105]. Smart contracts promote autonomous and deterministic transaction execution by directly encoding the terms of the contract into an algorithm, thus eliminating the need for manual intervention after deployment [7, 8]. This automation reduces processing time and associated operating costs by removing intermediaries such as lawyers, notaries, and financial institutions [7, 8, 105].

Another significant benefit is the immutability inherent in blockchain-based smart contracts. Once deployed, a smart contract cannot be modified, which enhances the integrity and trustworthiness of digital agreements. This feature ensures that the terms initially agreed upon by the involved parties remain unchanged throughout the contract's lifecycle [8]. Immutability, coupled with the auditability of blockchain systems, facilitates real-time inspection of transactions and contributes to the prevention of fraud and manipulation [8].

Smart contracts also promote precision and clarity [7,8,105]. Since contractual clauses are implemented as conditional logic, they must be explicitly defined, leaving less room for ambiguity [8]. In public blockchains, transparency is an additional benefit, as all network participants can access the contract's terms and monitor the transaction history [7,8]. This decentralised verification process fosters accountability and trust among parties without the need for centralised oversight [7, 8].

In the business world, efficiency and speed of execution are important. The speed of completion of processes is ensured by the absence of manual verification and automatic execution based on trigger events, such as the transfer of a certain amount of cryptocurrencies or a date [7, 8]. The security of smart contracts is supported by the decentralised nature of blockchain and cryptographic techniques. Since each transaction is validated by multiple nodes in a distributed validation approach, it is resistant to fraud attempts and malicious attacks [7]. Moreover, the application of encrypted data improves the integrity of the network and protects data from tampering or unauthorised access [7].

Lastly, cost savings is an important advantage [7,8]. As stated previously, smart contracts minimise administrative and legal challenges by eliminating intermediaries and automating the execution and verification of the terms of the contract. For large companies and international firms that manage a lot of contracts, this reduction leads to substantial savings and improved efficiency in operations [7].

But even with these benefits, smart contracts still face a number of technical and practical challenges that may prevent their adoption.

Their best characteristic, immutability, might be a drawback in the event of an error or the need for contract rewriting [7]. Once deployed, errors or logical flaws in the contract code cannot be corrected without deploying a new contract. This rigour introduces risks, particularly in complex or long-term agreements.

While techniques such as conditional update systems or *escape hatches*[1] exits, they require special care as they may undermine the security that immutability seeks to guarantee [7].

[1]Escape hatches are emergency measures used in rollups that enable users to continue interacting with the system or recover assets in cases where operators, such as state proposers or sequencers, become inactive, behave maliciously, or disregard their responsibilities [107, 108]. There are two types: forced withdrawal, which ensures that money is taken out of the second layer even if the state proposer fails to generate valid state updates or proofs, and forced inclusion, which allows transactions to be included despite censorship [108].

Privacy is another concern. Despite the pseudonymous nature of blockchain nodes, the public nature of the transaction record allows any network participant to inspect the content of smart contracts [7, 8]. Although some blockchain systems use pseudonymous public keys to enhance transaction privacy, most transaction data, such as balances, remain public [8]. This presents challenges in cases where the preservation of contractual confidentiality is important [7].

The high technological barrier to establishing and verifying smart contracts is yet another challenge. Developing trustworthy, auditable, and secure code requires knowledge of specific programming languages (e.g., Solidity, Go, Vyper, Rust) and an understanding of security best practices and contract logic [8]. The participation of regular users and startups is limited by this complexity, as they often do not have the financial and technological capacity to guarantee the integrity of the contracts they use [7].

Moreover, smart contracts are vulnerable to some kinds of attacks. Examples cases such as the Decentralised Autonomous Organisation (DAO) and the Parity wallet attacks demonstrated how small weaknesses such reentrancy vulnerabilities, can cause significant monetary losses [8, 109]. The importance of exhaustive testing, legal verification, and code auditing before deployment is underscored by such situations.

Some challenges are related to the environment in which smart contracts perform. For instance, in order for contracts to work adequately, they frequently need off-chain data, such as weather or other results. Considering blockchains cannot access external data directly, they rely on oracles, which are external services that input real-time data into the blockchain [8]. Oracle reliability and security have significance since inaccurate information can result in inappropriate contract behaviour [8].

Concerns about dependence on transaction ordering are also known. The order in which transactions are included in a block is decided by miners, and users are often able to influence this sequence (a problem called front-running [110]), which could impact the contract's result, particularly in financial applications [7, 8, 110]. Furthermore, execution efficiency is limited as smart contracts are often processed sequentially across the network. Performance is affected by this serialisation, and scaling can be challenging [8].

In summary, smart contracts provide considerable advantages for digital transactions and contractual automation, such as increased operational efficiency, lower costs, and more confidence. However, their technological complexity, lack of privacy safeguards, ineffective implementation, and potential for irreparable errors highlight the importance of careful planning and more extensive infrastructure improvements.

3.1.3 Use Cases

Despite the technological challenges mentioned before, the ability of smart contracts to privately and transparently automate trust-based interactions has led to their adoption across a variety of sectors [8, 23, 105]. The following use cases are found in areas such as finance, supply chain management, healthcare, energy, and others. These exemplify how the potential benefits of smart contracts are being implemented in practice while also exposing unaddressed domain-specific constraints.

Financial System

The financial sector is one of the most visible areas where smart contracts have demonstrated practical potential [8, 48, 105]. Smart contracts are being studied in the capital markets to improve efficiency and market liquidity by lowering settlement times [8, 48].

In addition to capital markets, the adoption of smart contracts can also help the mortgage loan business [8]. Smart contracts use digital documentation and rule-based execution to simplify and automate the mortgage process, from origination to maintenance [8]. This reduces errors, fraud, and delays [8].

In the insurance sector, smart contracts can improve transparency and speed up the resolution of claims [8,48,105]. One company that uses smart contracts to offer flight insurance is the French airline AXA [3, 111]. If a delay exceeds two hours, they employ smart contracts based on Ethereum to trigger automatic compensation [8, 48]. Furthermore, smart contracts can be used in trade finance and auto insurance, reducing the possibility of fraudulent claims and enabling real-time data access [48].

Susanto et al. [112] argue that smart contracts offer benefits for sophisticated financial processes, including capital market transactions, escrow agreements, loans, and inheritances. The authors do, however, raise attention to important technological constraints, including limited scalability, expensive implementation, and difficulties with compatibility with legacy systems.

Younu et al. [113] support this opinion by studying how blockchain technology and smart contracts can improve the efficiency, security, and interoperability of financial systems, especially in data sharing, international transactions, and supply chains. Nevertheless, the study also acknowledges challenges, such as cultural opposition inside banking institutions, high development costs, legal and standardisation concerns. Also, security considerations including data privacy protection and private key management remain as a major concern.

To improve the security and efficiency of decentralised financial systems, Prabanand et al. [114] offer a novel solution that combines deep learning models optimised with hybrid algorithms with smart contracts and blockchain networks. However, the authors point out challenges include excessive computational use, communication overwork, and robustness constraints in unusual conditions. Additional optimisations in terms of latency, traceability, and scalability are identified as vital for crucial applications in financial settings.

Internet of Things (IoT)

The integration of smart contracts with IoT infrastructure increases automation and confidence in M2M communication. Traditional IoT systems tend to be restricted by resources and centralised. According to [8, 48, 105], the smart contracts provide a decentralised paradigm that improves efficiency and scalability.

For example, smart contracts can minimise dependence on central servers in industrial manufacturing by automating firmware updates and enabling devices to verify update hashes stored on the blockchain [8]. Moreover, smart contracts support independent interactions such as service verification, access control, and payment processing between devices in smart cities, supply chains, and smart homes [48, 105].

With the use of smart contracts for transactions and decision-making processes, Dis-

tributed Autonomous Corporations (DACs) present an alternative to centralised control for IoT devices [8]. An example of this integration is IoTeX, which provides a blockchain-based platform for IoT applications that prioritise privacy [48, 115].

According to recent research, the integration of blockchain, IoT, and smart contracts enables increased efficiency, transparency, and security by enhancing visibility and trust throughout the supply chains [116]. IoT sensors linked to smart contracts have the ability to automatically record position, weather conditions, or transit events. This allows digital contracts to be triggered in order to verify shipment, apply taxes, or release payments.

However, most of research remains theoretical and lacks practical insights and validations in real world situations [116, 117]. Furthermore, challenges relating to implementation costs, compatibility concerns across heterogeneous systems, and organisational reluctance to transformation persist [117]. These drawbacks highlight the need to conduct further practical studies that may ensure technical integration, economic viability, and operational sustainability while converting technology potential into workable solutions for particular supply chain management environments [116, 117].

In addition to industrial applications, the integration of smart contracts with interactive ecosystems such as decentralised gaming demonstrates new potential within the IoT paradigm. Blockchain-based gaming systems employ smart contracts to provide automated activities, transparent reward distribution, and verifiable ownership of digital assets, consequently enhancing security and trust in human-machine interactions [52]. These applications reflect distributed environments where devices, digital identities, and tokens interact independently, although often being entertainment-oriented.

Blockchain-based solutions can be applied in educational and cultural contexts, encouraging user engagement in interactive networks in a safe and transparent way, as demonstrated by projects such as the "GENERA Web3 Game" [51]. However, these systems have the same challenges that other IoT systems: the demand for user-friendly interfaces, scaling challenges, and accessibility concerns [51, 52].

Healthcare System

Smart contracts help to improve healthcare data security, patient privacy, and overall system efficiency. Smart contracts can be triggered by wearable health devices based on patient data to update healthcare records, schedule appointments, or send out notifications [105, 118].

Patient data can be safely kept on a blockchain, where access is controlled by configurable terms in smart contracts, ensuring adherence to confidentiality regulations while offering data owners authority. This promotes digital health systems that are tamper-proof and interoperable.

Additionally, smart contracts can enhance data provenance and minimise the risk of data manipulation in clinical research [8]. Systems like as ProvChain contribute to maintain data integrity and confidence by tracking the origin, access, and transformation of research data in real-time [8, 118].

According to recent research, smart contracts have the potential to modernise clinical data integrity, access control, and electronic health record management [119–121]. Rejeb et al. [119] and Khezr et al. [120] state that although smart contracts show promise in

improving data security and patient-centered management, structural issues prevent their widespread adoption. Scalability problems, safety hazards, high consumption of energy, technological complexity and a lack of regulations are some of the concerns.

Khezr et al. [120] divides blockchain applications in healthcare into three categories: clinical data management, traceability in the pharmaceutical supply chain, and integration with connected medical devices in the context of the Internet of Medical Things (IoMT). The confidential transfer of clinical data is simplified by smart contracts in this context, which provide holders with authority over access permissions. However, since various jurisdictions have different laws, the study warns of the challenges in sharing health data across borders. Additionally, it highlights the performance constraints of blockchains, in particular latency and validation time in high transaction volume scenarios.

Vargas et al.'s study [121] explores the application of smart contracts to the IoMT in two distinct systems. To optimise transaction costs and data management security, the first system employs a fog-cloud architecture. The second solution uses blockchain technology and machine learning to identify cyberattacks and promote global patient data transfer. Although these models show different methods to data protection, they fall short in legal analysis.

Governance and Digital Rights

The potential of smart contracts to automate decision-making, increase transparency, and reduce dependence on intermediaries has been explored in digital governance contexts [8]. Applications such as digital identity management, DAOs, and decentralised voting systems are a few examples for traditional governance structures. However, there are relevant legal and social obstacles to the actual application of these models.

Through decentralised cryptographic techniques, e-voting systems and self-tallying protocols guarantee voter privacy and result integrity [8]. Initiatives for digital identity management, like Tsinghua University User Reputation System (TURS), provide users authority over their credentials and reputations on-chain while protecting their privacy using smart contract logic [8].

Similarly, smart contracts enable fair compensation and ownership protection in the creative industries [48, 105]. Through platforms such as Ujo Music, artists get paid directly when their work is used, eliminating intermediaries and delays [48, 122]. Additionally, the incorporation of digital watermarks into smart contracts enables legal action in cases of unauthorised distribution and the traceability of content usage.

Pelt et al. [56] presents a comprehensive model of blockchain governance with six dimensions: creation and context, roles, incentives, members, communication, and decision-making, implemented in three layers: off-chain community, off-chain development, and on-chain protocol. In blockchain ecosystems, the framework seeks to aid in the systematic evaluation of governance processes. However, some drawbacks such as specialist selection bias, a lack of established accountability processes, and ambiguous interpretation requirements were highlighted.

Zachariadis et al. [55] and Yusuf & Martinez [57] highlight how smart contracts are transforming institutional and legal relationships. The potential for automated, auditable, and trustless transactions poses significant concerns about control mechanisms, distributed

decision-making, and code vulnerabilities. Further limitations regard scalability, legislative ambiguity, deterioration of traditional legal principles, and incompatibility with legislation in different jurisdictions.

Kayode et al. [58] research how smart contracts can be used to enhance corporate governance by automating procedures including stakeholder interaction, voting, and compliance auditing. The study examines how blockchain-based platforms, such as DAOs and digital voting systems, can improve institutional and operational confidence. Drawbacks are highlighted, nevertheless, such as implementation costs, security vulnerabilities, and challenges integrating with present legislation.

According to [56–58], the incorporation of these technologies ought to be supported by flexible regulations, responsibility, and democratic control. The dilemma of oracles and the inflexible nature of automated contracts underscore the importance for hybrid models that combine rules codification with human mediation [57].

Apart from their applications in digital identity and electronic voting, smart contracts have also been used to improve transparency, trust, and fund traceability in charitable donation scenarios. Platforms proposed at [22, 54] use smart contracts and technologies such as InterPlanetary File System (IPFS) and Decentralised Digital Identifiers (DIDs) to record immutable transactions, automate fund transfers, and reduce dependence on intermediaries. These solutions aim to increase transparency, reduce fraud, and improve investor trust.

However, these attempts face technical constraints, such as high transaction costs on public networks, limitations with scalability, and the complexity of integrating with legacy systems [22, 53, 54]. According to Wu et al. [53], accessibility for non-technical users and the requirement for digital literacy are also major barriers to wider adoption, particularly by smaller organisations or in emergency situations.

Energy and Sustainability

Through decentralised P2P energy trading, electric vehicle charging systems, carbon emissions certification, running of virtual power plants, and smart grid automation, smart contracts encourage development in the energy industry [46–49].

Users can lease solar panels to communities through platforms such as The Sun Exchange, with blockchain technology automatically tracking payments and performance [48]. Smart contracts are also being integrated into electric vehicle charging networks to manage price bidding and selection of charging stations according to availability and proximity [48].

Wu et al. [46] state that energy transactions can be made more transparent and safe by leveraging blockchain technology in the Energy Internet, an architecture that integrates distributed agents with a variety of renewable energy sources. However, the study points out some drawbacks, including limited scalability in networks with a large number of nodes, sophisticated integration between digital and physical systems, a lack of appropriate legal structures, and an absence of specialists in sustainable energy and blockchain.

Similarly, Borkovcová et al. [47] present a comprehensive analysis of the developing role of blockchain in the energy industry, highlighting how it can help the objectives of decentralisation, decarbonisation and digitalisation. Private initiatives with minimal public sector involvement are driving applications such as active consumer participation and de-

centralised microgrids. However, the authors emphasise the lack of standardisation, the fragility of legal structures, and the immature nature of research, which is still isolated from academic communities with limited access.

Honari et al. [49] research how smart contracts influence smart grid performance, highlighting advantages such as demand response management, secure energy trading, and automation of auxiliary services. However, the study also identifies some weaknesses, such as duplicate data injection attacks, concerns about the compatibility of the blockchain platform, and privacy issues. Another obstacle to its practical use is the lack of integration with current regulations and legal restrictions.

In general, the studies examined suggest that smart contracts have the potential to impact a wide range of industries, including finance, health, energy, digital governance, supply chains, and charities, by automating processes, reducing intermediaries, and increasing transparency. However, they still face multiple challenges to practical adoption, such as concerns about scalability, compatibility, implementation costs, regulatory gaps, and organisational resistance. The consolidation of these solutions will be based on coordinated efforts in the development of technology, legal feasibility, and the advancement of hybrid models that combine automated efficiency with human flexibility.

3.2 United Extensible Firmware Interface (UEFI)

The Basic Input/Output System (BIOS) has been the most widely used standard for booting laptops since the 1980s [123]. This 16 bit firmware was responsible for loading the OS loader, executing a series of hardware authentication processes, and offering basic input and output interfaces [124]. Its position as an intermediary between the hardware and OS, as well as its wide adoption and standardisation over the years, made it a vital component of a laptop's architecture [123].

But as technology evolved, BIOS started to reveal several weaknesses. It was challenging to adapt to new technologies due to its inflexible architecture and its dependence on legacy code [123]. Furthermore, the 1MB address space constraint, the lack of native support for modern interfaces (such as USB at boot), and the lack of strong security measures were significant limitations [123]. These weaknesses presented concerns about the integrity and performance of the systems at a time when the goals for scalability, reliability, and security against firmware attacks were increasing.

As concerns about firmware weaknesses grew, it became clear that a new strategy was required. Concurrently, the increasing integration of technologies such as high capacity storage, cryptographic modules, and secure networks demanded a firmware platform that was not only flexible, but also robust and compatible with modern execution environments [125].

The UEFI emerged as an extension of Intel's Extensible Firmware Interface (EFI) initiative, originally launched in the 1990s [124, 126]. EFI attempted to address the technical constraints of BIOS by offering a flexible, expanding interface that could operate in 32-bit and 64-bit configurations [10, 124, 127]. The UEFI Forum, an independent group, received

control of the technology as the specification evolved and is currently responsible for its continuous improvement and standardisation [126, 128].

UEFI evolved from a technical concept to an open specification that is widely used by hardware manufacturers, operating system suppliers, and security businesses [9, 10, 123]. Nowadays, it is the most widely used standard in embedded devices, workstations, and servers. Its architecture enables constant flexibility to new demands, especially in areas such as IoT, DaaS, and trusted computing, in addition to notable performance and compatibility improvements [123, 129].

Given this progress, it is helpful to examine the UEFI architecture, its cryptographic and network abilities, and potential for its future application for novel distributed security scenarios.

3.2.1 UEFI Architecture and Functionality

UEFI established a modular architecture based on standardised drivers and services, allowing pre-boot, boot, and runtime services to be logically decoupled. This functional separation enables secure integration with the OS and peripheral devices, while improving the manageability and scalability of the boot process.

The UEFI System Table[2] is a data structure that serves as a point of reference between the firmware and the rest of the system, and it is at the core of this architecture [10]. This table contains pointers to the primary services and protocols that permit UEFI applications and the OS to interact with the firmware in a controlled way [10]. It introduces two sets of features: (1) boot services[3], which provide support during the boot process, and (2) runtime services[4], which remain accessible after control goes to OS, ensuring, for example, access to persistent variables or the system time [10, 127].

The Pre-EFI Initialisation (PEI)[5] module manages the pre-boot phase and is in responsible for performing the fundamental system configuration tasks such as memory detection and Central Processing Unit (CPU) initialisation [9, 124]. After, the Driver Execution Environment (DXE)[6] phase loads the drivers required to connect to the devices. DXE phase has significance in the UEFI architecture, as it enables the dynamic loading of firmware modules, such as disk controllers or network interfaces, independently of OS, promoting flexibility and adaptability to new technologies [9, 124].

A feature of UEFI is the support for the GUID Partition Table (GPT)[7], extending the previous limits on the volume and number of supported partitions, as well as boot environments with graphical interfaces, support for networking and running native applications before loading the OS [127].

Another advance of UEFI is its compatibility with multiple processor architectures, including `x86`, `x64`, `ARM` and `AArch64` [130]. An abstraction layer between the firmware and the hardware itself enabled this feature, enabling reuse of the same source code base across multiple platforms. This method reduces the time required for development and makes it

[2] `https://uefi.org/specs/PI/1.8/V2_UEFI_System_Table.html`
[3] `https://uefi.org/specs/UEFI/2.11/07_Services_Boot_Services.html`
[4] `https://uefi.org/specs/UEFI/2.11/08_Services_Runtime_Services.html`
[5] `https://uefi.org/specs/PI/1.8/V1_Overview.html`
[6] `https://uefi.org/specs/PI/1.8/V2_Overview.html`
[7] `https://uefi.org/specs/UEFI/2.11/05_GUID_Partition_Table_Format.html`

easier to maintain heterogeneous systems. Tools such as EDK II[8] support this modular logic and allow the development of secure, lightweight, and reliable firmware [124, 130].

The modularity of UEFI extends beyond the development stage. Once implemented, this architecture allows for the integration of extensions via drivers, protocols, or specific applications that can be introduced in conformity with the consumer's demands [10]. For example, this covers support for new storage controllers, remote management systems, or specialised hardware interfaces. This flexibility ensures that the UEFI can be upgraded and adapted to new technological conditions without re-engineering the current firmware.

Table 3.2 outlines the main components of the UEFI architecture and their functional roles during the boot process, offering a comprehensive perspective of the modularity and flexibility of the firmware environment.

Table 3.2: Components of the UEFI architecture and their roles in the boot process.

Component	Description
UEFI System Table	A data structure that aggregates pointers to protocols, available services, and system configuration tables. It allows OS loaders, other firmware modules, and programs to have structured access to UEFI functionality.
Boot Services	A set of features that are accessible during the boot process, such as memory management, device identification, driver loading, and image execution. These services are stopped when control is transferred to the OS.
Runtime Services	A subset of features that are accessible prior to and after the boot process, providing platform-level functionality required for OS performance, such as persistent firmware variables and system time.
PEI	The firmware phase responsible for basic hardware initialisation, including CPU, chipset, and temporary memory setup. Additionally, it creates the PEI Foundation and transfers control to the DXE phase.
DXE	The firmware phase responsible for creating the execution environment's structure after PEI. The DXE Foundation is loaded first, followed by the loading of modular drivers and protocols to initialise the platform hardware. This provides dynamic and scalable platform setup that is independent of the OS.
GPT	An advanced partitioning system that allows for more partitions, redundancy features, and storage capacities larger than 2TB. It is required for UEFI-compliant booting and replaces the outdated Master Boot Record (MBR) format.
UEFI Drivers	Independent modules that manage hardware resources during the pre-boot environment. These drivers enable the usage of a variety of gadgets, including storage, USB, graphics, and network interfaces, prior to the loading of the OS.

[8]The EFI Development Kit II (EDK II) is an open-source firmware development environment maintained by the TianoCore community. It supports UEFI-compliant firmware development across multiple architectures, providing a modular and reusable codebase [130].

UEFI protocols	Standardised software interfaces that define how components that specify the communication between components in the UEFI. These protocols allow drivers, services, and applications to interact in a modular and consistent way, abstracting low-level hardware details and enabling flexibility.
UEFI programs	Executable programs that run in the UEFI environment before the OS loads. They are usually kept on the EFI System Partition and accessed by the UEFI Boot Manager. Examples of these include OS bootloaders, firmware setup tools, diagnostics, or update tools.

In addition to the internal modular structure described above, the UEFI architecture can also be represented through a layered abstraction model, which illustrates the sequential positioning of hardware, firmware, and software elements during the system's boot sequence. This conceptual model is shown in Figure 3.2.

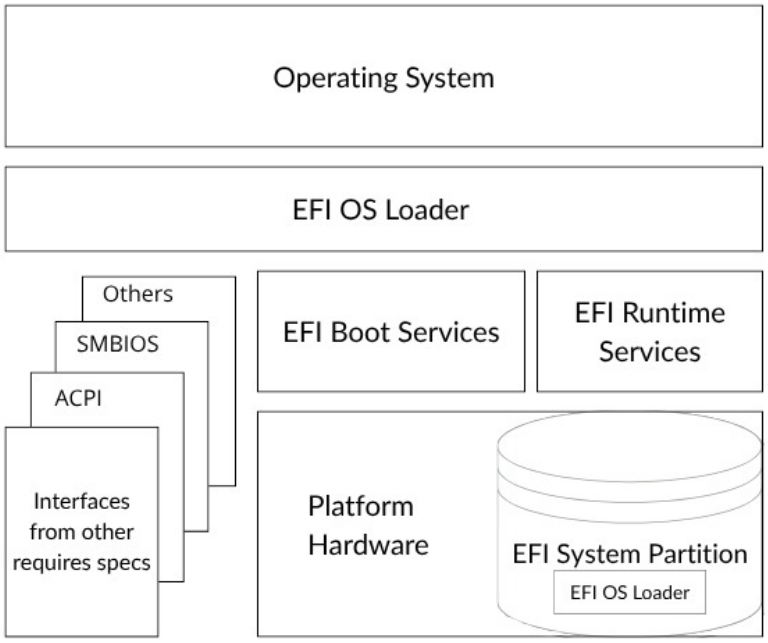


Figure 3.2: Layered abstraction model of UEFI architecture. Source: [9, 10].

Although it does not represent communication flows, the layered diagram clarifies the logical arrangement of responsibilities across the platform. Each layer encapsulates a specific role in platform initialisation, from low-level configuration to OS execution, with dependencies on external specifications. Table 3.3 complements this visual structure with a functional description of each layer, highlighting its relevance to the compatibility, performance, and flexibility of the platform.

Table 3.3: Functional roles of each layer in the UEFI architecture.

Layer	Description
Platform Hardware	The physical components of the system, such as the CPU, main memory, chipset, buses, controllers, and peripherals. It serves as the foundation for firmware operations and is the focus of the abstraction pushed by the upper layers of UEFI.
EFI System Partition	A standardised disk partition, often in FAT32 format, is where the OS loader, diagnostic tools, UEFI applications, and other relevant tools are stored. During boot, the firmware loads and manages it.
EFI OS Loader	An executable application that starts the transition between the firmware environment and the OS. It is responsible for loading the kernel, supplying inputs, and customising the system's state via UEFI services. The boot variables specify where it will be located and how it will run.
External Interfaces	Sets of defined specifications that enable the firmware to supply the OS with structured platform information. For example, SMBIOS and ACPI define system topology, device setup, and power management characteristics.
Operating System (OS)	After the boot procedure is complete, this software component takes full control of the system. It manages resources, interacts with hardware, and executes user applications using the data and structures that the prior levels provided. The firmware provides information that the OS uses to ensure stability and compatibility.

3.2.2 Support for Cryptography and Networking

Incorporation of cryptographic features and network protocols in UEFI has proven vital to ensuring the integrity, confidentiality, and authenticity of the system components prior to loading of OS. UEFI is indispensable for building a trusting execution base from the initial phase of system initialisation in current systems, particularly if they require distributed trust structures, safe upgrades, and remote leasing.

UEFI supports `IPv4` and `IPv6`, allowing connectivity during the pre-boot phase for essential operations such as dynamic Internet Protocol (IP) configuration, domain name resolution, and remote resource access [9]. The use of protocols such as `EfiDhcp4Protocol`, `EfiDns4Protocol`, and `EfiHttpProtocol` enables Hypertext Transfer Protocol (HTTP) connectivity, dynamic network initialisation, and the retrieval or validation of boot images from trustworthy servers [131].

This feature plays a role in scenarios where the system relies on remote validation techniques or network booting (Preboot Execution Environment (PXE) over UEFI), where system components need to be validated against remote trust anchors prior to OS transfer [132]. Furthermore, UEFI's support for Domain Name System (DNS) and HTTP-based services enables smooth interaction with cloud-based systems and decentralised device provisioning.

In the pre-OS setting, the `EfiHttpProtocol` offers secured channels of communication when paired with Hypertext Transfer Protocol Secure (HTTPS) ability [131]. Due to

this, UEFI firmware can safely receive updates, validate digital signatures, or carry out remote attestation without exposing executable software or sensitive metadata at risk of tampering.

By establishing secure communication channels before OS assumes control, UEFI decreases the attack surface of the system and aids in the prevention of man-in-the-middle or downgrade attacks, especially in scenarios involving firmware updates or remote diagnostics [133]. In this instance, the application of Transport Layer Security (TLS)-based protocols ensures data integrity and confidentiality during transmission [133].

A wide diversity of cryptographic techniques and protocols are incorporated into UEFI to enable secure firmware and boot component validation, authentication, and confidentiality [134]. This includes support for symmetric and asymmetric cryptographic algorithms, which are used for operations such as digital signature validation, certificate validation, and secure key exchange [134].

These skills offer essential features such as validation of image signatures (e.g., OS loaders or drivers), using signed access policies for variable security, key management and secure storage using UEFI settings or embedded Trusted Platform Modules (TPMs) [134].

These cryptographic operations are actively used in Secure Boot workflows, where the digital signatures of bootloaders and drivers are verified against a trusted certificate database stored in UEFI variables. Additionally, platforms like Microsoft Windows use the key infrastructure UEFI for kernel-mode driver enforcement and BitLocker integration, depending on measured boot support from embedded TPMs [135].

UEFI platforms often employ Public Key Infrastructure (PKI) to establish trust chains between the manufacturer of the platform, the firmware components, and OS. This strengthens the defences of the system against firmware-level attacks by ensuring that only trustworthy software can be executed.

To achieve a balance between security and computing performance, UEFI applications often employ lightweight encryption in resource-constrained scenarios, such as embedded systems or OS devices [134, 136]. Cryptographic operations can be performed with low performance and memory usage if adopting algorithms design for restricted settings, such as those detailed in Table 3.4.

Libraries, such as `CryptoPkg`, offer a modular and portable cryptographic layer that abstracts underlying solutions in the EDK II tool [130]. Additionally, when available, these libraries can be customised for hardware acceleration, offering a versatile framework for safe firmware development on a spectrum of hardware platforms [130].

Table 3.4: Lightweight cryptographic algorithms for UEFI.

Algorithm	Type	Application in UEFI
Advanced Encryption Standard (AES)	Symmetric	Encrypts stored variables, protects boot data, and provides memory protection. `CryptoPkg` library supports AES-128 and AES-256 variants, as well as hardware acceleration.

Lightweight Encryption Algorithm (LEA)	Symmetric	Designed to provide efficient encryption for low-resource embedded systems. It can be used to preserve boot-time data or UEFI configuration parameters while consuming few resources.
Rivest Shamir Adleman (RSA)	Asymmetric	Used in signature verification during Secure Boot and key exchange operations. It is required to validate UEFI drivers and bootloaders using PKI-based trust chains.
Elliptic Curve Cryptography (ECC)	Asymmetric	Provides stronger security with smaller key sizes than RSA. Commonly used for digital signatures, particularly on modern devices with TPM 2.0 potential.

UEFI is anticipated to apply post-quantum cryptography fundamentals and more dynamic trust policies through real-time TPM-backed evaluations in response to emerging legal demands and novel means of attack [137]. This development ensures that firmware will continue to be a fundamental component of secure, scalable computer systems.

3.2.3 Flexibility and Upcoming Potentials

The UEFI architecture is designed to evolve. Its modular layered nature allows it to include emerging technologies and security paradigms without requiring extensive changes to the system firmware. Due to its adaptability, UEFI is suitable for current computer environments that are characterised by distributed trust models, diversity, and an increasing need for self-verifying and autonomous systems [123, 129].

As previously stated, a key component of this flexibility is splitting between protocol-level services and hardware-specific initialisation [10]. UEFI enables the continuous implementation of new features, such as authentication techniques, remote attestation methods, or support for decentralised key management, without demanding a firmware core recompile through the application of driver models and protocol interfaces [10].

In terms of security, this flexibility proves essential for integrating cryptographic algorithms, blockchain-based verification approaches, or attestation solutions that use secure enclaves or remote TPMs [123]. As an essential aspect of source of trust of the system, UEFI is being positioned not only as a bootloader but also as an instrument that can interact with external validation services and implement restrictions that extend beyond basic secure boot.

Projects such as LinuxBoot [138], which replaces DXE drivers with a Linux kernel, or TrenchBoot [139], which enables dynamic root of trust establishment via late launch mechanisms, demonstrate how the UEFI environment can be extended and strengthened. Similarly, UEFI capsule updates allow remote firmware upgrades via authenticated HTTPS, supporting over-the-air secure updates in modern infrastructure [9, 133].

Moreover, as autonomous and adaptive boot environments become more prevalent, UEFI could evolve into an intelligent policy engine that responds to real-time security monitoring, enforces legal requirements during boot, or even rolls back to trusted firmware

images when an anomaly is detected. Future secure computing systems are anticipated to be based around UEFI given its capacity to communicate with remote servers, validate system health, and enforce conditional access decisions, all before the OS is fully operational.

Given this, the place of UEFI as a platform that is prepared for the future is highlighted by how its modularity performs in parallel with technologies such as dynamic enforcement of laws, decentralised trust infrastructures, and lightweight cryptography [129, 140]. Far from being limited to boot-time processes, UEFI has the potential to act as a persistent, flexible, and intelligent layer in the overall system security paradigm.

Chapter 4

Related Work

The DaaS paradigm represents a change in the way computing resources are acquired and managed. A DaaS refers to an environment in which services and devices are interconnected, with devices acting as abstract units that can be remotely monitored and managed through a service platform [19,141]. This approach integrates sensors, actuators, and electronic controllers connected to a centralised monitoring system, enabling them to carry out tasks and provide measurement data [19].

DaaS is gaining popularity among businesses, as it allows them to subscribe to a service that provides equipment rather than purchasing hardware outright, resulting in lower capital spending, faster upgrades, and smoother IT logistics [141, 142]. These services often also include lifecycle services, remote management, and integrated maintenance [141].

Although DaaS is gaining ground in industry, it remains underexplored in academic research. Barrangán-Gil et al. [19] describe DaaS as a paradigm in which physical devices are managed via networked services, notably in local or home control systems. However, their work is limited to interoperability and abstraction in controlled circumstances, without making reference to broader concerns such as data integrity, remote supervision, or global scalability. In addition, it fails to address security, trust and auditing, within DaaS applications.

This need emerged during the COVID-19 pandemic, when governments launched initiatives to provide students with devices to enable access to digital education. The Portuguese Government, for example, provided students with leased computers during lockdowns [16–18]. These programmes followed a DaaS model but faced some challenges such as theft, unauthorised use, and a lack of ways to monitor or revoke access in (near) real-time.

The economic and operational modelling of DaaS also poses challenges. Hruby & Scheller [143] explore this topic by comparing two frameworks to model device leasing ecosystems: Resources, Events, Agents (REA) and Possession, Ownership, Availability (POA). POA provides a straightforward framework for depicting possession and control, but the REA model is notable for its semantic depth and ability to describe economic events and agent interactions. Both have weaknesses when it comes to replicating ownership and possession flows, for example, or when there are numerous agents and concurrent transactions.

Maree et al. [141] explores the use of Digital Twins (DTs) in the management of devices inside the DaaS platforms. As dynamic and self-evolving digital models, DTs provides continuous supply, maintenance planning, and device monitoring consistent with the DaaS lifecycles. The study emphasises the improved interoperability and efficiency of the platforms as well as their flexibility. However, there are concerns about the integration of scalability, validation, and security in distributed environments.

In a DaaS model, security is a factor. System for remotely interacting and operating devices, makes them more vulnerable to tampering or attack. Current centralised management solutions in academic environments rely on user authentication and network trust [19]. However, as Sookhak et al. [23] point out, these measures are insufficient to protect the infrastructure from external threats, particularly in public networks such as those present in academic institutions.

The adoption of decentralised technology, such as blockchain, presents a feasible response to such challenges. Blockchain technology ensures transparency, immutability, and resistance against attacks by offering a permanent and auditable record of all transactions. Smart contract integration can automate and enforce security regulations without the involvement of a centralised manager, as evidenced by studies on blockchain in IoT and access control [144–146].

Sultana et al. [146] study the potential of smart contracts to offer trustworthy, distributed access control. Their research reveals that blockchain, when combined with smart contracts, may overcome the trust issue in device management. Still, there is a demand for near real-time access control and revocation solutions, which are essential in DaaS models. A benefit, particularly when there are many participants involved, is blockchain's potential to produce an immutable transparent log of each device interaction.

Furthermore, decentralised auditing of device activities in addition to smart contract access control ensures that actions such as locking, unlocking, and denial are clearly stored and resistant to tampering [145]. This addresses the challenges of accountability and transparency that may arise in centralised systems.

Several studies [144–148] have explored blockchain-based concepts for enforcing access control and auditability in distributed systems. These studies demonstrate the advantages and challenges of incorporating blockchain technology into DaaS applications.

Maesa et al. [144] present an Ethereum-based access control architecture that transforms Extensible Access Control Markup Language (XACML) rules into immutables smart contracts. Their architecture includes components such as a Policy Enforcement Point (PEP) and a off-chain handler, which provide safe, transparent, and auditable access rule enforcement. The system raises concerns regarding transaction costs, latency, and implementation complexity even as it shows increased confidence and transparency.

In [145], the authors refined the previous model [144] by integrating contract customisation strategies to reduce gas consumption and enhance scalability. They also address privacy concerns with encryption techniques and the deployment of private blockchains, demonstrating the viability of blockchain for data-sensitive scenarios, including DaaS.

A blockchain-extended access control system for IPFS-based file sharing, has been proposed by Steichen et al. [147]. This approach combines Ethereum's contract-based permissions with IPFS distributed storage abilities, resulting in secure, selective, and auditable

data sharing. Their approach is applicable to DaaS as it supports secure data access from distributed devices, although with higher latency and dependency on public blockchain networks.

A multi-level access control framework for the IoT is designed by [146]. Their solution offers flexible smooth device control that is in line with DaaS demands through the use of dynamic permissions and behaviour monitoring. Compared to other approaches, their cost evaluations show increased efficiency. But, this approach has the same flaws with privacy, complexity, and blockchain dependency.

Moreover, Cruz et al. [148] suggest an Ethereum smart contract-based RBAC system. It is appropriate for multi-tenant DaaS configurations since it provides dynamic role assignment and verification without the need for intermediaries. The solution provides scalable and secure delegation of permissions across organisational boundaries. However, like the previous one, the constraints of public blockchains and privacy and regulatory issues are still unresolved.

Another area of study [149, 150] focusses on blockchain-based approaches to firmware integrity, version control, and secure software distribution, all of which are key parts of device lifecycle management in DaaS.

A blockchain-based technique to perform trustworthy firmware updates in IoT contexts is proposed by Lee et al. [149]. Their approach leverages a P2P network, digital signatures, and secure hashing to distribute and validate the firmware on its own. By ensuring the integrity and authenticity of the version, blockchain technology eliminates the need for centralised update servers. Although the decentralised structure of the scheme increases resilience, it has concerns about latency, device resource limitations, and the complexity of blockchain connectivity.

Baza et al. [150] provide a decentralised firmware update system for autonomous vehicles that uses blockchain, Attribute Based Encryption (ABE), and Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) to protect sensitive data and authenticate upgrades. This architecture supports multi-manufacturer ecosystems and incorporates smart contract-based reputation and reward mechanisms to encourage engagement. These characteristics highlight service-orientated approaches, especially in distributed environments that demand high levels of trust and security. However, there are limitations with scalability and implementation due to computational costs and dependence on cryptographic techniques.

To manage and safeguard firmware upgrades for smart inverters used in photovoltaic systems, Bere et al. [151] apply the Hyperledger Fabric platform in the renewable energy industry. Their permissioned blockchain approach uses smart contracts to handle firmware validation, patching, and rollbacks. The framework stores an immutable record and stores firmware hashes to enhance security and ensure continuous operations. Despite being an example of service-like management that is compatible with DaaS principles, the system has only been tested in controlled environments, and further research must be conducted to determine how effectively it performs in different environments.

Choi et al. [140] present a further solution in which they establish a decentralised architecture that combines IPFS and the blockchain to safely deliver firmware upgrades. Their solution uses immutable blockchain hashes to validate and monitor firmware images. This approach addresses maintenance limitations and promotes long-term device autonomy by

eliminating the need for the manufacturer during the update process. Despite its strength, the architecture poses new infrastructure demands and scalability challenges in public blockchain deployments.

Firmware-level security is the subject of another relevant research field [152–154], namely related to the UEFI. These studies address weaknesses in the early boot stages as well as secure boot methods and provide insights on protecting the device boot process, which is an essential feature when devices are viewed as remotely managed services, although they are not specifically focused on DaaS.

Chen et al. [154] conducted research on vulnerabilities in UEFI firmware for power monitoring devices, employing fuzzing techniques, static analysis, and optimisation strategies such as simulated annealing[1]. By combining these techniques, 44 vulnerabilities were found in 12 firmware modules and validated against national vulnerability databases. Although there are drawbacks, such as false positives and the need for additional automation, the scalability of the solution and practical validation are its strongest features. This work emphasises how important automated validation and continuous monitoring are in DaaS scenarios.

The different phases of UEFI (such as PEI, DXE, among others) are analysed by [152], who also emphasises the potential vulnerabilities in these stages, such as driver replacement, bootkits, and rootkits. In addition, the article highlights the importance of secure key management and secure boot as key mitigating strategies. It does not target DaaS, but as it encourages protection from boot up to normal system operation, the principles apply to device management in this model.

Ayedh et al. [153] examine security and privacy in Bring Your Own Device (BYOD) scenarios and provide a strategy that extends beyond the traditional Confidentiality, Integrity, Availability (CIA) model to incorporate transparency, confidentiality, and trust. Even though it focusses on personal devices, the approach is feasible for DaaS, for example in cases when users, businesses, or departments share devices. Some technologies that can improve secure remote management in DaaS include the use of Machine Learning (ML) and cryptography [153].

All of these research findings emphasise how devices are treated as dynamic service endpoints that need to be managed carefully, flexible, and transparently throughout their lifecycle instead of as distinct hardware components. The introduction of blockchain technology improves trust, decentralisation, and resilience in these systems. However, the adoption of these technologies remains constrained by several barriers, including latency, computer resources, the operational complexity of monitoring decentralised networks, and the complexity of cryptographic techniques. Addressing these challenges remains a critical limitation for DaaS to reach its full potential.

Table 4.1 offers a structured comparison of the related work, highlighting the importance of this study. This study outlines the distinct contributions of the proposed solution and places it within the current research environment.

[1]Simulated annealing is an optimisation approach based on the physical process of thermal tempering. The premise is to allow the local optimal solution to be escaped by admitting suboptimal solutions with a specific probability at the start of execution. This technique is successful for combinatorial optimisation problems with a large and irregular set of solutions, such as the selection of seeds in fuzzing [154].

Table 4.1: Comparison of research on DaaS, access control, and firmware security.

Authors	Focus Area	Application Domain	Overview
Barrangán-Gil et al. [19]	DaaS in smart environments	Smart homes and domotic systems	The study explores an abstraction paradigm for device management using service oriented architectures, focusing on gateways, network nodes, and communication protocols to enable device interoperability and integration inside smart home systems. Limited to closed environments, lacks discussion on scalability, trust, or security.
Hruby & Scheller [143]	Economic modeling of DaaS using REA and POA frameworks	Device leasing ecosystems	Presents an economic model that uses the REA and POA frameworks to describe DaaS ecosystems, covering agent roles, resource flows, and transaction modeling challenges, with an emphasis on consistency and simplicity over traditional systems. Lacks effectiveness in complex, multi-agent environments.
Maree et al. [141]	Digital Twins for IoT Device Management	IoT device management	The study explores the potential to embrace DTs to improve the provisioning, management and monitoring of IoT devices, focussing on their lifecycle management, hierarchical structure and integration frameworks. Scalability and security integration are limited.
Steichen et al. [147]	Blockchain-based decentralised access control	IPFS and distributed file systems	Presents a blockchain-enabled access control system for IPFS and evaluates its effectiveness and influence on file sharing activities. It addresses access control and blockchain storage options, emphasizing how smart contracts can be integrated with IPFS to improve security and decentralisation. Suitable to DaaS, but performance depends on public blockchain latency and availability.
Maesa et al. [144, 145]	Blockchain-based decentralised access control	IoT data management	With a reference implementation using XACML and Solidity, it presents an approach for access control by smart contracts on blockchain allowing for distributed policy evaluation and attribute manager maintenance via smart contracts. Provides transparency but raises gas cost, latency, and complexity concerns.

Cruz et al. [148]	Smart Contract-based RBAC	Multi-tenant DaaS	Provide an Ethereum-based smart contract RBAC framework, to enhance secure role management and verification across businesses, including performance evaluation and prototype deployment. Scalable for organisations, but lacks privacy guarantees on public blockchains.
Sultana et al. [146]	Blockchain-based access control and data sharing	IoT and decentralised networks	Proposes a blockchain-integrated system with smart contracts for secure access control and data sharing across IoT devices, highlighting trustworthiness, authentication, and cost efficiency.
Bashun et al. [152]	UEFI security vulnerabilities and threats	Firmware security	The paper discusses technical benefits and security procedures of UEFI, describing potential vulnerabilities and attacks. Supports secure key handling and early boot protections.
Lee et al. [149]	Blockchain-based secure firmware update	Embedded devices in IoT environments	Proposes a blockchain-based strategy for secure firmware version verification, validation, and distribution, addressing security and scalability concerns in IoT firmware updates. Latency and resource demands are factors to keep in mind.
Baza et al. [150]	Blockchain-based secure firmware update	Autonomous vehicles and automotive industry	Proposes a blockchain-based system for secure, scalable firmware upgrades in autonomous vehicles, leveraging multi-authority attribute-based encryption, smart contracts, and zero-knowledge proofs to provide secure distribution and verification of updates without requiring confidence among autonomous vehicles. High computational and cryptographic complexity.
Choi et al. [140]	Blockchain-based firmware update	IoT firmware updates	Propose a blockchain-based distributed firmware update architecture to handle security challenges in IoT devices, thus improving the reliability and safety of firmware delivery. Public blockchain raises scalability and infrastructure costs.

Bere et al. [151]	Blockchain-based firmware security and management	Renewable energy systems	Introduces a Hyperledger Fabric architecture that uses smart contracts to improve firmware integrity, automate security checks, and enable secure recovery, as proven by real-time status updates and experimental validation. Effectiveness in a variety of situations has not been demonstrated.
Ayedh et al. [153]	Trust and risk management models	BYOD and shared devices	Provides an overview of BYOD compliance theories without delving into specific privacy technologies. Encourages trust, transparency, and privacy when using personal devices. It suggests cryptography and machine learning algorithms.
Chen et al. [154]	Vulnerability discovery in devices	IoT devices	The impact of Industry 4.0 on production technology is discussed, with a focus on vulnerabilities in power system firmware and the proposal of fuzzy testing methodologies to detect security flaws. Relevant to remote trust in DaaS.
(This work)	Blockchain-assisted DaaS with UEFI firmware integration	DaaS environments	A layered architecture is proposed, combining Ethereum smart contracts with UEFI firmware. Provides decentralised device control, immutable logs, lifecycle tracking, and real-time auditing. Even before the operating system boots, the device is checked for conformity. Highly relevant for distributed, trust-critical DaaS environments.

Comparative research shows that while previous works address the maintenance or access control of secure firmware in distributed systems, none of them fully integrate firmware-level security, dynamic access control, and remote management in a DaaS model with multiple players. The approach described in this dissertation distinguishes itself by integrating these components into a cohesive system.

It uses blockchain for immutability and auditability, smart contracts for the enforcement of decentralised access policies, and UEFI validation to ensure the integrity of the device from the moment it starts. This unified approach addresses significant gaps in the literature, especially in large-scale deployments such as educational device distribution initiatives.

The suggested approach presents a novel and viable solution to DaaS management by integrating operational, security, and lifecycle concerns into an integrated structure.

Chapter 5

DaaS Solution

This chapter outlines the specifications that are necessary for implementing the blockchain-based remote equipment monitoring and control solution into practice, as well as the architecture of the proposed solution. The purpose is to explain the system's components, how they work together, the technological decisions that were taken, and the requirements that the solution must satisfy.

The architecture of the system was designed to incorporate smart contracts, which automate device control, provide immutability and transparency in transactions, and allow remote management by multiple individuals, notably contractors and suppliers. In order to achieve the goal, the system was designed based on clear and prioritised requirements, which will be discussed in the following section.

5.1 Requirements

The goal of this project is to develop a novel approach to remote equipment management employing blockchain technology. Establishing requirements in priority order helps to ensure that development is focused on the features that will be most useful to the project and its stakeholders. In order to do this, the MoSCoW technique (see Appendix A.1) was selected, a prioritisation technique that is widely used in software projects and that proved to be particularly useful for the project's circumstances.

The MoSCoW approach offers for an organised and strategic perspective of the key aspects of the solution by classifying requirements into four prioritisation categories. The first category, **must have**, stands for the essential components required for the basic functionality of the system. These requirements ensure that the project satisfies its essential standards and that the solution would not be able to achieve its main goal without them.

The second category, **should have**, includes features that, although not necessarily required for basic performance, enhance the project and should be integrated whenever feasible. These requirements aim to enhance the solution's functionality, making it more extensive and advantageous for the different users and stakeholders.

Furthermore, there are **could have** requirements for extra features. Such requirements are considered as lesser priority even if they could increase the solution's breadth and diversity and enhance the user experience. They can be delayed without compromising

the primary goals of the system, but their achievement is conditional on the availability of time and resources.

Lastly, the **won't have** category describes features that, despite their potential attraction, will not be given priority at the current stage in the project. Future iterations of the solution could take these features into account if there is an excellent reason to increase the scope and available resources.

The potential of the MoSCoW approach to clarify the definition of priorities, align with stakeholder expectations, and focus on deliverables that enhance the value of the project justifies its application in this case. The flexibility of this technique, which allows improvements throughout development, is particularly pertinent to initiatives requiring challenging and novel technologies, such as the adoption of blockchain for equipment management.

The MoSCoW matrix is shown in Figure 5.1. This matrix offers an organised list of the characteristics that have been identified as necessary, desired, and voluntary for the successful deployment of the proposed system. Additionally, the matrix supports the clear visualisation of priorities and guides development, ensuring that efforts focus on the demands that will be most beneficial to stakeholders. In the following sections, each category of requirements will be discussed in detail.

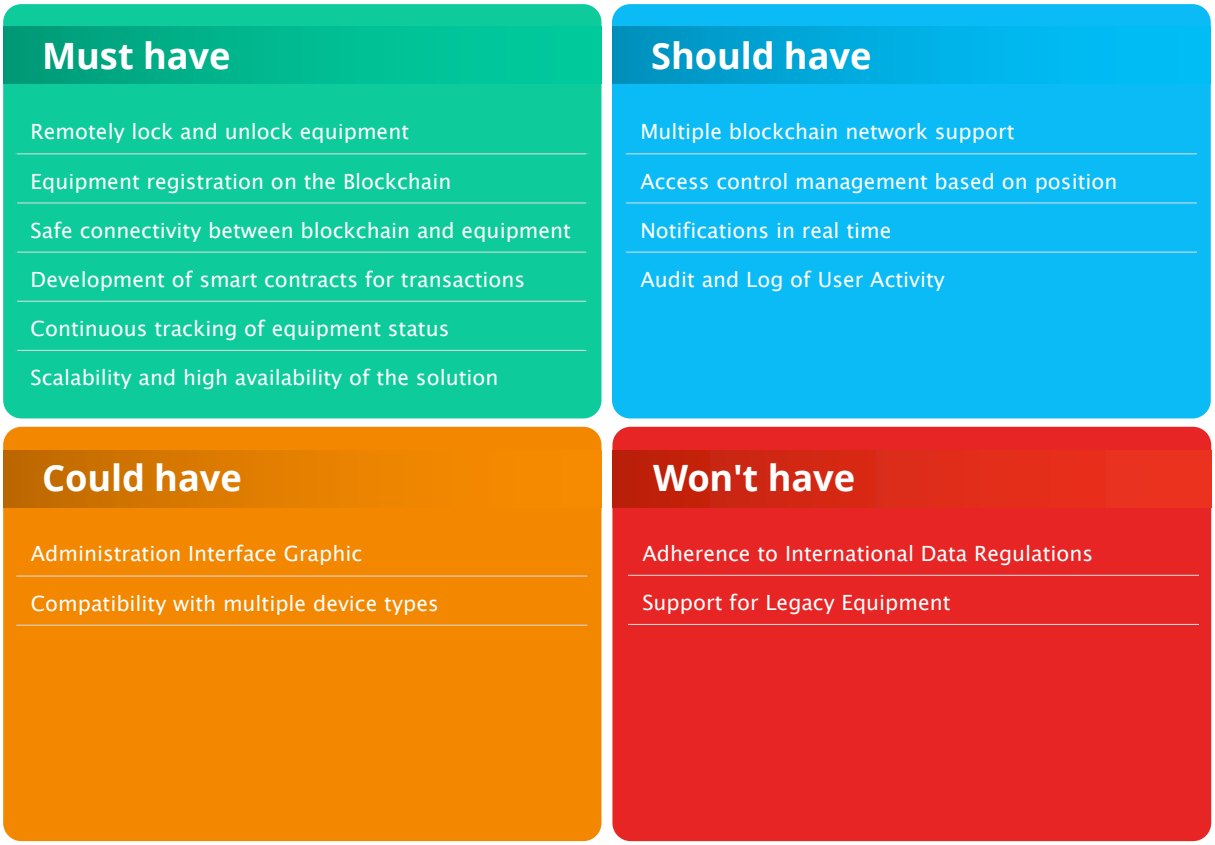


Figure 5.1: MoSCoW Prioritisation Matrix applied to blockchain-based remote equipment management system.

5.1.1 Must Have Requirements

For the system to be effective, some features must be implemented. They are seen to be essential for achieving the main goals of the project and ensuring the bare minimum of functionality required for the solution.

REQ-001 Remotely lock and unlock equipment
The solution must have robust capabilities to remotely lock and unlock devices using smart contracts on blockchain. This ensures security and control over the equipment even while it is being locked or unlocked remotely by requiring that all interactions be recorded in an immutable and transparent way.

REQ-002 Equipment registration on the blockchain
Each unit of equipment must be registered on the blockchain, establishing an immutable digital identity that centralises configuration and usage data. This record will enable transparent and verified monitoring of all operations performed on the equipment.

REQ-003 Safe connectivity between the blockchain and the equipment
The equipment and the blockchain network must communicate securely and encrypted to prevent unauthorised access and malicious alteration of the data being provided. This ensures the system's reliability and ensures data integrity.

REQ-004 Development of smart contracts for transactions
To record all transactions and interaction between parties, smart contracts must be developed. Smart contract automation minimises manual involvement and strengthens user confidence by ensuring transparency, efficiency, and integrity in transactions.

REQ-005 Continuous tracking of equipment status
Each device's locked or unlocked state must be constantly monitored by a specific module in the UEFI firmware. This continuous monitoring is required to ensure that the equipment corresponds to the blockchain record and enables early detection and rectification of inconsistencies.

REQ-006 Scalability and high availability of the solution
Scalability and high availability must be ensured by the system's architecture in order to support an increasing number of devices and transactions without compromising performance. The system must be able to handle a large number of transactions simultaneously, as well as an expanding fleet of operational devices.

5.1.2 Should Have Requirements

Although not mandatory, these requirements are encouraged to strengthen the system and increase its utility for all parties involved. They are a valuable addition to the project and should be implemented wherever possible.

REQ-007 Multiple blockchain network support
Along with Ethereum, the system should be able to operate on multiple blockchain networks, enhancing future integration flexibility and ensuring that the structure is flexible to different situations and implementation requirements.

REQ-008 RBAC management

The system should allow the management of access and permissions according to different levels of user responsibility, especially suppliers and contractors. RBAC management provides a layer of security and simplifies task assignment based on responsibilities.

REQ-009 Notifications in real time

Real time alerts should be sent by the system to users to improve event management and user experience by notifying them about significant changes in device status or important incidents such as access attempts or configuration changes.

REQ-010 Audit and log of user activity

A thorough record of all actions that users perform on the DaaS platform should be maintained by the system. This includes smart contract execution and equipment status updates, enabling thorough audits and enhancing operational transparency.

5.1.3 Could Have Requirements

Desired requirements that potentially expand the system's potential fall under this category. Even if they are not equally essential, bringing them into practice can enhance the solution's performance and user experience.

REQ-011 Graphical User Interface (GUI)

To make it simpler to maintain and keep an eye on the devices and transactions recorded on the blockchain, a GUI could be developed for administrators. The system would be easier to use with this interface, enabling efficient management and a user-friendly interface.

REQ-012 Compatibility with multiple device types

A wider range of devices, including smartphones and vending machines, could be integrated and managed by the system. This requirement would increase the scope and applicability of the solution in many industries.

5.1.4 Won't Have Requirements

This section details the requirements that have been recognised but will not be implemented in the project due to time, resource, or strategic boundaries. However, when the solution expands, these requirements could be taken into account for subsequent iterations.

REQ-013 Adherence to internacional data regulamentations

Despite its long-term value, the adherence to international data protection regulations is outside the scope of this project. Later, after the system is operating consistently, the requirement might be reevaluated.

REQ-014 Support for legacy equipment

Compatibility with outdated devices that don't support modern features, such as UEFI networking, will not be considered. Supporting legacy equipment would require costly adaptations that are currently unfeasible but might be reviewed in future versions of the system.

5.2 Architecture

The blockchain-based architecture of the proposed system, which is depicted in Figure 5.2, has been designed to ensure the efficacy of remote equipment monitoring and control. In fulfilment of the established requirements, the architecture was organised in layers and connected modules to provide security, traceability, and the ability to remotely control devices.

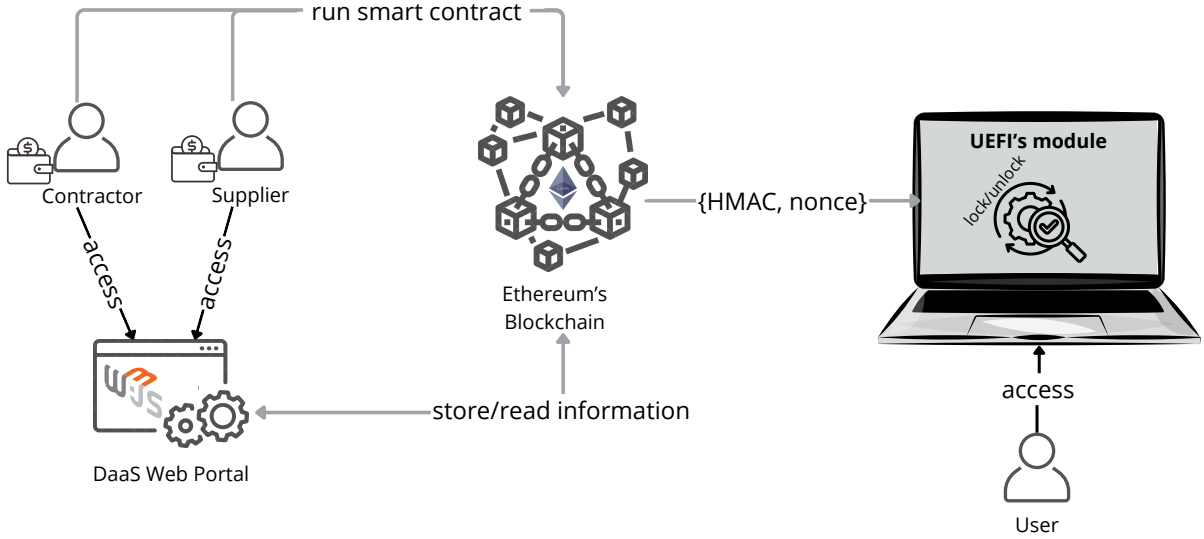


Figure 5.2: Blockchain-based proposed architecture.

The Blockchain, which ensures the system's security and transparency, is the core of the solution. Here, all device interactions are automated and recorded through the use of smart contracts. A smart contract stores all of the device's data, including its identifier number and current state, on the network when a device is registered.

These contracts ensure that all transactions are transparent and permanent, allowing for remote equipment management, especially the ability to lock and unlock devices as authorised. The Supplier and the Contractor are the two main entities that interact with the contracts; they have the authority to establish the contracts to register new devices and modify the lock status of the equipment, respectively.

In addition, the blockchain is responsible for preserving and validating each transaction that occurs within the system. Since each operation is documented, any changes to the device's status are auditable and immutable. The Ethereum network was selected for its resilience and effective smart contract support.

The Verifier module is implemented in the UEFI firmware component at the device's base. This module behaves as a state machine that communicates with the blockchain network to manage the device status instructions before the operating system boots up and to verify the device's current status.

In conformity with registered smart contracts, this module performs a monitoring task, ensuring that only allowed devices are able to boot up. The firmware will prevent a blocked device from initialising. This module ensures a high level of security for field operations by running independently of the operating system.

The solution also includes a DaaS Web Portal that the main entities can access, providing a user experience for system interaction. Authorised users can use this interface to manage their daily activities, including buying equipment and maintaining commercial contracts, as well as to view the current status of all registered devices and perform actions such as locking or unlocking them.

Furthermore, the interface supports auditing and log visualisation features, allowing the generation of detailed reports on the performed transactions. This increases transparency and control over the system's operations by making it simpler to keep track of past interactions with the devices.

Through the centralisation of essential functions in a single smart contract that is currently running on the blockchain, communication between the many layers of the system is intended to enhance the process of remotely monitoring and managing equipment. The lock and unlock of the device serves as an example of how communication flows between layers. The blockchain's responsible smart contract receives a request to lock or unlock a device from an approved user, such as a supplier or contractor, via the web portal.

When triggered, the contract records the state change, enabling any transaction to be instantly recorded in the device's history. The UEFI firmware of the device synchronously verifies and adopts the new state by continuously reading the blockchain. The solution eliminates the need for manual inspections and prevents any divergence by synchronising the device's state with the immutable status recorded on the blockchain. This ensures that only devices that have been effectively unlocked can work.

The design of the system architecture prioritised security, document immutability, and scalability to accommodate multiple devices. Blockchain technology improves operational security and transparency since each transaction is permanently stored, resulting in an audit history that is trustworthy and impenetrable to fraud. This fosters an environment of rigour and trust by enabling suppliers and contractors to keep up-to-date on activities in real time.

In terms of scalability, the system was designed to accommodate a growing number of smart contract interactions and devices without sacrificing performance. Since each new device is simply registered to the current contract, the blockchain-based architecture eliminates the need to deploy new contracts, making it easy to incorporate new devices into the system. As new devices are added, the system can increase its monitoring and control features while preserving data integrity and efficiency.

5.3 Device Management Smart Contract

The smart contract developed for the proposed system plays a role in ensuring security, transparency, and automation within the blockchain-based architecture. Implemented on the Ethereum network, written in Solidity and designed with modularity to enable maintenance and scalability, the smart contract is responsible for managing device registration, state control, and access permissions.

In the proposed system, the smart contract uses data structures to manage device-related data. The primary data structure used in the system is shown in pseudocode in the Listing 5.1. The `Device` data structure is in the role of modelling a device and contains several of important pieces of information, including the timestamp of the device registration within

the DaaS system (`registrationTimestamp`), the User's address (`user`), the Contractor's address (`contractor`), an authorisation token (`token`), a `lease` validity, and the device's lock (`isLocked`) and registration (`isRegistered`) status.

All device data is stored in the `devices` mapping and is indexed by a unique device identifier. This mapping makes it easier to retrieve the data and to assess the lock status of the device. Furthermore, the Supplier address (`supplier`) is stored for smart contract usage control.

```
1  struct Device {
2      bool isRegistered;
3      bool isLocked;
4      address user;
5      address contractor;
6      uint registrationTimestamp;
7      string token;
8      string lease;
9  }
10
11 address public supplier;
12 mapping(string => Device) public devices;
```

Listing 5.1: A model of the Device data structure and its key characteristics.

The system specifies two events for the purpose of providing transparency and enabling auditing of the operations performed on the devices. These events, which occur at specific times throughout the functionality of the system, are described in Table 5.1.

Every time a new device is successfully registered in the system, the first event is released, called `DeviceRegistered`. This action establishes a record of the device's registration, which means that the DaaS platform will begin monitoring the device immediately thereafter.

Table 5.1: A model of smart contract events and their properties.

Designation	Parameters	Trigger
DeviceRegistered	Device Identifier, User Address, Contractor Address	On device registration
DeviceStatusChanged	Device Identifier, Locked status, Cryptographic Hash	On lock status change

Anytime a device's lock status is modified either by the Supplier or the Contractor, the second event `DeviceStatusChanged`, is triggered. The purpose of this event is to log status change activities so that they may be analysed later, ensuring transparency in device management.

In addition to events, the system also uses modifiers to control how smart contracts are performed. The modifiers used in this proposed system to control access to contract performance are shown in Table 5.2. The purpose of these modifiers is to ensure that only authorised entities are able to carry out specific assignments.

Table 5.2: A model of smart contract modifiers and their properties.

Designation	Parameters	Purpose
OnlySupplier	None	Restrict smart contract execution to the Supplier
OnlySupplierOrContractor	Device Identifier	Restrict smart contract execution to the Supplier or Contractor

Since the Supplier is the only entity allowed to register new devices in the DaaS system, the first modifier, `OnlySupplier`, guarantees that the smart contract will only be executed by them. In contrast, the second modifier, `OnlySupplierOrContractor`, grants permission to both entities, the Supplier and the Contractor, to execute the terms of the contract.

The process of registering new devices is carried out by the `register device` function (see Listing 5.2). This smart contract requires three parameters as input: the device identifier, the User address, and the Contractor address. To ensure the security and integrity of the system, only the Supplier has permission to register new devices.

```
1 function registerDevice(string memory _id, address _userAddress, address _contractorAddress
      ) public onlySupplier {
2     require(!devices[_id].isRegistered, "Device already registered.");
3     devices[_id] = Device(true, false, _userAddress, _contractorAddress, block.timestamp, "", "");
4     emit DeviceRegistered(_id, _userAddress, _contractorAddress);
5 }
```

Listing 5.2: A model of the device register function and its key characteristics.

The device register function, as displayed in the code above, builds a new entry into the data structure, in a scenario in which the device is not yet registered in the system, comprising its registered state (true), its lock state (false), the User's address, the Contractor's address, and a timestamp. After that, the `DeviceRegistered` event records the operation by releasing details about the registered device, ensuring operational traceability and transparency.

The device is initially set up as unlocked upon registration. As stated in Listing 5.3, to change the lock state of a device, the `change device status` function must be executed. This function receives as parameters the device identifier number, the latest lock status, the lease validity and the authorisation token.

In the decentralised system, the authorisation token (`token`) is crucial for safeguarding and validating communications between individuals. It is created with an HMAC, which ensures the integrity of the data being sent by using a secret key. This `token` ensures that malicious agents haven't changed the data related to the device's state shift while it was being transmitted. Furthermore, the token confirms the request's origin, assuring that only authorised users can alter the device's status through the Web Portal. Thus, by keeping confidential details away the network, it also preserves the confidentiality of sensitive data.

On the other hand, the `lease` validity plays a role in repelling replay attacks, which pose hazards to decentralised systems. The lease prevents a previously recorded action from being illegally copied or reused by assigning a distinct temporary value to every activity. This value must be recalculated for every shift in the device's state, ensuring that every

transaction is unique and immutable. In addition to strengthening security by rendering all activities permanent, this approach ensures that all interactions are transparent and tracked, sustaining system transparency and trust. When combined, the authorisation `token` and `lease` validity offer a strong layer of security that ensures that any modifications applied to the device's state are legitimate, safe, and unique.

```
1 function changeDeviceStatus(string memory _id, bool _locked, string memory _lease, string
      memory _token) public onlySupplierOrContractor(_id) {
2     require(devices[_id].isRegistered, "Device is not registered.");
3     devices[_id].isLocked = _locked;
4     devices[_id].lease = _lease;
5     devices[_id].token = _token;
6     emit DeviceStatusChanged(_id, _locked, _hmacHash);
7 }
```

Listing 5.3: A model of the device change status function and its key characteristics.

The above method employs the use of the `onlySupplierOrContractor` modifier, which limits access to the function's execution and ensures that only the Contractor or Supplier has the ability to alter the device's status. The `DeviceStatusChanged` event is released at the conclusion of execution, documenting the device status shift and ensuring that the system keeps a transparent history of the shifts applied.

The smart contract includes the `check device status` function so that anyone has access to the lock status of a specified device. The check device status function, as displayed in Listing 5.4, allows to any interested party to determine if the device is locked or unlocked by returning the lease validity and the authorisation token.

This feature is essential when allowing transparent real-time device state querying, ensuring that each party can audit the current state without requiring more interaction with the device management entities.

```
1 function checkDeviceStatus(string memory _id) public view returns (string memory, string
      memory) {
2     require(devices[_id].isRegistered, "Device is not registered.");
3     return (devices[_id].lease, devices[_id].token);
4 }
```

Listing 5.4: A model of the device check status function and its key characteristics.

Our design takes advantage of this data extracted from the smart contract to safely validate the device's state through the Verifier module incorporated into the UEFI. By recalculating the HMAC and comparing it with the received token, this component should be able to recreate the validation message. When the values match, the data's integrity is verified, and the device's status (locked or unlocked) is safely verified. This approach ensures trustworthy system-to-system collaboration and prevents the likelihood of information tampering.

Chapter 6

Validation and Results

This chapter examines the validation process of the smart contract developed for the device management system. A critical first step in ensuring the smart contract's efficiency, security, and effectiveness before deploying it in a real network is validation. To achieve this, a security assessment of the code was carried out, accompanied by the execution of various types of tests.

The chapter begins by describing how the solution was evaluated and validated to ensure compliance with established requirements, as outlined in Section 5.1. Following this, the practical results of the solution are presented, with an emphasis on the findings obtained, including visual components that illustrate the success of the solution. Finally, the conclusions derived from the tests carried out are discussed.

6.1 Validation

Using blockchain technology to provide security, transparency and automation, the development of the smart contract and the adoption of the system were envisioned to completely meet the requirements specified in Section 5.1.

The core functionality of the system is the ability to remotely lock and unlock the equipment (REQ-001). This is achieved through smart contract transactions that store the lock status immutably on the blockchain. To ensure security and transparent management, the contract can be triggered by an entity with the required authorisations to alter the device state.

To ensure that each equipment is uniquely identifiable and registered on the blockchain (REQ-002), the smart contract includes a device registration function (Listing 5.2). In addition, a unique identifier number is linked to each registered device, making tracking straightforward.

Cryptographic protocols are used to protect communication between the blockchain and the equipment (REQ-003). An HMAC-based validation mechanism is vital to each device interaction as a way to prevent unauthorised data manipulation. Furthermore, lease legitimacy criteria is used to prevent replay attacks and ensure that only valid transactions are carried out.

The smart contract handles all interactions with the equipment (REQ-004). For instance,

the Contractor can either contact the Supplier or submit a lock instruction on his own if he desires to lock an equipment. By introducing smart contracts, the proposed solution aims to decrease the number of processes and inspections that people must handle.

Querying the smart contract for the current device status supports continuous equipment status tracking (REQ-005). The blockchain logs any changes made to a device's lock state, enabling the system to keep an accurate verified history of all operations.

Scalability and high availability (REQ-006) are inherent to the chosen blockchain infrastructure. The decentralised nature of the blockchain ensures that the system remains operational even if some nodes fail. Furthermore, the smart contract was designed to manage several concurrent transactions, allowing the management of an increasing number of devices without experiencing a decline in performance.

The design of the system encourages future improvements, such as supporting various blockchain networks (REQ-007). Although development and testing were conducted on Ethereum's Sepolia testnet, the contract structure can be easily adapted to be implemented on other networks with minor adjustments.

The smart contract's assignment of particular rights, often referred to as modifiers (Table 5.2), addresses RBAC (REQ-008). The contract distinguishes users with different duties (Contractors and Suppliers) by verifying their addresses prior to granting them access to specific features.

Due to the asynchronous nature of blockchain transactions, real-time notifications (REQ-009) were taken into account but not included in the smart contract. To alert users to relevant events, such as status changes or unauthorised access attempts, DaaS Web Portal might use external monitoring technologies.

All user actions are fully audited by the system (REQ-010), as each transaction is permanently stored on the blockchain. This ensures total transparency and accountability, enabling any interested party to confirm previous interactions.

Some requirements were considered outside the scope of the implemented prototype. However, the system was designed to support some of these features in future versions. Although it was not created as part of the core implementation, a GUI (REQ-011) could have been included to improve usability. Device management would be simplified through a Web3JS front-end dashboard that would enable user interaction with the smart contract.

Similarly, although laptops were the main category of devices for which the system was designed, it is possible to expand compatibility to include mobile devices, as well as vending machines (REQ-012). Since there are no hardware restrictions imposed by the smart contract design, additional firmware implementations could allow a wider adoption.

However, there are some requirements that the proposed solution does not support. For example, due to the inherent challenges that blockchain presents with respect to international data protection and other laws, adherence to international data standards (REQ-013) was not explicitly addressed. Furthermore, the dependence on UEFI networking firmware, which is unavailable from prior models, led to the disregarded support for legacy equipment (REQ-014).

In general, the application of blockchain technology and smart contracts in the presented solution successfully meets the core requirements. The decentralised structure of the

system ensures automation, security, and immutability, meeting the goals outlined in the initial concept of the project.

6.2 Results

Following validation, with the specified requirements met, the smart contract was subjected to a set of tests to evaluate the efficiency, security and performance of the solution. This section presents the results from the tests that were conducted, with a focus on the metrics that were examined, the data that was collected, and the proof that the implementation was successful.

A public Ethereum testnet, Sepolia[1], was used to conduct the smart contract testing in a controlled setting. Due to its similarity to the real network, Sepolia was selected to ensure the tests could be performed under more production-like configurations. The Remix IDE[2], a platform for developing and deploying smart contracts, was used to deploy the smart contract. During the testing, transactions and results were tracked with tools like Etherscan[3] to verify the contract's state and finished transactions.

6.2.1 Test Scenarios

The tests performed to validate the smart contract are detailed in the subsequent sections, in combination with the inputs, expected outputs, and actual outputs. All of the input parameters and values used can be found in Appendix B, which also serves as an overview of all the values used in the tests to prevent misunderstanding. Each test scenario was designed to assess a specific smart contract feature and its agreement with the defined requirements.

Test Case 1: Device Registration

The aim of this test is to prove that the smart contract executes as expected through the device registration process on the blockchain. In particular, when a new device is inserted into the system, it is expected that (1) it will be registered on the blockchain through the device identifier number provided, the user, and the contractor address associated; (2) an event will be emitted confirming the device's registration; and (3) the device's related data will be accurately stored in the mapping devices.

To execute the test, the register device function was invoked (see Listing 5.2). Three parameters must be provided as input: device identification (`ID1`), the user address (`U1`), and the contractor address (`C1`), as was previously stated in Section 5.3. The relevant details of the transaction output can be seen in Table 6.1.

Table 6.1: Output details of the device registration transaction.

Field	Value
status	`0x1 Transaction mined and execution succeed`

[1] `https://rpc.info/ethereum-sepolia`
[2] `https://remix.ethereum.org`
[3] `https://sepolia.etherscan.io/`

transaction hash	0x66c1e13688a5a6b52038d5dc885ada42e311b5847d1f955227efadb26903a069
block hash	0xc5b736aeeb7b46074f1f36184623fcde746d13d37c075db24b6794634a710e3c
block number	201
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	DeviceManagement.registerDevice(string,address,address) 0xE682b91c33B76918BDD0143267E28B64BE1093E9
gas	118726 gas
transaction cost	103240 gas
execution cost	80472 gas
decoded input	{ "string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433", "address _userAddress": "0 x583031D1113aD414F02576BD6afaBfb302140225", "address _contractorAddress": "0 xCA35b7d915458EF540aDe6068dFe2F44E8fa733c" }
logs	[{ "from": "0xE682b91c33B76918BDD0143267E28B64BE1093E9", "topic": "0 xfd9debad2fafb11807eb091e1e920dd86a1eb845e94bd015ae2cc1e0890e2137", "event": "DeviceRegistered", "args": { "0": "520cc938-c90b-4a76-b81f-1d3b11f6d433", "1": "0x583031D1113aD414F02576BD6afaBfb302140225 ", "2": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c ", "id": "520cc938-c90b-4a76-b81f-1d3b11f6d433", "user": "0 x583031D1113aD414F02576BD6afaBfb302140225", "contractor": "0 xCA35b7d915458EF540aDe6068dFe2F44E8fa733c" } }]

The transaction was successfully mined and executed, as evidenced by the `status` field, meaning that the device registration process was carried out without errors. The `block hash` gives an identifier of the block in which the transaction was included, letting evidence that the transaction was registered in a specific block (`block number`), thereby promoting the traceability of the transaction within the distributed system of the blockchain. The `transaction hash` is a unique identifier that allows the transaction to be tracked inside the blockchain.

The address of the account (`Supplier`) that triggered the transaction is represented by the `from` field, and the address of the smart contract that carried out the device's registration

logic is indicated by the `to` field. To ensure the process's integrity, it is important to verify that the transaction was delivered and received by the right addresses.

The amount of computational power needed to mine and process the transaction is reflected in the cost of `gas`. The total cost of the transaction, including the quantity of gas used for operations apart from the registration function, is referred to as the `transaction cost`. The `execution cost`, which indicates the process's efficiency, is the cost directly associated with executing the registration function in the smart contract.

We can conclude from this output that the test was successful as the device was correctly recorded on the blockchain. The function's successful execution was evidenced by the `Device Registered` event that was released (`logs`). The data was appropriately recorded, according to an analysis of the transaction details, including the transaction hash and event logs.

Test Case 2: Alter Device Status

This test was carried out to ensure that the system could accurately update the status of a device that was previously registered on the blockchain and to validate that it was able to alter the status of a device's block. In particular, when an existing device's status is changed, it is expected that (1) the device's status is changed to the new state upon request; (2) an event will be emitted confirming the device's status change; and (3) the contract should appropriately reflect the updated data, including the new lock status.

This process was initiated by calling the change device status function (see Listing 5.3), supplying the device identification (`ID1`), new lock status (`S1`), lease validity (`Lease`), and authorisation token (`H1`) as inputs. These parameters have the aim preserving the operation's immutability and security.

Table 6.2 summarises the outcomes of this execution. The successful recording of the transaction reflected the device's updated status. The `status` field of the transaction returned success, indicating that the transaction was mined successfully once more.

Table 6.2: Output details of the device status change transaction.

Field	Value
status	`0x1 Transaction mined and execution succeed`
transaction hash	`0x5f933481168e243beb8cd58d58e149bafb19730f7211e78f09e19f183419b807`
block hash	`0x1415ba2a70cd3e5edf58c58a3fab8b3fe708d8f0f528392a2290853f864c2b89`
block number	`202`
from	`0x5B38Da6a701c568545dCfcB03FcB875f56beddC4`
to	`DeviceManagement.changeDeviceStatus(string,bool,string,string) 0xE682b91c33B76918BDD0143267E28B64BE1093E9`
gas	`148710 gas`
transaction cost	`129313 gas`
execution cost	`105297 gas`

decoded input	{
	"string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433",
	"bool _locked": true,
	"string _lease": "1619885200",
	"string _token":
	"eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d20805d4e324f20
	633fdd109"
	}

logs	[
	{
	"from":
	"0xE682b91c33B76918BDD0143267E28B64BE1093E9",
	"topic":
	"0xb143203a4d248516d8adea8b157e3b2e7df5afe81167469cbfa61
	ab82db3ab6c",
	"event": "DeviceStatusChanged",
	"args": {
	"0": "520cc938-c90b-4a76-b81f-1
	d3b11f6d433",
	"1": true,
	"2":
	"eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d20805d4e32
	4f20633fdd109",
	"id": "520cc938-c90b-4a76-b81f-1
	d3b11f6d433",
	"isLocked": true,
	"token":
	"eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d20805d4e32
	4f20633fdd109"
	}
	}
	]

Furthermore, after the function finished running, the `Device Status Changed` event was released. This event not only proved that the device's status had changed, but also provided all the information required to monitor this shift, such as the device identifier, the new status (`true` for locked), and the corresponding authorisation token.

The computational cost is another crucial consideration. A total of `129313 gas` was needed for the transaction, and the execution cost (`105297 gas`) demonstrated how effective the status update process was. Operations that require updated information and event broadcasts are suitable for this gas value.

The test was successful and the system demonstrated effectiveness in reliably and transparently recording and reporting the device's state change, according to evidence gathered.

Test Case 3: Querying Device Status

This test was conducted to ensure that the system behaves properly when a device's status is queried. The goal was to ensure that anyone who wanted to know a device's lock status could do it without having to engage in more involved interactions, such as communicating directly with the Supplier.

The device's identifier (`ID1`) was supplied as a parameter to the `check device status`

function (see Listing 5.4). The current lock status of the device was monitored using this identification. It is expected that the authorisation token and the lease's validity would be returned for increased security. This will make it more difficult to modify the data and allow the lock status to be verified directly on the device afterward.

The query output is shown in Table 6.3. Since the query operation simply returns data and does not alter the contract state, the transaction was successful and no further logs were produced and there was no associated cost.

Table 6.3: Output details of the device status check transaction.

Field	Value
from	`0x5B38Da6a701c568545dCfcB03FcB875f56beddC4`
to	`DeviceManagement.checkDeviceStatus(string)` `0xE682b91c33B76918BDD0143267E28B64BE1093E9`

decoded input	`{`
	`"string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433"`
	`}`

decoded output	`{`
	`"0": "string: 1619885200",`
	`"1": "string:`
	`eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d20805d4e32`
	`4f20633fdd109"`
	`}`

The validity of the `Lease` and the authorisation token (`H1`), two crucial components that ensure the integrity and security of the query and confirm the authenticity of the received data, are reflected in the `decoded output` data.

Even though there were no events in this test, the lack of logs also shows that the query was executed successfully, minimising processing costs and maintaining transparency by not requiring events to be recorded on the blockchain.

We can conclude from the findings that the system performs correctly, enabling efficient device status calls with trustworthy and safe data, as expected. This test further verifies the accuracy and integrity of the data supplied by the system and acts as a continuation of Test Case 2.

Test Case 4: Cryptographic Security

With an emphasis on the decision of the hash calculation model, the goal of this test was to analyse cryptographic security in the device management process. As pointed out in Section 5.3, the cryptographic security of the project is ensured by employing HMAC. The following data are used to construct the hash value: device serial number (SN), device identifier (ID), nonce, and state, in the format of *SN:ID:nonce:state*.

The DaaS Web Portal is the one that performs the hash calculation in the current scenario, however, an alternate scenario was also tested, in which this operation would be performed directly on the contract itself, as illustrated in the Appendix C. The initial scenario was selected due to a variety of technical considerations and challenges experienced while

performing these duties directly on the smart contract, but also by introducing additional costs.

In the alternate scenario, it is assumed that the serial number will be known only to the device and the Web Portal. Consequently, the smart contract would need to get external data to be able to calculate the hash. As provided in Appendix C, the smart contract takes advantage of an Oracle Chainlink to obtain external data, namely, the serial number of a device, to facilitate this connection.

The Chainlink platform is a decentralised network of oracles that allows smart contracts on the blockchain to interact with external data sources, such as Application Programming Interfaces (APIs), without relying on a single point of failure [155]. In this instance, Oracle is used to retrieve the serial number of a specific device by querying an external API. The Chainlink network acts as an intermediary between the contract and the Oracle, ensuring that the data provided is accurate and immutable on the blockchain.

Some essential variables were configured to successfully implement the solution, such as the `Link Token` address (Appendix C, line 36), which is used to pay for requests made to Oracle, and the `Oracle` address (Appendix C, line 37), which identifies Oracle inside Chainlink network.

The function responsible for requesting the serial number of the device is shown in Listing 6.7. This method builds a request for the Chainlink Oracle (line 79), which then sends a GET request to an external API endpoint (line 86). The Beeceptor[4] dummy service, which simulates the connection to the Web Portal and manipulates the endpoint used to retrieve the serial number, has the API Uniform Resource Locator (URL) set up (line 80). The device identifier is used to submit the request, and the contract privately preserves the serial number when Oracle provides it (line 90).

```
78  function requestSerialNumber(string memory _id) public {
79      Chainlink.Request memory req = _buildChainlinkRequest('7
            d80a6386ef543a3abb52817f6707e3b', address(this), this.fulfill.selector);
80      string memory url = string.concat('https://webportaldaas.free.beeceptor.com/', _id);
81
82      req._add('get', url);
83      req._add('path', 'serial_number');
84
85      uint fee = (1 * LINK_DIVISIBILITY) / 10;
86      _sendChainlinkRequest(req, fee);
87  }
88
89  function fulfill(bytes32 _requestId, string memory _serialNumber) external
        recordChainlinkFulfillment(_requestId) {
90      serialNumber = _serialNumber;
91      requestProcessed = true;
92  }
```

Listing 6.7: A model of the device serial number request function and its key characteristics extracted from Appendix C.

In the alternate scenario, the smart contract's primary functions, such as `register device` and `change device status`, remain straightforward, registering a device and just updating the lock status value to preserve system clarity. However, the `check device`

[4]https://beeceptor.com/

`status` method has been modified to rely on getting the serial number through Oracle, which adds a layer of complexity to its execution.

It is very important to note that the `request processed` variable (line 91) ensures that there is a brief delay between the time the serial number is requested and the moment the data is available, which is necessary to ensure that the interactions between the smart contract and the Oracle are executed correctly. To ensure that the Oracle answer has been processed prior to the serial number being used in the HMAC calculation, this control is crucial.

The `check device status` function, shown in Listing 6.8, ensures that the device is registered and that the external request has been processed correctly before proceeding with the hash calculation. If these conditions are met, the function generates a *nonce* value in Universally Unique Identifier (UUID) format and uses it together with the serial number and device identifier to calculate the hash.

```
60  function checkDeviceStatus(string memory _id) public view returns (bool, string memory,
        bytes32) {
61          require(devices[_id].isRegistered, "Device is not registered.");
62          require(requestProcessed, "The request has not yet been processed. Try again soon.");
63
64          string memory nonce = this.generateUUIDformat();
65          bytes32 hmac = this.generateHMAC(serialNumber, _id, nonce, devices[_id].isLocked);
66
67          return (devices[_id].isLocked, nonce, hmac);
68      }
```

Listing 6.8: A model of the device check status function and its key characteristics extracted from Appendix C.

After providing a quick overview of the alternate smart contract, let's discuss the details of the outcomes. The `request serial number` function, which is expected to retrieve external data using the Chainlink oracle, was first tested (see Listing 6.7). The request was successful, according to the transaction analysis, which is shown in Table 6.4. At a cost of `134679 gas` units, the contract was executed efficiently. This amount of gas reflects the time required to complete a request to Oracle, including the time necessary to connect to the external API and verify the information returned.

The transaction log confirms that the request was sent to Oracle confirming that the `Chainlink Requested` event was triggered. This log is important since it confirms that the request to get the device's serial number was sent and that the interaction with Chainlink was successful. Along with the request identifier (`args`), the event indicates that the smart contract has successfully initiated the serial number retrieval process, which is a prerequisite for further verifying the device's status.

Table 6.4: Output details of the request device serial number function in the alternate smart contract version.

Field	Values
status	`0x1 Transaction mined and execution succeed`
transaction hash	`0x3751cb281cf89d211d90416d7e3015861f8f4844b99b155262d1db451aff95a8`

to	DeviceManagement.requestSerialNumber(string) 0x0ff88a088e1c5eaa6067e7e6b77447f6a45cd53c
gas	136915 gas
transaction cost	134679 gas

decoded input	{
	"string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433"
	}

logs	[
	{
	"from": "0
	x0ff88a088e1c5eaa6067e7e6b77447f6a45cd53c",
	"topic":
	"0xb5e6e01e79f91267dc17b4e6314d5d4d03593d2ceee0fbb452b75
	0bd70ea5af9",
	"event": "ChainlinkRequested",
	"args": {
	"0":
	"0xe2a9c205867970b85fbd90f86e0361a0bc94b391f4154dae6
	9d8c468246bb1d9",
	"id":
	"0xe2a9c205867970b85fbd90f86e0361a0bc94b391f4154dae6
	9d8c468246bb1d9"
	}
	}
	]

The next step was to validate the `check device status` functionality (see Listing 6.8). As already stated in Test Case 3, there is no related fee since it is a query to the blockchain and does not alter its status. An accurate and straightforward summary of the transaction outputs for both versions, which return the status of the device using slightly different strategies, is given in Table 6.5.

The transaction was executed successfully in both versions. Nevertheless, we have two distinct outputs for the same input. As anticipated in the actual solution, where the HMAC computation is carried out directly by the Web Portal, the main version's transaction output consists of a string value that represents the timestamp of a date (`Lease`) and a string value that corresponds to the HMAC generated and inserted in the change device status function (`token`).

The outcome of the transaction is more complicated in the alternate scenario, since the process has been adjusted to include the use of the Chainlink Oracle to access external data. A boolean value was the first result returned reflecting the device's lock status (true for locked or false for unlocked). The second value was a string with the nonce produced (Listing 6.8, line 64) and the third value was a `bytes32` value reflecting the HMAC calculation (Listing 6.8, line 65).

Table 6.5: Comparison of transaction outputs for the device status check function in the main and alternate smart contract versions.

Field	Atual Solution	Secondary Version
status	0x1 Transaction mined and execution succeed	
to	0xE682b91c33B76918BDD0143267E28B64BE1093E9	0x0ff88a088e1c5eaa6067e7e6b77447f6a45cd53c
decoded input	{ "string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433" }	{ "string _id": "520cc938-c90b-4a76-b81f-1d3b11f6d433"
decoded output	{ "0": "string: 1619885200", "1": "string: eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d20805d4e324f20633fdd109" }	{ "0": "bool: true", "1": "string: 001ee3b438f61f001d6a32211a270036", "2": "bytes32: 0x d6d23cbc6b580d831cb7c a7ddec689421b97a4f77b1f04ea40fdaacfcbb26039" }

In light of these assessments, the complexity and costs related to data processing within the blockchain are the principal motives for avoiding hash computation directly in the contract. The execution costs in terms of gas are increased when cryptographic functions are computed directly in the contract. This is particularly true on a blockchain like Ethereum, where transaction costs are directly influenced by the volume and complexity of the operations performed. Placing these computations on the DaaS Web Portal makes the process more cost-effective and efficient.

Another drawback of employing Chainlink Oracles is that it relies on outside sources, which could pose security and availability challenges. Despite being a trustworthy and decentralised platform, Chainlink is susceptible to external glitches that can affect system performance, such as API unavailability or concerns with Oracle's network [156]. The gas costs related to queries sent to the Oracle should be taken into account. In addition to the gas costs required to execute the transaction on the blockchain, LINK tokens are used as a payment for using Oracle, and the amount required for each request fluctuates based on the network and difficulty of the job.

6.2.2 Overview

The efficiency of the evaluation was demonstrated by the successful execution of essential activities such device registration, device status change, and device status query. Furthermore, the cryptographic security tests proved the resilience of the system by using HMAC to ensure the authenticity and integrity of the data being submitted. As an alternative scenario, communication with Oracles was also successfully examined, showing the contract's reliability in integrating external information.

Despite the effectiveness of individual components and the transparency of the processes, scalability and concurrency, in the smart contract's ability to handle multiple simultaneous interactions and the blockchain's capacity to process a growing volume of transactions, become vital elements that ensure system sustainability in real-world environments, become important topics when analysing the behaviour of the system in a high-demand setting.

Ethereum was chosen as the platform for the smart contract's development and execution due to its decentralisation, security, and blockchain network maturity. However, scalability and competitiveness are significant challenges for Ethereum. As a public blockchain, the Ethereum network can only handle a limited quantity of transactions per second[5]. This limit, referred to as network throughput, might end up in bottleneck and higher transaction costs during intervals of heavy demand.

The price of gas was observed throughout the testing phase and, while simple transactions such as device registration and status updates went adequately, more competition can result in a significant increase in operational costs. The cost of gas rises due to increased network congestion, as users must pay higher fees to ensure that their transactions are included in the next block during periods of high activity. As the solution grows to embrace more devices and interactions, this occurrence can evolve into a bottleneck.

Even while Remix is an adequate tool for testing and development in controlled environments, it is incapable to replicate multiple concurrent transactions. This factor posed an obstacle to the evaluation of the system's concurrency and scalability. The studies of these two components are therefore based on Ethereum's acknowledged constraints and provide predictions about how the network would behave in situations of congestion.

Another important concern was the use of external oracles to supply the smart contract with external data, particularly in the cryptographic security test. Oracles give the system flexibility and reliability, but using them also leads to the system to be more dependent on outside sources, which can compromise its availability and security. Furthermore, in addition to the gas charges in Ethereum, the use of Oracles implies extra costs in LINK tokens. These fees need to be taken into account, particularly in situations with high demand, when using oracles often might raise system running costs significantly.

Even while the testing revealed that the solution performed well on several of aspects, such as device registration, status update, and status query, the scalability and concurrency assessment identified significant shortcomings, especially considering the Ethereum network's boundaries. There were benefits and drawbacks to using oracles for external data integration; although they increased flexibility, they also increased costs and raised dependency on outside sources.

These factors demonstrate that although the solution is trustworthy and efficient in situations with low demand, large-scale sustainability need a more thorough examination of the blockchain's processing power and financial effects. Ethereum was chosen as the foundational platform because it is secure and decentralised, but as the system grows, its scalability issues need to be closely monitored.

[5]By March 2025, Ethereum's throughput on the main network is estimated to fluctuate from 15 to 30 Transactions Per Second (TPS) [157]. Future improvements like sharding and Layer 2 solutions should significantly increase scalability and help it finally achieve its 100,000 TPS target [158, 159].

Chapter 7

Conclusion

This dissertation proposes the development, implementation and validation of a blockchain-assisted DaaS solution that employs smart contracts and firmware integration to provide safe, remote and transparent management of distributed devices. The inspiration came from current problems experienced in large-scale public projects, such as the supply of laptops to students during the COVID-19 pandemic. Although these programmes are essential for digital inclusion, they exposed weaknesses in device management, especially in unsupervised environments, that led to loss, theft, misuse, and a lack of accountability and traceability.

The proposed approach addresses these problems by employing a decentralised, transparent, and tamper-resistant architecture. Fundamentally, the solution combines an UEFI module that is directly incorporated into the firmware of the device with smart contracts that are deployed on the Ethereum blockchain. Through this connectivity, the device itself can verify its operational state using the blockchain during the boot process, preventing unauthorised use as soon as possible, before any human contact or participation of the operating system occurs.

Several novel contributions were achieved throughout the development of this project. First, a decentralised DaaS model was designed and implemented, replacing traditional backend infrastructure with smart contracts that manage access and operational state automatically. Second, the approach connects the application and firmware by incorporating a verifier module into the UEFI boot process. This ensures an immutable connection between access policy and device activity by enabling direct synchronisation between the device and the blockchain. Third, the solution presents a scalable architecture that supports a wide range of devices and interactions while ensuring transparency and confidentiality.

The evaluation phase, executed on the Ethereum Sepolia testnet, proved that the system met its main requirements. Smart contracts were tested in realistic usage scenarios that included device registration, state change, access control, and auditing. Each contract interaction was validated using emitted events, on-chain state changes, cryptographic validation with HMAC, and lease validity. Furthermore, devices in a blocked state were successfully prevented from booting by the firmware verifier module, confirming the integrity of the control loop.

The system also proved to be resistant to common risks such as unauthorised access and data tampering. The adoption of blockchain technology eliminated the need for cen-

tralised management by ensuring its activities were immutable and transparent. The tests additionally validated the feasibility of integrating Chainlink oracles to the architecture, which would allow for cooperation with external sources of data. It highlights the flexibility of the architecture, even though it remains at Proof of Concept (POC) phase.

However, there were some limitations. The lack of a GUI limited the ability of non-technical stakeholders to use the system, even if the firmware connection and smart contracts performed as envisioned, the lack of frontend prevents wider adoption. Additionally, the use of Ethereum introduces gas fees to the operational costs, which could make it challenging to expand in scenarios with frequent demand or low profit margins. The use of Chainlink oracles also increases cost through the consumption of LINK tokens and dependence on external services. Lastly, the system is currently confined to UEFI-enabled devices with networking abilities, disqualifying legacy systems and restricting its use to specific deployment scenarios.

This dissertation fills some gaps in the existing literature. Previous researches address DaaS from a business abstraction or systems integration perspective, with fewer emphasis on firmware-level control, decentralisation, or real-time auditing. Furthermore, secure firmware upgrades and blockchain-based access control have been researched individually, with no solution integrating both fields into a single field-deployable model. The present research brings these fields together by combining decentralised access management, secure firmware interaction, and real-time state validation into a single cohesive system.

This dissertation offers an insightful implementation of a reference that can be used as an anchor for further research and business improvement. Beyond education, its architecture can be applied to the healthcare industry (e.g., patient-accessible medical devices), logistics industry (e.g., secure mobile asset tracking), or even vending machines. Auditability, control, and resilience in untrusted environments are basic demands shared by all of these subjects.

7.1 Future Work

Several prospects for further research and advancement are noted, building on the solid system established in this dissertation. These seek to increase the proposed DaaS architecture's scalability, resilience, and value while also expanding its potential use across many industries and technical environments.

1. **Improvements to Access Control Models:**
   Although RBAC is supported in the current smart contract implementation, flexibility and operational rule enforcement could be improved by finer-grained control models. Upcoming versions may consist of context-aware rules (e.g. device health or geographic location) and time-based access restrictions (e.g. enabling access only during educational hours);

2. **Implementation of a GUI:**
   A intuitive web-based interface is essential to encouraging practical adoption, especially in educational environments where technical knowledge could be limited. To provide global accessibility, the interface should work on desktop and mobile devices. The platform should provide device registration and deactivation, real-time audit log inspection, and role management monitoring.

3. **Domain Adaptation and Cross-Sector Application:**
   Despite the prototype's dedication to applications in the educational field, the architecture is adaptable to other domains. Potential future modifications could consist of healthcare, securing diagnostic or therapeutic devices that patients can access, with authorisation only given under clinical supervision; logistics, enabling remote control and tamper-evident tracking of portable assets (such as rental equipment and shipping containers); and automatic vending, integrating smart vending machines that work only upon remote authorisation, pay-per-use models, or limited areas. These use cases share the same requirements: high transparency, distributed confidence, and performing in untrustworthy settings.

4. **Deployment on Private Blockchains:**
   Although the Ethereum public testnet was used to develop and verify this project, other blockchain platforms might have advantages in terms of cost-effectiveness, management, and performance. Future iterations should explore private blockchain platforms such as Quorum or Hyperledger Fabric, which offer improved efficiency, lower transaction costs, and more influence over consensus mechanisms.

In conclusion, this research has demonstrated that it can be feasible and advantageous to apply blockchain technology, particularly smart contracts and public auditability, to address the issue of remote device management. The proposed solution overcomes the constraints in existing DaaS applications by integrating firmware-level enforcement, decentralised management, and cryptographic trust.

Even if there remain concerns with usability, cost, and scalability, the framework established in this dissertation offers a strong basis for future development. The idea has the potential to have an impact on society in addition to contributing to academic research by rendering it feasible to manage digital infrastructure in a method that is safer, more transparent, and more trustworthy in a world whose digital backbone is growing ever more critical to public trust and operational security.

Chapter 8

Bibliography

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 6 2017, pp. 557–564. [Online]. Available: http://ieeexplore.ieee.org/document/8029379/

[2] N. Hassan, N. Jain, and V. K. Chandna, "Blockchain, cryptocurrency and bitcoin," *International Journal of Computer Engineering and Applications*, vol. XII, 4 2018. [Online]. Available: https://www.researchgate.net/publication/334279715_BLOCKCHAIN_CRYPTOCURRENCY_AND_BITCOINwww.ijcea.com

[3] M. K. Shrivas, "The disruptive blockchain: Types, platforms and applications," *TEXILA INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH*, pp. 17–39, 4 2019. [Online]. Available: https://www.texilajournal.com/academic-research/article/1216-the-disruptive-blockchain

[4] A. Chopra, "Blockchain technology in food industry ecosystem," *IOP Conference Series: Materials Science and Engineering*, vol. 872, p. 012005, 6 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/872/1/012005

[5] M. M. Ahmed and A. C. Shakir, "Review study: Blockchain application in payroll system," *Al-Kitab Journal for Pure Sciences*, vol. 7, pp. 83–99, 8 2023. [Online]. Available: https://isnra.net/index.php/kjps/article/view/928

[6] R. B. Uriarte, R. de Nicola, and K. Kritikos, "Towards distributed sla management with smart contracts and blockchain," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 2018-December. IEEE, 12 2018, pp. 266–271. [Online]. Available: https://ieeexplore.ieee.org/document/8591028/

[7] S. Nzuva, "Smart contracts implementation, applications, benefits, and limitations," vol. 9, 2019. [Online]. Available: www.iiste.org

[8] S. Jani, "Smart contracts: Building blocks for digital transformation," Tech. Rep., 4 2020.

[9] Magnus, N. Nystrom, V. Martin, and Zimmer, "Uefi networking and pre-os security," *Intel ® Technology Journal —*, vol. 15, 10 2011.

[10] U. Forum, "Uefi specification 2.11," 11 2024. [Online]. Available: https://uefi.org/specs/UEFI/2.11

[11] P. K. Paul, P. S. S. Aithal, R. Saavedra, and S. Ghosh, "Blockchain technology and its types-a short review," *Blockchain Technology and its Types-A Short Review. IJASE*, vol. 9, pp. 189–200, 12 2021.

[12] A. Rahman, A. Montieri, D. Kundu, M. R. Karim, M. J. Islam, S. Umme, A. Nascita, and A. Pescapé, "On the integration of blockchain and sdn: Overview, applications, and future perspectives," *Journal of Network and Systems Management*, vol. 30, p. 73, 10 2022. [Online]. Available: https://link.springer.com/10.1007/s10922-022-09682-4

[13] Z. Ouyang, J. Shao, and Y. Zeng, "Pow and pos and related applications," in *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*. IEEE, 9 2021, pp. 59–62. [Online]. Available: https://ieeexplore.ieee.org/document/9588080/

[14] J. Reis, M. Amorim, N. Melão, Y. Cohen, and M. Rodrigues, *Digitalization: A Literature Review and Research Agenda*. Springer Nature, 3 2020, vol. Part F201, pp. 443–456. [Online]. Available: http://link.springer.com/10.1007/978-3-030-43616-2_47

[15] P. Europeu, "Moldar a transformação digital: a estratégia da ue explicada," 11 2023. [Online]. Available: https://www.europarl.europa.eu/news/pt/headlines/society/20210414STO02010/transformacao-digital-importancia-beneficios-e-politica-da-ue

[16] Lusa, "Escola digital prevê distribuição de 100 mil equipamentos na 1.º fase," 8 2020. [Online]. Available: https://www.noticiasaominuto.com/tech/1567506/escola-digital-preve-distribuicao-de-100-mil-equipamentos-na-1-fase

[17] T. Europa, "Ministério da educação disponibiliza 100 mil computadores para escola digital," 11 2020. [Online]. Available: https://www.tveuropa.pt/noticias/ministerio-da-educacao-disponibiliza-100-mil-computadores-para-escola-digital/

[18] G. da República Portuguesa, "Escolas receberam instruções sobre entrega de computadores a alunos," 11 2020. [Online]. Available: https://www.portugal.gov.pt/pt/gc22/comunicacao/noticia?i=escolas-receberam-instrucoes-sobre-entrega-de-computadores-a-alunos

[19] R. Barrangán-Gil, J. A. Holgado-Terriza, and J. M. G. Guerrero, "Daas (device as a service): Un paradigma de abstracción de dispositivos basados en servicios aplicado a sistemas domóticos," *XXIV Jornadas de Concurrencia y Sistemas Distribuidos*, pp. 21–35, 6 2016. [Online]. Available: https://www.researchgate.net/publication/304102442

[20] S. Higgins, "Intel unveils 'sawtooth lake' proposal at hyperledger meeting," 4 2016. [Online]. Available: https://www.coindesk.com/markets/2016/04/07/intel-unveils-sawtooth-lake-proposal-at-hyperledger-meeting/

[21] A. Alamsyah, G. N. W. Kusuma, and D. P. Ramadhani, "A review on decentralized finance ecosystems," *Future Internet*, vol. 16, p. 76, 2 2024. [Online]. Available: https://www.mdpi.com/1999-5903/16/3/76

[22] C. Nairi, M. Cicioğlu, and A. Çalhan, "Smart blockchain networks: Revolutionizing donation tracking in the web 3.0," *Computer Communications*, vol. 228, p. 107972, 12 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0140366424003190

[23] M. Sookhak, M. R. Jabbarpour, N. S. Safa, and F. R. Yu, "Blockchain and smart contract for access control in healthcare: A survey, issues and challenges, and open issues," *Journal of Network and Computer Applications*, vol. 178, p. 102950, 3 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804520304045

[24] I. Mironov, "Hash functions: Theory, attacks, and applications," Microsoft Research,, Tech. Rep., 11 2005.

[25] X. Fan, B. Niu, and Z. Liu, "Scalable blockchain storage systems: research progress and models," *Computing*, vol. 104, pp. 1497–1524, 6 2022. [Online]. Available: https://link.springer.com/10.1007/s00607-022-01063-8

[26] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, pp. 99–111, 1 1991. [Online]. Available: http://link.springer.com/10.1007/BF00196791

[27] R. C. Merkle, *A Digital Signature Based on a Conventional Encryption Function*, 1988, pp. 369–378. [Online]. Available: http://link.springer.com/10.1007/3-540-48184-2_32

[28] E. Mykletun, M. Narasimha, and G. Tsudik, "Providing authentication and integrity in outsourced databases using merkle hash tree's," Tech. Rep., 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:10853829

[29] S. N. Institute, "Rpow - reusable proofs of work," 2004. [Online]. Available: https://nakamotoinstitute.org/finney/rpow

[30] C. Staff, "What is double-spending?" 6 2021. [Online]. Available: https://www.gemini.com/cryptopedia/double-spending-problem-crypto

[31] B. Academy, "Double spending explained," 1 2023. [Online]. Available: https://academy.binance.com/en/articles/double-spending-explained

[32] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008. [Online]. Available: www.bitcoin.org

[33] K. Qin, L. Zhou, Y. Afonin, L. Lazzaretti, and A. Gervais, "Cefi vs. defi – comparing centralized to decentralized finance," 6 2021. [Online]. Available: http://arxiv.org/abs/2106.08157

[34] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform." Tech. Rep., 2014. [Online]. Available: https://ethereum.org/

[35] Z. Liu, N. C. Luong, W. Wang, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "A survey on applications of game theory in blockchain," 2 2019. [Online]. Available: http://arxiv.org/abs/1902.10865

[36] M. Ubaidullah and Q. Makki, "A review on symmetric key encryption techniques in cryptography," *International Journal of Computer Applications*, vol. 147, pp.

43–48, 8 2016. [Online]. Available: http://www.ijcaonline.org/archives/volume147/number10/bokhari-2016-ijca-911203.pdf

[37] P. P. Santoso, E. Rilvani, A. B. Trisnawan, K. Adiyarta, D. Napitupulu, T. Sutabri, and R. Rahim, "Systematic literature review: comparison study of symmetric key and asymmetric key algorithm," *IOP Conference Series: Materials Science and Engineering*, vol. 420, p. 012111, 10 2018. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/420/1/012111

[38] J. I. Ahmad, R. Din, and M. Ahmad, "Analysis review on public key cryptography algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, p. 447, 11 2018. [Online]. Available: http://ijeecs.iaescore.com/index.php/IJEECS/article/view/14423

[39] R. de Best, "Size of the bitcoin blockchain from january 2009 to april 11, 2024," 4 2024. [Online]. Available: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/

[40] Blockchain.com, "Blockchain size," 2024. [Online]. Available: https://www.blockchain.com/explorer/charts/blocks-size

[41] R. Nagayama, R. Banno, and K. Shudo, "Trail: A blockchain architecture for light nodes," 7 2020. [Online]. Available: http://arxiv.org/abs/2007.08066

[42] N. C. K. Yiu, "An overview of forks and coordination in blockchain development," 2 2021. [Online]. Available: http://arxiv.org/abs/2102.10006http://dx.doi.org/10.13140/RG.2.2.36579.07207

[43] A. Council, "Central bank digital currency tracker," 2024. [Online]. Available: https://www.atlanticcouncil.org/cbdctracker/

[44] P. K. Ozili, "Decentralized finance research and developments around the world," *Journal of Banking and Financial Technology*, vol. 6, pp. 117–133, 10 2022. [Online]. Available: https://link.springer.com/10.1007/s42786-022-00044-x

[45] R. Moro-Visconti and A. Cesaretti, *Decentralized Finance (DeFi).* Springer Nature Switzerland, 10 2023, pp. 287–340. [Online]. Available: https://link.springer.com/10.1007/978-3-031-42971-2_9

[46] J. Wu and N. K. Tran, "Application of blockchain technology in sustainable energy systems: An overview," *Sustainability*, vol. 10, p. 3067, 8 2018. [Online]. Available: https://www.mdpi.com/2071-1050/10/9/3067

[47] A. Borkovcová, M. Černá, and M. Sokolová, "Blockchain in the energy sector—systematic review," *Sustainability*, vol. 14, p. 14793, 11 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/22/14793

[48] M. I. Zacky, S. Helmi, and I. D. Cella, "Smart contracts on the blockchain: Design, use cases, and prospects," *Blockchain Frontier Technology*, vol. 3, pp. 54–73, 6 2023. [Online]. Available: https://journal.pandawan.id/b-front/article/view/363

[49] K. Honari, S. Rouhani, N. E. Falak, Y. Liu, Y. Li, H. Liang, S. Dick, and J. Miller, "Smart contract design in distributed energy systems: A systematic review," *Energies*, vol. 16, p. 4797, 6 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/12/4797

[50] G. Zhu, D. He, H. An, M. Luo, and C. Peng, "The governance technology for blockchain systems: a survey," *Frontiers of Computer Science*, vol. 18, p. 182813, 4 2024. [Online]. Available: https://link.springer.com/10.1007/s11704-023-3113-x

[51] D. Stamatakis, D. G. Kogias, P. Papadopoulos, P. A. Karkazis, and H. C. Leligou, "Blockchain-powered gaming: Bridging entertainment with serious game objectives," *Computers*, vol. 13, p. 14, 1 2024. [Online]. Available: https://www.mdpi.com/2073-431X/13/1/14

[52] G. Jaferian, D. Ramezani, and M. G. Wagner, "Exploring the impact of smart contracts on decentralized gaming ecosystems," 3 2025. [Online]. Available: https://doi.org/10.48341/f592-nx97

[53] H. Wu and X. Zhu, "Developing a reliable service system of charity donation during the covid-19 outbreak," *IEEE Access*, vol. 8, pp. 154 848–154 860, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9170634/

[54] P. D. Patil, D. J. Mhatre, N. H. Gharat, and J. Tinsu, "Transparent charity system using smart contracts on ethereum using blockchain," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, pp. 743–748, 4 2022. [Online]. Available: https://www.ijraset.com/best-journal/transparent-charity-system-using-smart-contracts-on-ethereum-using-blockchain

[55] M. Zachariadis, G. Hileman, and S. V. Scott, "Governance and control in distributed ledgers: Understanding the challenges facing blockchain technology in financial services," *Information and Organization*, vol. 29, pp. 105–117, 6 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1471772719300284

[56] R. van Pelt, S. Jansen, D. Baars, and S. Overbeek, "Defining blockchain governance: A framework for analysis and comparison," *Information Systems Management*, vol. 38, pp. 21–41, 1 2021. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10580530.2020.1720046

[57] A. Yusuf and R. Martinez, "Smart contracts and legal enforceability: Decoding the political philosophy of code as law," *Interdisciplinary Studies in Society, Law, and Politics*, vol. 4, pp. 292–302, 2025. [Online]. Available: https://journalisslp.com/index.php/isslp/article/view/321

[58] O. K. Akinsola and B. J. Mary, "Smart contracts and corporate governance: Automation, legal risks, and benefits," Tech. Rep., 2025. [Online]. Available: https://www.researchgate.net/publication/388406514

[59] J. Sedlmeir, H. U. Buhl, G. Fridgen, and R. Keller, "The energy consumption of blockchain technology: Beyond myth," *Business & Information Systems Engineering*, vol. 62, pp. 599–608, 12 2020. [Online]. Available: https://link.springer.com/10.1007/s12599-020-00656-x

[60] "Bitcoin energy consumption index," 2024. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[61] J. Quitzon, "51% attack: Definition, risks, examples," 7 2023. [Online]. Available: https://atomicwallet.io/academy/articles/51-attack

[62] G.-J. Glasbergen, J. Lovejoy, and A. Ouyang, "51% attacks." [Online]. Available: https://www.dci.mit.edu/projects/51-percent-attacks

[63] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg, *Byzantine Fault Tolerance, from Theory to Reality.* Springer Verlag, 2003, vol. 2788, pp. 235–248. [Online]. Available: http://link.springer.com/10.1007/978-3-540-39878-3_19

[64] M. Nasreen, A. Ganesh, and C. Sunitha, "A study on byzantine fault tolerance methods in distributed networks," *Procedia Computer Science*, vol. 87, pp. 50–54, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1877050916304641

[65] W. Zhong, C. Yang, W. Liang, J. Cai, L. Chen, J. Liao, and N. Xiong, "Byzantine fault-tolerant consensus algorithms: A survey," *Electronics*, vol. 12, p. 3801, 9 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/18/3801

[66] C. Dwork and M. Naor, *Pricing via Processing or Combatting Junk Mail.* Springer Berlin Heidelberg, 1993, pp. 139–147. [Online]. Available: http://link.springer.com/10.1007/3-540-48071-4_10

[67] M. Jakobsson and A. Juels, *Proofs of Work and Bread Pudding Protocols(Extended Abstract).* Springer US, 1999, pp. 258–272. [Online]. Available: http://link.springer.com/10.1007/978-0-387-35568-9_18

[68] A. M. Antonopoulos, *Mastering Bitcoin: Programming the Open Blockchain*, 2nd ed. O'Reilly Media, Inc., 2017. [Online]. Available: https://dl.acm.org/doi/10.5555/3164842

[69] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," Tech. Rep., 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:42319203

[70] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 10 2017. [Online]. Available: http://arxiv.org/abs/1710.09437

[71] W. Zhao, S. Yang, X. Luo, and J. Zhou, "On peercoin proof of stake for blockchain consensus," in *2021 The 3rd International Conference on Blockchain Technology.* ACM, 3 2021, pp. 129–134. [Online]. Available: https://dl.acm.org/doi/10.1145/3460537.3460547

[72] S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Systems with Applications*, vol. 154, p. 113385, 9 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417420302098

[73] S. Gao, Z. Li, Z. Peng, and B. Xiao, "Power adjusting and bribery racing," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 11 2019, pp. 833–850. [Online]. Available: https://dl.acm.org/doi/10.1145/3319535.3354203

[74] D. Karakostas, A. Kiayias, and T. Zacharias, "Blockchain bribing attacks and the efficacy of counterincentives," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security.* ACM, 12 2024, pp. 1031–1045. [Online]. Available: https://dl.acm.org/doi/10.1145/3658644.3670330

[75] M. Castro and B. Liskov, "Practical byzantine fault tolerance," *Proceedings ofthe Third Symposium on Operating Systems Design and Implementation*, 1999.

[76] A. K. Yadav, K. Singh, A. H. Amin, L. Almutairi, T. R. Alsenani, and A. Ahmadian, "A comparative study on consensus mechanism with security threats and future scopes: Blockchain," *Computer Communications*, vol. 201, pp. 102–115, 3 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0140366423000269

[77] J. R. Douceur, *The Sybil Attack*, 2002, pp. 251–260. [Online]. Available: http://link.springer.com/10.1007/3-540-45748-8_24

[78] D. Larimer, "Bitshares whitepaper," 2015. [Online]. Available: https://bitshares.github.io/docs/#/whitepaper

[79] H. Xiong, M. Chen, C. Wu, Y. Zhao, and W. Yi, "Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms," *Future Internet*, vol. 14, p. 47, 1 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/2/47

[80] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, *Proof os Space*, R. Gennaro and M. Robshaw, Eds. Springer Berlin Heidelberg, 2013, vol. 9216. [Online]. Available: http://link.springer.com/10.1007/978-3-662-48000-7

[81] Crypto, "Proof of capacity," 2024. [Online]. Available: https://crypto.com/glossary/pt/proof-of-capacity

[82] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, pp. 34–37, 12 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2695533.2695545

[83] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, pp. 1243–1248, 12 1968. [Online]. Available: https://www.science.org/doi/10.1126/science.162.3859.1243

[84] NEM, *Proof of Importance*, 2 2018, pp. 26–43. [Online]. Available: https://nemproject.github.io/nem-docs/pages/Whitepapers/NEM_techRef.pdf

[85] Sawtooth, "Hyperledger sawtooth," 2016. [Online]. Available: https://github.com/hyperledger-archives/sawtooth-poet

[86] W. Docs, "Lpos — leased proof of stake," 2018. [Online]. Available: https://docs.waves.tech/en/blockchain/waves-protocol/leased-proof-of-stake-lpos#core-concepts-of-waves-lpos

[87] I. Stewart, "Proof of burn," 2012. [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_burn

[88] K. Karantias, A. Kiayias, and D. Zindros, "Proof-of-burn," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12059 LNCS, pp. 523–540, 2020. [Online]. Available: http://link.springer.com/10.1007/978-3-030-51280-4_28

[89] N. Community, "Nxt whitepaper," Tech. Rep., 7 2014. [Online]. Available: https://bitbucket.org/JeanLucPicard/nxt/src

[90] S. Popov, "The tangle," Tech. Rep., 2 2018. [Online]. Available: https://www.allcryptowhitepapers.com/iota-whitepaper/

[91] "Iot chain," Tech. Rep., 2018. [Online]. Available: https://www.allcryptowhitepapers.com/iotchain-whitepaper/

[92] Z. Hussein, M. A. Salama, and S. A. El-Rahman, "Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms," *Cybersecurity*, vol. 6, p. 30, 11 2023. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-023-00163-y

[93] I. Foundation, "The tangle." [Online]. Available: https://wiki.iota.org/get-started/introduction/iota/introduction/

[94] CryptoWallet.com, "What is orphan block?" [Online]. Available: https://cryptowallet.com/glossary/orphan-block/

[95] M. Grant, "Orphan block: What it is, how it works, faq," 5 2023. [Online]. Available: https://www.investopedia.com/terms/o/orphan-block-cryptocurrency.asp

[96] K. Shah, D. Lathiya, N. Lukhi, K. Parmar, and H. Sanghvi, "A systematic review of decentralized finance protocols," *International Journal of Intelligent Networks*, vol. 4, pp. 171–181, 1 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666603023000179

[97] D. Metelski and J. Sobieraj, "Decentralized finance (defi) projects: A study of key performance indicators in terms of defi protocols' valuations," *International Journal of Financial Studies*, vol. 10, p. 108, 11 2022. [Online]. Available: https://www.mdpi.com/2227-7072/10/4/108

[98] Crypto.com, "Fiat." [Online]. Available: https://crypto.com/glossary/fiat

[99] "Lite coin white paper," Tech. Rep., 2011. [Online]. Available: https://github.com/litecoin-project/litecoinwww.litecoin.org

[100] B. Chase and E. MacBrough, "Analysis of the xrp ledger consensus protocol," 2 2018. [Online]. Available: http://arxiv.org/abs/1802.07242

[101] R. Kumar, "Staking vs liquidity pools: Best strategies for 2025," 1 2025. [Online]. Available: https://industrywired.com/cryptocurrencies/staking-vs-liquidity-pools-best-strategies-for-2025-8609522

[102] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, 9 1997.

[103] V. Dwivedi, V. Pattanaik, V. Deval, A. Dixit, A. Norta, and D. Draheim, "Legally enforceable smart-contract languages," *ACM Computing Surveys*, vol. 54, pp. 1–34, 6 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3453475

[104] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: foundations, design landscape and research directions," 8 2016. [Online]. Available: http://arxiv.org/abs/1608.00771

[105] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th International Conference on*

Computing, Communication and Networking Technologies (ICCCNT). IEEE, 7 2018, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/8494045/

[106] Icertis, "What is the contract lifecycle management process diagram?" [Online]. Available: https://www.icertis.com/contracting-basics/contract-lifecycle-management-process-diagram/

[107] J. Gorzny, L. Po-An, and M. Derka, "Ideal properties of rollup escape hatches," 3rd International Workshop on Distributed Infrastructure for Common Good, Tech. Rep., 2022.

[108] F. G. Figueira, M. Derka, C. L. Chiu, and J. Gorzny, "A practical rollup escape hatch design," 3 2025. [Online]. Available: http://arxiv.org/abs/2503.23986

[109] S. Palladino, "The parity wallet hack explained," 7 2017. [Online]. Available: https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7

[110] W. Zhang, L. Wei, S.-C. Cheung, Y. Liu, S. Li, L. Liu, and M. R. Lyu, "Combatting front-running in smart contracts: Attack mining, benchmark construction and vulnerability detector evaluation," *IEEE Transactions on Software Engineering*, vol. 49, pp. 1–17, 6 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10108045/

[111] "Axa drops ethereum-based flight insurance platform," 11 2019. [Online]. Available: https://www.artificiallawyer.com/2020/10/08/axa-scraps-fizzy-insurance-smart-contract-but-still-interested-in-the-tech/

[112] B. H. Susanto, M. N. Masrek, and I. E. Khairuddin, "Implementation of smart contract technology in financial services institutions," *Environment-Behaviour Proceedings Journal*, vol. 7, pp. 249–254, 11 2022. [Online]. Available: https://ebpj.e-iph.co.uk/index.php/EBProceedings/article/view/4129

[113] D. Muayad and M. Abumandil, "Role of smart contract technology blockchain services in finance and banking systems: Concept and core values," *SSRN Electronic Journal*, 2022. [Online]. Available: https://www.ssrn.com/abstract=4078566

[114] S. C. Prabanand and M. S. Thanabal, "Advanced financial security system using smart contract in private ethereum consortium blockchain with hybrid optimization strategy," *Scientific Reports*, vol. 15, p. 6764, 2 2025. [Online]. Available: https://www.nature.com/articles/s41598-025-89404-3

[115] T. I. Team, "Iotex a decentralized network for internet of things powered by a privacy-centric blockchain," Tech. Rep., 7 2018. [Online]. Available: https://docs.iotex.io/readme/whitepaper

[116] J. Moosavi, L. M. Naeni, A. M. Fathollahi-Fard, and U. Fiore, "Blockchain in supply chain management: a review, bibliometric, and network analysis," *Environmental Science and Pollution Research*, 2 2021. [Online]. Available: http://link.springer.com/10.1007/s11356-021-13094-3

[117] K. Chakraborty, A. Ghosh, and S. Pratap, "Adoption of blockchain technology in supply chain operations: a comprehensive literature study analysis," *Operations Management Research*, vol. 16, pp. 1989–2007, 12 2023. [Online]. Available: https://link.springer.com/10.1007/s12063-023-00420-w

[118] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 10 2017, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/8292361/

[119] A. Rejeb, H. Treiblmaier, K. Rejeb, and S. Zailani, "Blockchain research in healthcare: a bibliometric review and current research trends," *Journal of Data, Information and Management*, vol. 3, pp. 109–124, 6 2021. [Online]. Available: https://link.springer.com/10.1007/s42488-021-00046-2

[120] S. Khezr, M. Moniruzzaman, A. Yassine, and R. Benlamri, "Blockchain technology in healthcare: A comprehensive review and directions for future research," *Applied Sciences*, vol. 9, p. 1736, 4 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/9/1736

[121] C. Vargas and M. M. da Silva, "Case studies about smart contracts in healthcare," *DIGITAL HEALTH*, vol. 9, 1 2023. [Online]. Available: https://journals.sagepub.com/doi/10.1177/20552076231203571

[122] A. Attar, "The ujo platform: A decentralized music ecosystem," 7 2018. [Online]. Available: https://blog.ujomusic.com/the-ujo-platform-a-decentralized-music-ecosystem-e530c31b62bc

[123] LightNode, "Uefi vs bios: A comprehensive guide to modern firmware technologies," 11 2024. [Online]. Available: https://go.lightnode.com/tech/uefi-vs-bios

[124] T. Latzo, F. Hantke, L. Kotschi, and F. Freiling, "Bringing forensic readiness to modern computer firmware," 5 2025. [Online]. Available: http://arxiv.org/abs/2505.05697

[125] L. Catuogno and C. Galdi, "Secure firmware update: Challenges and solutions," *Cryptography*, vol. 7, p. 30, 6 2023. [Online]. Available: https://www.mdpi.com/2410-387X/7/2/30

[126] I. Corporation, "Unified extensible firmware interface." [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/tool/unified-extensible-firmware-interface.html

[127] S. M. Al-Mutoki, A. A. ali Hadi, and A. F. Mohammed, "Unified extensible firmware interface (uefi) between speed and security," Tech. Rep., 2017. [Online]. Available: https://www.researchgate.net/publication/368690729

[128] U. Forum, "Unified extensible firmware interface forum." [Online]. Available: https://uefi.org/

[129] R. Wilkins and B. Richardson, "Uefi secure boot in modern computer security solutions," Tech. Rep., 9 2013. [Online]. Available: https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2019.pdf

[130] TianoCore, "Edk ii project." [Online]. Available: https://github.com/tianocore/edk2

[131] U. Forum, "Uefi specification - network protocols — arp, dhcp, dns, http and rest." [Online]. Available: https://uefi.org/specs/UEFI/2.11/29_Network_Protocols_ARP_and_DHCP.html#

[132] ——, "Uefi specification - boot manager." [Online]. Available: https://uefi.org/specs/UEFI/2.11/03_Boot_Manager.html#network-booting

[133] G. Olaoye, "Authentication and integrity of uefi capsule updates," 2 2025.

[134] U. Forum, "Uefi specification - secure technologies." [Online]. Available: https://uefi.org/specs/UEFI/2.10/37_Secure_Technologies.html

[135] F. O. for Information Security, "Sisyphus win10: Analysis of tpm integration and uefi "secure boot" in windows 10." [Online]. Available: https://www.bsi.bund.de/dok/12818680

[136] M. Usman, I. Ahmed, M. Imran, S. Khan, and U. Ali, "Sit: A lightweight encryption algorithm for secure internet of things," *International Journal of Advanced Computer Science and Applications*, vol. 8, 4 2017. [Online]. Available: http://thesai.org/Publications/ViewPaper?Volume=8&Issue=1&Code=ijacsa&SerialNo=51

[137] P. KAMPANAKIS, P. PANBURANA, M. CURCIO, C. SHROFF, and M. M. ALAM, "Post-quantum lms and sphincs + hash-based signatures for uefi secure boot," Tech. Rep., 6 2021. [Online]. Available: https://eprint.iacr.org/2021/041.pdf

[138] T. L. project, "Linux boot book." [Online]. Available: https://book.linuxboot.org/

[139] T. Developers, "Trenchboot." [Online]. Available: https://trenchboot.org/

[140] S. Choi and J.-H. Lee, "Blockchain-based distributed firmware update architecture for iot devices," *IEEE Access*, vol. 8, pp. 37 518–37 525, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9007435/

[141] J.-M. Maree, K. Kruger, and A. Basson, "Opportunities for digital twins for the provisioning, management and monitoring of heterogeneous iot devices," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*. ACM, 9 2024, pp. 378–389. [Online]. Available: https://dl.acm.org/doi/10.1145/3652620.3688251

[142] M. R. Private, "Device-as-a-service market size." [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/pc-as-a-service-market-155153641.html

[143] P. Hruby and C. V. Scheller, "Device as a service-an economic model," 2019.

[144] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control services," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 7 2018, pp. 1379–1386. [Online]. Available: https://ieeexplore.ieee.org/document/8726511/

[145] ——, "A blockchain based approach for the definition of auditable access control systems," *Computers & Security*, vol. 84, pp. 93–119, 7 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404818309398

[146] T. Sultana, A. Almogren, M. Akbar, M. Zuair, I. Ullah, and N. Javaid, "Data sharing system integrating access control mechanism using blockchain-based smart contracts for iot devices," *Applied Sciences*, vol. 10, p. 488, 1 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/2/488

[147] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 7 2018, pp. 1499–1506. [Online]. Available: https://ieeexplore.ieee.org/document/8726493/

[148] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12 240–12 251, 3 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8307397/

[149] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an internet of things environment," *The Journal of Supercomputing*, vol. 73, pp. 1152–1167, 3 2017. [Online]. Available: http://link.springer.com/10.1007/s11227-016-1870-0

[150] M. Baza, M. Nabil, N. Lasla, K. Fidan, M. Mahmoud, and M. Abdallah, "Blockchain-based firmware update scheme tailored for autonomous vehicles," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 4 2019, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/8885769/

[151] G. Bere, B. Ahn, J. J. Ochoa, T. Kim, A. A. Hadi, and J. Choi, "Blockchain-based firmware security check and recovery for smart inverters," in *2021 IEEE Applied Power Electronics Conference and Exposition (APEC)*. IEEE, 6 2021, pp. 675–679. [Online]. Available: https://ieeexplore.ieee.org/document/9487468/

[152] V. Bashun, A. Sergeev, V. Minchenkov, and A. Yakovlev, "Too young to be secure: Analysis of uefi threats and vulnerabilities," in *14th Conference of Open Innovation Association FRUCT*. IEEE, 11 2013, pp. 16–24. [Online]. Available: http://ieeexplore.ieee.org/document/6737940/

[153] A. T. A. M, A. W. A. Wahab, and M. Y. I. Idris, "Systematic literature review on security access control policies and techniques based on privacy requirements in a byod environment: State of the art and future directions," *Applied Sciences*, vol. 13, p. 8048, 7 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/14/8048

[154] M. Chen, Y. Yu, G. Xie, C. Zeng, and Z. Xu, "Uefi-based research on the inner operation mechanism and characteristics of firmware vulnerabilities in key devices of electric power monitoring systems," *Applied Mathematics and Nonlinear Sciences*, vol. 9, 1 2024. [Online]. Available: https://www.sciendo.com/article/10.2478/amns-2024-0136

[155] C. Foundation, "What is a blockchain oracle?" [Online]. Available: https://chain.link/education/blockchain-oracles

[156] T. Hodges, "Chainlink white paper — section 3— oracle security," 11 2017. [Online]. Available: https://medium.com/chainlink/chainlink-white-paper-section-3-oracle-security-c88825fe56ac

[157] M. Editorial, "Scaling solutions for ethereum: Overcoming the bottleneck," 3 2025. [Online]. Available: https://metana.io/blog/scaling-solutions-for-ethereum-overcoming-the-bottleneck/#:~:text=March\%205\%2C\%202025-,TL;DR:,consensus\%20mechanism\%20and\%20implementing\%20sharding.

[158] A. Sergeenkov, "Ethereum's surge roadmap targets 100,000+ transactions per second," 11 2024. [Online]. Available: https://www.forbes.com/sites/digital-assets/2024/10/17/ethereums-surge-roadmap-targets-100000-transactions-per-second/

[159] Y. A. Szaniecki, "The dencun upgrade is live: Ethereum's evolution continues at full throttle," 3 2024. [Online]. Available: https://hashdex.com/en-KY/insights/the-dencun-upgrade-is-live-ethereum-s-evolution-continues-at-full-throttle

[160] A. B. C. Limited, "Moscow prioritisation." [Online]. Available: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html

[161] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, "Fuzzy_moscow: A fuzzy based moscow method for the prioritization of software requirements," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. IEEE, 7 2017, pp. 433–437. [Online]. Available: http://ieeexplore.ieee.org/document/8342602/

[162] A. B. C. Limited, "Dsdm agile project framework." [Online]. Available: https://www.agilebusiness.org/dsdm-project-framework.html

[163] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, "Requirements prioritization techniques comparison," *Modern Applied Science*, vol. 12, p. 62, 1 2018. [Online]. Available: http://www.ccsenet.org/journal/index.php/mas/article/view/72718

[164] R. Izhar, K. Cosh, and S. N. Bhatti, "Enhancing agile software development: A novel approach to automated requirements prioritization," in *2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 6 2024, pp. 286–293. [Online]. Available: https://ieeexplore.ieee.org/document/10613648/

[165] N. Kano, N. Seraku, F. Takahashi, and S. Tsuji, "Attractive quality and must-be quality," *Journal of the Japanese Society for Service Quality Control*, vol. 14, 1984.

[166] J. Dalton, *Kano Model*. Apress, 2019, pp. 189–190. [Online]. Available: http://link.springer.com/10.1007/978-1-4842-4206-3_37

[167] S. Mote, V. Kulkarni, and D. B. Narkhede, "Kano model application in new service development and customer satisfaction," *IOSR Journal of Business and Management*, vol. 18, pp. 10–14, 8 2016. [Online]. Available: http://iosrjournals.org/iosr-jbm/papers/Vol18-issue8/Version-1/B1808011014.pdf

[168] T. Riemann, A. Kreß, L. Roth, S. Klipfel, J. Metternich, and P. Grell, "Agile implementation of virtual reality in learning factories," *Procedia Manufacturing*,

vol. 45, pp. 1–6, 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2351978920310672

[169] H. Jesse, "The kano model: how to wow your customers," 1 2021. [Online]. Available: https://dmexco.com/stories/kano-model/

[170] F.-H. Lin, S.-B. Tsai, Y.-C. Lee, C.-F. Hsiao, J. Zhou, J. Wang, and Z. Shang, "Empirical research on kano's model and customer satisfaction," *PLOS ONE*, vol. 12, p. e0183888, 9 2017. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0183888

[171] "Design management tools and techniques," 2016. [Online]. Available: https://www.ifm.eng.cam.ac.uk/research/dmg/tools-and-techniques/

[172] D. Zacarias, "The complete guide to the kano model," 9 2022. [Online]. Available: https://foldingburritos.com/blog/kano-model/

[173] I. S. Planes, "Kano model." [Online]. Available: https://www.launchnotes.com/glossary/kano-model-in-product-management-and-operations

[174] K. Mayan, "What is the kano model prioritization framework? overview, guide, and templates," 4 2023. [Online]. Available: https://www.savio.io/product-roadmap/kano-model/

Appendix A

Feature Prioritisation Techniques

Accurate feature prioritising has become crucial in modern project management, especially in agile methodologies, to ensure successful project delivery within budget and schedule constraints. Several techniques have been proposed to help teams identify the components or behaviours that are vital to the achievement of a project.

In this section, an outline of some prioritisation techniques is given, such as the MoSCoW technique, Kano model, Eisenhower matrix, and others. Through knowledge of these methodologies, project managers can make informed decisions about which features to produce, when to develop them, and under what constraints to work, ensuring that the features are in alignment with stakeholder expectations and its goals.

A.1 MoSCoW Technique

One popular prioritisation methodology in project management, especially in the realm of agile development, is the MoSCoW technique. Its goal is to help teams decide which features or needs, in order to ensure project success, should be executed with the highest priority. MoSCoW is an abbreviation for four priority categories: *must have* (mandatory), *should have* (important but not critical), *could have* (desirable but not required), and *won't have this time* (not included in this iteration, but perhaps in the future) [160, 161].

The MoSCoW approach offers a clear framework to manage the scope of the project and align stakeholder expectations by establishing requirements in this way. By highlighting important components and allowing for adjustments in lower priority items subject to time and resource constraints, it aids decision-making.

As a component of the Dynamic Systems Development Method (DSDM) [162], a framework designed to support agile software development, the MoSCoW technique initially emerged in the late 1990s. The development of DSDM was motivated by the necessity to satisfy the demands of stakeholders while delivering software under tight financial and schedule constraints. Within this framework, the MoSCoW technique emerged as an approach to ensure that the most significant project requirements, those required for the system to operate efficiently, were given precedence over smaller features. MoSCoW was first implemented in DSDM, but it has now spread to many other sectors, including product design, engineering, and even business strategy and software development.

In software development, for example, it is frequently used to organise sprints or iterations such that the *must have* features are finished before moving on to less important functionality. In this way, teams can have greater control over scope creep and make sure they offer value even under pressure by focussing on what is really essential for the project to succeed.

One of the main benefits of this technique is its ability to improve communication between project teams and stakeholders [163, 164]. It promotes a shared understanding of what is required for a project to be successful by clearly prioritising goals. In addition, it offers a structured strategy for settling alternatives in the event of unexpected limitations such as insufficient resources or modified project timelines. Teams are able to modify their emphasis without compromising their primary goals due to the adaptability which the *could have* and *won't have* categories offer.

Apart from its usefulness in agile environments, MoSCoW has also been applied in more general project management contexts. By defining priorities, teams can effectively allocate resources and concentrate on producing high-impact results [164]. It also helps to steer clear of typical project mistakes, such as overcommitting teams to high-priority activities or mismanagement of stakeholder expectations.

Although this approach has some advantages, there are also drawbacks. The subjective nature involved in requirement categorisation is one of the major problems [164]. Some stakeholders may feel that something is a *should have* or even a *could have* while others may consider it a *must have*. In challenging projects involving several partners with different priorities, this might result in disagreements or conflicts.

Another limitation is the potential overuse of the *must have* category [160]. The tendency to include too many requirements in this category is common, especially when stakeholders have a strong desire to have their specific demands satisfied [160]. This could result in an unrealistic project scope, making it difficult for teams to complete all *must have* tasks in the allotted time or budget. Therefore, a failure to maintain appropriate discipline while setting priorities can compromise the effectiveness of the technique.

The MoSCoW technique can also become challenging in complex projects with a lot of interrelated needs [163]. Managing and continually improving the prioritisation of requirements may become difficult as projects grow in scope, particularly when needs change over time as a result of changing business demands or improvements in technology. In these situations, additional prioritisation strategies such as cost-benefit analyses or weighted scoring models could be needed to support MoSCoW and offer a more sophisticated evaluation of priorities.

A.2 Kano Model

The Kano Model is a framework promoting client satisfaction that was created in 1984 with the goal to help business entities determine which features will satisfy clients and which may be irrelevant or cause disappointment, it focusses on identifying and classifying product features according to their influence on client satisfaction [165, 166].

The model acknowledges that not all features have the same impact on the public's view and makes a distinction between different products and their needs. Through feature classification into basic, performance, and excitement categories, the Kano Model offers

a sophisticated approach on how feature prioritisation and product design might affect client satisfaction and, eventually, business success [164, 166].

This technique classifies product characteristics into five distinct categories: *must be* (essential and expected by clients), *one-dimensional* (which linearly increases satisfaction), *attractive* (which, when present, attracts clients), *indifferent* (which do not influence satisfaction), and *reverse* (which, when present, create disappointment) [167, 168].

The main advantage is that it gives teams insight into how different features influence client happiness, allowing them to focus their efforts on areas that will have the most impact on client loyalty and competitive distinction. Numerous sectors use the Kano Model extensively, but it has the greatest adoption in marketing [169], user experience design [170, 171], and product development [171, 172].

In marketing, the technique helps highlight the features that could serve as Unique Selling Points (USPs) [169]. Client interest and competitive distinction are frequently driven by excitement requirements [169]. Client feedback analysis also relies on this model [170]. Product teams can make better judgements about which features to prioritise, maintain, or improve by classifying client feedback into different feature categories.

In product development, the Kano Model is used to achieve a balance between addressing performance demands (to satisfy and exceed client expectations) and basic needs (to prevent client discontent) [172]. In software development, for example, performance demands can be more concerned with speed or advanced functionality, whereas basic requirements might include functional reliability.

The Kano Model also has drawbacks. The constantly evolving expectations of clients are one of the main challenges [173, 174]. Touch screens are a good example of an earlier exciting feature, but have since become a standard feature, falling into the category of basic needs.

Furthermore, the way the model classifies features may be a little arbitrary because different clients or market segments might evaluate particular characteristics differently [174]. In global or diversified marketplaces, a characteristic that may delight a particular group of consumers may be considered a basic need by another [174]. This diversity makes decision-making more challenging and could require additional feature categorisation among different clients.

The Kano Model may not apply as well in contexts where it can be challenging to innovate or add novel features due to financial or technological constraints [173]. In some cases, it could be necessary to keep the emphasis on satisfying essential and functional requirements, even if doing so reduces the product's potential to surprise and delight clients.

A.3 Eisenhower Matrix

The Eisenhower Matrix, also known as the Urgent-Important Matrix, is a prioritisation framework that assists team members in managing tasks and responsibilities by categorising them based on their urgency and importance. It is attributed to Dwight D. Eisenhower, who was recognised for prioritising efficient time management.

This matrix categorises activities into four quadrants based on two factors: importance (whether the activity adds to long-term goals or objectives) and urgency (whether the activity requires urgent focus).

The first quadrant includes requirements that are urgent and important. These are crucial tasks that require immediate attention and are closely related to the achievement of the main objectives. This category can include responding to emergencies, meeting deadlines, or dealing with unexpected challenges that could have severe consequences if left untreated.

The second quadrant contains requirements that are essential but not urgent. Although these assignments do not always require immediate attention, they are important for long-term performance and achievement of goals. This quadrant frequently includes strategic planning, skill development, connection building, and other proactive actions. Although these chores may be postponed without immediate effects, it is critical to schedule them and assign time to ensure that they do not become lost in favour of additional concerns.

The third quadrant contains requirements that are urgent but not important. These requirements often require immediate attention but do not significantly contribute to long-term goals or organisational development. Typical scenarios are delays, irrelevant communications, or modest requests from others that do not align with fundamental priorities. The recommended approach for these quadrant requirements is to assign them wherever possible or to manage them efficiently to reduce their influence on more important activities.

Finally, the fourth quadrant has requirements that are neither urgent nor important. These requirements tend to be alternatives or time wasters that make no tangible contribution to personal or professional goals. The Eisenhower Matrix distinguishes between these aspects, allowing team members to focus on high-value tasks, avoid redundant distractions, and manage time more efficiently.

The Eisenhower Matrix is often employed for personal productivity, project management, and team cooperation. One of its key benefits is its capacity to encourage deliberate, proactive decision making, allowing individuals and organisations to focus on requirements that provide long-term value rather than reactive, low-value activities.

In project management, the matrix aids in prioritising operations within complicated processes by recognising which operations are crucial and time-sensitive. For example, in software development, fixing critical system weaknesses (Quadrant I) takes precedence over future planning activities (Quadrant II), although routine administrative responsibilities can be regularly assigned (Quadrant III).

The matrix also makes assignment easier inside teams, ensuring that team members focus on high-value work while less significant activities are given elsewhere. Furthermore, by identifying tasks in Quadrant IV, organisations can reduce redundancies while increasing total productivity.

Although the Eisenhower Matrix is a simple and useful tool, it has boundaries. A significant issue is the subjective nature of evaluating the urgency and relevance of activities. What someone believes urgent or important may differ from another's, leading to possible conflict within the team. Another restriction is the constantly shifting nature of the activities. A task that is initially classified as "Not Urgent" may become urgent over time

if left left behind, requiring periodic re-evaluation of priorities.

Furthermore, the Eisenhower Matrix emphasises on individual activities rather than their interdependencies. Prioritising tasks in isolation in larger projects with several interconnected activities can ignore the larger implications of specific actions. Furthermore, the matrix places a strong focus on time management, which may not always coincide with resource limits or organisational goals. For example, even if a task is urgent and critical, insufficient resources can prohibit immediate action. As a result, the matrix must be used in combination with other resource availability-based decision-making frameworks.

A.4 RICE Scoring Model

The RICE Scoring Model is a prioritisation framework developed to help product teams make objective judgements about which features, projects, or initiatives to pursue. The acronym RICE stands for Reach, Impact, Confidence, and Effort, which are four key components applied to evaluate tasks. Each of these components plays an important role in determining the potential value and feasibility of a project or feature, leading to a comprehensive approach to task prioritisation.

The first component, reach, aims to measure the magnitude of the effect of a project. It predicts the number of individuals who will be influenced by the feature or project within a certain timeframe. The second component of the RICE model is impact, a measure of how much a product or project will benefit all those it reaches. While impact can be challenging to define precisely, it is usually evaluated using qualitative descriptions. These qualitative decisions are then transformed into numerical scores to allow comparative between efforts. High-impact initiatives are often strongly aligned with long-term organisational goals, making this factor critical for strategic decision making.

Confidence, the third component of the model, indicates the level of knowledge of the team in their estimations of reach, impact, and effort. This component is a vital risk mitigation tool that allows teams to modify priorities based on the accuracy of their estimates. Confidence is measured as a percentage, with larger percentages representing greater security. The last component, effort, estimates the resources needed to complete the feature. Importantly, effort acts as a denominator in the RICE formula (see Figure A.1), which means that tasks that require less effort will therefore score higher for prioritisation. This technique guarantees that the model strikes a balance between the prospective effect and resource efficiency, encouraging a realistic approach to project selection.

$$Score = \frac{Reach \times Impact \times Confidence}{Effort}$$

Figure A.1: RICE formula for feature prioritisation.

The RICE model is frequently used in product development, project management, and strategic planning due to its balanced, quantitative nature. Its main benefit is that it allows teams to prioritise tasks based on multiple factors rather than just urgency or stakeholder viewpoints.

The method supports teams in prioritising features that are projected to have the largest impact with the least amount of work, which is important when managing limited resources. It offers an organised approach to evaluating activities inside complicated projects, ensuring that resources are allocated efficiently. The RICE model also aids in strategic planning by linking efforts to organisational goals. Furthermore, confidence criteria promote a data-driven approach, which reduces the chance of selecting initiatives with low certainty or high risk.

In software development, for example, if a team is evaluating multiple feature upgrades, RICE scores can be used to evaluate which changes would benefit the most users, increasing customer satisfaction and product adoption rates.

A drawback of this approach is the subjectivity involved in measuring reach, effect, and effort. Different team members may interpret these criteria differently, which might lead to inconsistent scoring. In addition, the model assumes that all tasks are independent and does not take into account project interdependence. To address this, teams often require calibration meetings to ensure alignment with scoring standards, which can be time consuming.

Furthermore, while confidence criteria are useful for risk assessment, it can be difficult to measure effectively, particularly when data is few or teams lack experience evaluating impact and reach. This might lead to overestimated or deflated confidence levels, skewing the prioritising process. Finally, the RICE model is fundamentally resource-focused and does not take into account other strategic elements like brand alignment or competitive positioning, which may be essential in certain scenarios. As a result, teams might consider combining RICE with other frameworks, such as the Kano Model for customer satisfaction.

A.5 A Comparative Overview of Prioritisation Techniques

In addition to the popular techniques described, organisations can explore different prioritisation techniques based on the project's setting, resources, and goals. Other alternatives include the ICE Scoring Model, the Value vs. Complexity Matrix, and the Weighted Scoring Model.

Each of these techniques provides a distinct method for prioritising activities and projects. For example, the ICE Scoring Model simplifies the RICE method by emphasising impact, confidence, and ease. The Value *versus* difficulty Matrix employs a two-dimensional grid to evaluate the work value and difficulty. The Weighted Scoring Model, on the other hand, enables for more flexible customisation of the assessment criteria to meet business requirements.

Although each technique has its own set of advantages, identifying the best technique is influenced by the project's needs, timeline, and team dynamics. For guidance in identifying the most appropriate method, a comparative table (Table A.1) summarises the characteristics, advantages, and disadvantages of the setting priorities strategies covered in this section.

Table A.1: Comparison of prioritisation techniques: key criteria, use cases, advantages, disadvantages.

Technique	Criteria	Use Cases	Advantages	Disadvantages
MoSCoW	Must have Should have Could have Won't have	Agile project management	Simple and understandable Efficient for scope management	Subjective nature Lacks quantitative prioritising
Kano Model	Must Be One Dimensional Attractive Indifferent Reverse	Product Design and Client Satisfation	Concentrates on consumer happiness Useful for feature analysis	Limited strategic or operational prioritisation
Eisenhower Matrix	Urgency Importance	Task and time management	Helpful for routine tasks prioritising Straightforward standards	Subjective nature Lack of capacity to evaluate larger projects
RICE Score Method	Reach Impact Confidence Effort	Product and project management	Balanced approach Tangible Objective	Subjective nature Interdependencies are not considered adequately
ICE Score Method	Impact Confidence Ease	Quick prioritisation decisions	Beneficial for urgent need Quick and simplified version of RICE method	Potential indifference of fundamental components Need for constant calibration
Value *versus* Complexity	Value Complexity	Project roadmaps and resource allocation	Aids in identifying low complexity and highly valuable objects Visual approach	Restricted to the value complexity axis Could overlook focused on clients objectives
Weighted Scoring Model	Personalised standards according to organisational goals	Prioritising strategic and extensive projects	Highly flexible and configurable It can fit different goals	Costs time to assign weights and set up criteria Can be challenging

The prioritising techniques discussed in this section can be used in many different project configurations and demands since each has unique benefits and drawbacks. Given its simple classification, the MoSCoW technique performs well in agile environments, enabling teams to choose priorities according to their needs and available resources.

Conversely, the Kano Model offers insightful data on client happiness and can be used

to identify features that encourage user engagement. The RICE and ICE models provide data-driven frameworks that satisfy the demands of product management for wider reach and impact evaluation, while the Eisenhower Matrix is efficient at simplifying decision-making for daily priorities in task management.

The weighted scoring model and value *versus* complexity matrix allow for an in-depth analysis of multiple factors, providing flexibility for the alignment of strategies and the balance of resources over time. The present study highlights the relevance of identifying a prioritisation framework that is consistent with the project scope, stakeholder requirements, and overall strategic objectives by reviewing the advantages and potential applications of each technique.

Appendix B

Mapping of Test Parameters and Input Values for Smart Contract Testing

Table B.1: Mapping of test parameters and input values.

Type	Description	Designation	Value
Contractor Address	Each address identifies a distinct Contractor with the authority to carry out specific duties on the devices.	C1 C2 C3	0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c
User Address	The addresses in the system stand in for the users. These individuals are the owners or those in charge of the lent devices.	U1 U2 U3	0x583031D1113aD414F02576BD6afaBfb302140225 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 0x77fB05fb6e9d2b43B50a0E4d3bf6dc0353874Bc4
Supplier Address	The address representing the Supplier of the equipment.	Supplier	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Laptops' Serial Number	Each device's own serial number.	SN1 SN2 SN3	SN1U1C1 SN2U2C1 SN3U3C2
Laptop's Identifier Number	Each device in the system is assigned a unique identification number that is used for device management and tracking. This data is connected to the serial number of the device.	ID1 ID2 ID3	520cc938-c90b-4a76-b81f-1d3b11f6d433 baf135d3-bc6b-45d2-9e09-725c63a515b6 87e7d82a-2ade-4865-9781-c11482f666d2

State	Indicates the current state of a device, represented by either '0' (unlocked) or '1' (locked).	`S0` `S1`	`0` `1`
Lease	The timestamp representing the validity of the lease associated with the device.	`Lease`	`1619885200`

HMAC hashes	The HMAC values used to guarantee security when locking a device.	`H1`
		`H2`
		`H3`
			`eb5d8510ac82a60cf7a86f3b1c4e2e17b00bb0f6d208 05d4e324f20633fdd109` *(SN1:ID1:Lease:S1)*
			`189e4a26edeb345596a87171a651b3c255a957b3cdce caf4605d50a2866fef29` *(SN2:ID2:Lease:S1)*
			`7bfcd932e7417b7f405091424936d82a09ea326a6592 b671eb8f35d486188761` *(SN3:ID3:Lease:S1)*

	The HMAC values used to guarantee security when unlocking a device.	`H5`	`ccff0b29b90fec92b610bb13daf076a86ce2fec02d2f 887a72a9eb0216781e05` *(SN2:ID2:Lease:S0)*

Appendix C

Smart Contract Development with ChainLink Oracle

```
1  pragma solidity ^0.8.3;
2
3  import "@chainlink/contracts/src/v0.8/ChainlinkClient.sol";
4
5  contract DeviceManagement is ChainlinkClient {
6      using Chainlink for Chainlink.Request;
7
8      struct Device {
9          bool isRegistered;
10         bool isLocked;
11         address user;
12         address contractor;
13         uint registrationTimestamp;
14     }
15
16     address public supplier;
17     mapping(string => Device) public devices;
18
19     string private serialNumber;
20     bool private requestProcessed;
21
22     event DeviceRegistered(string id, address user, address contractor);
23     event DeviceStatusChanged(string id, bool isLocked);
24
25     /**
26      * @notice Initialize the link token and target oracle
27      *
28      * Sepolia Testnet details:
29      * Link Token: 0x779877A7B0D9E8603169DdbD7836e478b4624789
30      * Oracle: 0x6090149792dAAeE9D1D568c9f9a6F6B46AA29eFD (Chainlink DevRel)
31      * jobId: 7d80a6386ef543a3abb52817f6707e3b // string
32      *
33      */
34     constructor () {
35         supplier = msg.sender; // to change to softi9's address
36         _setChainlinkToken(0x779877A7B0D9E8603169DdbD7836e478b4624789);
37         _setChainlinkOracle(0x6090149792dAAeE9D1D568c9f9a6F6B46AA29eFD);
38     }
39
```

```
40      modifier onlySupplier() {
41          require(msg.sender == supplier, "Permission denied, is not the supplier.");
42          _;
43      }
44
45      modifier onlySupplierOrContractor(string memory _id) {
46          require(msg.sender == supplier || msg.sender == devices[_id].contractor, "Permission
                denied, is not the supplier nor contractor.");
47          _;
48      }
49
50      function registerDevice(string memory _id, address _userAddress, address
            _contractorAddress) public onlySupplier {
51          require(!devices[_id].isRegistered, "Device already registered.");
52          devices[_id] = Device(true, false, _userAddress, _contractorAddress, block.timestamp);
53          emit DeviceRegistered(_id, _userAddress, _contractorAddress);
54      }
55
56      /**
57      * @notice Check if the device is registered and get the status
58      * Pre-requisite: Run the requestSerialNumber method first and wait a few seconds.
59      */
60      function checkDeviceStatus(string memory _id) public view returns (bool, string
            memory, bytes32) {
61          require(devices[_id].isRegistered, "Device is not registered.");
62          require(requestProcessed, "The request has not yet been processed. Try again soon.");
63
64          // fazer pedido para ter serial number
65          string memory nonce = this.generateUUIDformat();
66          bytes32 hmac = this.generateHMAC(serialNumber, _id, nonce, devices[_id].isLocked);
67
68          return (devices[_id].isLocked, nonce, hmac);
69      }
70
71      function changeDeviceStatus(string memory _id, bool _locked) public
            onlySupplierOrContractor(_id) {
72          require(devices[_id].isRegistered, "Device is not registered.");
73          devices[_id].isLocked = _locked;
74          emit DeviceStatusChanged(_id, _locked);
75      }
76
77      // Oracle Functions
78      function requestSerialNumber(string memory _id) public {
79          Chainlink.Request memory req = _buildChainlinkRequest('7
                d80a6386ef543a3abb52817f6707e3b', address(this), this.fulfill.selector);
80          string memory url = string.concat('https://webportaldaas.free.beeceptor.com/', _id);
81
82          // Set the URL to perform the GET request on
83          req._add('get', url);
84          // Set the path to find the desired data in the API response
85          req._add('path', 'serial_number');
86
87          // Sends the request
88          _sendChainlinkRequest(req, (1 * LINK_DIVISIBILITY) / 10); // 0,1 * 10**18 (Varies
                by network and job)
89      }
90
```

```
91      function fulfill(bytes32 _requestId, string memory _serialNumber) external
            recordChainlinkFulfillment(_requestId) {
92          serialNumber = _serialNumber;
93          requestProcessed = true;
94      }
95
96      // Extra Functions
97      function generateUUIDformat() external view returns (string memory uuid){
98          // generate 256bits' hash with block timestamp, difficulty and current gasprice
99          bytes32 buf = keccak256(abi.encode(block.timestamp, block.prevrandao, tx.gasprice));
100         // converts bits for bit manipulation
101         uint128 _uuid = uint128(bytes16(buf));
102
103         // set byte 6 to version 4
104         _uuid = _uuid & 0xFFFFFFFFFFFFFF00FFFFFFFFFFFFFF00FF; // clears the bits
105         _uuid |= uint128(uint8(0x40)) << 56; // set the version bits
106
107         // set byte 8 to variant 4 (randomness)
108         _uuid = _uuid & 0xFFFFFFFFFFFFFFF3FFFFFFFFFFFFFFFF; // clears the bits
109         _uuid |= uint128(uint8(0x40)) << 48; // set the variant bits
110
111         // converts back to bytes16
112         return bytesToHex(bytes16(_uuid));
113     }
114
115     function bytesToHex(bytes16 data) private pure returns (string memory) {
116         bytes memory hexChars = "0123456789abcdef";
117         bytes memory result = new bytes(2 * 16); // 2 hex chars per byte
118         for (uint i = 0; i < 16; i++) {
119             result[2 * i] = hexChars[uint(uint8(data[i] >> 4))];
120             result[2 * i + 1] = hexChars[uint(uint8(data[i] & 0x0f))];
121         }
122
123         return string(result);
124     }
125
126     function generateHMAC(string memory _serialNumber, string memory _id, string
            memory _nonce, bool _state) external pure returns(bytes32) {
127         string memory message = string.concat(_serialNumber, ":", _id, ":", _nonce, ":", (_state
                ? "1" : "0"));
128         string memory key = "kEy_SK_YeIck4V25a7d8b1pzW6loR=zu&t0jUhkD&
                j50H8J9Jz8wq_@_5NwEgaCi";
129         bytes memory b_key = bytes(key);
130
131         if (b_key.length > 64) {
132             b_key = abi.encode(sha256(b_key));
133         } else {
134             if (b_key.length < 64) {
135                 bytes memory new_key = new bytes(64);
136                 for (uint i = 0; i < b_key.length; i++) {
137                     new_key[i] = b_key[i];
138                 }
139                 b_key = new_key;
140             }
141         }
142
143         bytes memory ipad = new bytes(64);
144         bytes memory opad = new bytes(64);
```

```
145
146         for (uint i=0; i < 64; i++) {
147             ipad[i] = b_key[i] ^ 0x36;
148             opad[i] = b_key[i] ^ 0x5c;
149         }
150
151         // sha256(opad || sha256(ipad || mensagem))
152         //inner: sha256(ipad || mensagem) & final: sha256(opad || innerhash)
153         bytes32 innerhash = sha256(abi.encodePacked(ipad, message));
154         return sha256(abi.encodePacked(opad, innerhash));
155     }
156
157 }
```