



# ANALYSING SECURITY RISKS IN THE ARCHITECTURE OF BLOCKCHAIN-BASED SYSTEMS AND SMART CONTRACTS

By

SABREEN AHMADJEE

A thesis submitted to  
the University of Birmingham  
for the degree of  
DOCTOR OF PHILOSOPHY

School of Computer Science  
College of Engineering and Physical Sciences  
University of Birmingham  
May 2023

UNIVERSITY OF  
BIRMINGHAM

**University of Birmingham Research Archive**

**e-theses repository**

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

---

## ABSTRACT

Blockchain is a revolutionary technology that aims to provide secure, decentralised distributed systems where users can share, store and verify transactional data without the need for a central authority to perform authentication or verification. However, the widespread use of this technology, especially after the emergence of smart contracts, the blockchain-based computer programs, has incentivised attackers to exploit its existing security challenges. Moreover, the distinguishing properties and internal complex structure of the technology increase the chance of making poorly informed architectural design decisions, which might introduce security weaknesses to the systems supported by blockchain. Malicious attacks with severe consequences result from weak designs in blockchain systems and smart contracts. For instance, in recent years, the decentralised finance (DeFi) sector experienced a series of high-profile attacks resulting in multi million-dollar losses. These concerns advocate the need for architecture-centric approaches to abstract the complexity of the blockchain components, address architectural-level security risks specific to smart contracts and blockchain-based systems, and make the development of such systems secure, easier, and more organised.

Within this context, we propose architectural-centric analysis approaches for security risk assessment that allow security to be incorporated into blockchain-based systems from the ground up. We present a classification of the state-of-the-art that provides secure architectural design approaches and supports blockchain security risk assessment methods. We also provide a taxonomy of blockchain architecture design decisions and map these decisions to related security attacks and threats. Additionally, we explore the use of the security

---

technical debt metaphor to identify smart contracts' security issues related to sub-optimal design decisions and to estimate the accumulation of the security risk ramifications. By leveraging security debt, we contribute to a technical debt-aware approach to design secure smart contracts, and we provide a decision support model to select a secure and cost-effective blockchain oracle platform.

As part of the demonstration and evaluation, we use three case studies that represent blockchain-based systems and decentralised applications; we leverage a dataset of representative vulnerable smart contracts; and we distribute a survey and conduct interviews with smart contract experts to assess and refine our approaches. The significance of this work is that it uses architecture-centric approaches that provide a systematic guide for blockchain systems and smart contract software engineers to make justifiable design decisions that result in more secure implementations and reduced security complications.

## ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Rami Bahsoon, for his unwavering support in helping me grow as a confident researcher. From the very beginning, he encouraged me to develop my own ideas, provided guidance in structuring and writing my own papers, and helped me publish my work in some of the top journals and conferences. Rami, I cannot express how much I appreciate your mentorship throughout my PhD journey, your valuable insights, and most importantly, your empathy and patience.

I want to express my sincere gratitude to my thesis group members, Dr. Dave Parker and Dr. Leandro L. Minku, for their ongoing feedback and valuable insights on my research. Additionally, I would like to thank Prof. Rick Kazman and Dr. Siamak Farshidi for generously sharing their time and expertise, providing insightful conversations, and offering constructive feedback on the paper we co-authored. Finally, a special thanks to Dr. Carlos Mera-Gómez for his useful inputs and suggestions, as well as his valuable advice and guidance throughout my PhD journey.

I am confident that I could not have completed this PhD research without the support and love of my family. I am indebted to my parents, Mohammedsiraj and Safia, and my sister, Saleha, for their support and continuous prayers; to my husband, Ahmed, for his endless support, love, and encouragement throughout my PhD journey; to my brothers; and my friends in Birmingham, Weam, Hanouf, and Ohoud; in Ecuador, Mercy; and in Saudi Arabia, Shorouq and Wasaif.

# Contents

	<b>Page</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Problems to be Addressed . . . . .	4
1.3 Research Methodology . . . . .	6
1.4 Research Contributions . . . . .	8
1.5 Thesis Structure . . . . .	10
1.6 Publications Linked to this Thesis . . . . .	12
<b>2 Security Architectural Approaches for Blockchain Systems: A Systematic Literature Review</b>	<b>14</b>
2.1 Overview . . . . .	14
2.2 Background . . . . .	16
2.2.1 Blockchain Overview . . . . .	16
2.2.2 Smart Contracts . . . . .	17
2.3 Research Methodology . . . . .	17
2.3.1 Research Questions . . . . .	18
2.3.2 Search Strategy . . . . .	18
2.3.3 Study Selection . . . . .	20
2.3.4 Quality Assessment . . . . .	22
2.3.5 Data Extraction . . . . .	23

---

2.3.6	Data Synthesis . . . . .	24
2.4	Results . . . . .	25
2.4.1	Demographics of Selected Studies . . . . .	25
2.4.2	Purpose of Selected Studies . . . . .	26
2.5	Analysis of the Selected Publications . . . . .	27
2.5.1	Security Architectural Design Approaches (RQ1) . . . . .	27
2.5.2	Blockchain Security Risk Assessment Methods (RQ2) . . . . .	34
2.6	Discussion . . . . .	38
2.6.1	An Outlook for Future Directions . . . . .	38
2.6.2	Gap Analysis . . . . .	40
2.6.3	Threats to Validity . . . . .	41
2.7	Related Work . . . . .	42
2.8	Summary . . . . .	45
<b>3</b>	<b>Blockchain Major Architectural Design Decisions and their Security At-</b>	
	<b>tacks and Threats</b>	<b>47</b>
3.1	Overview . . . . .	48
3.2	Research Methodology . . . . .	50
3.2.1	Surveying the Literature . . . . .	50
3.2.2	Mapping Approach . . . . .	53
3.3	Taxonomy of Dimensions for Architectural Decisions in Blockchain-based Sys-	
	tems . . . . .	55
3.3.1	Blockchain Access Type . . . . .	57
3.3.2	Storage and Computation . . . . .	65
3.3.3	Consensus Mechanism . . . . .	67
3.3.4	Block Configuration . . . . .	69
3.3.5	Key Management . . . . .	71

---

3.3.6	Cryptographic Primitives . . . . .	73
3.3.7	Chain Structures . . . . .	76
3.3.8	Node Architecture . . . . .	79
3.3.9	Smart Contracts . . . . .	80
3.4	Mapping Threats and Attacks with Blockchain Architectural Decisions . . .	82
3.4.1	Attacks and Threats Classification in Blockchain-based Systems . . .	84
3.5	Application of the taxonomy and mapping approach . . . . .	107
3.5.1	Key management as a case to demonstrate instantiating the taxonomy and its mapping to attacks and threats . . . . .	108
3.6	Discussion . . . . .	113
3.6.1	Validation . . . . .	113
3.6.2	Threats to Validity . . . . .	115
3.7	Related Work . . . . .	116
3.8	Finding and Summary . . . . .	121
3.8.1	Summary . . . . .	122
<b>4</b>	<b>A Novel Approach for Assessing Smart Contracts' Security Technical Debts</b>	<b>123</b>
4.1	Overview . . . . .	124
4.2	Preliminaries . . . . .	126
4.2.1	Ethereum Platform . . . . .	126
4.2.2	The Common Weakness Scoring System . . . . .	127
4.2.3	Technical Debt . . . . .	128
4.3	Our Approach for Assessing Technical Debts . . . . .	128
4.3.1	Identification of Security Design vulnerabilities . . . . .	129
4.3.2	Measurement of Negative Consequences of Design vulnerability in Smart Contracts . . . . .	135



---

4.3.3	Replication Package for Replicability . . . . .	140
4.4	Experimentation . . . . .	140
4.4.1	Experiment Setup . . . . .	141
4.4.2	Experimental Study . . . . .	141
4.5	Results and Discussion . . . . .	145
4.5.1	Identification of Design Vulnerabilities (RQ1) . . . . .	145
4.5.2	Estimation of Negative Consequences (RQ2) . . . . .	148
4.6	Evaluation . . . . .	151
4.6.1	Survey Questions . . . . .	151
4.6.2	Key Findings . . . . .	151
4.6.3	Threats to Validity . . . . .	154
4.7	Related Work . . . . .	156
4.8	Summary . . . . .	158
<b>5</b>	<b>Decision Support Model for Blockchain Oracle Platform Selection</b>	<b>159</b>
5.1	Overview . . . . .	159
5.2	Preliminaries . . . . .	162
5.3	Research approach . . . . .	164
5.4	MCDM For Blockchain Oracle Platform Selection . . . . .	165
5.4.1	Knowledge Acquisition and Mapping . . . . .	166
5.4.2	Inference Engine . . . . .	175
5.5	Application of the model using two case studies . . . . .	182
5.5.1	Application one: Dynamic Legal Agreements (DLA) . . . . .	185
5.5.2	Application two: Decentralised Auctions (DA) . . . . .	188
5.5.3	Results and Analysis . . . . .	191
5.6	Evaluation . . . . .	197
5.6.1	Sensitivity Analysis . . . . .	197

5.6.2	Comparison with Previous Work . . . . .	199
5.6.3	Comparison with Ad-hoc Methods . . . . .	201
5.7	Discussion . . . . .	204
5.7.1	Threats to Validity . . . . .	206
5.8	Summary . . . . .	208
<b>6</b>	<b>Reflection and Appraisal</b>	<b>210</b>
6.1	Overview . . . . .	210
6.2	Analysis of the Research Questions . . . . .	210
6.3	Reflection on the Evaluation . . . . .	214
6.3.1	Evaluation Criteria . . . . .	214
<b>7</b>	<b>Conclusion Remarks and Future Work</b>	<b>220</b>
7.1	Contributions . . . . .	220
7.2	Future Work . . . . .	222
7.2.1	Enhancing Current Security Analysis Tools . . . . .	222
7.2.2	Extending the Approaches Proposed . . . . .	223
7.2.3	Fully Automating the Approaches Proposed . . . . .	224
7.2.4	Addressing Security and Architectural Dimension Limitations. . . . .	225
<b>A</b>	<b>Quality Assessment of Selected Studies</b>	<b>227</b>
<b>B</b>	<b>Mapping Selected Publications with the Taxonomy</b>	<b>229</b>
<b>C</b>	<b>Survey Questioners and Responses</b>	<b>232</b>
<b>D</b>	<b>Interviewee Structure</b>	<b>239</b>
<b>E</b>	<b>Linking Oracle Features with Platforms and Quality Attributes</b>	<b>241</b>

References

243

# List of Figures

1.1	Structure of the Chapters and Related Research Questions . . . . .	12
2.1	Search and Academic Papers Selection Procedure . . . . .	20
2.2	Quality Scores of the Included Studies . . . . .	24
2.3	Chart of Focus Areas of Selected Studies . . . . .	26
2.4	Chart of the Classification of Selected Publications . . . . .	35
3.1	Search and Paper Selection Procedure . . . . .	53
3.2	Taxonomy of Major Dimensions for Blockchain Architectural Decisions . . . . .	56
3.3	Mapping Threats, Attacks and Architectural Decisions . . . . .	84
3.4	Architectural Decisions for EMR System . . . . .	111
4.1	Experiment Execution Steps . . . . .	144
4.2	Identified Vulnerabilities per Category by each Tool. . . . .	148
4.3	Principal and Interest of Ten Vulnerable Contracts. . . . .	149
5.1	Decision Model Structure for Oracle Selection . . . . .	166
5.2	Distribution of Chosen Studies Throughout the Document Analysis Phase . . . . .	167
5.3	A Sample of the Mapping Between: <b>a.</b> the Boolean Oracle Features and Platforms and <b>b.</b> the Boolean Oracle Features and Quality Attributes . . . . .	174
5.4	The Percentages of the Q that are Desired for each Application and the Percentages that Tellor Provides . . . . .	194

# List of Tables

2.1	Extracted Data Items . . . . .	23
2.2	Publication Venues of Selected Studies . . . . .	25
2.3	Main Purpose and Focus Area of the Primary Studies . . . . .	28
2.4	Categories of Secure Architectural Design Approaches . . . . .	37
2.5	Blockchain Security Risk Assessment Methods . . . . .	38
2.6	Related Works, their Methods, and Focus Area . . . . .	45
3.1	Selected Articles . . . . .	52
3.2	Quality Attributes per Dimension of Architectural Decisions . . . . .	58
3.3	Linking Attack Categories with Blockchain Architectural Decisions . . . . .	93
3.4	Linking STRIDE with Blockchain Architectural Decisions . . . . .	98
3.5	Linking Attacks with Threats . . . . .	103
3.6	Analysis of Alternative Decisions for EMR System . . . . .	109
3.7	Summary of Related Work . . . . .	117
4.1	An Example of Vulnerabilities Mapping for DoS to CWE Categorisation . . . . .	133
4.2	Description of Design Flaws Categories . . . . .	134
4.3	Experiment Dataset. For each Vulnerable Contract, We Provide Design Flaws Categories of its Flaws (DFC), Number of Design Vulnerabilities (Vulns), and Lines of Code (LOC). . . . .	142
4.4	Overall Severity Level . . . . .	147
4.5	Survey Questions . . . . .	153

5.1	Explanation of Attacks that Target Centralised (C) and/or Decentralised (DC) Oracle Types . . . . .	176
5.2	Deployment and Transactional (tx.) Cost of Integrating each Oracle Platform	182
5.3	STDI Quantification of Attacks for Price Feeds Smart Contract . . . . .	183
5.4	Final Cost and Security Value of Oracle Platforms . . . . .	183
5.5	Entire List of Oracle Features of the Two Case Studies . . . . .	192
5.6	Inference Engine Outcomes based on Features' Priorities and Pareto-Optimal Solutions (POS) . . . . .	195
5.7	Sensitivity Analysis of WSM Results of DLA Application . . . . .	196
5.8	Sensitivity Analysis of STDI Results . . . . .	202
5.9	Our Work in Comparison to Previous Works . . . . .	203
5.10	Comparison of our Model and Ad-hoc Methods . . . . .	204
A.1	Quality Assessment . . . . .	228
B.1	Mapping Selected Publications with the Taxonomy . . . . .	230

# Chapter One

## Introduction

### 1.1 Overview

Blockchain is an append-only data structure that connects blocks of data consecutively following the chronological sequence, ensuring that this distributed ledger cannot be tampered with or cryptographically fabricated. Blockchain technology has received widespread attention in industry and academia since the success of Bitcoin, a seminal application based on this technology [376]. Smart contracts, pieces of programming code that enforce trustworthy transactions and agreements between several anonymous participants, are among the most crucial blockchain applications.

Indeed, the usage of blockchain and smart contracts has gone beyond cryptocurrency systems to underlie many dependable mainstream software systems, including finance, integrity verification, the Internet of Things (IoT), healthcare, data management, security, and privacy [55]. Organisations leverage blockchain as a critical component within software system architectures to provide more dependable and secure computation and storage [55, 325]. Despite this widespread adoption, blockchain is considered at an early stage of its development since the incubation period of software engineering technology usually requires

15 to 20 years to reach maturity [313, 285]. Thus, anticipating the attack landscape and the mindset of the adversary is still challenging. Therefore, to satisfy most security requirements, blockchain applications must be well-designed, and security risks surrounding blockchain must be recognised and assessed at an early development stage.

Although blockchain-based systems are not inherently secure [211, 152, 166], there is a lack of a systematic guide to designing secure systems based on blockchain technology. This situation leads to architects operating in an ad-hoc manner [276], relying on the wisdom and trust of peers, which may introduce security flaws to the system. Additionally, the main focus of current research is providing security at the code level [218, 294, 101], proposing techniques [145, 157] and developing tools [225, 248, 334] that detect contract code issues. However, concentrating just on code makes it challenging to obtain high system quality [297]. Architectural concerns can overwhelm the most thorough coding efforts, and neglecting such issues leads to vulnerable systems that can be easy to breach [176]. The consequent security breaches may span many dimensions leading to costly repairs to stabilise the security of the system (i.e. security maintenance costs), loss of trust in the system, and costs in terms of credibility and reputation damage. Moreover, exploiting the security vulnerabilities in blockchain systems may lead to significant financial losses [166, 40].

Malicious attacks with severe consequences result from weak designs in blockchain systems and smart contracts. As this technology gains traction, attackers are drawn to exploit its security gaps. In July 2020, a breach of Ledger’s website led to a massive data compromise. Attacker used this to phish users for crypto wallet keys and even sent ransom emails. Storing sensitive data insecurely enabled these attacks. This underscores the need for architectural secure approaches in building secure blockchain systems. A strong architectural foundation is pivotal in such methods, integrating security from the outset.

In this thesis, we contribute to architecture-centric analysis approaches for security



risk assessment in blockchain-based systems and smart contracts. *Architectural analysis* [297] is a structured method for examining the design decisions in a complex system concerning the desired quality attribute goals, such as security and cost-effectiveness. The idea behind architectural analysis for security is that it is too late to fix security design issues identified during coding or maintenance stages because doing so might lead to severe security risks and costly repairs. We aim to equip blockchain systems decision-makers, architects, and non-technical individuals with systematic approaches to building secure-by-design blockchain-based systems and smart contracts. *Secure-by-design* [192, 302] refers to a systematic process that ensures that a system and all its components are developed from the ground up with security quality attributes in mind. Our work focuses on secure-by-design processes that operate at the architectural design stage of blockchain systems and smart contracts.

To achieve our aim, this thesis first contributes to a systematic literature review that identifies and categorises existent approaches regarding architecting and designing secure blockchain-based systems and smart contracts. This review confirmed the lack of systematic architectural techniques for blockchain software engineers to make well-informed design decisions and build secure systems.

Second, the thesis investigates and classifies the key architectural decisions regarding blockchain-based systems and maps such decisions to potential security attacks and threats. We leveraged several techniques, including threat modelling, to identify and classify security issues surrounding blockchain architectural decisions. *Threat modelling* [359] is a systematic analysis of the design of a system. It helps to identify, rate, and prioritise design-level security threats.

Third, the thesis provides a technical debt-aware approach for assessing security design vulnerabilities in smart contracts. *Technical debt* is a metaphor devised to capture how the value of software engineering decisions evolves over time [240]. Specifically, the metaphor

helps identify the root causes of issues by highlighting the decisions that led to the debt. The metaphor also helps estimate the debt value and monitor the environmental trade-offs in which the decision is made [193]. Security debt approaches have proven to be applicable [292] and effective [167, 260] in assessing software security risks. Employing the security debt metaphor to express security risks assists in making such a risk more visible and, consequently, more justifiable [291].

We found that multiple security debts might ship to smart contracts from external third parties, such as off-chain data feeders, known as blockchain oracles. *Oracle* is an agent that collects and provides data from external resources to the contracts. The secure decision-making process becomes increasingly complex as the number of oracle alternatives and their features increases. Thus, finally, the thesis creates a decision support model for the oracle selection problem.

## 1.2 Problems to be Addressed

The architectural design of blockchain systems and smart contract applications should follow systematic approaches and adopt security architectural-centric analysis techniques. They make the security implications of design decisions more visible in the final architecture, making the system less vulnerable to potential threats and attacks that may emerge from simplified assumptions or unjustified architectural choices.

However, existing research and methods on blockchain systems and smart contracts security are mainly concerned with coding and testing, providing little insight into creating a secure blockchain architecture [101, 195, 218, 409]. For instance, Durieux et al. [101] systematically evaluated several state-of-the-art automated tools for analysing smart contracts code and discussed their accuracy and efficiency. Similarly, Kushwaha et al. [195]

reviewed smart contracts analysis tools and explored several code analysis techniques, such as symbolic execution and taint analysis. Liu et al. [218] proposed a taxonomy regarding the security verification of blockchain smart contracts. The study presents the pros and cons of each existing security verification technique. Zou et al. [409] conducted interviews with smart contract developers. However, the questions mainly concerned smart contract implementation and testing. Other works [306, 212, 152] focused on aggregating security vulnerabilities and attacks regarding blockchain and smart contracts. These studies have not considered assessing the security implications of such attacks.

This thesis bridges that gap by developing a suite of systematic architectural-centric approaches for assessing the security risks in blockchain-based systems and smart contracts, making novel use of security threat modelling and technical debt analysis.

We pose and address the following research questions to solve this issue:

- **RQ1:** a) What are the common security architectural design approaches used when architecting blockchain-based systems and smart contracts? b) What are the existing frameworks, models, and methodologies for security risk assessment in blockchain-based systems and smart contracts?
- **RQ2:** a) What are the architecturally significant design decisions in blockchain-based systems? b) How can potential threats and attacks be traced to blockchain architectural decisions and to which components?
- **RQ3:** a) How to identify design vulnerabilities in smart contracts? What are the specific analysis techniques and tools? b) How to quantify the impact of technical debts related to design vulnerabilities in smart contracts?
- **RQ4:** How can we advance the oracle selection problem by designing decision support models that assist in systematically selecting secure and cost-effective oracle platforms

feasible for decentralised applications?

Any system based on blockchain and smart contracts is prone to security risks [212, 60, 353]. Therefore, the security of blockchain systems calls for approaches capable of increasing the visibility of such risks and assessing their consequences to the entire system if left unaddressed. This work explores architectural-centric approaches from a security perspective, employee security risk assessment techniques, and security technical debt metaphor. They support the feasibility of our approaches in locating the design root of security issues and measuring the ramifications of sup-optimal decisions over time. Combining risk assessment techniques and technical debt processes produces a robust method for the assessment of both security risks and quality issues [291, 292].

### 1.3 Research Methodology

Design Science Research (DSR) is a problem-solving-focused research model [89]. It aims to create and assess artefacts to solve identified problems by facilitating the transformation from the current state, such as following ad hoc methods, to the desired state, such as employing systematic approaches. Since our work aims to create systematic architectural-centric analysis solutions that assist in making informed security architectural design decisions, we found that our research aligns with design science research. Therefore, we adopt the iterative process of Design Science Research Methodology (DSRM) proposed by Peffers et al. [275]. The primary steps appear below:

- **Problem identification and motivation:** Understanding security architecture analysis and its approaches in the context of blockchain-based systems is the initial step in our research. Thus, we conducted a Systematic Literature Review (SLR) to deepen

our knowledge of the subject and to identify potential future challenges. According to the results, we found a need to provide systemic approaches that allow software engineers to build secure blockchain-based systems. Therefore, we focused our research on the architectural-centric analysis of security approaches regarding blockchain-based systems.

- **Define the objectives for a solution:** The primary aim of this thesis is to devise systematic architectural analysis approaches that guide blockchain-based systems' software engineers in making informed architectural design decisions concerning security quality attributes. We intend to raise awareness of security issues, their root causes, and the implications for blockchain systems. This intention allows systems architects to make early interventions, starting from the inception and design stages. Furthermore, our approaches aim to assist in making feasible and secure design decisions that are also cost-efficient.
- **Design and development:** We have leveraged several well-known security techniques to develop our approaches, including threat modelling and security technical debt. These techniques effectively identify security issues and increase the visibility of their negative impacts on systems. In particular, the technical debt metaphor has proven to measure the impact of security weaknesses' exploitation in terms of damage to business value, the ramifications of incorrect decisions over time, and locating the design root of security vulnerabilities [168, 167]. The metaphor also allows for quantifying the monetary consequences of making a sup-optimal design decision. In this thesis, we establish the foundations for integrating the built-in decision support of technical debt analysis into blockchain-based systems.
- **Demonstration:** To demonstrate the applicability of our proposed approaches, several case studies that represent blockchain-based systems and decentralised applications have been used, including electronic medical record blockchain-based systems

[99], a dynamic legal agreement decentralised application [293], and a decentralised auctions application [264]. Additionally, we use a dataset of representative vulnerable smart contracts that are either actual vulnerable contracts or explicitly programmed to demonstrate a specific vulnerability.

- **Reflective Evaluation:** To reflect on the evaluation of this work, we follow the assessment strategy proposed by Kitchenham et al. [187]. Multiple subfeatures are adopted to assess our proposed approaches: (i) completeness and organisation to evaluate the approach’s documentation; (ii) ease of implementation and applicability to evaluate the quality of the approaches; (iii) usefulness, clarity, comparison with alternative approaches, and cost-effectiveness to evaluate the benefits of the approaches. We consulted experts through an online survey and interviews to assess and refine our approaches.

## 1.4 Research Contributions

The work described in this research contributes to the field of security architectural design decisions for blockchain and smart contract-based systems. In particular, the work contributes novel security architectural approaches that assist software engineers in designing secure and cost-effective blockchain and smart contract systems. Our work employed several methods of security-centric architectural analysis [297], including threat modelling classification, vulnerability assessment, security risk analysis, and attack surfaces. In addition, it uses the security technical debt metaphor [168] to quantify the implications and cost of potential security risks. In summary, this thesis makes the following contributions:

1. **A thorough analysis of existing literature that proposes architectural design approaches to architect secure blockchain systems and smart contracts.** We

conducted a systematic literature review (SLR) to identify existing approaches to architecting and designing secure blockchain-based systems and smart contracts. This review has established a lack of systematic techniques that provide a clear guide for creating secure blockchain and smart contract systems. Based on this review, we advocate architectural analysis of security approaches for blockchain systems. As a result, we formulated a set of security architectural design approaches to assist in making secure and optimal decisions when designing blockchain systems and smart contracts.

2. **A taxonomy of the primary dimensions of blockchain architectural decisions and a mapping approach that associates each dimension with potential security attacks and threats.** We conducted a review partially guided by the SLR method to derive this taxonomy, which defines, illustrates, and classifies the key architectural decisions regarding blockchain-based systems. It describes each architectural decision from the security perspective and discusses the quality attribute trade-offs entailed by each architectural choice. We mapped each dimension of our taxonomy with potential security attacks and their posed threats. Mapping the proposed taxonomy with the related security ramifications allows software architects to fully comprehend the impact and scope of the security challenges associated with developing blockchain systems.
3. **A technical debt-aware approach to designing secure smart contracts.** We formulated a technique for assessing security architectural design vulnerabilities in smart contracts. This approach contributes to a Vulnerability-oriented Architectural Analysis (VoAA) approach [297] for smart contracts. Our technique involves two steps: (i) identification of design vulnerabilities using security analysis techniques; and (ii) an estimation of the ramifications of the identified vulnerabilities leveraging the technical debt metaphor, its principal, and interest. Developers can use our approach to inform the design of more secure contracts and reduce unintentional debts caused by a lack of

awareness of security issues.

4. **A decision support model for blockchain oracle platform selection.** The model supports smart contract decision-makers in selecting a secure, cost-effective, and feasible oracle platform for their applications. We investigated existing blockchain oracle platform alternatives and analysed their related features. We formulated our model by leveraging Multi-Criteria Decision-Making (MCDM) techniques. Additionally, we used a combination of security technical debt and Multi-Objective Optimisation (MOO) techniques, as the former allows one to estimate the security consequences of each alternative, and the latter guides in selecting the best ones, considering the monetary cost and security value. The model assists architects in comparing and assessing various platforms precisely and making a secure and cost-effective optimal decision.

## 1.5 Thesis Structure

Figure 1.1 demonstrates the structure of the chapters and related research questions. Below is an overview of the remaining parts of this thesis.

**Chapter 2** reviews existing architectural analysis of security approaches related to blockchain-based systems and smart contracts. Based on the findings, this chapter broadly classifies existing publications that contribute to providing security by design approaches, as well as the chapter shows a categorisation of publications that support blockchain risk assessment methods. The chapter concludes by illuminating the architectural design issues with blockchain security that have not yet been solved and need further research.

**Chapter 3** provides a taxonomy of the major dimensions for blockchain architectural decisions and demonstrates a mapping approach that associates such a dimension with potential security attacks and threats. Attack tactics categorisation and threat modelling



classification classify attacks and their posed threats, respectively. This chapter derives from our paper presented in [6].

**Chapter 4** proposes a debt-aware approach to assessing the security technical debts incurred in smart contracts design. The assessment involves detecting design issues and quantifying their consequences to security if they remain unfixed. Moreover, the chapter shows that employing our approach increases the visibility of security design issues and allows developers to concentrate on resolving smart contract vulnerabilities through technical debt impact analysis and prioritisation. This chapter derives from our paper presented in [5].

**Chapter 5** provides a blockchain oracle decision support model. The chapter compiles information on features, attack surfaces, and quality aspects of cutting-edge blockchain oracle platforms to inform oracle selection decision-making. This chapter shows how a combination of prioritisation techniques and a multi-objective optimisation valuation mechanism guide selecting the best alternatives, considering the requirements specification, monetary cost, and security technical debts.

**Chapter 6** evaluates the thesis concerning the extent to which our work in earlier chapters answered the stated research questions and includes a reflection on how we evaluated each contribution.

**Chapter 7** summarises the key contributions of the thesis and discusses possible directions for further research in the security architectural analysis of blockchain-based systems and smart contracts.

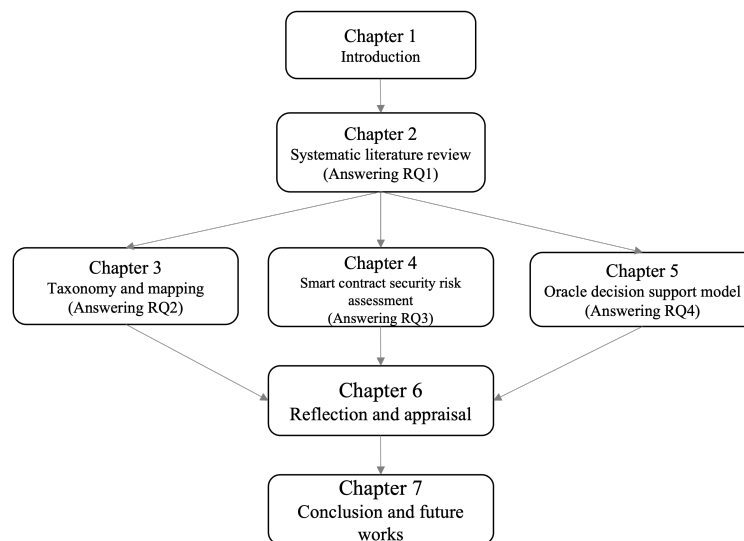


Figure 1.1: Structure of the Chapters and Related Research Questions

## 1.6 Publications Linked to this Thesis

This thesis has a definitive reference of research that has either already been published or is presently being submitted to top-tier journals or a well-established conference. The research concepts and outcomes of the thesis appear in the articles below.

- S. Ahmadjee, C. Mera-Gomez, R. Bahsoon and R. Kazman, "A Study on Blockchain Architecture Design Decisions and their Security Attacks and Threats", ACM Transactions on Software Engineering and Methodology (TOSEM) continuous Special Section on Security and SE.
- S. Ahmadjee, C. Mera-Gomez and R. Bahsoon, "Assessing Smart Contracts Security Technical Debts", in 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021 pp. 6-15.
- S. Ahmadjee, C. Mera-Gomez, S. Farshidi, R. Bahsoon and R. Kazman, "Decision Support Model for Blockchain Oracle Platform Selection", ACM Transactions on Soft-

ware Engineering and Methodology (TOSEM) continuous Special Section on Security and SE, (under review for publication).

- S. Ahmadjee, C. Mera-Gomez and R. Bahsoon, "Security architectural approaches for blockchain systems: A systematic literature review", ACM Computing Surveys (CSUR), (under review for publication).

# Chapter Two

## Security Architectural Approaches for Blockchain Systems: A Systematic Literature Review

### 2.1 Overview

Blockchain is a disruptive technology intended to implement secure, decentralised distributed systems, in which transactional data can be shared, stored and verified by participants without needing a central authentication or verification authority. The distinguishing properties and internal complex structure of blockchain technology enable the development of new security risks. Moreover, the widespread usage of this technology, especially after the emergence of smart contracts, blockchain-based computer programs, has incentivised attackers to exploit their existing security challenges [338].

Numerous malicious attacks have occurred because of poorly designed or vulnerable blockchain-based systems and/or smart contracts. For instance, in July 2020, an unauthorised third party accessed the e-commerce and marketing database of the Ledger company

website [200]. This cyberattack caused a massive data breach. Scammers used this data and applied phishing attacks to trick users into revealing the keys to the company's crypto wallet. Scammers also sent emails that included users' data and threatened them, asking them to pay a ransom. Making a poor decision to store sensitive data in an insecure, off-chain, centralised database facilitated these attacks. This sort of incident emphasises the necessity of secure-by-design approaches to orchestrate the creation of secure blockchain-based systems. A robust architectural design is the first step in secure-by-design processes, which allow security to be incorporated into the system from the ground up.

In this chapter, we provide a Systematic Literature Review (SLR) that identifies existent approaches to architect and design secure blockchain-based systems and smart contracts. The following are this chapter's main contributions:

- A classification of existing publications that contribute to providing secure architectural design approaches. Four commonly used approaches are identified: decision models; taxonomies; design patterns; and guidelines.
- Categorisation of publications that support blockchain risk-assessment methods. The publications contribute to several security risk-assessment phases that comprise security risk identification, risk analysis and/or risk calculation.
- Determination of certain blockchain security architectural design challenges that are yet unresolved and require more investigation.

Existing studies focus on reviewing the approaches, frameworks and/or automation tools that are leveraged in blockchain and smart contract testing [343, 148]. However, the approaches for assessing security issues' root causes at the early design stages are neglected. Even though there is a study [97] that reviewed the state of knowledge on perceived risk related to the adoption and application of blockchain technology, approaches related to secu-

ity risk assessment were not investigated. To the best of our knowledge, our efforts represent a pioneering step in conducting an SLR to investigate, classify and analyse the current approaches and methods for architecting and designing secure blockchain-based systems and smart contracts.

The rest of this chapter is structured as follows. Section 2.2 presents a brief overview of blockchain and smart contracts. Section 2.3 provides the research methodology used to conduct the systematic literature review and Section 2.4 presents the findings of the review. Analysis of the findings is provided in Section 2.5. Section 2.6 presents the future directions, the gap analysis, and the potential threat to the validity of the work. Related reviews are contrasted with ours in Section 2.7, and finally, Section 2.8 summarises the chapter.

## **2.2 Background**

This Section provides a brief overview of blockchain and smart contracts. Chapter 3 will provide a thorough explanation of the architectural design decisions and components of a blockchain and smart contracts.

### **2.2.1 Blockchain Overview**

Blockchain is a chain of ordered blocks, which are distributed across thousands of nodes, each block connecting to the previous block via a cryptographic hash of its content [131]. The block is seen as immutable because it cannot be modified retroactively without the modification of all the subsequent blocks. Generally, each block contains a list of transactions, a hash of the current block, a hash of the previous block, a timestamp and other information such as a nonce value. Each node participating in the blockchain network can

create a cryptographically signed transaction and then exchange it with peers in the network to provide non-repudiation to the stored transactions. Cryptographic mechanisms used by blockchain technology add integrity to the system. This technology is based on a decentralised peer-to-peer network that dispenses with the need to trust a centralised controller. Trust in the blockchain is built by relying on its protocols, mechanisms and cryptographic algorithms. Transparency and visibility are high in blockchain because data stored in the chain is publicly accessed by all the participants in the network.

### 2.2.2 Smart Contracts

A smart contract is a decentralised code agreement designed to impose an automatic negotiation of a series of instructions without requiring approval by a central authority [356]. The structure of a smart contract is similar to the structure of the class in object-oriented languages. The contract could consist of state variables, functions and events. Additionally, the contract can leverage other contracts by using inheritance. A smart contract code is stored and run on top of the blockchain and the correct execution of the contract is enforced by the blockchain properties, namely transparency and immutability. Once a contract is deployed, its program code is fixed and cannot be modified. This condition distinguishes smart contract programs from regular computer programs.

## 2.3 Research Methodology

Based on Kitchenham and Charters [177], who offered widely recognised SLR guidelines in software engineering, we conducted the review in several distinct stages: (i) identifying the review research questions; (ii) establishing the search strategy; (iii) determining the inclusion and exclusion criteria; (iv) applying the study selection procedure; (v) assessing the quality

of the final set of included studies; and (vi) extracting and analysing the data.

### 2.3.1 Research Questions

We intend to examine the following Research Questions (RQ):

- RQ1: What are the common security architectural design approaches used when architecting blockchain-based systems and smart contracts?
- RQ2: What are the existing frameworks, models and methodologies for security risk assessment in blockchain-based systems and smart contracts?

RQ1 is designed to provide an overview of the existing security architectural design approaches used in architecting blockchain systems and smart contracts. Several blockchain architectural methods have been developed in recent years, and we aim to investigate to what extent security aspects are considered in these approaches. Additionally, we want to understand the purpose and limitations of these approaches. RQ2 is devised to determine current methods for assessing the security risks linked to blockchain and smart contracts. It is important to emphasise that our research focuses on the security risk assessment methodologies for blockchain adaption rather than on how blockchain technology is employed as a solution for use in risk management or as a method to provide security to applications such as blockchain-based Internet of Things (IoT) applications.

### 2.3.2 Search Strategy

Studies were selected by entering keywords into the search feature of five major publishers or search engines: (i) IEEE Explore; (ii) ACM Digital Library; (iii) Science Direct; (iv) ISI



Web of Science; and (v) Scopus. The keywords were chosen to encourage the emergence of research findings that would aid in addressing the two RQs. The search strings for the search are as follows:

- (“Blockchain” OR “Smart contract”) AND (“Secure”) AND (“Architecture Analysis” OR “Architectural Analysis” OR "Architectural Design") AND ("Methodology" OR "Frameworks" OR "Approach" OR “Model”)
- (“Blockchain” OR “Smart contract”) AND (“Secure”) AND (“Architecture” OR “Architectural”) AND (“Risk”) AND ("Assessment" OR "Analysis")

We filtered the results of these searches by applying the inclusion and exclusion criteria, which are presented in the following Section. To complement and enhance the search process, a snowballing strategy was employed as defined by Wohlin [361]. This strategy refers to identifying more articles by utilising a paper’s reference list, which is known as ‘backwards snowballing’, or citations, which is known as ‘forwards snowballing’. We conducted forwards and backwards snowballing iterations until no more papers fulfilling the inclusion criteria were found.

As the blockchain technology topic is actively growing in the industry and the technology is being rapidly adapted, we considered including relevant grey literature, specifically industry sources and standards from the leading global consultancy firms that have published white papers, guidelines or approaches that discuss the security architectural design of blockchain systems.

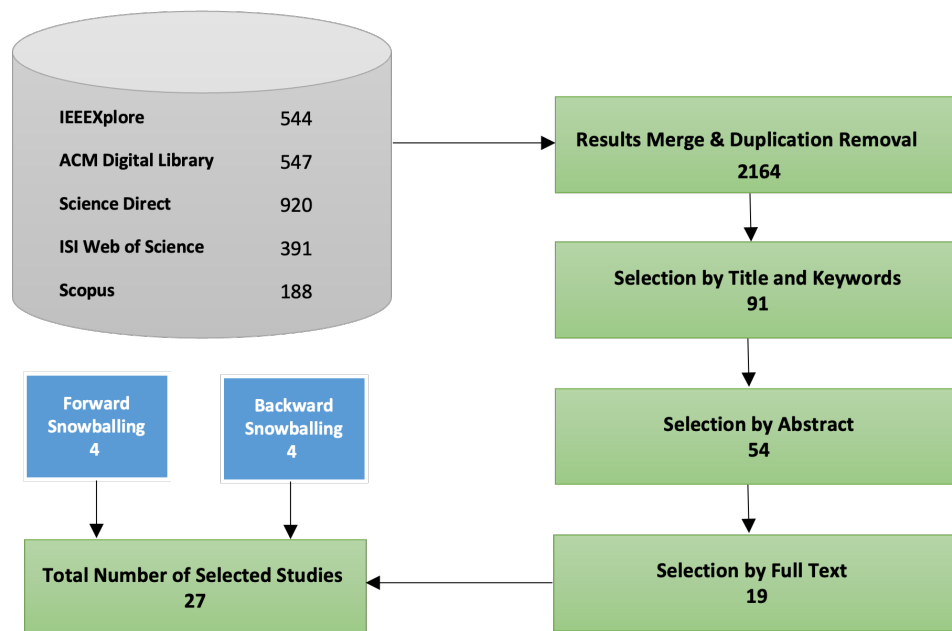


Figure 2.1: Search and Academic Papers Selection Procedure

### 2.3.3 Study Selection

Because not all the papers returned by the search were relevant to the study questions, they had to be screened first. As a result, we identified the selection criteria that were used to verify that the outcomes were objective. The following are the inclusion and exclusion criteria that we established:

#### Inclusion Criteria (I)

**I1:** Papers published in peer-reviewed journals, conference proceedings, workshops, or book chapters.

**I2:** Papers containing information related to blockchain and/or smart contracts.

**I3:** Papers explicitly related to the topics of analysis, assessment or management of security risks in blockchain and/or smart contract applications.

**I4:** Papers explicitly related to security architectural design approaches for blockchain and/or smart contracts.

#### Exclusion Criteria (E)

**E1:** Papers from disciplines other than computer science, in which blockchain was used merely as a component of the application.

**E2:** Papers written in languages other than English.

**E3:** Papers that were not freely available.

Three rounds of filtering were used to determine the final selection of research papers. Figure 2.1 presents the search and selection processes.

**First round:** We selected papers based on metadata, including title, venue name and keywords. In this round, we considered the criteria I1 and E1.

**Second round:** We independently chose papers by reading the papers' abstracts. In this round, we considered criteria I2 and E2.

**Third round:** We independently chose papers by reading the full text of the papers selected in the previous round. In this round, we considered criteria E3, I2, I3 and I4.

**Snowballing:** Additional articles were discovered using forwards and backwards snowballing. All inclusion and exclusion criteria were considered.

After round three and the snowballing step, we applied Equation 2.1 to calculate Cohen's Kappa ( $k$ ) [320], which is a statistical measure to assess the agreement between two reviewers deciding the same issue:

$$k = (P_o - P_e)/(1 - P_e), \quad (2.1)$$

where  $P_o$  is the probability of the observed agreement, and  $P_e$  is the probability of random agreement. We got a  $P_o$  of 0.90, and a  $P_e$  of 0.60, which produced a  $k$  of 0.75, indicating substantial agreement. To resolve the disagreements, a discussion that involved all the reviewers was conducted. As a result of this discussion, 27 studies were included in our SLR.

**Manual search:** We started searching for publications of well-known institutions and organisations that regularly produce publications related to blockchain, such as Deloitte [88], KPMG [191] and NIST (National Institute of Standards and Technology) [257]. Additionally,

we expanded our research by conducting a manual search using the Google search engine to cover other industrial publications. Finally, we selected eight grey publications that matched our inclusion criteria.

### 2.3.4 Quality Assessment

In this study, the criteria presented by Yang et al. [384] were customised to assess the quality of the selected studies. Three essential characteristics of empirical studies were considered in this research: rationality, rigour and credibility. *Rationality* assesses whether the research context and purpose are clearly stated; *rigour* investigates whether the research approach is practical, scientific and complete; and *credibility* determines whether the research findings are reliable and meaningful. We selected the most widely used criteria for each characteristic, as determined by previous software engineering SLRs. The following are the eight quality criteria that were examined for each selected study:

#### **Rationality**

1. Is the paper based on empirical research?
2. Is the context of the study stated clearly?
3. Is there a clear description of the research objectives?

#### **Rigour**

1. Does the method adequately address the research objectives?
2. Is the data collection method fully described?
3. Is data analysis sufficiently described?

#### **Credibility**

1. Is there a clear description of the results?
2. Do the researchers discuss limitations or threats to the validity of the results?

The reviewers separately scored each of the eight studies' criteria based on a Boolean

Table 2.1: Extracted Data Items

Item	Association
Title of the study	Demographic
Year of the study	Demographic
Venue of the study	Demographic
Type of security architectural method	RQ1
Purpose of the security architectural method	RQ1
Security risk assessment method	RQ2
Purpose of the risk assessment method	RQ2

matrix (0 or 1), in which 1 indicates that the study fulfils the quality assessment question and 0 indicates that the study fails to fulfil the quality assessment question. A discussion was held if there were any discrepancies between the reviewers, and the study was reviewed again to determine the final score. Table A in Appendix 1 illustrates the scores of each study.

### 2.3.5 Data Extraction

Data extraction is the process of gathering data items that were used to analyse the final studies and answer our two research questions. This study’s data extraction mostly comprised demographic information, information related to blockchain security architectural design approaches and information related to security risk assessment methods. Table 2.1 shows the extracted items. Demographic information can be statistically demonstrated, while the information related to the research questions requires in-depth analysis. The data extraction method was first applied to a collection of 10 highly cited studies in the field of blockchain architecture. The collected data were merged and classified. We selected the main concepts and aspects that resulted in the first draft of our classification. Next, we examined the entire collection of selected studies to develop the classification.

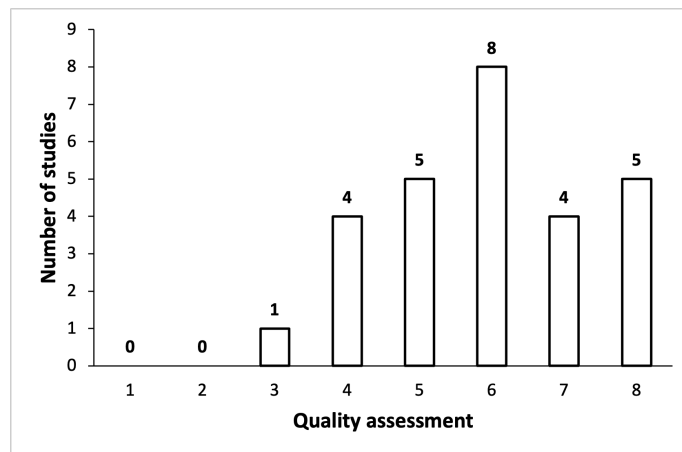


Figure 2.2: Quality Scores of the Included Studies

### 2.3.6 Data Synthesis

This Section aims to provide a concise summary of the data that was extracted to fulfil the study's objectives. The data that was extracted for this study is both quantitative and qualitative. A descriptive analysis method was applied to synthesise a set of quantitative data, including publication venues of selected studies and their quality scores. The purpose and focus area of each selected study was also described. The thematic synthesis method [161] was used to synthesise the qualitative data and answer the study's research questions. The qualitative data refers to the type of blockchain security architectural approaches and security risk assessment methods. To answer RQ1, a categorisation process was conducted, and the blockchain security architectural design approaches were classified into categories that have similar characteristics. Then the purpose of each approach was explained. To answer RQ2, the security risk identification and security risk analysis and calculation approaches were identified and thoroughly explained. Section 2.4 (Result) and Section 2.5 (Analysis) present all the extracted and synthesised information.

Table 2.2: Publication Venues of Selected Studies

Publication Channel	Selected Studies
Journals	[119, 124, 51, 382, 159, 154, 8, 398, 34, 308, 9, 128, 44]
Conferences	[373, 376, 369, 378, 318, 303, 336, 289, 220, 374, 183, 247, 228]
Workshop	[364]

## 2.4 Results

This Section demonstrates the distribution of publications in different venues over time, the distribution of selected studies' quality assessment scores and the aim and focus of each paper.

### 2.4.1 Demographics of Selected Studies

Only peer-reviewed articles were included in this study, Table 2.2 illustrates the selected studies in each publication venue. As shown, half of the included studies were published at conferences and the other half were published in journals. However, most of the journal papers received high scores on the quality assessment. For instance, only journal papers received a full quality assessment score of eight, while only one conference paper received a score of seven, as shown in Figure 2.2. It demonstrates the quality scores of the included studies according to the quality criteria presented in Table A. Figure 2.2 shows eight studies received a score of six. Most of these studies provided no details on their data collection methods, and they did not discuss limitations or threats to the validity of their results. Only one study received the lowest score, which is three. This study was not empirical research; it provided no information about the data collection methods; and there was no clear description of the results and study limitations.

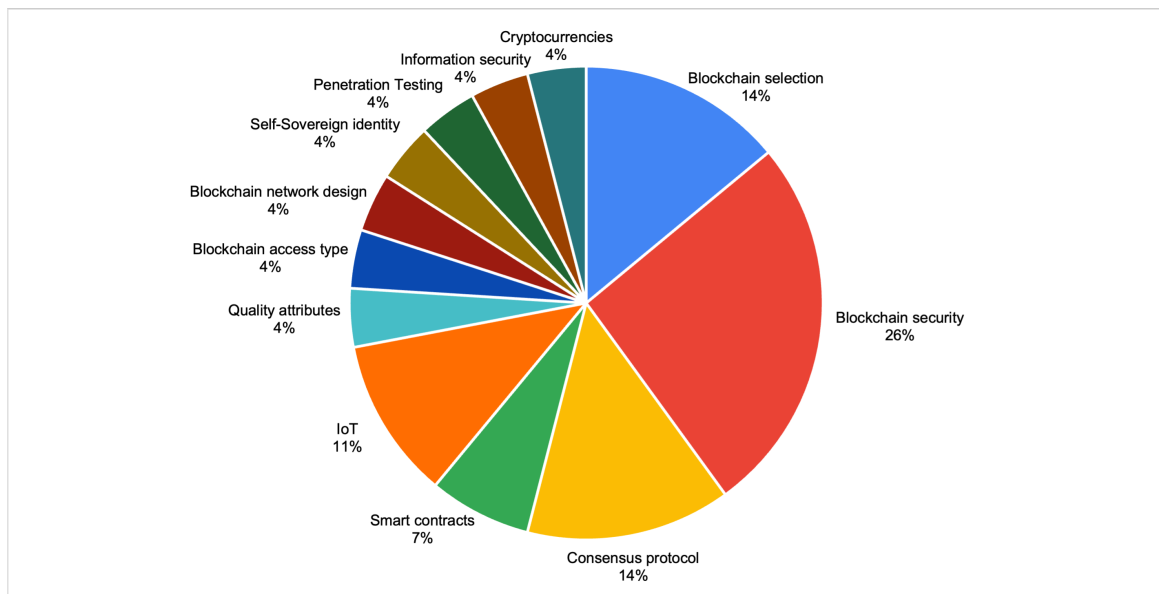


Figure 2.3: Chart of Focus Areas of Selected Studies

## 2.4.2 Purpose of Selected Studies

Each selected article was fully read before essential information was retrieved and summarised in Table 2.3. It explains the aim of each study as well as its focus area. Figure 2.3 shows the percentages of different focus areas of the 27 selected studies. The focus areas identified in the selected studies highlight that a quarter (25%) of all studies are mostly concerned with blockchain security issues as they discuss popular blockchain security attacks and threats. Blockchain selection and blockchain consensus mechanisms are both the second-most-popular areas, at 14%. The studies related to blockchain selection aim to help select a suitable blockchain platform, while the studies related to consensus mechanisms are mostly concerned with exploring multiple consensus protocols. Some studies also aim to assist decision-makers in selecting the most-suitable one. IoT is the third-most-common focus area, at 11%, and is mostly concerned with security issues and quality attributes related to blockchain-based IoT. Smart contracts are the fourth-most-common area, with 7%. The studies focus on a few architectural design aspects of smart contracts. The least-common ar-



as on our list are related to quality attributes, blockchain access types, blockchain network design, self-sovereign identity, penetration testing, information security and cryptocurrencies, each accounting for 4%.

## 2.5 Analysis of the Selected Publications

The two research questions are analysed in-depth in this Section. Figure 2.4 depicts the classification of the selected publications and the percentage of studies per category.

### 2.5.1 Security Architectural Design Approaches (RQ1)

Based on our results, we broadly classify the selected studies into four commonly used approaches that support the secure architectural design of blockchain-based systems. These are (i) decision models; (ii) taxonomies; (iii) design patterns; and (iv) guidelines. A representative selection of studies that belong to these categories can be found in Table 2.4. Since some studies contribute to multiple categories, they appear more than once in the table.

*Decision models.* In our context, decision models refer to the models that support the analysis and inform architecture-related design decisions with regard to blockchain-based systems and smart contracts. One of the aims of a decision model is to link components from the problem space to their solution space counterparts. Four studies [119, 373, 124, 369] contributed to the architectural design decision model.

Farshidi et al. [119] provided a multi-criteria decision model (MCDM) for the blockchain platform selection problem. The authors stated that their decision model involves 121 blockchain features that can be linked to 28 blockchain platform alternatives and multiple quality attributes, including security. However, the blockchain features were briefly explained

Table 2.3: Main Purpose and Focus Area of the Primary Studies

Study	Purpose	Focus Area
[119]	Equips blockchain decision-makers with a decision support model to select a suitable blockchain platform for their applications	Blockchain selection
[378]	Provides an experience report about blockchain architectural decisions to decide or discard the adoption of a decentralised solution based on blockchain	
[318]	Provides a methodology that assists the selection of blockchain for a given set of requirements and also offers guidance throughout blockchain configuration	
[154]	Provides a risk analysis methodology to facilitate understanding of the security implications in the adoption of blockchain	
[159]	Introduces a layered security reference architecture for blockchain that identifies origins of known security threats and their potential mitigation mechanisms	Blockchain security
[398]	Discusses the blockchain's basic architecture and its potential security and trust issues at the data, network, consensus, smart contract and application layers	
[308]	Presents a discussion on a set of attack vectors and security threats to blockchain-based solutions	
[9]	Presents a security risk assessment methodology that enables a systematic quantification of the risk associated with blockchain technology and its ecosystem	

*Continued on next page*

Table 2.3 Main Purpose and Focus Area of the Primary Studies (*Continued from previous page*)

Study	Purpose	Focus Area
[373]	Provides architects with a decision model to assist them in selecting the appropriate architectural design patterns for blockchain-based applications	
[247]	Identifies the main threats to blockchain and assesses how these threats may adversely impact a blockchain-based solution	
[374]	Categorises design patterns for blockchain-based applications, including one category of security patterns	
[124]	Introduces a multi-criteria framework for selecting the most-suitable consensus protocols depending on the identified criteria, priorities and other requirements	Consensus protocol
[228]	Assesses attacks that target consensus protocols with respect to their potential implementation in an IoT blockchain environment	
[128]	Introduces an evaluation framework of blockchain consensus algorithms and discusses security design principles to deal with different attacks	
[44]	Proposes a classification framework of consensus protocols to serve as a comprehensive and integrated taxonomy	
[289]	Provides a brief discussion of architectural aspects of smart contracts and a security mechanism to be followed when designing smart contracts	Smart contracts
[364]	Elaborates a set of security design patterns for smart contracts	
[303]	Presents a discussion of security issues related to blockchain adoption in IoT environments	

*Continued on next page*

Table 2.3 Main Purpose and Focus Area of the Primary Studies (Continued from previous page)

Study	Purpose	Focus Area
[382]	Provides a catalog of architectural tactics for achieving a set of required quality attributes in the design of IoT systems based on blockchain	IoT
[51]	Presents an analysis of security issues at each layer of a blockchain architecture and the potential impact of security attacks against a blockchain system	
[376]	Proposes a taxonomy that captures some architecturally relevant blockchain characteristics in relation to their support for various quality attributes	Quality attributes
[369]	Provides a methodology to identify whether blockchain is useful depending on the problem requirements, and if so, what type of blockchain might be appropriate	Blockchain access type
[336]	Presents a map with the main design dimensions for blockchain networks	Blockchain network design
[220]	Provides design guidelines that detail a number of Ethereum design patterns and their map to Solidity coding practices	Self-Sovereign identity
[34]	Presents a penetration testing framework for smart contracts and decentralised apps that assesses the security risk of several attacks	Penetration Testing
[8]	Complements the information security controls framework for blockchain established by the International and National Information Security Standards	Information security
[183]	Provides risk analysis and risk management guidelines based on NIST and ISO standards for cryptocurrency exchange	Cryptocurrencies

as the study focused on the methodology of building the decision model. Although the study considered security and listed a few security issues, the authors failed to analyse any of these issues and did not explicitly discuss how blockchain features are associated with security quality attributes.

Xu et al. [373] proposed a decision model that allows designers to choose suitable patterns for blockchain applications. The work provided patterns for several blockchain architectural components, including smart contracts and blockchain oracles. It proposed two security decision models, one for authentication and the other for authorisation. Several security quality attributes were considered with regard to the patterns, including integrity, availability and confidentiality. However, security issues that might arise when employing these patterns were not discussed.

Only one element of blockchain architecture was discussed in previous studies [124, 369]. Filatovas et al. [124] proposed the use of MCDM to select an appropriate consensus protocol for blockchain applications. Wüst et al. [369] provided a decision model for choosing a suitable blockchain type: permissioned or permissionless. Because neither study's main focus was security, they only discussed a few security attributes, such as integrity and availability, and mentioned some security issues.

***Taxonomies.*** In software engineering, taxonomies help with knowledge categorisation and organisation, enabling practitioners and researchers to comprehend and analyse a complicated design space as well as assess and compare design solutions.

Xu et al. [376] proposed taxonomies of the architectural components of blockchain systems and demonstrated how these components affected system quality attributes such as performance. However, impacts on security attributes, possible attacks and their subsequent threats were missing. Attacks were classified in the study by Zhang et al. [398] based on blockchain layers: data, networks, consensus, smart contracts and applications.

Xu et al. [378] classified design decisions into blockchain decisions and application decisions and then conducted a trade-off analysis of the related quality attributes. Their research briefly analysed security and discussed a few security attacks. Homoliak et al. [159], classified security vulnerabilities and threats in four layers: the consensus layer, the network layer, the state machine layer, and the application layer. Brotsis et al. [51] only analysed blockchain architectural components that are suitable for IoT applications. The study also classified attacks based on several aspects including identity, service and manipulation.

Previous studies [128, 44] focused solely on the architectural decisions of the blockchain consensus protocols. Bouraga [44] classified 28 consensus protocols based on four criteria: origin, design, performance and security. In terms of security, they showed whether the considered consensus protocol addressed Sybil attacks, denial of service attacks, 51% attacks and eclipse attacks. Fu et al. [128] introduced three evaluation dimensions of blockchain consensus algorithms: effectiveness, decentralisation and security. The study reviewed the security design principles for resisting several attacks, including double spending attacks, Sybil attacks and eclipse attacks.

***Guidelines.*** The guidelines refer to well-established concepts and outlines that work in practice and, as a result, offer knowledge and insights.

Staderini et al. [318] presented a guide to selecting the most appropriate type of blockchain. The proposed guidelines considered some blockchain architectural criteria, including consensus protocols and smart contracts. The study discussed a number of attacks, such as 51% attacks and mining attacks. Richard et al. [289] briefly explored smart contract architecture. The study suggested a smart contract development model that is divided into three classes in the form of the development cycle, security mechanism and development support. Several tasks were assigned to the security mechanism class, including input filtering, bug detection and vulnerability metrics. The study failed to examine or discuss any

security-related issues.

Previous studies [336, 303] analysed types of blockchain; Tran et al. [336] analysed the consensus protocol, while Sargsyan et al. [303] discussed node architecture. Both studies discussed security briefly. Despite the fact that these studies claimed to provide blockchain architectural design guidelines, their proposed guidelines failed to provide a precise and clear set of steps to be followed by blockchain systems architects.

***Design patterns.*** These refer to repeatable solutions that can be applied to common blockchain or smart contracts' architectural design problems.

Xu et al. [374] presented security patterns that enhance the immutability, integrity and non-repudiation of blockchain systems. The study also proposed design patterns that provide solutions to common security issues related to smart contracts built on Ethereum blockchain. Two studies [382, 228] presented a catalogue of architectural solutions for blockchain-based IoT applications. Yáñez et al. [382] discussed multiple blockchain architectural elements and their related security attributes; however, the security issues of these architectural elements were not discussed. Mackenzie et al. [228] only discussed several blockchain consensus protocols for IoT application and their related attacks. In addition, Liu et al. [220] only presented and discussed design patterns for blockchain-based self-sovereign identity, and the study briefly analysed the security issues of some presented patterns.

Based on our findings, there are only four industry sources that provide approaches for architecting blockchain systems and for analysing security. NIST [380] provided an overview document of blockchain technology to assist practitioners in understanding how this technology works. The document organises concepts, components, models and other elements related to blockchain technology, as well as discusses several security attacks including 51% attacks, Sybil attacks, DDoS attacks and double spending attacks. The American Council

for Technology-Industry Advisory Council (ACT-IAC) [327] also provided a blockchain playbook that defines a process incorporating several phases, including a technology selection phase, to help adopt the technology. In the selection phase, the playbook discusses several architectural components such as smart contracts and consensus protocols and discusses the security attributes of such components. However, both documents lack a thorough analysis of security problems that are associated with blockchain architectural components.

The European Union Agency for Cybersecurity (ENISA) [76] produced a report that explained several blockchain components, including consensus mechanism, smart contracts and sidechains. Their report also discussed several traditional and blockchain-specific cybersecurity issues. Finally, each issue was mapped to essential best practices to aid practitioners in developing secure blockchain systems. The report, however, is only concerned with blockchain-related challenges in the financial sector.

The German Federal Office for Information Security [165] produced a document that highlighted the security attributes of several blockchain components and discussed possible attacks that targeted each component. The document also assessed and compared the security attributes of public and private blockchains.

## 2.5.2 Blockchain Security Risk Assessment Methods (RQ2)

According to ISO 27001 [1], risk assessment involves five main steps: (i) risk identification; (ii) assigning risk owners; (iii) risk analysis; (iv) risk calculation; and (v) risk evaluation. Based on our results, we found that the selected studies related to RQ2 contribute to blockchain risk identification, risk analysis and risk calculation methods. Table 2.5 lists these studies. Some of the research helps to either identify security risks or analyse and calculate such risks. A few publications, however, have an impact on both areas, as Table 2.5 shows. The



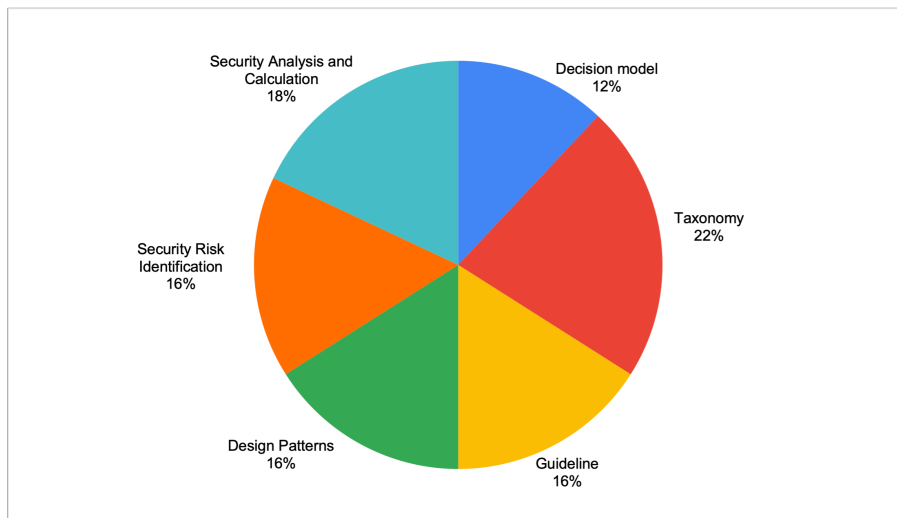


Figure 2.4: Chart of the Classification of Selected Publications

primary contribution of each study is described in the following paragraphs.

***Security risk identification.*** ISO 27001 defines risk identification as the process of determining the assets that have value and could be targeted by attackers; finding the potential threats to these assets; and determining the vulnerabilities that might be exposed to these threats. We found several studies that contributed to blockchain security risk identification methods.

Homoliak et al. [159] proposed a threat-risk model that involves six elements: (i) the kind of blockchain users (owners); (ii) assets that are present at the application layer; (iii) threat agents or malicious users; (iv) threats that emerge from blockchain architectural components' vulnerabilities; (v) countermeasures; (vi) risks that are caused by threats and malicious users and lead to asset corruption or losses. However, the model fails to provide clear steps or a guide to blockchain practitioners on how to employ such a model in their system design.

In Schlatt et al. [308], the attack vectors were classified based on several blockchain architectural components, including consensus protocols, application wallets and smart con-

tract language. The work proposed a research framework from an information security perspective to help analyse the identified attacks. Kim et al. [183] only concentrates on cryptocurrency exchange platforms. The study analyses their related vulnerabilities and provides security enhancement recommendations based on NIST and ISO/IEC 13187:2011 [163] standards. Hebert and Di Cerbo [154] proposed a partial risk assessment methodology to identify security risks in various elements of blockchain architecture. It also suggested using threat modelling methodology to analyse the identified risks. However, the evaluation and ranking of the consequences of these risks were not considered in the methodology.

***Security risk analysis and calculation.*** ISO 27001 defines risk analysis and calculation as the process of assessing the impact of security risks and their likelihood of occurrence and then determines the severity of each risk. We found several studies that proposed blockchain security risk analysis and/or calculation methods.

A penetration testing framework for blockchain applications was proposed by Bhardwaj et al. [34]. The study performed penetration testing on a commercial blockchain application to identify and rank the consequences of the potential threats. However, the study only considered a few security issues that were solely related to smart contracts. Al Ketbi et al. [8] proposed blockchain security controls and their implementation guidelines to fill the gap in the existing national and international standards. The study evaluated, ranked and recommended security controls that could be implemented. However, the security attacks that were imposed as a consequence of the blockchain architectural components were not been considered.

Al Mallah et al.[9] classified blockchain security threats into four categories: network threats; double spending threats; private key threats; and smart contract threats. The study also assessed the threat risk impact based on the author's observations and opinions. Case studies of blockchain applications were not employed to show how to identify and

Table 2.4: Categories of Secure Architectural Design Approaches

Category	Studies
Decision Model	[119, 373, 124, 369]
Taxonomy	[376, 398, 378, 159, 51, 128, 44]
Guidelines	[318, 289, 336, 303, 378]
Design Patterns	[374, 382, 228, 220, 373]

assess threats in a real work example. Morganti et al. [247] proposed a cybersecurity risk assessment framework for blockchain-based smart mobility. The framework first determined the impact and probability of occurrence of each threat and then ranked the corresponding risks.

We found four industrial sources that contributed to blockchain risk assessment methods. In 2017, KPMG [184] released a white paper that investigated two specific blockchain attacks and explained how these attacks can be avoided. The paper also proposed a security framework that blockchain architects can use to identify and mitigate security risks that have arisen as a result of the use of blockchain. In 2018, KPMG [103] published another white paper that identified 10 specific blockchain risk areas and provided a five-level approach to assess the identified risks. Security attributes and issues were briefly discussed.

Deloitte provides a risk management framework that involves three risk considerations, including standard risks, value transfers and smart contracts [279]. The information security risks of blockchain wallets and smart contracts are superficially explained, as the main aim of this framework is not security. The World Economic Forum (WEF) provides a toolkit that involves a five-step approach for blockchain cybersecurity risk management [250]. The risk assessment template and the guide for filling it out are also provided. The toolkit provides a risk identification checklist involving multiple risk factors. However, security issues related to blockchain architectural and design decisions were missing.

Table 2.5: Blockchain Security Risk Assessment Methods

Method	Studies
Security Risk Identification	[159, 308, 183, 154, 34]
Security Analysis and Calculation	[34, 8, 9, 247, 159, 183]

## 2.6 Discussion

Based on the results of this review and our observations and analysis, in this Section we discuss research directions that deserve further investigation; describe specific limitations to be addressed in this thesis; and finally, present potential threats to the validity of our work and how we mitigate them.

### 2.6.1 An Outlook for Future Directions

According to our results, only a few studies [119, 124, 376, 369, 336] have provided systematic approaches with clear steps for architecting and designing secure blockchain-based systems. However, security is not the focus of these studies. Security tends to be briefly discussed as an architectural quality attribute and in the context of blockchain architectural design decisions for integration.

Lacking systematic security in design approaches and security standards that assist practitioners at the early blockchain development stages may lead to catastrophic incidents such as the attacks that caused a permanent freeze of 280M USD [60] because of a design problem in the smart contracts. Moreover, because blockchain-based systems contain several architectural components, each with a relatively complex internal structure, a lack of systematic methods can over-complicate the design of secure blockchain systems and smart contracts. Therefore, researchers need to establish comprehensive and systematic secure-

by-design approaches to orchestrate the decision-making and architectural design processes with regard to blockchain systems and smart contracts. This is also one of the recommendations made by Wan et al. [353], in which the authors conducted a comprehensive study to investigate how practitioners view and practice smart contract security.

As the results show, there is a lack of complete methodologies that provide a clear guide to the identification, quantification and management of security risks related to blockchain systems' architectural design decisions. The selected publications either provide partial risk assessment methods with no clear guide on how to employ them or only focus on a few security issues that relate to blockchain architectural components. Moreover, security attributes are not the focus of some publications [119, 124, 369, 376], and security is only discussed in a briefly or superficially. Due to the lack of methods for identifying, analysing, assessing and managing a broad set of security issues, a large percentage of blockchain projects have failed [97]. Consequently, there is an urgent need for a general framework, approaches and methodologies that address security risks in the context of blockchain architecture. The availability of such approaches enables the architecting of blockchain systems that are secure-by-design, have manageable security risks and lower the chances of security breaches and project failure.

There have been several initiatives in smart contract security, resulting in the definition of several vulnerabilities for smart contracts and tools for finding them. However, smart contract security is still in its early stages and many other challenges require attention, such as the security implications of design decisions in smart contracts, including programming languages and off-chain integration. Despite the number of smart contract vulnerabilities discovered by researchers [159, 212, 152, 24, 60], there is no reference classification that organises and collects security issues based on criteria such as implications, ramifications, cost of exploitation and resolution. Therefore, systematising the process of smart contract security from the early design stage, including the standards, best practices, tools and approaches, is an essential step towards designing secure smart contracts.

## 2.6.2 Gap Analysis

Drawing on the findings of the systematic literature review and the outlook for the future, we identified the following gaps that steered our investigations in the thesis:

- **A systematic approach to assist architects in making secure architectural design and configuration decisions for blockchain-based systems:** The results of our review indicate that although there are several initiatives that discuss blockchain architectural components, these initiatives either focus on only one of these components or briefly discuss a few of them. Additionally, our results show there are few studies that discuss the security of blockchain components or that provide a systematic approach to identifying security issues that are associated with such components. This knowledge gap may increase the occurrence of unsuitable design decisions and produce configurations prone to potential security risks. To address this limitation, we derive a taxonomy of commonly used architectural design decisions in blockchain-based systems. We map each of these decisions to potential security attacks and their posed threats. We provide a systematic way to apply the taxonomy and leverage the mapping approach. A thorough justification of our solution is given in Chapter 3.
- **An approach for assessing smart contracts' security risks:** Assessing the security risks of smart contracts at the early design stage is an essential step to mitigate potential vulnerabilities and deploy secure and robust contracts. However, the findings of our SLR revealed that there is a lack of security-based design approaches that focus on the architectural design elements of smart contracts and assist in identifying, analysing and quantifying the associated security risks. Experience shows that many deployed contracts are vulnerable to exploitation due to their poor design, which allows attackers to steal valuable assets from the involved parties. An assessment approach that allows developers to recognise the consequences of deploying vulnerable contracts

is needed. The technical debt metaphor has proven to be effective in measuring the impact of security weaknesses, exploitation, ramifications of negative decisions, over-time and locating the design root of security vulnerabilities. Therefore, in this thesis we leverage this metaphor, as a potential approach, to propose a debt-aware approach for assessing security design vulnerabilities in smart contracts. Chapter 4 provides a detailed explanation of the approach.

- **Decision support model for blockchain oracle platform selection:** Smart contracts use agents known as blockchain oracles to collect and provide data feeds to the contracts. The functionality and compatibility of oracle with smart contract applications need to be considered when selecting the best-fit oracle platform. As the number of oracle alternatives and their features increases, the decision-making process becomes increasingly complex. Selecting the wrong or sub-optimal oracle is costly and may lead to severe security risks. Although there are several studies that propose decision-model approaches for selecting blockchain architectural components or blockchain platforms, approaches that evaluate the security issues of blockchain oracles have been neglected in existing publications. To address this limitation, we provide a guided decision support model for the oracle selection problem. The model leverages the security technical debt metaphor to quantify the security risks of possible attacks on the examined blockchain oracles and assists in eliminating solutions that can manifest into potentially costly and risky security technical debts. Henceforth, the model can support smart contract decision-makers in selecting a secure, cost-effective and feasible oracle platform for their applications. The proposed decision model is explained in depth in Chapter 5.

### 2.6.3 Threats to Validity

Based on Wohlin et al. [362], we have considered the following potential threats to validity:

**External validity.** Though an effort to include studies on architecture design approaches for building secure blockchain systems and smart contracts, there is the risk of missing some papers. To mitigate this potential threat, in addition to the results of the automated search, we applied backwards and forwards snowballing search methods to examine additional papers, where the snowballing strategy was explained in Subsection 2.3.2. Moreover, we considered including grey literature to enhance the generalisability of the results.

**Construct validity.** There are potential biases during the study selection and data extraction processes. To mitigate this threat, we worked independently on the selection paper process. Any disagreement was resolved by having a group discussion. The possibility of bias introduced during the data extraction process was reduced by ensuring that everyone reviewing the studies had a common understanding. We also made sure that the data extraction procedure matched the research questions.

**Conclusion validity.** Another threat arose because we cannot guarantee the completeness of our classification of blockchain secure architectural design approaches and security risk assessment methods, as there might be additional categories that could enrich the classification. To mitigate this threat, we iteratively refined our classification each time a new concept was encountered in the literature. Nevertheless, the classification is adaptable to evolve and to cope with new additions and changes.

## 2.7 Related Work

The academic community is becoming increasingly interested in problems related to blockchain technology. Unfortunately, there is still a dearth of comprehensive literature reviews that look at certain areas of software engineering, including architectural design approaches for building secure blockchain-based systems. Based on the methods employed to apply the re-



views by the existing literature, we categorise them into three groups: systematic literature reviews, surveys, and comprehensive reviews. The studies, their methodologies, and their focus areas are listed in Table 2.6.

**Systematic Literature Reviews.** Drljevic et al. [97] conducted a systematic literature review that demonstrates the state of knowledge with regard to the perceived risk related to the adoption and application of blockchain technology. The study sheds light on risk definitions that are related to technology, business and project management. The connection between blockchain technology adoption and risks is also clarified. The study emphasises the importance of standards-based approaches for the effective adoption and application of blockchain. The study concluded that there are significant research gaps in the field of risk management with regard to the use of blockchain technology, a conclusion that is consistent with our findings. However, in contrast to our study, security risk assessment approaches have not been investigated in this study.

In 2016, Yli-Huumo et al. [391] conducted a mapping study to investigate the available topics and challenges that were related to blockchain technology. They found most of the studies focused on Bitcoin systems, and only a few papers investigated other blockchain topics such as smart contracts. According to that study, blockchain security architectural design approaches were ignored in the considered studies. This is nearly consistent with our findings, as we found no studies investigating security architectural design methods prior to 2016, and we only found one study [378], published in 2016, which sheds light on several blockchain design decisions.

The improvement of smart contracts and decentralised application development was the focus of a systematic literature study carried out by Vacca et al. [343]. The study included the frameworks and automation tools that are employed in smart contract testing. They found that many of the currently used methods and tools only deal with particular

smart contract-related problems and challenges.

**Surveys of Literature.** Guo and Yu [148] conducted a survey on the security of blockchain technology. The study evaluated blockchain security through risk analysis to identify extensive blockchain security risk categories, examined actual attacks against the blockchain and possible defects and their underlying causes, as well as presented newly developed blockchain security countermeasures. Leng et al. [203] conducted a survey to review the state of blockchain security research. Based on the selected papers, the study classified the security of the blockchain into three levels: the process level, the data level, and the infrastructure level. The study also investigated the existing solutions for addressing security issues. Studies [148] and [203] both presented an extensive review of blockchain security issues, and their reviews can be used to support the building of secure blockchain systems. However, security design issues and security risk assessment methodologies, which assist in avoiding security problems at the early stages, have not been investigated in these studies.

**Comprehensive Reviews.** König et al. [189] provided a comparative analysis of 19 blockchain standards published by organisations that work and focus on information security. Security management and technical security are among the criteria that are used to compare the content of the standards. The study, however, only considered organisational standards in the comparison and excluded other types of grey literature such as white papers and industrial reports. They also ignored academic literature, which can also aid in the adoption of reliable and secure blockchain technology.

To the best of our knowledge, and in contrast to other efforts, our work explores and categorises the current architectural design approaches and security risk assessment methods used to construct secure-by-design blockchain-based systems.

Table 2.6: Related Works, their Methods, and Focus Area

<b>Study</b>	<b>Methods</b>	<b>Focus Area</b>
[97]	SLR	Risk management of blockchain
[391]	SLR	Blockchain topics and challenges
[343]	SLR	Smart contract applications
[148]	Survey	Security issues of blockchain
[203]	Survey	Security issues of blockchain
[189]	Comprehensive Review	Blockchain standards

## 2.8 Summary

This chapter presented a systematic literature review to investigate current approaches and methods that assist in architecting and designing secure blockchain-based systems and smart contracts. We selected 27 academic studies and eight industrial reports that satisfied the defined inclusion criteria. We found that the approaches provided by a set of selected studies can be classified into four categories: (i) decision models; (ii) taxonomies; (iii) design patterns; and (iv) guidelines. The other set of selected studies contributed to several security risk assessment phases: (i) security risk identification and/or (ii) security risk analysis and calculation.

We argued that the development of secure blockchain systems requires leveraging security architectural design approaches. However, based on our review results, we found there is a lack of systematic security architectural-centric approaches, security standards and complete security risk assessment methodology. We concluded that there is a critical need for a generic framework, methods and approaches that handle security risks in the context of blockchain architecture at the early architectural and design stages.

In this chapter, we presented several research directions that deserve further investigation in the field of security architectural design decisions for blockchain systems and smart contracts. These are as follows: (i) a systematic approach to assist architects in making secure architecture design and configuration decisions for blockchain-based systems; (ii) an approach for assessing smart contracts' security risks; and (iii) a decision support model for blockchain oracle platform selection. We will provide a thorough explanation of each research direction in the upcoming chapters.

## Chapter Three

# Blockchain Major Architectural Design Decisions and their Security Attacks and Threats

Chapter 2 has identified certain blockchain security architectural design challenges that are yet unresolved and require more investigation. Among the identified challenges, we found a need for a systematic approach to assist architects to make secure architectural design and configuration decisions for blockchain-based systems. Therefore, in this chapter, we present a taxonomy of commonly used architectural design decisions in blockchain-based systems. We identify and classify the potential security attacks and their posed threats and map them to the architectural design decisions of blockchain systems. The mapping approach aims to guide architects to make justifiable design decisions that will result in more secure implementations. The result of the mapping feeds into the requirements of the upcoming chapters as it assists in determining the attack surfaces of the smart contracts and understanding their impacts.

This chapter reports in full the contents of a published manuscript [6] by the thesis

author Ahmadjee et al. It includes verbatim text from the manuscript and some changes employed for the purpose of this thesis.

### 3.1 Overview

Blockchain is a decentralised technology that claims to provide several security properties, including immutability, integrity, non-repudiation, and availability [376]. Additionally, the technology helps alleviate multiple security risks that threaten traditional centralised systems such as single points of failure. However, since a blockchain-based system has several architectural components, each with complex internal structures, the chance of introducing security vulnerabilities by making poorly informed design decisions or misconfiguring any of those components is non-trivial. There is a lack of a systematic guide to design secure systems based on blockchain technology [351, 409]. This situation leads to architects operating in an ad-hoc manner [276], relying on the wisdom and trust of peers [409]. Moreover, a lack of awareness of security attacks and the consequent impacts on blockchain system architecture can deter practitioners from addressing security issues at the early development stage [352]. As a result, attackers might discover security flaws and breach the system.

The novel contributions of this chapter are:

- A taxonomy that defines, illustrates, and classifies the key architectural decisions regarding blockchain-based systems including access type, data storage, and transaction computation decisions. This taxonomy is the result of an approach partially guided by a systematic literature review that identifies the major architectural design properties and choices related to blockchain-based systems.
- A mapping approach that associates architectural decisions of blockchain-based sys-

tems with potential security attacks and threats. A threat classification model is used to categorise threats associated with the attacks and the architectural choices. Specifically, we use the Microsoft STRIDE threat model [199] because it classifies threats based on the ramifications of their realisation, such as a denial of service (DoS), disclosure of information, or elevation of privilege.

Mapping the proposed taxonomy with the related security ramifications helps software architects to fully comprehend the impact and scope of the security challenges associated with blockchain systems. The security implications of threats can be directly associated with the likelihood and impact of potential attacks on the whole system. This approach provides a basis for evaluating the potential security risks in all dimensions of the system. We advocate security risk assessment at early stages to provide the engineers with more objective guidance that can assist in further analysis for potential threats and attacks, refining the design and testing for security.

Previous studies have illustrated various security vulnerabilities and the consequent attacks on blockchain-based systems [211, 152, 217]. However, to the best of our knowledge, our work represents a novel initiative in presenting a thorough categorisation of threats and their correlation with attacks, as well as with blockchain architectural decisions that are susceptible to those attacks. Other studies [376, 403] have presented taxonomies of the architectural properties of blockchain systems and showed the impact of these properties on quality attributes of the system such as performance. However, their effects on security properties— possible attacks and their subsequent threats—have not been covered. Moreover, these prior studies did not systematically determine the major architectural decisions that are considered when designing such a system. To the extent of our knowledge, this work is among the initial efforts to present a taxonomy that categorises architectural decisions of blockchain-based systems based on a systematic review of the literature.

The rest of the chapter is organised as follows. Section 3.2 outlines the procedure followed to conduct our review. Section 3.3 provides a comprehensive taxonomy of architectural decisions for blockchain-based systems, followed by Section 3.4 that introduces our mapping approach, and links the classified threats and attacks with our taxonomy. Section 3.5 demonstrates an application of our work, followed by Section 3.6 shows the validation of our work and discusses the threats to validity to our approach. Section 3.7 discusses the related works. Finally, Section 3.8 represents future research directions and concludes the chapter.

## 3.2 Research Methodology

### 3.2.1 Surveying the Literature

Our empirical route [255] is partially guided by a systematic literature review research method [186]. In this way, we increase the possibility of producing an unbiased exploration of the current body of work on the field and select representative articles that can reflect the major architectural decisions for blockchain-based systems along with their characteristics, attributes, and variants. To achieve this aim we conducted the following research question (RQ):

**RQ1:** *What are the architecturally significant design decisions in blockchain-based systems?* The search period for this survey starts in 2008 since the original paper describing blockchain appeared that year [251]. We conducted a trial search first to select and refine the query terms used. We tested several possible queries which resulted in either a huge number of papers or research papers that were unrelated to our survey goals. Finally, we settled on *Blockchain AND (Architecture OR Architectural)*. This query has been designed as open-ended, to provide exhaustive coverage of the literature where the terms



architecture or architectural often preceded many of the keywords that relate to architecture such as design, tactic, quality, model, or concerns. Searches were performed in five electronic databases: IEEEExplore, ACM Digital Library, Science Direct, ISI Web of Science, and Scopus.

Since not all the resulting papers from the search were related to the research questions, they needed to be filtered first. Hence, we identified several selection criteria that were applied to ensure that the outcomes were objective. The inclusion and exclusion criteria we defined are as follows:

**Inclusion criteria (I)**

**I1:** Papers published in peer-reviewed journals, conference proceedings, workshops, or book chapters.

**I2:** Papers focused on one or more blockchain architectural design decisions.

**I3:** Papers reporting on fundamental research into software architecture for blockchain and/or their applications.

**Exclusion criteria (E)**

**E1:** Papers from disciplines different than computer science, in which they use blockchain merely as a component of the application.

**E2:** Papers without full text, papers written in other language than English, and duplicate papers.

**E3:** Papers related to blockchain-based applications without any substantial architectural discussion.

Our study selection procedure employed three rounds to filter the research papers and select the final set.

**First round:** We selected papers based on metadata including title, venue name and keywords. In this round, we considered the criteria I1 and E1.

**Second round:** We independently chose papers by reading the abstracts of the papers. In

Table 3.1: Selected Articles

Publication Type	References
Surveyed and SLR (Secondary study)	[24], [217], [64], [152], [60], [211], [83], [299], [307], [50], [18], [23], [123], [381]
Primary Study	[38], [106], [266], [201], [284], [21], [390], [155], [335], [63]
Grey Literature	[314], [109], [284], [74], [93], [277]

this round, we considered criteria I3, E2, and E3.

**Third round:** We independently chose papers by reading the full text of the papers selected in the previous round. In this round, we considered criteria I2, I3, E2, and E3.

Additionally, we performed a manual search to include relevant papers informed by experience or citations from seminal research. Figure 3.1 illustrates the search and the selection procedure followed.

### Data Extraction

The data extraction process was initially applied to a set of 10 studies that are highly cited and considered the most seminal in the area of blockchain architecture. We read their full text to collect information regarding architecting blockchain-based systems. The considered aspects were: blockchain-based systems *architectural components*, their *variants*, *characteristics*, and *design decisions*. Then, all the extracted information was combined and categorised. We identified the key concepts and dimensions that led to an initial version of our taxonomy. Next, we analysed the full set of selected papers to revise and refine the taxonomy. Therefore, the approach that we followed to build our taxonomy can be described as an empirical to conceptual approach. Publication *titles* and *aims* were collected from all included studies.

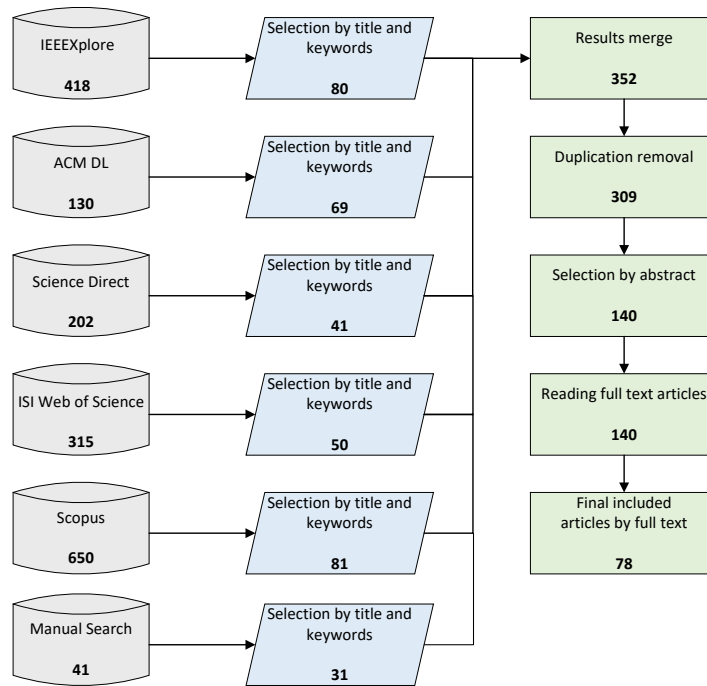


Figure 3.1: Search and Paper Selection Procedure

### 3.2.2 Mapping Approach

The second objective of our study is to map the architectural decisions of blockchain-based systems to threats and attacks. To achieve this, we decided to address the following research question:

**RQ2:** *How can potential threats and attacks be traced to blockchain architectural decisions and to which components?*

There are various strategies that can be used to investigate ways in which blockchain-based systems may be threatened. We focus on common and widely acknowledged security threats with a malicious purpose which are often posed by cyberattacks, compiled from existing literature. We propose attack and threat classification models with the aim of identifying and tracing the threats to the architectural aspects of blockchain systems. The adversarial tactics categorisation proposed by MITRE [245] is used to classify attacks and the STRIDE threat model is used to classify threats. MITRE provides adversarial tactics and techniques

based on community contributions from real-world observations.

We used a general search string (*Blockchain AND (“Security Attacks” OR “Cyberattacks”) AND (“Security Threats”)*) for identifying blockchain-specific threats and attacks. We have specifically searched for seminal and widely cited surveys and reviews on the topic to help us identify commonly documented blockchain attacks and threats. At the time of writing, we found 14 articles that surveyed blockchain-specific threats and attacks. Additionally, we have considered 10 primary studies on the topic, where additional attacks were identified. Grey literature is also considered by searching and analysing the first 200 top results from Google. From the selected results, we have filtered the most relevant reports, unpublished research, and articles that relate to the investigation query and published by public or private institutions or organisations. We included those results that identified new/recent issues that were not covered by the standard academic literature. We excluded generic reports related to blockchain technology without describing specific issues and those that provide superficial and non-elaborating mentioning about attacks and threats. Additionally, when grey literature provides redundant information to that of the peer-reviewed one - peer-reviewed ones were used. Table 3.1 shows the selected articles. Our search identified 56 distinct blockchain specific attacks that are defined and classified. Each of the identified attacks and threats can be traced back to its source(s).

Before the classification and mapping process, we first extracted blockchain-specific attacks and threats from the selected articles, along with their definitions, to ensure that we had a shared understanding. We have only focused on commonly used strategies for launching the attacks as attackers may take novel approaches in combining different tactics to launch/execute the attack. The classification was straightforward for the attacks that are not only targeting blockchain as the MITRE team has classified them under the related tactics. To minimise inherent biases in the mapping process, two reviewers with security backgrounds and expertise worked independently on the rest of identified attacks and threats

to categorise and map them to blockchain architectural dimensions. Once the reviewers completed the task, the results were discussed and verified with the third reviewers, and for each disagreement, all reviewers discussed their rationale to consolidate the results. The final result was reviewed by the fourth reviewer. After analysing the result, we found that several attack tactics proposed by MITRE are unrelated to any of the attacks in our set. Thus, we excluded these tactics from our mapping. Finally, the result was sent to an expert in blockchain security, who reviewed and provided feedback on our classification.

### **3.3 Taxonomy of Dimensions for Architectural Decisions in Blockchain-based Systems**

Taxonomies have an essential role in the software engineering discipline because the categorisation and organisation of the knowledge enable practitioners and researchers to understand and analyse a complex design space and to evaluate and compare design options.

Our work is not only aligned with academia but also with the industry. The industry has organised concepts, components, models, and other elements around blockchain technology to facilitate its understanding [380]. The industry has also proposed a playbook of blockchain [327] that defines a process with five phases to assist in the adoption of the technology: (i) problem assessment; (ii) organisational readiness; (iii) technology selection; (iv) blockchain implementation; and (v) blockchain integration. In this context, our work fits in the third phase since an organisation of architectural decisions in dimensions is intended to support the construction of a platform architecture and an operational model, which are two of the outputs of the referred phase.

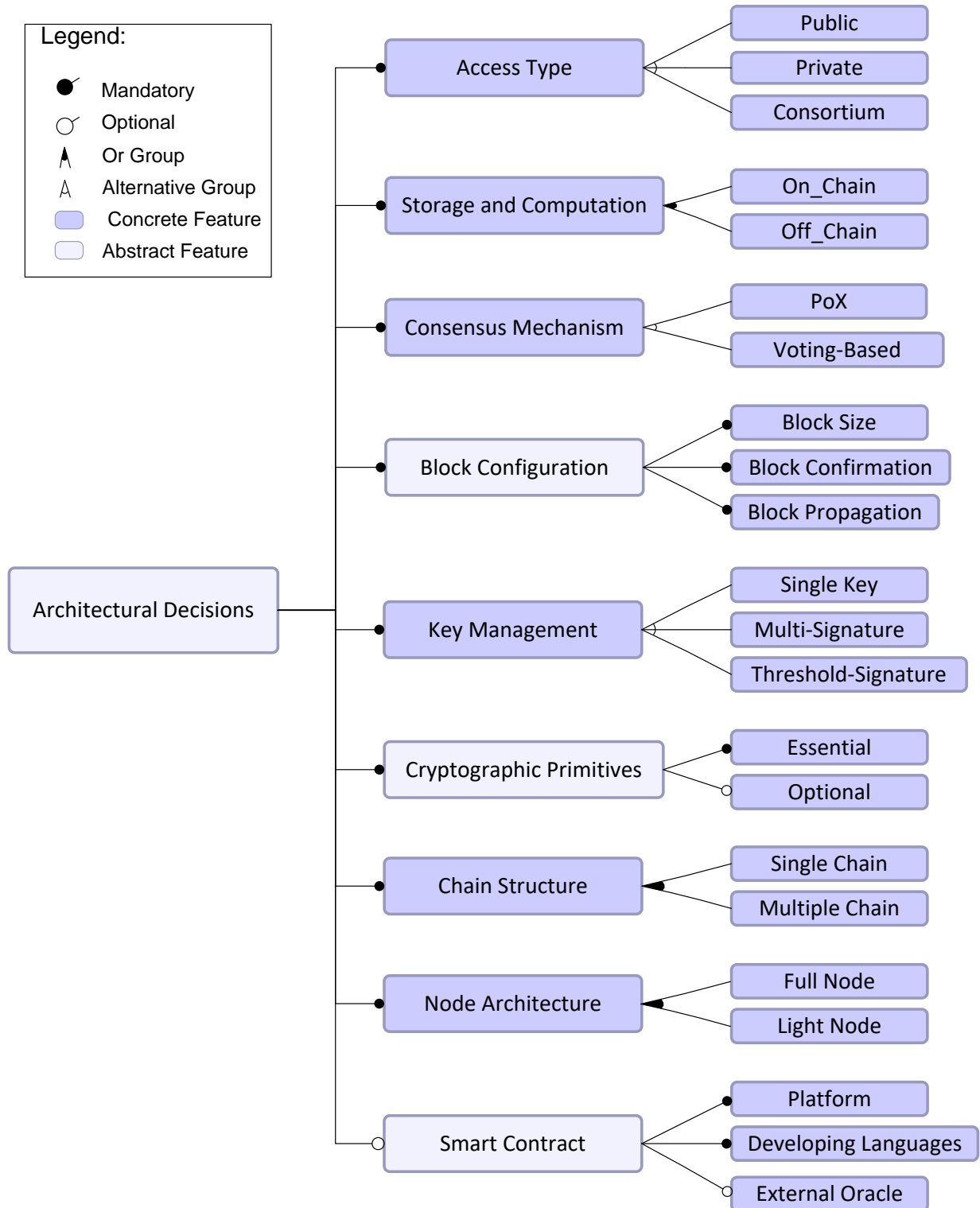


Figure 3.2: Taxonomy of Major Dimensions for Blockchain Architectural Decisions

As Figure 3.2 <sup>1</sup> illustrates, the results of our survey indicate that the dimensions of key architectural decisions are:

(i) blockchain access type; (ii) data storage and transaction computation; (iii) consensus mechanism; (iv) block configuration; (v) key management; (vi) cryptographic primitives; (vii) chain structure; (viii) node architecture; and (ix) smart contract. In the following Subsections, we describe each architectural decision from the security perspective and provide a discussion of the quality attribute trade-offs entailed by each architectural choice. Table 3.2 illustrates these attributes. Table B.1 in Appendix 2 represents selection of studies that belong to each dimension of the taxonomy.

### 3.3.1 Blockchain Access Type

Among the crucial design decisions for blockchain-based systems is the choice of blockchain access, which identifies who is permitted to participate in the network and in what transactions. Blockchain access is classified into three categories: public, private and consortium [404]. Each has its own security properties and limitations.

#### Public Access

In this access type, any node can read, send and verify transactions. This type is often known as a permissionless digital ledger, where nodes are also able to create new blocks of transactions [231]. A public blockchain is fully decentralised; there is no central authority that controls the system [207]. Additionally, information is distributed, shared and recorded in all nodes that participate in the network. Therefore, this type is widely available since they have no single point of failure. Importantly, immutability and integrity of information

---

<sup>1</sup>The diagram follows an extended feature model notation

Table 3.2: Quality Attributes per Dimension of Architectural Decisions

Dimensions of Architectural Decisions		Quality Attributes	
		Upside	Downside
Access Type	Public	<ul style="list-style-type: none"> <li>• Fully decentralised</li> <li>• Widely available</li> <li>• High immutability</li> <li>• High integrity</li> <li>• High level of transparency</li> </ul>	<ul style="list-style-type: none"> <li>• Low privacy</li> <li>• Low confidentiality</li> </ul>
	Private	<ul style="list-style-type: none"> <li>• Centralised administration</li> <li>• High privacy</li> </ul>	<ul style="list-style-type: none"> <li>• Low integrity</li> <li>• Low immutability</li> </ul>
	Consortium	<ul style="list-style-type: none"> <li>• Partially decentralised</li> <li>• Better availability than in private ones</li> <li>• Better privacy than in public ones</li> </ul>	<ul style="list-style-type: none"> <li>• Low integrity</li> <li>• Low immutability</li> </ul>
Storage and Computation	On-Chain	<ul style="list-style-type: none"> <li>• Based on the applied blockchain access type</li> </ul>	

*Continued on next page*



Table 3.2 Quality Attributes per Dimension of Architectural Decisions (*Continued from previous page*)

Dimensions of Architectural Decisions		Quality Attributes	
		Upside	Downside
	Off-Chain	<ul style="list-style-type: none"> <li>• High privacy</li> <li>• High confidentiality</li> <li>• Reduced cost</li> <li>• Low latency</li> </ul>	<ul style="list-style-type: none"> <li>• Low availability in case of a centralised administration</li> <li>• Low integrity</li> <li>• Low immutability</li> </ul>
<b>Consensus Mechanism</b>	Proof-based (PoX)	<ul style="list-style-type: none"> <li>• Unlimited number of nodes</li> <li>• High decentralisation</li> <li>• Free join</li> <li>• Award</li> <li>• Low energy consumption in case of PoS</li> </ul>	<ul style="list-style-type: none"> <li>• Mismanaged node identity</li> <li>• High energy consumption in case of PoW</li> </ul>
	Voting-based	<ul style="list-style-type: none"> <li>• Managed node identity</li> <li>• Low energy consumption</li> </ul>	<ul style="list-style-type: none"> <li>• Not free join</li> <li>• Mostly no award</li> <li>• Limited number of nodes</li> <li>• Low decentralisation</li> </ul>

*Continued on next page*

Table 3.2 Quality Attributes per Dimension of Architectural Decisions (*Continued from previous page*)

Dimensions of Architectural Decisions		Quality Attributes	
		Upside	Downside
<b>Block Configuration</b>	Block Size	<ul style="list-style-type: none"> <li>• Higher throughput in case of large size</li> </ul>	<ul style="list-style-type: none"> <li>• Network congestion and slower propagation in case of a large size</li> </ul>
	Block Confirmation	<ul style="list-style-type: none"> <li>• Low latency in case of fast confirmation</li> </ul>	<ul style="list-style-type: none"> <li>• Increase fraud attempts in case of fast confirmation</li> </ul>
	Block Propagation	<ul style="list-style-type: none"> <li>• Based on the block size</li> </ul>	
<b>Key Management</b>	Single Key	<ul style="list-style-type: none"> <li>• Better accessibility than in other types</li> <li>• Low latency</li> </ul>	<ul style="list-style-type: none"> <li>• Low availability</li> <li>• Low integrity</li> <li>• Repudiation</li> </ul>
	Multi-Signature	<ul style="list-style-type: none"> <li>• High integrity</li> <li>• Non-repudiation</li> </ul>	<ul style="list-style-type: none"> <li>• High latency</li> <li>• Low confidentiality</li> </ul>
	Threshold-Signature	<ul style="list-style-type: none"> <li>• High confidentiality</li> <li>• High integrity</li> <li>• Non-repudiation</li> </ul>	<ul style="list-style-type: none"> <li>• High latency</li> </ul>

*Continued on next page*

Table 3.2 Quality Attributes per Dimension of Architectural Decisions (*Continued from previous page*)

Dimensions of Architectural Decisions		Quality Attributes	
		Upside	Downside
<b>Cryptographic Primitives</b>	Essential Primitive	<ul style="list-style-type: none"> <li>• Hash: immutability, integrity, high efficiency</li> <li>• Digital signature: authentication, non-repudiation, medium efficiency</li> </ul>	
	Optional Primitive	<ul style="list-style-type: none"> <li>• Confidentiality, privacy, anonymity, unlinkability</li> </ul>	
<b>Chain Structure</b>	Single Chain	<ul style="list-style-type: none"> <li>• Manageability</li> <li>• Low privacy</li> </ul>	<ul style="list-style-type: none"> <li>• High latency</li> <li>• Low scalability</li> <li>• Low throughput</li> </ul>
	Multiple Chains	<ul style="list-style-type: none"> <li>• High privacy</li> <li>• Better scalability and throughput than in single chain</li> </ul>	<ul style="list-style-type: none"> <li>• Complex to manage</li> </ul>
<b>Node Architecture</b>	Light Node	<ul style="list-style-type: none"> <li>• Low storage cost</li> </ul>	<ul style="list-style-type: none"> <li>• Partial dependency on full nodes</li> </ul>
	Full Node	<ul style="list-style-type: none"> <li>• High availability</li> </ul>	<ul style="list-style-type: none"> <li>• Costly</li> </ul>
<b>Smart Contract</b>	Platform	<ul style="list-style-type: none"> <li>• Based on the applied blockchain access type and consensus mechanisms</li> </ul>	

*Continued on next page*

Table 3.2 Quality Attributes per Dimension of Architectural Decisions (*Continued from previous page*)

Dimensions of Architectural Decisions		Quality Attributes	
		Upside	Downside
	Developing Lan- guages	<ul style="list-style-type: none"> <li>• Based on the applied blockchain access type and consensus mechanisms</li> </ul>	
	External Oracle	<ul style="list-style-type: none"> <li>• High availability and trustworthy in case of a decentralised oracle</li> <li>• High efficiency in case of a centralised oracle</li> </ul>	<ul style="list-style-type: none"> <li>• Low availability and low trust in case of a centralised oracle</li> <li>• Low efficiency in case of a decentralised oracle</li> </ul>

are also supported in public blockchains, as any node can verify that the data has not been tampered with and, once the information is written in the block, it cannot be altered without detection because it is stored in different nodes in the decentralised network [197]. The level of transparency of this type of blockchain is high as well, since the records and their updates are available to the public. While public blockchains are highly transparent and the chained information is visible to other peers, privacy is difficult to achieve; extra cryptographic mechanisms are required to strike a balance between transparency and privacy. Hence, the decision for adopting this type of blockchain requires architects to consider the trade-offs between privacy and transparency in the given context.

### **Private Access**

In this type, often known as permissioned blockchains [20, 231], only one organisation has written permissions, while read permissions can be public or restricted to a preselected set of readers [207]. Since private blockchains are controlled by a single group, they are known as centralised blockchains. In this type, all validators are known as they are all members of a single group- and a centralised authority controls the network by verifying each node, and allowing or rejecting requests to join the network. These specific features of private blockchains give them some security advantages over the public type. Private blockchains provide a greater level of privacy [197], especially when read permission is also restricted [402]. However, private blockchains require trust in identities, especially when the number of nodes in the network is low as the parties might act in collusion to threaten the system. Incorrect trust assumptions, when selecting participants in the network, can have security ramifications. Moreover, as public verifiability is not required, the integrity of the system can only be ensured if the system is not breached.

## **Consortium Access**

This type of access is partially decentralised, meaning that rather than the system being controlled by a single organisation, a group of pre-selected nodes from multiple organisations are responsible for consensus and block validation [94]. Read and write permissions can be determined by the consortium; they can be public or limited to selected nodes in the network [231, 283]. This type of access provides a balance between public and private. Since it is partially decentralised, it provides better availability than private access, while it has better privacy than public one, as it is partly private. The possibility of tampering, however, is one of the significant security limitations of this type of access. Since the chain is controlled by a group of nodes, they can collude and alter or reverse the transactions, which negatively affects the immutability promise of the technology. *Hybrid* is a type of blockchain that combines the features of public permissionless and private permissioned classes [367]. Participants can manage the access feature and decide who gets access to which data on the blockchain. This allows systems to operate with transparency without having to reduce system privacy [90].

Recognising the security strengths and possible risks of each type of blockchain access assists architects to choose the model that best matches the requirements of the systems they are attempting to build. In practice, an access type that is optimal for a particular case, such as financial systems, may be sub-optimal for another case, such as healthcare systems. In [368] authors provided a methodology to determine the appropriate access type for a particular blockchain systems scenario. Additionally, [231] and [197] explain the differences between each type.

### 3.3.2 Storage and Computation

Blockchain technology choices have implications on storage space and computation. Choosing to place data and computation on-chain or off-chain is a critical architectural decision that involves several trade-offs such as cost, privacy, and integrity of information.

#### Data Storage

Several properties must be considered when deciding to store data on- or off-chain, such as scalability, performance, privacy, and confidentiality [377]. Confidentiality and privacy of sensitive information stored on a public blockchain cannot be guaranteed, as the content is visible to every node joining the chain. In some applications, data is required to be visible to specific nodes, not to all the nodes in the blockchain network. In this case, storing data off-chain can be helpful to overcome and mitigate such limitations [383]. Commonly, when there is a decision to include off-chain storage, it will be used to store raw data, while meta-data and hashes of raw data will be stored on-chain. A set of on-chain data management patterns have been proposed in [375].

The decision to select off-chain storage needs to be taken deliberately, because inadequate analysis of the security consequences of any storage type can lead to security risks. For example, an architect might decide to use a centralised solution, such as private cloud storage, as it is easy to configure and manage; yet this solution could become a single point of failure. Moreover, if the raw data which is stored off-chain is deleted or lost, it cannot be recovered from its hash value which is permanently stored on-chain. Another option would be to use a peer-to-peer decentralised file sharing platform such as IPFS (InterPlanetary File System) [28, 372, 401], Swarm [151], OrbitDB [265] or Filecoin [280]. Due to the decentralised nature of these kinds of storage, if one node disconnects, the data can still be

accessed. One of the drawbacks of off-chain storage is that the integrity of the raw data is based on the soundness and the security of the hash algorithm that is applied to hash the raw data, as Subsection 3.3.6 discusses.

## Transaction Computation

Transaction execution, validation and consensus mechanisms in blockchain increase the response time as they require communication and execution overheads. Moreover, mining processing – the process of assigning new blocks to the chain – is expensive because it typically involves a fee. Some blockchain applications require micropayment transactions, payments of small amounts of money, often just a few cents. This kind of transaction is very costly to be done on-chain because the transaction fee that is required to execute the transaction might be higher than the cost associated with the transaction. As it is infeasible to execute and store each micropayment transaction on-chain, several applications which require such kinds of transactions have been established to create a separate off-payment channel between the participants [405]. This construction helps to reduce latency and cost and increases throughput. The constructions are also commonly known as Layer-2 channels or state channels.

There are several protocols available for implementing off-chain payment channels, each of which has its own advantages and disadvantages. Lightning Network is a protocol that allows the routing of payments through several intermediary nodes. This approach reveals information about the nodes and the performed transactions, as the intermediary nodes can see the flow of funds through the channel. To mitigate this problem, Bolt protocol has been proposed [144]. This protocol includes a set of techniques for building anonymous payment channels. There are also other methods that have been proposed to preserve the privacy of users of the off-chain channel. A comprehensive analysis of the off-chain channel



and its related protocols can be found in [173]. A set of off-chaining patterns were also proposed in [104, 246]. The objective of these patterns is to move computation and data off-chain, without compromising blockchain features such as trustlessness.

### 3.3.3 Consensus Mechanism

While blockchain is a decentralised technology that relies on a decentralised authority to manage, authorise and verify the transactions, a fault-tolerant consensus protocol is required, which is a set of rules to assure that all nodes agree on the new block that is appended to the blockchain. Transaction verification and immutability depend on the selected consensus mechanism. There are several consensus mechanisms in use in the existing blockchain technologies. One crucial point is to understand the type of faults and trade-offs relating to the application domain to select an optimal consensus protocol that helps to secure the blockchain. When a sub-optimal protocol decision has been taken, replacing the consensus protocol in blockchains will be challenging, requiring a serious code rewrite. Researchers have suggested that a general-purpose permissioned blockchain should be built with pluggable consensus mechanisms, and it has been emphasised that there is no “one-size-fits-all” solution [345]. Hyperledger fabric is the first blockchain that has come with a pluggable (not hard-coded) consensus protocol.

A consensus protocol inside a blockchain can be classified into two classes [253]. The first class is the proof-based consensus mechanism, also known as Proof-of-X (PoX). This type of mechanism is preferable when the expected number of nodes joining the network is large as in the case of public blockchain. The second class is a voting-based consensus mechanism, also known as Byzantine Fault Tolerant (BFT)-based consensus. This type is more applicable to consortium and private blockchains where the number of nodes is often restricted and all the nodes inside the network are known and adjustable. However, PoX

is also applicable to the consortium and private blockchains and is not restricted to public ones.

The core concept of PoX consensus is to give a right to the node or set of nodes that show or accomplish a specified proof to be allowed to append a new block to the chain and receive a subsequent incentive. Several variants of this consensus mechanism have been proposed in the literature. The main one is Proof-of-Work (PoW) [137] which was proposed when Bitcoin appeared in 2009. This consensus type requires solving a puzzle with adjusted difficulty, demanding high computational power consumption. The solution of the puzzle involves a group of nodes; the nodes that first solve the puzzle are then allowed to broadcast their blocks to the blockchain network. The second popular protocol in this type of category is Proof-of-Stake (PoS) [180, 30] which was proposed in 2011 and claimed to mitigate the high resource consumption and the limitations of PoW. The main idea of this consensus type is utilising stake to determine the possibility of nodes mining the subsequent block of the chain. Hence, the nodes with a higher stake have a higher chance of validating and broadcasting the block. Although PoS requires lower energy consumption than PoW, the latter provides better decentralisation [45]. Often, the number of miners in the PoW is much larger than the number of validators in PoS. There are several variants have emerged to tackle the drawbacks of PoS including Delegated Proof-of-Stake (DPoS) [209], CloudPoS [45], and Proof-of-Supply-Chain-Share (PoSCS) [45]. In addition to PoW and PoS, there are multiple consensus protocols that can be classified under PoX category including Proof-of-Space [102], Proof-of-Activity [31], Poof-of-elapsed time [62], and more [253, 383, 241].

The main concept behind a voting-based consensus mechanism is that agreement to append the block to the chain is based on the majority of node decisions. In particular,  $K$  nodes, where  $K$  is a given threshold, are required to show the same proposed block before accepting it in the chain. Most of these algorithms require at least one process to receive and validate the votes from all other processes and then broadcast the result. There are

several consensus protocols proposed under this category such as Practical Byzantine Fault Tolerance (PBFT) [56] which has been used by the Hyperledger blockchain platform, Ripple [310], R3 Corda [71] with BFT-SMART [33], Quorum with Raft [198], and more [253, 283, 407, 323]. More about consensus algorithms can be found in [253, 357].

### 3.3.4 Block Configuration

There are three main aspects to be considered regarding block configuration: block size, confirmation, and propagation.

#### Block Size

This refers to the maximum number of transactions aggregated within a block. The optimal block size is still a debatable subject. The system's throughput is sensitive to the size of the block as increasing its size is a way to enhance the throughput of the blockchain system. However, arbitrary increases in the block size without carefully analysing the consequences of this increase can adversely affect the system. Large block sizes can cause network congestion and slower propagation speeds, which, in turn, result in raising the number of stale blocks [137]; these are blocks that are not joined to the longest chain because of a conflict or concurrency. This results in wasted computational effort of the miners, who simultaneously create large blocks, and could be exploited by attackers seeking to disrupt the network. Moreover, an attacker could exploit the size of a large block to multiply the impact of their malicious actions. This could involve embedding harmful transactions or smart contracts within a large block, causing more extensive damage when validated. A malicious actor can also intentionally create a large block containing numerous fraudulent transactions. This could slow down the validation process and potentially lead to a backlog of transactions.

A trade-off between security risks and throughput requires an analysis at an early stage when deciding on the block size. Alternative solutions need to be taken into account to enhance the throughput of the blockchain systems such as second layer payment channels [375].

### **Block Confirmation**

Commonly, in blockchain-based systems, a transaction is confirmed after waiting for a certain period, which can be a specific number of blocks that have been created once the transaction has joined the blockchain. Deciding on the required number of blocks for confirmation is a critical design decision. This strategy has been used to guarantee that a transaction is attached to the longest chain securely. However, researchers [405, 383] have argued that real-world businesses often require an immediate response, as no one wants to be at risk of losing assets during the waiting time. Therefore, an immediate confirmation has been proposed in some consensus algorithms such as PBFT and Proof-of-Familiarity [383]. Another strategy for transaction confirmation is to add a checkpoint to the blockchain [376]. The transaction is accepted once the checkpoint is valid; otherwise, if the fork chain starts before the checkpoint appears, it will be rejected by all nodes.

### **Block Propagation**

In a blockchain network, a broadcast protocol is needed to distribute blocks to the peers in the network. The architectural decision of the underlying broadcast protocol affects the security, reliability, and scalability of the network [115]. Several protocols have been proposed to deliver blocks to the nodes in the network. Advertisement dissemination [137] is one of the common protocols that has been used by the PoW blockchain. In this protocol, if node A receives a new block from another node in the network, it advertises the hash of the block to

its other connections. If one of the nodes, e.g. node B, has not previously received this block, B will demand it from block A which will then send the contents of the block to B. Other propagation mechanisms are proposed, such as send header, unsolicited block push, and relay networks, with the aim of reducing the propagation delay as blockchain forks are caused due to long propagation time [137]. Obviously, there is a correlation between the propagation latency and the size of the block as a large block is propagated slowly in the network which allows an adversary to leverage this delay. Multiple ways have been introduced for enhancing the propagation of the blocks such as minimise verification, pipelining block propagation, and connectivity increase [86].

Block configuration aspects are often based on the blockchain platform. At the time of writing the average Ethereum block size is 20 to 30 KB, transactions are confirmed every 15 seconds, and an advertisement hybrid propagation mechanism is used for block propagation. Therefore, it is essential that the developer knows these details to select an appropriate platform for their system.

### 3.3.5 Key Management

This Section explains the various ways for signing transactions and different options for storing the users' keys.

There are alternative signature schemes to sign the transaction in a blockchain: using a single key, a multi-signature scheme, or a threshold signature scheme. Storing and depending on only a single key to sign sensitive transactions, such as financial or cryptocurrency transactions, introduces a single point of failure, which contradicts decentralisation and the distributed trust concepts of blockchain technology [96]. To mitigate this threat a multi-signatures mechanism was proposed [142]. This mechanism requires multiple secret

keys to generate the signatures,  $M$  signatures for  $N$  private keys are required to sign any transaction. A simple example is a two-factor wallet [233] that requires two devices, such as the user's mobile phone and laptop, to sign any transaction. However, this scheme increases the transaction size linearly with the number of signatures which subsequently increases the transmission time. Additionally, this scheme negatively affects the confidentiality of the transaction, since it will be visible in the public block that a multi-signature transaction has been used.

Threshold signatures are an alternative signature scheme where the transaction can be signed using shares of a single private key. These shares are split among  $N$  parties using threshold cryptography. This scheme provides the same  $M$ -of- $N$  security but increases confidentiality since transactions are indistinguishable from non-threshold transactions on the blockchain and the parameters  $M$  and  $N$  are kept private. ECDSA threshold signature has been introduced by [135] and [141] to enhance Bitcoin security. Other research [319] proposed RSA and BLS threshold signatures for the Hyperledger Fabric blockchain platform. The security of the threshold signature scheme is based on a cryptographic algorithm that is used to apply the digital signature as described in Subsection 3.3.6.

There are several alternative ways to manage and store private keys, each of which has its own security implications [299]. In most blockchain applications users use a piece of software, called a wallet, to store their private keys securely. Public keys and associated addresses can also be stored in the wallet [52]. Keys can be stored in an off-line wallet which is the most secure type; however, it is inconvenient to use, and it is commonly used as a backup. Alternatively, online wallets can be used, which is more convenient, but the server can steal such keys. A local or device wallet is another kind of wallet where the keys are stored directly in a specific file; thus users can have full control over their keys.

### 3.3.6 Cryptographic Primitives

Cryptography is a key component of blockchain technology, as the security of the whole blockchain system is based on the security of its underlying cryptographic primitives. Compromising one of them adversely affects the security of the entire blockchain system. Cryptographic primitives in blockchains are classified into: essential and optional.

#### Essential Cryptographic Primitive

This type includes hash functions and digital signatures.

**Cryptographic Hash Function:** In blockchain, the hash function is used in many operations such as addresses and block generations, message digests in signatures, and in some consensus mechanisms such as PoW. A secure hash function should be collision resilient, tamper-resilient and should be a one-way function, where its result should be easy to verify and hard to invert [355]. These properties provide two security attributes to the blockchain system: immutability and integrity. Secure Hash Algorithms (SHA256) and RIPEMD160 [305] are popular hash functions in blockchain and have been used in most blockchain platforms [355]. However, there are also several hash functions that have been used in different platforms such as Ethash [344] in Ethereum, SCrypt in Litecoin [117] and other platforms. Based on [196] results, SHA256 and RIPEMD160 are among the fastest algorithms and have the best performance. This is because these algorithms can validate blocks without occupying a lot of memory space and processing power. As [355] stated the efficiency of hash algorithms is the highest comparing to other types of cryptographic primitives. In [355, 196], authors comprehensively analysed the performance of hash functions suitable for use in the blockchain.

**Digital Signature:** Asymmetric-key cryptography primitive is used to generate a digital signature, where each node should have a pair of a private key and a public key. The private

key should be kept secret since it is used to sign the transaction, while the corresponding public key can be used by any node in the system to confirm the ownership of the signed transaction and to verify that the transaction has not been modified or tampered with. One of the security properties that secure digital signatures provide is authentication where a valid signature indicates that the transaction is signed by a known user. Non-repudiation is another security property where the node that sent the transaction cannot deny it. Moreover, a valid signature can guarantee the integrity of the transaction and that it has not been altered in transmission. The key generation algorithm of a digital signature scheme should have a good randomness source to generate different key pairs for different users. A weak randomness source could allow an attacker to recover the user's private key and sign transactions. In [237] a vulnerability is reported in an elliptic curve digital signature algorithm (ECDSA) [305], which is used in the most common algorithm in several blockchain platforms, such as Bitcoin and Ethereum. This algorithm fails to generate enough randomness during the signature process, which allows an attacker to discover the user's private key. However, based on [116] analysis, ECDSA requires lower computation and it is much faster than other type of digital signature including RSA and DSA. These two algorithms require longer keys to provide a safe level of encryption protection compared with ECDSA which requires much shorter keys and that leads to much better performance. As stated in [355] the efficiency of digital signature is medium comparing to other type of cryptographic primitives. Alternative digital signatures, listed and explained in detail in [355], are also used by several blockchain platforms.

The choice of secure cryptographic primitives is essential as a vulnerable one can weaken the entire blockchain system. The main cryptographic schemes used in blockchain, including ECDSA, depend on the difficulty of prime factorisation and the discrete logarithm problem. However, these schemes can be threatened by applying quantum algorithms that work in a polynomial time to break such schemes. Thus, using quantum-resistant cryptog-



raphy and post-quantum mechanisms, which utilise cryptosystems that stay secure under the assumption that an attacker is in possession of a large-scale quantum computer, are recommended by [129, 390, 206]. However, the authors in [182] argued that the robustness of these algorithms is based on unproven assumptions, and they are computationally intensive. They suggested the use of quantum key distribution (QKD), which provides security based on the laws of quantum physics where a secret key is distributed using the quantum channel. As quantum physics is fundamentally random [129], the bit stream generated by a quantum random number generation (QRNG) is provably random. Thus, this ensures the generation of truly random encryption keys. A thorough analysis of QKD and QRNG can be found in [182].

### **Optional Cryptographic Primitive**

This includes the symmetric algorithms and other cryptographic primitives that are used mainly for enhancing the confidentiality, privacy, anonymity, and unlinkability of blockchain-based systems. One or more of these properties might require to be provided by the system under consideration, especially in the case of a public blockchain, as the chain's state is transparent, and everyone can access the chain without restriction. Therefore, cryptographic primitives that preserve identity and transaction privacy need to be considered in the decision-making process.

There are several cryptographic methods that aim to protect identity and transaction privacy in the blockchain, including zero-knowledge proof (NIZK) [146] and Homomorphic encryption [136]. Zerocash [304] is a privacy protocol that makes use of NIZK proof and a homomorphic scheme to achieve both anonymity and transaction privacy. However, it involves high computational costs for generating transaction proofs. Lelantus [171] protocol has emerged to enhance the confidentiality and privacy of the blockchain and mitigate the

disadvantages of zerocash. Multiple privacy protocols leveraging one or more of the cryptographic schemes have emerged [190]. Authors in [355], systematically discussed, analysed, and compared the cryptographic schemes.

### 3.3.7 Chain Structures

Creating single chains or multiple chains is one of the essential architectural decisions to design a secure, reliable, and efficient blockchain-based system.

#### Single Chain

All transaction types generated in a blockchain-based system are recorded together in a unique chain when this type is selected. Clearly, a single chain is easier to manage; yet it increases latency and negatively affects the scalability and the throughput of systems [379], especially when a sub-optimal data structure is implemented. In a classical blockchain such as Bitcoin, the data structure is a linked list of blocks that creates a chain and, when a conflict appears, the longest chain is selected by the nodes in the network. This selection rule increases the chances of double spending threats. The Greedy Heaviest-Observed Sub-Tree (GHOST) protocol was proposed by [316] and has been used by Ethereum, but it changed the selection rule to select the side whose subtree has the most work. This protocol enhances security by increasing mining fairness. Directed Acyclic Graph (DAG) is an alternative structure that has been proposed by [205] to enhance the security of a traditional blockchain structure. They changed the structure from a list to a graph where each block references multiple predecessors. This structure is better suited when the block size is large or when blocks are frequently created. Analysis showed that this structure considerably enhances the mining power utilisation [205]. TrustChain [266] is another data structure which can be used in permissionless blockchain systems. This structure includes a NetFlow algorithm

that calculates the trustworthiness of the nodes using the prior transaction as input. This algorithm makes blockchain systems more secure as it prevents untrusted fake identities from joining the network. There are several data structures that have been demonstrated in the literature including LeapChain [286] and segregated witness [376].

It is worth noting that a longer chain in a blockchain can introduce security vulnerabilities as it takes more time to propagate across the network. This can lead to variations in the blocks seen by different nodes, potentially causing temporary forks [114]. Attackers could exploit these forks by intentionally creating conflicting transactions, resulting in confusion and potential double-spending. A longer chain also increases the time and resources required for synchronisation. If synchronisation is inefficient or slow, it could create a window of opportunity for attackers to exploit inconsistencies or vulnerabilities in the synchronisation process. Moreover, a longer chain can lead to increased storage costs and bandwidth requirements for participants. This economic burden might prompt some participants to opt for lightweight or simplified validation methods, reducing overall network security and decentralisation. Therefore, it's essential for blockchain software engineers to address these vulnerabilities through strong security practices.

## Multiple Chains

Instead of storing and executing all types of transactions and information on one chain, information and transactions can be classified, executed, and stored in more than one chain. There are several structures involving multiple chains that have emerged to overcome scalability and throughput issues in the blockchain. The sharding scheme, also known as a layer-1 scalability method, is one of the solutions aimed at improving the performance of the blockchain. It does this by splitting the processing of transactions among smaller sets of nodes, called shards. These shards operate separately and in parallel to enhance throughput

and decrease storage and computation overheads. However, the possible corruption of the shards is one of the security issues that need to be tackled, in that malicious nodes can easily dominate a single shard. To tackle this concern a sharding scheme requires the random division of the network into small shards to prevent any shard from accepting an overwhelming number of adversaries. Ethereum 2.0 and RapidChain [394] are sharding mechanisms that randomly attach nodes to specific shards, and regularly reassign them at random intervals. A detailed explanation and systematic analysis of several sharding protocols can be found in [78].

Another scalability solution is called sidechain [223], which is an independent blockchain that has its own ledger. However, it is not a standalone platform as it is linked to the main chain to allow the assets to flow from one chain to other. This scheme can extend the blockchain to support multiple applications without increasing the load on the main chain. Satellite chains [210] are type of multiple chains, where each interconnected but independent sub-chain can have a different consensus mechanism running privately in parallel in single blockchain system. Double chain [204] is another instance that aims to ensures the privacy of the user information by storing users' information in one chain, while transactions are stored in another chain. Multiple chain structures are intended to enhance performance and privacy of blockchain systems. However, the nodes classification in this type of chain might be complex and it requires a specific management strategy [170]. Additionally, not considering secure protocols for exchanging information, assets or tokens among different chains can weaken the security and soundness guarantees of blockchain systems. Unitary Inter-chain Network Protocol on Transport Layer (UINP) supports cross-chain [366][216] schemes from the transport layer. This protocol gives low latency convenience and provides high security similar to Transport Layer Security (TLS) [366]. UINP employs point-to-point data transmission that can match transactions without requiring a third party for validation.

### 3.3.8 Node Architecture

Blockchain is a peer-to-peer decentralised network where the participant nodes can be full nodes or light nodes. The job of full nodes is to keep and verify a local copy of the entire chain of transactions. Each full node can access all the historic transactions and verify if the new one is valid and consistent with the existing transactions. As all records are replicated in each full node, blocks and transactions can be verified locally, if one of the full nodes is down the blockchain network is not affected because there are other active full nodes in the network [40]. These properties contribute to the availability of the blockchain network and reduce a single point of failure threat. Pruned node is a full node that erases some data when it reaches a particular limit to allow the new blocks to be stored and preserve blockchain size [125]. Even though having full nodes adds robustness to the blockchain network by omitting the need for a centralised party, this type of node requires a substantial amount of storage and processing cost to download a full copy of the entire chain and to execute the verification operations. Since often the size of the blockchain increase linearly due to the immutability and append-only properties, the overhead of the full nodes continually grows. As a result low-end users, who have mobile and smart devices, may be reluctant to participate in a blockchain.

To overcome the aforementioned drawbacks and allow end users with limited resource devices to join blockchain systems, light nodes were introduced [251]. In this type of node, downloading and verifying the full set of blockchain transactions is unnecessary as the light nodes only need to store block headers that are requested from a full node to employ transaction verification. Simple Payment Verification Protocol (SPV) can be utilised by light clients. They request the block header from a full node which, in turn, provides them with the required information. However, requesting a block header from a single full node introduces a serious security risk as the full node can behave maliciously and provide a fake

header to the light nodes. To mitigate this risk, light nodes can request block header from multiple full nodes and then compare the received results. This approach adds overheads to the light node as it needs to establish a secure connection to each full node and that can slow the verification operation and add complexity. There are several strategies presented to address this issue such as the Distributed Lightweight Client Protocol (DLCP) [72]. In this protocol, the request can be encrypted before sending it to a predetermined number of full nodes which can access and process it and then relay it to the light client with a single response. Then the light client is required to decrypt the response only once for verification irrespective of the number of contributing full nodes to ensure that all full nodes agree in one response. This approach reduces computing and communication complexity on the client's side and provides lower latency.

There are several protocols that have been proposed to mitigate the computation overhead and storage size from the client's side, such as Blockstack [12], distributed hash tables (DHTs) [3] and more [72]. The architects need to investigate and analyse the security properties of these protocols to mitigate the potential risks, especially if the application domain needs a blockchain with hundreds of gigabytes in size and lightweight users participating in the blockchain application.

### 3.3.9 Smart Contracts

Smart contracts offer a general-purpose computing platform to provide more complex programmable transactions [211] [152]. A smart contract is a decentralised piece of code designed to impose the negotiation of a contract's instructions automatically without the need of a central authority to approve it. Importantly, once the contracts are deployed, they cannot be modified and the author of the contracts will not have any control over them. Thus, a cautious approach is required when designing and structuring smart contracts.

The reliability and security of smart contracts are based on the consensus mechanism and the decisions of the underlying blockchain platform, programming language, and including an external oracle. Selecting of blockchain platform depends on the access type that the developer decides to choose. There are multiple platforms for developing smart contracts, each of which provides various features. In [119], the authors provide a decision module to assist the developers in selecting the suitable platform based on its criteria and quality attributes. Each platform supports one or more than one programming languages. Some platforms, such as Hyperledger fabric, support general-purpose programming languages such as Java, Python, and GO [20] [294]. As claimed in [345], using such a language facilitates and accelerates the development process as the developers are not required to learn a domain-specific language and they can continue with familiar languages. Some applications require domain-specific languages [218] [262], such as Solidity [81] and Vyper [346], the two most active and maintained languages used in the Ethereum platform, to enhance the security of smart contracts, making them like traditional contracts and more straightforward to understand. In [79] and [294], the authors provide a comparison of the available platforms and discuss the supporting languages. Considering security patterns [364] [219] and adhering to best practices [91] [140] are also crucial to guarantee the correctness of the contracts.

One of the critical design decisions related to smart contracts is introducing an external oracle to the isolated blockchain environment [376]. While smart contracts cannot access any data from outside the blockchain environment, a trusted third-party oracle can provide the contracts with the required information. Blockchain oracle is an external data agent that accesses real-world data and transmits it to the blockchain to be leveraged by smart contracts [364]. Moreover, the external oracle role is not restricted to fetching the data from outside of the blockchain, but it can ensure the validity and the authenticity of the fetched data to guarantee a valid execution of the contracts [232]. Blockchain oracles, such as Provable [282], can retrieve data from a centralised server. The efficiency of this

type is high, but a single point of failure is introduced, which might affect the availability and accessibility to the data. A distributed type of oracle, such as ChainLink [107], resolves these issues as it contains several redundant oracles servers. These servers are trusted by the whole blockchain network and do the same job of checking the external state. However, the efficiency of this type is low as it leads to higher latency for data processing [221].

### 3.4 Mapping Threats and Attacks with Blockchain Architectural Decisions

Even though the security properties of blockchain technology help to make the system based on it resistant to some kinds of attacks, they do not make it completely immune. This is because the system may be subject to a number of security threats if inappropriate architectural design decisions are made.

The architecture of a software system is the set of structural design decisions that serve as a blueprint for the construction and evolution of the system [27]. Decisions that transversely impact the system include the selection of technological platforms, the selection of structural components, and quality attributes (e.g. security, performance) [290]. In particular, architecture-related security concerns can be fixed more efficiently if they are identified and assessed at an early stage.

Multiple studies have shown how weaknesses in a software system's architecture may have a greater influence on numerous security concerns that allow adversaries to attack the system [168, 167]. Schemes for architectural security analysis have also been proposed to identify potential attacks and threats before the system is developed [13]. Additionally, our previous work [5] proposes an approach to assess smart contract security design weaknesses.



This increases the awareness of the developer of the potential security issues before publicly deploying the contract.

Making use of threat modeling is a crucial part of the development process when it comes to enhancing the security of the system. Microsoft reported that security vulnerabilities significantly decrease after including threat analysis in the development process [330]. Threat modeling assists in focusing on firstly addressing threats that represent the highest risk by following appropriate mitigation strategies. Threat modeling involves a structured method that is more cost efficient and effective than conducting security analyses in a haphazard manner without recognising distinct threats in each architectural component of the system.

In this study, we use the STRIDE threat modeling classification approach proposed by Microsoft to classify different threat types into six categories as shown in Figure 3.3. This approach classifies the threats based on the implications of their realisation, such as the manipulation of information, denial of service, and elevation of privilege. The ramification of threats can be mapped to the impact of their incidence, as such mapping is crucial for the assessment of the security risks in blockchain systems. Each class of the STRIDE model covers the unique sort of attacks that lead to a specific type of threat. Noticeably, one attack can pose several threats in such a threat classification; for instance, a majority attack – also known as a 51% attack – can pose multiple threats such as tampering with transactions, disclosing sensitive information and/or the elevation of privilege [252]. The STRIDE threat model has been used in the blockchain context by [175] to classify and analyse the risks associated with blockchain-based records management. However, to the best of our knowledge, we are at the forefront of identifying and classifying the threats at a low-level of the blockchain-based systems to link them to specific architectural decisions.

We categorise the possible attacks with regard to blockchain systems based on the

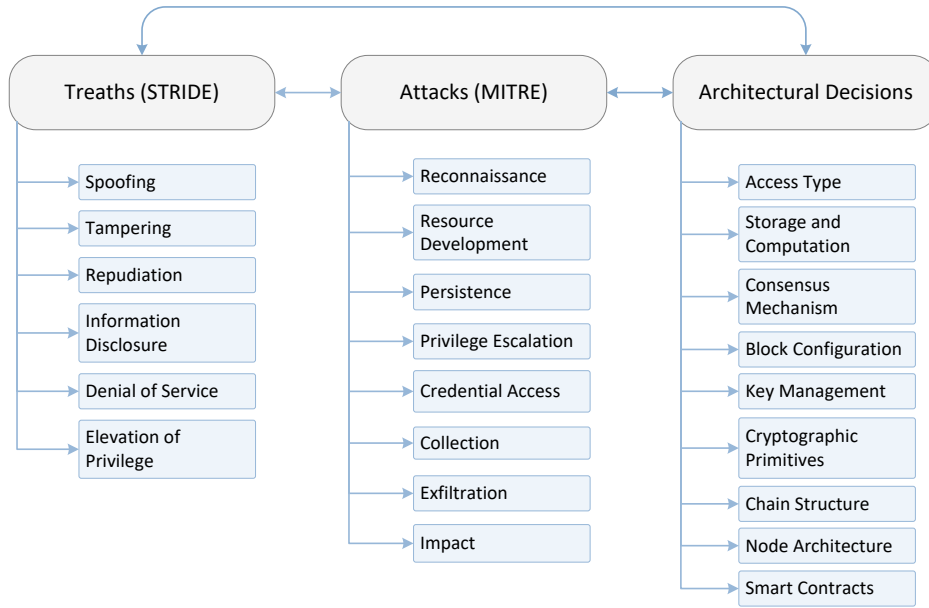


Figure 3.3: Mapping Threats, Attacks and Architectural Decisions

adversarial tactics categorisation proposed by MITRE [245]. Each tactic characterises a high-level description of an attack’s behavior. Previous studies have categorised the attacks in terms of organisation and accessibility point of view [211, 152], and some have illustrated possible attacks for specific blockchain applications [68, 123]. To the best of our knowledge, this work is among the initial efforts that attempt to shed light on not only the attacks but also the threats posed by each form of attack.

### 3.4.1 Attacks and Threats Classification in Blockchain-based Systems

This Section shows how an architect can use the categorisation of attack information to identify threats in a blockchain-based system by considering the following steps: **(i)** determine architectural dimensions of the blockchain based-system and how each attack category can breach them; **(ii)** determine the threats that affect the architectural dimensions

of blockchain-based system; and **(iii)** determine the security threats caused by each attack category using the STRIDE threat model. This way, an architect can map all applicable attacks and their associated threats in the blockchain system. Figure 3.3 captures the essence of the mapping.

### Linking Attack Categories with Blockchain Architectural Decisions

Attacks that target blockchains are aggregated and classified based on MITRE's attack tactics. In this work, we select attack tactics categories that are applicable to blockchain systems. Each tactic represents the objective that adversaries attempt to accomplish. Table 3.3 shows attack tactic categories and the related techniques that each architectural dimension of a blockchain-based system might be prone to. The selected attack categories are illustrated with examples of attacks that are applicable to the blockchain systems as follows:

**Reconnaissance:** The adversary's goal here is to aggregate sensitive information. To achieve this goal, an adversary might apply active scans to gather information by probing victim infrastructure via network traffic. This helps the attacker to accomplish further attacks such as *deanonymisation* [38] attacks to a public blockchain, in hopes of identifying nodes' identities and grabbing useful information that should not be known to the adversary. *Man-in-the-middle (MITM)* attacks [7, 106] target private and consortium blockchains to violate node privacy or gather data exchanged between nodes and these blockchain networks. The adversary can also leverage *malware attacks* [83] to gather the information that might help to achieve other objectives, such as gathering users' wallet information to steal their private keys. Also, there are multiple attacks belonging to this tactic that target smart contracts. Attacks classified under this tactic are mostly intended to compromise pseudo-anonymity, confidentiality, or the privacy of targeted blockchain-based systems.

**Resource Development:** The adversary's goal in this tactic is to establish or compromise

resources that can be exploited to support further malicious operations. Such resources include a large number of pseudonymous identities, or fake identities, that appear to be different nodes when, in reality, they are all under the control of a single party. Therefore, the attacker can gain influence and control a majority of nodes in the network; this action is known as a *Sybil attack* [266] and targets public blockchains. Block withholding [299] and *Finney attacks* [7] are intended to create conflicting views about a blockchain. These lead to hiding, forging, or withholding important information that must be transmitted across the network. Reputation-based attacks [83], such as *whitewashing* and *hiding block attacks*, are considered a part of this tactical approach. A malicious node can change its reputation from negative to positive by eliminating its current identity and creating a new one. Noticeably, attacks using this tactic mostly target the mining process and consensus mechanisms. As a result, this might affect block configuration, chain structure, and several aspects of smart contracts.

**Persistence:** This shows a tactical adversary's objective to maintain its presence in the system. This goal can be achieved when a single attacker keeps creating Sybil nodes to dominate a majority of the network's hash rate and manipulate blockchain transactions to their advantage. This is known as a *majority attack* [211] and is mostly used to target public blockchains. The adversary might also target smart contracts and exploit access control vulnerabilities to change the contract owner and control every transaction invoked in the contract. *Parity multi-signature wallet attacks* [60] is a well-known type of attack that targets Ethereum smart contracts. This hack is an instance of exploiting a well-written library code once it is used in a non-intended context. The library's initialisation function could be externally called, which allowed the attacker to set himself as the owner of the contract. After taking over the contract, the attacker called the suicide method to kill the contract; this was once done, causing a permanent freeze of US\$280M in the affected wallets. Attacks belonging to this tactic can also target consensus mechanisms, cryptographic primitives, and multiple chains structure architectural dimensions.

**Privilege Escalation:** In this sort of attack, the adversary is attempting to obtain a higher level of permission in a blockchain system. Adversaries can exploit smart contract vulnerabilities to elevate their permissions and perform unauthorised actions. *Parity multi-signature wallet attacks* are one instance of an attack that falls under this category. Attacks belonging to this tactic can also target public access types, cryptographic primitives, and multiple chains structure. It is worth mentioning that this category of attack often overlaps with persistence attacks, as the exploited weaknesses that let an adversary persist can be exploited in an elevated context.

**Credential Access:** Attacks in this category aim to gain access to resources by exploiting the vulnerabilities within the system identification and authentication mechanisms to expose sensitive data or transactions and/or manipulate them. Such attacks include quantum attacks [390] that can derive private keys from public keys, brute-force attacks [152], *man in the browser (MITB)* [109] and *malware attacks* that can steal the keys and credentials of users' online wallets. Adversaries may attempt to hijack network traffic using *MITM* techniques to collect nodes' sensitive information. Employing legitimate credentials allows adversaries to gain access to the system and makes attackers' actions harder to detect. These types of attacks can also target private and consortium blockchains and centralised off-chain storage.

**Collection & Exfiltration:** We combined these two adversarial tactics into one Section, as adversaries often need to gather sensitive data by applying collection techniques before attempting to steal data via exfiltration techniques. Such attacks include *MITB*, where an adversary injects malware into a node's browser to collect sensitive data such as wallet credentials, which then allows the adversary to steal users' private keys. Additionally, an adversary can change the unique digital signature of a transaction before it is assigned to the chain, a process known as a *malleability attack* [152]. *DNS hijacking attacks* [299] aim to redirect users to malicious websites to collect seed phrases and private keys from users to allow the attacker to access users' wallets and steal their funds. According to [65], in 2021, two cryptocurrency portals faced this type of attack. Attacks under this category mostly tar-

get users' private keys to manipulate their transaction and steal their money. Additionally, attackers can target smart contracts and identify their weaknesses to steal the cryptocurrencies stored in them. Such attacks have included *DAO attacks* [152], *GovernMental attacks* [23], and *King of the Ether Throne attacks* [211]. In each of these attacks, attackers were able to drain millions from compromised contracts. Attacks belong to this tactic can also target access type and centralised off-chain storage.

**Impact:** The adversary's objective in this tactic is to damage, disrupt, or manipulate the blockchain system and its transactions. In particular, *DDoS attacks* [152] are a part of this category. The adversary uses legitimate operations to make connections, but then consumes resources to prevent other legitimate connections from requesting a particular service. Such an attack can be applied in a centralised off-chain storage system to prevent legitimate users from gaining access to stored records. Additionally, the attacker might use reconnaissance and collection techniques to hijack the connection between the external server and the blockchain system to tamper with the data. *Selfish mining attacks* [211], which target public blockchains, also fall under this category, where malicious miners collude to increase their benefits by causing honest miners to waste processing power creating blocks that will not eventually be linked to the chain. Meanwhile, the selfish miners can keep their mined blocks private, in an effort to maintain a private branch that is longer than the public branch. These selfish miners can then reveal their branch, and the honest miners will switch to it. As a result, the selfish miners win and are rewarded, while the honest miners lose and waste their power. Attacks classified in this category can be linked to all the architectural dimensions of the blockchain system. Noticeably, attacks in the impact category include only those affecting the integrity or availability of blockchain systems' information or transactions.

One insight from Table 3.3 is that a notable proportion of attacks target public blockchains and the consensus mechanisms related to them. If an architect decides to design a public blockchain-based system, they need to recognise adversaries' tactics and attack

techniques that often target this type of blockchain access. As the Table shows, public blockchains are prone to a significant number of attacks compared with private ones. Because of the characteristics of private blockchains, several attacks – including Sybil attacks, selfish mining attacks, and majority attacks – are difficult to launch and easy to prevent. Furthermore, a PoW consensus mechanism, which is often applied in public blockchains, is prone to the last two mentioned attacks, while PoS has been proposed to mitigate the risks of these attacks. However, since the records are securely distributed in public blockchain networks, they are more resilient against DDoS. Moreover, ransomware attacks [109] are difficult to achieve as locking down redundant records across the whole public network is a complicated task.

Another insight is that in blockchain, a user’s private keys are the most vulnerable point of attack. These keys can be compromised not only through exploiting the vulnerability in the digital signature cryptographic primitive, but also by attacking the wallets where these keys are stored. In particular, online wallets, which store the private keys in web servers, are prone to attacks that often target web applications. Moreover, several attacks that target smart contracts have been reported which have resulted in losses of millions of dollars including DAO attacks and Parity Multi-Sig Wallet attacks. To this end, architects can leverage Table 3.3 to recognise the adversarial tactics and attacks that target each architectural dimension when they decide to design blockchain-based systems.

### **Linking STRIDE Threats with Blockchain Systems Architectural Decisions**

Exploration of the relationship between blockchain architectural decisions and their consequent threats is presented in Table 3.4. The explanation of each STRIDE threat category in the context of blockchain systems is as follows.

**Spoofing:** This threat refers to the attempt on the part of an adversary to access

a blockchain system, or even control the network, by using a false identity. This can be done by stealing or retrieving the private keys and the credentials of an authorised node. Subsequently, the adversary can successfully access the victim's account or wallet to engage in illegitimate activities such as abusing transactions or violating the victim's privacy. Moreover, the adversary can create multiple fake accounts, Sybil nodes, to gain control of a consensus protocol as explained in the previous Subsection (Resource Development paragraph) and accomplish malicious behaviour such as double-spending. When the adversaries with their malicious nodes control a majority of the network, they can alter the entries on the distributed ledger to make the payments disappear after they have been spent [252].

**Tampering:** In this threat, an attacker attempts to accomplish unauthorised alterations to data, transactions, or blocks that are recorded in the storage or those that are being transferred through the network. Particularly, when an online wallet is used to perform transactions, the attacker attempts to hijack the session and modify the outgoing transaction to his benefit. Furthermore, attackers equipped with quantum computers will be able to apply the Shor algorithm [206], which can find the prime factorisation of large numbers and solve discrete logarithms in polynomial time. Consequently, the digital signature algorithms utilised in most current blockchain-based systems can be breached. This will allow attackers to easily derive private keys from public keys to alter transactions and sign them on behalf of the victim.

**Repudiation:** This is the ability of malicious participants or attackers to leverage the inability of the system or other participants to track the malicious actions or transactions that they have performed. The attacker can modify the blocks in the chain and the hashed meta-data stored on the chain once he can crack the hash function by finding hash collisions. Furthermore, a quantum adversary, who can apply a quantum algorithm such as Grover's search algorithm, can search for hash collisions significantly faster  $O(\sqrt{n})$  than in the case of a classic brute force attack  $O(n)$  [206]; in the future, an attacker will also be able to replace



blocks in the chain without affecting the integrity of the blockchain system.

**Information Disclosure:** This refers to exposing private information to individuals who are not permitted to have access to it. If the attacker successfully derives the private key from the user's public key, as explained in the Section dealing with a tampering threat, users will lose their privacy. In a public blockchain, if effective privacy-preserving mechanisms are not in place, the attacker can trace transactions and eventually link the user's pseudonym to the user's real identity [38].

**Denial of Service:** The aim of this threat is to make a system unavailable when legitimate users request a service. This can be accomplished by causing network congestion which interrupts the service available to the user. Even though a blockchain network presents resistance against this threat [152], blockchain-based systems are prone to this category of threat. This is because the node can be flooded with a large amount of junk data to exhaust its computational resources and prevent it from performing normal transactions. Additionally, blockchain systems might have a single point of failure component which is vulnerable to a denial of service threat. In particular, if smart contracts request data from a single external oracle and wait for its response to accomplish subsequent operations, an attacker can target this server by bombarding it with requests to prevent it from responding to legitimate smart contract requests.

**Elevation of Privileges:** This threat occurs when malicious users with restricted privileges succeed in gaining access to a system or a network to perform unauthorised activities. For example, this can be accomplished by an attacker who can trick a user of an online wallet to install a malicious payload into a high-privilege extension to gain access to the victim's wallet and alter the transactions. Another example is the usage of large block sizes that may cause chain forks [217], resulting from the increasing number of stale blocks [137], which leads to a significant mining power loss and limit the growth of the main chain. As

a consequence, this decision allows malicious miners to elevate their privilege level through (i) colluding to compromise and control the consensus mechanism; and (ii) performing malicious activities such as establishing their private chain to create conflicting transactions with higher chances of double spending threats.

Table 3.4 shows that threats exist in almost all architectural aspects of blockchain systems. In particular, private keys and their management face significant threats. One important insight is that the tampering category may potentially threaten all architectural aspects of blockchain systems. Taking sub-optimal choices when engineering blockchain system leads to threats that could potentially affect the other architectural dimensions of the system. Blockchain systems architects can use this Table to better understand the potential threats that each architectural decision might face. The information in this Table can therefore serve as a checklist for architects as they make or review design decisions.

### **Linking Attack Categories with STRIDE Threats**

In Table 3.5 we have linked attack categories and STRIDE threats. One insight from the relationships shown in this Table is that most categories of attacks can cause nearly all threat types. These attack categories include resource development, privilege escalation, credential access, collection & exfiltration, and impact. Moreover, some of these categories may pose a significant number of threats under each threat type such as the impact category. Another insight is that other attack categories such as reconnaissance pose a specific threat type as spoofing and information disclosure. This classification supports the identification of potential attacks that can exploit known vulnerabilities in blockchain system components and specifies the posed threats. Vulnerability identification approaches and tools can then be used to determine the specific flaws in the chosen system components. Particularly, there are a set of static and dynamic analysis tools that can identify security vulnerabilities in smart

Table 3.3: Linking Attack Categories with Blockchain Architectural Decisions

Attack Tactics	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Reconnaissance</b>	<i>Public:</i> Deanonymisation attack [38], Replay attack [299]; <i>Private:</i> MITM [7] [106]; <i>Consortium:</i> MITM [7] [106]				<i>Single-key:</i> Malware attacks [83], MITM [7] [106]; <i>Multi-Signature:</i> Malware attacks [83], Deanonymisation attack [38]; <i>Threshold-Signature:</i> Malware attacks [83]				Malware attacks [83], Replay attack [299]; <i>Solidity:</i> Solidity: Overflow/underflow attack [299]; <i>Hyperledge Platform:</i> Range query risks [50], Log injection attack [50]; <i>External oracle:</i> MITM [7] [106]
<b>Resource Development</b>	<i>Public:</i> Sybil attack [266], Majority attack [211], Time-jacking attack [299]		<i>PoW:</i> Majority attack [211], Selfish mining [211], Vector76 attack [152]; <i>BFT:</i> Consensus 34% Attack [74]; <i>PoS:</i> Majority attack [211], Long-range attacks [152], Short-range attacks [152]; <i>Vote-based:</i> Whitewashing [83], Hiding block attack [83]	Block withholding [299], Finney attack [7]			Block withholding [299], Selfish mining [211], Vector76 attack [152], Finney Attack [7]		<i>EOS Platform:</i> Roll back attack [314], Replay attack [314], RAMsomware attack [201]; <i>External oracle:</i> Oracle Manipulation Attack [284]

Continued on next page

Table 3.3 Linking Attack Categories with Blockchain Architectural Decisions (*Continued from previous page*)

Attack Tactics	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Persistence</b>	<i>Public:</i> Majority attack [211], Sybil attack [266]		<i>PoW:</i> Majority attack; <i>PoS:</i> Majority attack [211]; <i>BFT:</i> Consensus 34% Attack [74]; <i>Vote-based:</i> Whitewashing [83]			Majority attack [211]	<i>Multiple Chains (Shards):</i> Majority attack [211]		<i>Solidity</i> & <i>Vyper:</i> Parity multi-signature wallet attack [60]
<b>Privilege Escalation</b>	<i>Public:</i> Majority attack [211], Sybil attack [266]					Majority attack [211]	<i>Multiple Chains (Shards):</i> Majority attack [211]		<i>Solidity</i> & <i>Vyper:</i> Parity Multi-signature wallet attack [60], BECToken Attack [60]; <i>Solidity:</i> Overflow/ underflow attack [299]; <i>Hyper-ledge Platform:</i> Sandboxing attacks [50]; <i>GOLANG:</i> Docker TOCTOU Bug [50]

*Continued on next page*

Table 3.3 Linking Attack Categories with Blockchain Architectural Decisions (*Continued from previous page*)

Attack Tactics	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
Credential Access	<i>Private:</i> MITM [7] [106], MITB [109]; <i>Consortium:</i> MITM [7] [106], MITB [109]	<i>Centralised Off-chain:</i> MITM [7] [106]			<i>Single-key:</i> Brute Force attack [152], Malware attacks [83], Phishing [21] [93], MITB [109], MITM [7] [106], Replay attack [299]; <i>Multi-Signature:</i> Brute Force attack [152], Malware attacks [83]; <i>Threshold-Signature:</i> Brute Force attack [152], Malware attacks [83];	Brute force attack [152], Quantum attack [390]			

*Continued on next page*

Table 3.3 Linking Attack Categories with Blockchain Architectural Decisions (*Continued from previous page*)

Attack Tactics	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Collection &amp; Exfiltration</b>	<i>Public:</i> Deanonymisation attack [38]; <i>Private:</i> MITM [7] [106], MITB [109], Wormhole attacks [18]; <i>Consortium:</i> MITM [7] [106], MITB [109]	<i>Centralised Off-chain:</i> MITM [7] [106]			<i>Online-Wallet:</i> DNS hijacking [299], <i>Single-key:</i> MITB [109], MITM [7] [106], Crypto jacking [307], Malware attacks [83]; <i>Multi-Signature:</i> Malware attacks [83]; <i>Threshold-Signature:</i> Malware attacks [83];	Malleability attack [152]			<i>Solidity &amp; Vyper:</i> Short address attack [307], King of the Ether Throne [211], Dynamic libraries [211]; <i>Solidity:</i> DAO attack [152], Overflow/ underflow attack [299], GovernMental attack [23]; <i>EOS Platform Fake EOS Attack</i> [314], Random number attack [314]

*Continued on next page*

Table 3.3 Linking Attack Categories with Blockchain Architectural Decisions (*Continued from previous page*)

Attack Tactics	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Impact</b>	<i>Public:</i> Finney attack [7], Selfish mining [211], Eclipse attack [155], Routing attacks [64], Stealthier attack [335]; <i>Private:</i> Ransomware attacks [109] [277], Tampering [123], DDoS [152], DoS on endorsers [18]; <i>Consortium:</i> DDoS [152]	<i>Centralised Off-chain:</i> DDoS [152]; <i>Off-chain:</i> Brute force attack [152], Tampering [123]	<i>PoW:</i> Finney attack [7], Selfish mining [211], Eclipse attack [155], BGP hijacking [299] <i>PoS:</i> Nothing-at-stake [152]; <i>PBFT:</i> DDoS [152]	Consensus delay [152], Block withholding [299], Time-jacking attack [299]	<i>Online-Wallet:</i> Flooding attack [152], DNS hijacking [299]	Brute force attack [152], Quantum attack [390]	Finney attack [7], <i>Multiple chains:</i> DDoS [152]	Tampering, Block withholding [299]; <i>Light node:</i> DDoS [152]	<i>Solidity &amp; Vyper:</i> DoERS [208], HYIP Attack [60], ERC-20 Signature Replay Attack [60], Under-priced DDoS Attacks [60], BECToken Attack [60], Exploit Inconsistent behaviours of ERC-20 [63]; <i>Solidity:</i> Governmental Attacks [23]; <i>Hyperledge Platform:</i> Currency Attacks [381]; <i>GOLANG:</i> Key generation attack [381]; <i>EOS Platform:</i> Transaction Congestion Attack [314], Random number attack [314], DoS by draining EOS resources [201]; <i>External oracle:</i> DDoS [152], Tampering [123], Oracle Manipulation Attack [284]

Table 3.4: Linking STRIDE with Blockchain Architectural Decisions

STRIDE Threat	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Spoofing</b>	Public: Double spending [252] Private/ Consortium: Privacy violation [19], Untrusted identities [19], steal-front end login information [215] [112]	Centralised off-chain storage: Gain access to the storage [375]	Voting-based: Out-vote by fake accounts [152]. PoS: Generating different blockchains with old accounts [152].		Local wallet and On-line wallet: Buggy software installation [194], Compromised private key [7], Packet spoofing [174].	Transaction spoofing [386], Recovering of the private key [211], Compromised private key [7]			Change contracts owner [162].

*Continued on next page*



Table 3.4 Linking STRIDE with Blockchain Architectural Decisions (Continued from previous page)

STRIDE Threat	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Tampering</b>	Private: Transaction manipulation [305]	On-chain: modify the hashed metadata [375]. Off-chain Computation: Transaction manipulation [305]	PoW: Control transaction's confirmation [211], Acquire dominance in the pools [152], Block modification [305].	Control transaction's confirmation [211].	Impersonation at future transaction [305], Transaction manipulation. Online-Wallet: Alter out-going data [109], Alter transaction history.	Digital-signature: Shor's quantum algorithm [129], Transaction malleability, Impersonation at future transaction [305], Transaction manipulation [305]. Hash: Grover's search algorithm [206], Transaction manipulation [305], Block modification [305]	Control the confirmation operation [211], Control transaction's confirmation. Off-chain Channel: Transaction manipulation [305]	Light node: Fake header [72]	Malicious external oracle [376], Change contracts owner [162], External oracle: Unfair income [24], Critical unwanted behaviors [61], Money frozen [284]

Continued on next page

Table 3.4 Linking STRIDE with Blockchain Architectural Decisions (Continued from previous page)

STRIDE Threat	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Repudiation</b>	Private: Transaction manipulation [305]		PBFT: Untrustworthy nodes [375]. PoW: Control the confirmation operation [211], Acquire dominance in the pool, Block modification [305].	Control the confirmation operation [211]	Impersonation at future transaction [305].	Hash: Grover's search algorithm [206], Transactions manipulation, Block modification [305]. Digital-signature: Shor's quantum algorithm [129].			Malicious external oracle [376], Untrustworthy data feeds [23], Unfair income [24].
<b>Information Disclosure</b>	Public: Sensitive data exposure [19], Privacy violation [19], Private/ Consortium: Steal-front end login information [215], Eavesdropping [7]; Private: leakage of confidential information [18].	On-chain: Sensitive data exposure, Privacy violation [19].			Single-key: Eavesdropping [7], Data exposure, Privacy violation; Multi-sig: Eavesdropping [7], Privacy violation; [19], On-line wallet: Bypass credential validation [52], Compromised key [135].	Compromised key, Transaction pattern exposure [122], Transaction graph analysis [122].	Single chain: Sensitive data exposure [19], Privacy violation [19]	Light node: Privacy violation [19], Untractability violation [19].	Information leakage.

Continued on next page

Table 3.4 Linking STRIDE with Blockchain Architectural Decisions (Continued from previous page)

STRIDE Threat	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Denial of Service</b>	Private/ Consortium: Exhausting computational resources [211], Nodes flooding/isolation [152], Massive transaction backlogs [152].	Centralised Off-chain: Compromise the availability [376], Data loss [375].	PoW: Exhausting computational resources [211], nodes flooding/isolation, massive transaction backlogs.	Increase Block Size: Network congestion [68], Decrease transaction throughput [137].	Single-Key: Compromise the availability, Online-Wallet: Server flooding [152].		Multiple chains: Compromise the availability.	Light node: Compromise the availability.	resource-consuming procedure [208], Untrustworthy external calls [152], Untrustworthy data feeds [23], Disturbing external oracle [23], Compromise the availability, temporary shutdown of token trading [60]

Continued on next page

Table 3.4 Linking STRIDE with Blockchain Architectural Decisions (Continued from previous page)

STRIDE Threat	Blockchain Architectural Dimensions								
	Access Type	Storage and Computation	Consensus Mechanisms	Block Configuration	Key Management	Cryptographic Primitives	Chain Structure	Node Architecture	Smart Contracts
<b>Elevation of Privileges</b>	Public: Splitting mining power [227], Engineering block races, Modifying transactions, Create blockchain forks [217], Race conditions by forking [298]. Private-Consortium: Untrusted identity.	Centralised Off-chain Storage: Gain access to the storage.	PoW: Double-spending, Modifying transactions, Control the confirmation operation [211]. PoS: Double-spending, Modifying transactions, Control the confirmation operation [211]. DPoS: Double-spending, Collude threats [211]	Double-spending, Blockchain forks [217], Conflicting, Stale Block [137].	Crafting malicious Payload into high privilege extension [194], Bypass credential validation [52]; Online-wallet: Crafting malicious Payload into high privilege extension [194], Bypass credential validation, Local-Wallet: Bypass credential validation.	Digital-signature: Shor's quantum algorithm [129]. Hash: Grover's search algorithm [206], Double spending [252].	Double spending [252], Blockchain forks [217].		Destroyable contract [49], Change contract owner [162], Stolen tokens [60].

Table 3.5: Linking Attacks with Threats

Attack Tactics	STRIDE Threats					
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
<b>Reconnaissance</b>	Privacy violation [19], Transaction spoofing [386]			Privacy violation [19], Information leakage, Sensitive data exposure [19], Transaction graph analysis [122], employment analysis, Eavesdropping [7], Untractability violation [19], Eavesdropping [7], Transaction pattern exposure [122]		
<b>Resource Development</b>	Out-vote by fake accounts [152], generating different blockchains with old accounts [152], Change contracts owner [162], Untrusted identities [19], Double spending [252]	Control the confirmation operation [211], Acquire dominance in the pools, Unfair income [24], Change contract owner, Alter transaction history	Control the confirmation operation, acquire dominance in the pool, Untrusted identity, Unfair income		Exhausting computational resources [211], Nodes flooding/ isolation, Data loss [375].	Double-spending, Splitting mining power [227], Engineering block races, Untrusted identity, Create blockchain fork, Race conditions by forking [298], Control the confirmation operation [211], Collude threats [211], Conflicting, Stale Block [137], Change contract owner

*Continued on next page*

Table 3.5 Linking Attacks with Threats (Continued from previous page)

Attack Tactics	STRIDE Threats					
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
<b>Persistence</b>	Double spending [252], Out-vote by fake accounts [152], Generating different blockchains with old accounts [152], Untrusted identities [19]	Impersonation at future transaction [305], Block modification [305], Control the Confirmation Operation [211], Acquire dominance in the pools, Fake header, Unfair income [24]	Block modification [305], Control the Confirmation Operation, Acquire dominance in the pools, Untrusted identities [19], Unfair income [24]			Block modification [305], Double-spending, Untrusted identity, Control the confirmation operation [211]
<b>Privilege Escalation</b>	Double spending, Generating different blockchains with old accounts [152], Change contracts owner [162], Untrusted identities [19], Out-vote by fake accounts [152], Privacy violation [19]	Control the Confirmation Operation, Acquire dominance in the pools, Unfair income [24], Change contracts owner [162]	Control the Confirmation Operation [211], Acquire dominance in the pools, Untrusted identities [19], Unfair income [24]	Privacy violation [19], Sensitive data exposure [19]		Double-spending, Untrusted identity, Control the confirmation operation [211], Change contract owner, Critical unwanted behaviors [61], Stolen tokens [60]
<b>Credential Access</b>	Privacy violation [19], steal-front end login information [215], Gain access to the storage [375], Buggy software installation [194], Compromised private key [7], Transaction spoofing [386], Recovering of the private key [211]	Transactions Manipulation, Shor's quantum algorithm [129], Transaction malleability, Alter out-going data [109], Grover's search algorithm [206]	Transaction manipulation [305], Shor's quantum algorithm, Grover's search algorithm [206]	Privacy violation [19], Sensitive data exposure [19], Eavesdropping [7], Steal-front end login information [215], Bypass credential validation [52], Compromised private key [7]		Transaction manipulation [305], Shor's quantum algorithm [129], Grover's search algorithm [206]

Continued on next page

Table 3.5 Linking Attacks with Threats (Continued from previous page)

Attack Tactics	STRIDE Threats					
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
<b>Collection &amp; Exfiltration</b>	Privacy violation [19], Change contracts owner [162], steal-front end login information [215], Compromised private key [7], Transaction spoofing [386], Identity theft, Gain access to the storage [375]	Transaction manipulation [305], Alter out-going data [109], Fake transaction, Identity theft, Change contract owner, Malicious external oracle [376]	Transaction manipulation [305], Malicious external oracle [376]	Privacy violation [19], Transaction graph analysis [122], Sensitive data exposure [19], Eavesdropping [7], Steal-front end login information [215], Bypass credential validation [52], Compromised private key [7], Untractability violation [19], Transaction pattern exposure [122], leakage of confidential information [18]		Transaction manipulation [305], Change contract owner, Gain access to the storage [375], Bypass credential validation [52]

Continued on next page

Table 3.5 Linking Attacks with Threats (Continued from previous page)

Attack Tactics	STRIDE Threats					
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
<b>Impact</b>	Double spending [252], Recovering of the private key [211]	Control the confirmation operation [211], Acquire dominance in the pools, Unfair income [24], Modify the hashed met-data [375], Shor's quantum algorithm [129], Grover's search algorithm [206], Untrustworthy data feeds [23], Malicious external oracle [376], Money frozen [284]	Control the confirmation operation [211], Acquire dominance in the pools, Unfair income [24], Grover's search algorithm [206], Shor's quantum algorithm [129], Malicious external oracle [376], Untrustworthy data feeds [23]	Compromised private key [7]	Compromise the availability [376], Exhausting computational resources [211], Nodes flooding/isolation [152], Massive transaction backlogs [152], Data loss [375], Compromise the availability [376], Network congestion [68], Disturbing external oracle [23], Data loss [375], Decrease transaction throughput [137], Server flooding [152], Blockchain ingestion, Untrustworthy data feeds [23]	Double spending [252], Splitting mining power [227], Engineering block races, Create blockchain forks [217], Race conditions by forking [298], Control the confirmation operation [211], Collude threats [211], Stale Block [137], Destroyable contract [49], Gain access to the storage [375], Shor's quantum algorithm [129], Grover's search algorithm [206], Acquire dominance in the pools, Critical unwanted behaviors [61].



contract components [294, 218]. The information on the identified vulnerabilities allows a determination of the attack patterns that might exploit them. Our mapping approach assists an analyst in identifying the attacks and their corresponding threats in each architectural dimension of blockchain systems.

The mapping approach also supports the prevention of zero-day attacks, as recognising current attacks provides insights into the behaviour of malicious actors, their motivations, and their preferred attack vectors. Security systems can then be configured to analyse deviations from normal user or system behaviour. This proactive behavioural analysis can raise flags for potential zero-day attacks, as attackers might employ new methods to achieve their goals. Moreover, current attacks often target common vulnerabilities. Educating blockchain software engineers about these ongoing threats helps them recognise suspicious activities, avoid risky behaviour, and report potential incidents promptly. This heightened engineer's awareness can serve as an additional layer of defence against zero-day attacks that might exploit familiar attack vectors.

### **3.5 Application of the taxonomy and mapping approach**

In this Section, we provide a three-step decision-making process to be applied to each architectural dimension illustrated in the taxonomy:

1. Determine the quality attributes that are provided and not provided by each design alternative (using Table 3.2) to recognise the quality trade-offs and select among the alternatives with quality rationale in mind.
2. Identify attack tactics and techniques (using Table 3.3) that might compromise each design alternative to understand the adversarial objectives and potential methods for

attacking the system.

3. Identify the potential threats to each design alternative (using Table 3.4) and the specific threats posed by each attack (using Table 3.5).

Systematically applying these steps helps in evaluating and balancing trade-offs to find the optimal solution that best aligns with project goals. Additionally, it supports security engineers in establishing a risk management approach and effectively prioritising and mitigating the highest security risks that threaten blockchain systems by implementing effective countermeasures. Although some attacks and threats might be missed in the provided tables or new attacks might emerge, the provided classification and mapping are general enough to be extended and to categorise many types of attacks. Moreover, our approach can help in predicting the behaviour of zero-day attacks, as recognising current attacks provides insights into the behaviour of malicious actors, their motivations, and their preferred attack vectors, as explained earlier in Subsection 3.4.1.

### **3.5.1 Key management as a case to demonstrate instantiating the taxonomy and its mapping to attacks and threats**

In this Section, a concrete example of an existing blockchain-based system is leveraged to show how the proposed taxonomy and guidelines can be applied in practice. This blockchain system manages and shares electronic medical record (EMR) data for cancer patient care. The framework was proposed by Dubovitskaya et al. [99] in collaboration with the Stony Brook University Hospital. They utilise blockchain technology to maintain immutable and verifiable records that keep track of all actions across the network. This helps to improve the integrity of sensitive medical data, reducing the time needed to share EMRs as well as the overall cost. Figure 3.4 shows the EMR system’s architectural decisions that we were

Table 3.6: Analysis of Alternative Decisions for EMR System

Analysis of Alternative Decisions		
Quality Attributes	<pre> graph TD     KM[Key Management] --&gt; D{+}     D --&gt; MS[Multi-Signature]     D --&gt; SK[Single-Key]     D --&gt; TS[Threshold-Signature]     </pre> <p>Key Management</p> <p>Multi-Signature: + Integrity, + Non-repudiation, - Latency, - Confidentiality</p> <p>Single-Key: + Accessibility, + Latency, - Availability, - Integrity, - Non-repudiation</p> <p>Threshold-Signature: - Latency, + Integrity, + Non-repudiation, + Confidentiality</p>	
Alternatives	Potential Attacks	Potential Threats
Single Key	<p><b>Reconnaissance:</b> Malware attacks, MITM.</p> <p><b>Credential Access:</b> Brute Force, MITM, MITB, Malware attacks, Replay attacks, phishing.</p> <p><b>Collection and Exfiltration:</b> MITM, MITB, Malware attacks.</p> <p><b>Impact:</b> Flooding attack.</p>	<p><b>Tampering:</b> Impersonation at future transaction, Transaction Manipulation.</p> <p><b>Repudiation:</b> Impersonation at future transaction.</p> <p><b>Information Disclosure:</b> Privacy violation.</p> <p><b>Denial of Service:</b> Compromise the availability of the key, server flooding.</p> <p><b>Elevation of Privileges:</b> Crafting malicious payload into high privilege extension, Bypass credential validation.</p>
Multi-Signature	<p><b>Reconnaissance:</b> Malware attacks, Deanonimisation attacks.</p> <p><b>Credential Access:</b> Brute Force, Malware attacks.</p> <p><b>Collection and Exfiltration:</b> Malware attacks.</p>	<p><b>Tampering:</b> Impersonation at future transaction, Transaction Manipulation.</p> <p><b>Repudiation:</b> Impersonation at future transaction.</p> <p><b>Information Disclosure:</b> Privacy violation.</p> <p><b>Elevation of Privileges:</b> Crafting malicious payload into high privilege extension, Bypass credential validation.</p>
Threshold Signature	<p><b>Reconnaissance:</b> Malware attacks,</p> <p><b>Credential Access:</b> Brute Force, Malware attacks.</p> <p><b>Collection and Exfiltration:</b> Malware attacks.</p>	<p><b>Tampering:</b> Impersonation at future transaction, Transaction Manipulation.</p> <p><b>Repudiation:</b> Impersonation at future transaction.</p> <p><b>Elevation of Privileges:</b> Crafting malicious payload into high privilege extension, Bypass credential validation.</p>

able to extract or infer from their paper. The identified decisions are presented based on our taxonomy.

The EMR blockchain system is a permissioned consortium access type as patients' data might be transferred to several hospitals. This access type safeguards the privacy of highly sensitive data about patients. A fast response time is essential in medical systems, which can be provided easily by this access type. Moreover, in a permissioned consortium blockchain, there is no need to pay for the execution of a transaction, and this increases the usability of the system.

Patient metadata is stored on-chain, while two off-chain storage locations are used to store patients' raw data: an in-hospital database that stores oncology-related data and a cloud storage database that organises patients' data and encrypts the saved data with a symmetric key for each patient. A doctor can access data in the cloud according to the permission policy specified by the patient. These two off-chain storage sites reintroduce centralisation into the blockchain system and can function as a single point of failure. The EMR blockchain system applies a PBFT algorithm, which is a vote-based consensus mechanism. This is because all users in a medical application are known (patients and doctors), and only a predefined set of nodes can participate in the consensus mechanism. This type of mechanism protects against Sybil and Majority attacks.

This medical application was built on top of the Hyperledger platform, which implements smart contracts in the form of chaincode, comprising logic and correlated state components. Chaincode is written in the Go programming language. In Hyperledger, the size of the block is often 98 MB, and the block confirmation takes one-second [172]. SHA-256 is used as the default hash in the Hyperledger platform, and the EdDSA scheme is used for digital signatures. A symmetric algorithm is used to encrypt clinical data stored in the cloud repository to provide data confidentiality.

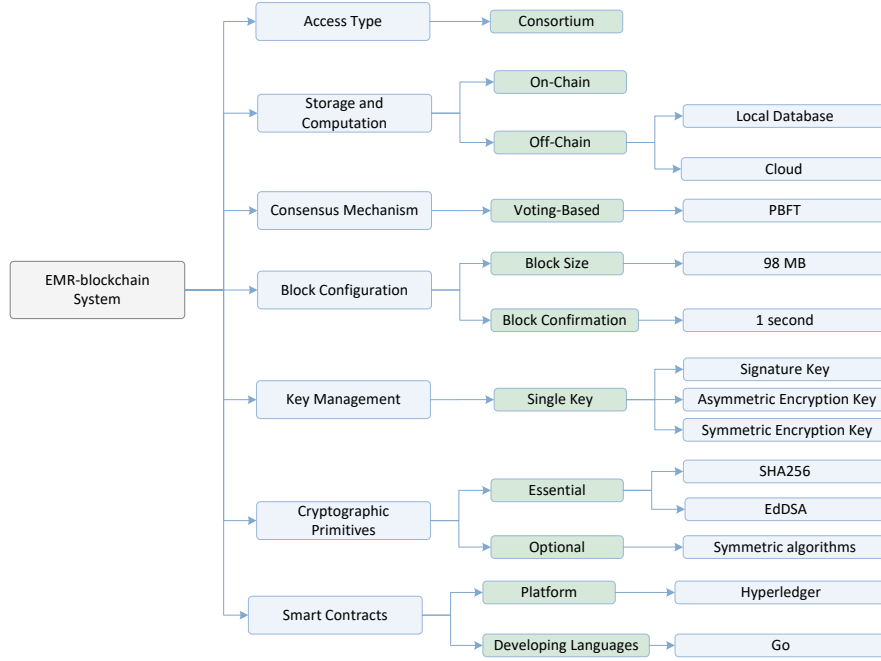


Figure 3.4: Architectural Decisions for EMR System

Each doctor has a public key  $pk_U^S$  and a private key  $sk_U^S$  for signing, as well as  $pk_U^\epsilon$  and  $sk_U^\epsilon$  for encryption. The patient can generate a metadata record on the chaincode, retrieve it, and specify permissions. In addition to the two key pairs, the patient also has an asymmetric encryption key  $SK^{AES}$  that encrypts and decrypts patient data. If a patient needs to enable a doctor to access his data, he should encrypt the  $SK^{AES}$  with the encryption public key of the doctor  $pk_D^\epsilon$ , and then share the encrypted value with the doctor. Since each user has a single key pair for signing and another single key pair for encryption, these keys become a single point of failure. If attackers compromise them, they could sign and encrypt data themselves. Moreover, if passive attackers compromise the patient's symmetric key, they could decrypt and observe data.

We aim to investigate alternative architectural choices regarding key management dimensions as the current choice is sub-optimal, and this might affect the security of the system. We apply our proposed three-step process to make secure and informative key

management choices, as shown in Table 3.6 and describe below.

**Step 1: Determine the quality attributes.** We used Business Process Model and Notation (BPMN) to represent the quality attributes provided and not provided by each key management choice. The single key option provides higher accessibility and reduced system latency, but it reduces system integrity, availability, and non-repudiation. The digital signature option provides higher integrity and non-repudiation but decreases system confidentiality and leads to higher latency. Although a threshold-signature may increase system latency, it increases system confidentiality, integrity, and non-repudiation.

**Step 2: Identify attack tactics and techniques.** We identify the tactics of potential attacks that each alternative key management design choice might be vulnerable to. Noticeably, the single key option is prone to larger sets of attacks than the other two options. Threshold signatures provide better mitigation against deanonymisation attacks than multi-signatures as the signed transactions are indistinguishable from non-threshold transactions on the blockchain.

**Step 3: Identify the potential threats.** We identify the potential security threats for each key management design option. The single key choice is prone to five threat categories and is the only choice that is threatened by denial of service. If this key is stored in an online wallet, the server might receive a massive number of requests with the aim of exhausting its resources and compromising the availability of the key. Threshold signature scheme is prone to the least number of threats compared with the other two choices. Applying this scheme protects users' privacy when compared to multi-signature scheme where the adversary can link users' identities by tracing their multiple keys.

Based on the three-step analysis, it appears that a threshold signature would enhance the security and privacy of the EMR system. Nevertheless, security architects need to conduct a risk assessment to quantify the potential risk exposures and thus make an informed

decision.

This brief example has illustrated that our approach provides a systematic way of assisting software engineers who are attempting to build a blockchain system. It also aids engineers who need to analyse and improve the security of an existing system. We have used key management as an example to demonstrate the instantiation of our taxonomy and its mapping to both threats and attacks. The same steps can be applied to analysing the attack tactics and the implied threats on any blockchain based technique. For the case of the blockchain-based EMR system, as an example, the security of other architectural choices as identified in Figure 3.4 can be analysed in the same way as that of key management. Our work provides systematic guidance for security engineers and architects, where the specific analysis techniques (e.g., prediction, estimation, etc.) for each step may vary based on the context and may be influenced by the availability of expertise in using these techniques.

## 3.6 Discussion

Here we discuss how we validate our work and the threats to validity to our approach.

### 3.6.1 Validation

A taxonomy can be validated in three ways to ensure its reliability and usefulness [342]: (i) orthogonality demonstration; (ii) benchmarking; and (iii) utility demonstration.

### **Orthogonality Demonstration**

Shows the dimensions and the categories of the taxonomy as Figure 3.2 and Section 3.3 demonstrated. We performed an iterative content analysis method to identify the dimensions of the proposed taxonomy. We continuously evolved our taxonomy whenever a new concept was encountered in the literature. We strove to ensure the generality of each dimension by noting when several terms appeared in the literature referring to the same concern (e.g. consensus protocol vs. consensus mechanism). Moreover, a new class is introduced only when clear-cut evidence of its relevance and significance justify its inclusion. As an example, we included consensus algorithm as a class because it has unique characteristics, properties, and well-defined terms as used in the literature. However, when there was an uncleared agreement on the term (e.g. node architecture), we came with a general class to encompass concerns and properties; nevertheless, some of the sub concerns can be further refined.

### **Benchmarking**

Compares the taxonomy to related classification schemes. Only two prior works have provided taxonomies of architectural aspects of blockchain [376, 301]. However, our taxonomy is novel as each dimension of our taxonomy has been discussed from a security perspective and mapped with the potential attacks and associated threats. Additionally, the taxonomy presented by Xu et al. [376] focuses only on six architectural components of blockchain systems, whereas the taxonomy presented here captures nine major dimensions for blockchain architectural decisions, and the discussions are explicitly focused on security. Different from Xu et al.'s taxonomy, three main architecture design dimensions – key management, cryptographic primitives, and node architecture – have been considered and thoroughly discussed in our study. Moreover, unlike theirs, our taxonomy has been derived by conducting a systematic literature review of studies related to architectural design decisions relevant to blockchain



systems. Salah et al. [301] presented a taxonomy that only targeted blockchain-based artificial intelligence applications; the architectural coverage of the paper was limited to the intelligence component. Conversely, the taxonomy proposed in this work is adequate for designing blockchain-based systems in general.

### Utility Demonstration

Is a mechanism to validate the benefits that could be gained from the taxonomy. There are several ways to demonstrate the benefits of the taxonomy as [342] stated, including expert opinion and instantiation. Our taxonomy has been reviewed by an expert who gave us substantial suggestions for refinements. Additionally, the instantiation of our taxonomy has presented in Subsection 3.5.1.

### 3.6.2 Threats to Validity

Based on [362], four potential threats to validity may affect our findings:

**Internal Validity.** One threat comes from the inherent nature of taxonomies: we cannot guarantee the completeness of our taxonomy since there may be additional architectural design decisions that could enrich or refine the taxonomy. To mitigate this threat, we iteratively refined our taxonomy each time a new concept was encountered in the literature. Furthermore, our taxonomy is adaptable and flexible to evolve and cope with new additions and changes. Another threat comes from the possibility of considering alternative methods for threat and attack categorisation. However, the Microsoft STRIDE threat model and MITRE's attack tactics categorisation were used in our study since they are widely used, they are consistent with current practice, and they ensure extensive coverage of potential threats and attacks.

**Construct validity.** Another threat arises because our taxonomy was mainly based on the

results of surveying the literature. We believe that additional sources of information could improve the completeness of this taxonomy. We mitigated this by searching the findings on multiple data sources. Additionally, our automated search was complemented by a manual search. Another threat arises from the provided set of attacks. Even though we have illustrated various kinds of attacks that pose threats to blockchain-based systems, it is impossible to cover all attacks because new attack types are always emerging. Therefore, the reader should be aware that the provided set of attacks is continually evolving as it is difficult to predict the state of the attackers. In this study, all presented attacks are informed not only by research papers but also by technical reports, developers blogs, and wiki pages. In any case, our proposed technique and methods for classification can be applied to categorise new and emerging attacks.

**Conclusion Validity.** There is a threat regarding the possibility that we interpreted the extracted data differently. The potential for bias introduced during the data extraction process was at least partially mitigated by ensuring a common understanding by all reviewers. We also ensured that the data extraction process was aligned with the research question.

**External Validity.** A final threat is related to the need to instantiate the taxonomy in different contexts, to assist in its refinement and validation. We demonstrated the utility of the taxonomy using an example of a blockchain-based health care system. We demonstrated the applicability of our guidelines by analysing the key management architectural dimension to enhance the security of such a system. Nevertheless, further validation of the utility of the taxonomy is needed to address this threat.

## 3.7 Related Work

A wide range of prior literature has discussed the properties, characteristics, and structure of blockchains. Some of them have focused on architectural components of a specific blockchain

Table 3.7: Summary of Related Work

Categorisation	Ref No	Year	Contribution	Focusing Area
<b>Blockchain Architecture</b>	[379]	2016	Explored blockchain from an architecture point of view. Compared blockchain with other software solutions	General Blockchain-Based System
	[376]	2017	Proposed taxonomy of some of the architectural components of blockchain systems. Showed how different architectural decisions affect the quality attributes of blockchain systems	General Blockchain-Based System
	[403]	2017	Reviewed the properties of blockchain systems. Mainly investigated, compared and analysed different consensus mechanisms	General Blockchain-Based System
	[130]	2018	Illustrated the blockchain frameworks. Reviewed some blockchain applications in details	General Blockchain-Based System
	[301]	2019	Provided a detailed survey on blockchain, platforms, consensus protocols and applications which are adequate for AI area	Artificial Intelligence
	[223]	2019	Reviewed some of the blockchain components. Provided a comprehensive discussion of the main properties of the state-of-the-art blockchain applications	Blockchain Application
<b>Security and Privacy Issues</b>	[211]	2017	Illustrated security risks and attacks over blockchains systems. Demonstrated several solutions protocols	General Blockchain-Based Systems
	[152]	2019	Extensively investigated vulnerabilities in each blockchain generation. Explained potential attacks. Highlighted possible countermeasures	General Blockchain-Based Systems
	[399]	2019	Explained the required security properties of blockchain-based cryptocurrency. Reviewed the existing techniques to achieve security and privacy for such systems	General Blockchain-Based Systems

*Continued on next page*

Table 3.7 Summary of Related Work (*Continued from previous page*)

<b>Categorisation</b>	<b>Ref No</b>	<b>Year</b>	<b>Contribution</b>	<b>Focusing Area</b>
	[354]	2019	Reviewed the security aspects and cyberattacks of each layer of blockchain-based systems. Summarised the existing mitigation techniques	General Blockchain-Based Systems
	[299]	2020	Investigated attack surface in multiple implementations of blockchains. Outline multiple defence techniques.	General Blockchain-Based Systems
	[68]	2018	Reviewed vulnerabilities in Bitcoin and related threats. Investigated the effectiveness of the proposed solutions. Reviewed privacy threats to Bitcoin's users. Analysed the existing privacy-preserving solutions	Bitcoin
	[123]	2018	Classified the potential attacks against blockchain-based IoT applications. Provided mechanisms to enhance their security and privacy	Internet-of-Things
	[11]	2019	Overviewed the main blockchain architecture components and characteristics from IoT perspective. Discussed how blockchain properties enhance IoT security	Internet-of-Things
	[40]	2017	Provided a taxonomy of the major security vulnerabilities in Ethereum smart contracts. Illustrated the significant attacks in Ethereum smart contracts	Smart Contracts
	[239]	2018	Provided a comprehensive classification of known security vulnerabilities in Ethereum smart contracts	Smart Contracts
	[229]	2018	Reviewed security and privacy issues related to smart contracts applications	Smart Contracts
	[60]	2020	Provided a comprehensive list of known security vulnerabilities, attacks, and defences in Ethereum smart contracts.	Smart Contracts

application, such as blockchain-based IoT [123], while others illustrated and discussed the security and privacy issues of blockchain technology [354]. To the best of our knowledge, no previous studies have classified the architectural design decisions of a blockchain-based system based on a systematic survey and then mapped them to threats and attacks. As shown in Table 3.7, we have classified the prior literature on blockchain into two major categories: blockchain architecture and security and privacy issues.

Regarding the first category, authors in [379] discussed several blockchain architectural design choices and compared decentralised blockchain with other software solutions. In [376], the authors presented a taxonomy of the architectural properties of blockchain-based systems and showed the impact of these properties on the performance and other quality attributes of the system. Yet, their impact on the security properties and their consequential security risks has not been covered. Moreover, a systematic review of the literature has not been conducted to represent the taxonomy. Similarly, in [403], the authors briefly discussed the types of blockchains and then discussed and compared different types of consensus protocols. Also, they provided a classification of different types of blockchain applications. A detailed dissection of blockchain applications and their properties, architecture, and issues was presented in [130]. Another review of blockchain applications was conducted by [223]. This study reviewed blockchain and its key components and comprehensively detailed application examples of blockchain-based IoT, security and data management. In [301], a survey on blockchain technology for Artificial Intelligence (AI) was conducted. This study provided a taxonomy of blockchain characteristics that can be leveraged by AI applications. It summarised existing blockchain platforms and protocols that could be adopted for AI applications.

Moving to the second category, security and privacy issues, researchers investigated this area and emphasised that blockchain is not completely secure and is prone to various vulnerabilities and security risks. One pioneering article in this category was [211], where

the authors reviewed the security threats to blockchain and the corresponding attacks and suggested some security solutions. Similarly, the authors in [152], classified security vulnerabilities based on blockchain accessibility. It also provided a detailed explanation of known vulnerabilities and subsequent potential attacks. Countermeasure techniques to improve the security of blockchain systems were also surveyed in this study. Investigation of security issues of blockchain was reviewed from other perspectives in [354], which analysed the security issues of each layer of blockchain: application, smart contracts, incentive, consensus, network and data layer. Another study [399] targeted security and privacy issues raised in blockchain-based cryptocurrency applications, highlighted security and privacy properties required in many blockchain applications and then reviewed the techniques and mechanisms to achieve them. In [299], authors investigated the attack surface in multiple implementations of blockchains in terms of cryptographic constructions, distributed system architecture, and applications. They also summarised defence measures carried out by blockchain technologies or recommended by researchers, to mitigate the impacts of these attacks. Several attacks that target Bitcoin and its underlying protocols, such as Proof-of-Work (PoW), were tabulated and analysed in [68]. Their survey also investigated the effectiveness of existing solutions. Moreover, it discussed privacy related threats and the privacy perceiving techniques against them.

A considerable number of surveys have reviewed the incorporation of blockchain in IoT systems. Three studies, [123, 11, 153], are related to our work, they reviewed architectural components and implementation of blockchain based-IoT and the related security issues regarding such integration. In [123], a detailed overview of blockchain protocols for IoT applications was provided. The authors classified and discussed the potential attacks against IoT applications implemented on a blockchain foundation. Likewise, a comprehensive survey regarding developing blockchain-based platforms, applications and services, which were adequate for IoT applications, was carried out in [11].

Since the emergence of smart contracts in blockchains the number of vulnerabilities and potential threats caused by wrong design and coding of smart contracts has increased. One review [40] aggregated the known vulnerabilities in Ethereum smart contracts and classified them into three categories based on the level: solidity level, Ethereum Virtual Machine (EVM) level, and blockchain level. Similarly, authors in [239] provided the same classification of vulnerabilities in Ethereum smart contracts; however, they provided a more comprehensive list. Authors in [60] utilised the same classification to classify 40 vulnerabilities and analysed their root causes. This study also discussed attacks and multiple defense techniques. Another study [229] provided an overview of the privacy and security issues that can arise in different smart contract applications. Noticeably, all presented blockchain surveys have discussed its architecture and security concerns from different perspectives. However, no one provided an in-depth survey of blockchain architectural design decisions and linked them to classified threats and potential attacks which can breach the system if ill-informed design decisions are taken.

### 3.8 Finding and Summary

**Methodological Limitations.** One of the major findings that require more investigation is a lack of academic studies with regard to conducting systematic approaches and the use of decision models to assist decision-makers and architects in choosing appropriate components, patterns, and features when constructing blockchain systems. Moreover, analysing the security risks encountered during blockchain systems' development, and how they influence the outcomes, has not been investigated in the literature. The programming flexibility of smart contracts opens opportunities for attackers to compromise them. Thus, there is a crucial need for best practices, standards, and frameworks to assess the security risks in smart contracts, as attacks against such contracts are unavoidable. In this thesis, we make a

contribution to a debt-aware approach that assesses security architectural design issues that manifest as a result of making ill-informed design decisions in the creation of smart contracts (in Chapter 4), as well as security issues that are shipped to the contract from a third party (in Chapter 5).

### 3.8.1 Summary

In this chapter, we surveyed architectural properties and aspects of blockchain-based systems and provided a taxonomy that captures their major architectural design decisions. The taxonomy illustrates nine dimensions of architectural decisions related to access type, data storage and transaction computation, consensus mechanism, block configuration, key management, cryptographic primitive, chain structure, node architecture, and smart contracts. We provided a mapping that links attacks and the posed threats to the architectural decisions in our taxonomy. We systematically classified the attacks in blockchain systems following MITRE's attack tactics categories and then associated the attacks with their posed threats using the STRIDE threat model.



## Chapter Four

# A Novel Approach for Assessing Smart Contracts' Security Technical Debts

In Chapter 2, we found that there is a lack of security-based design approaches that focus on the architectural design elements of smart contracts and assist in identifying, analysing and quantifying the associated security risks. A smart contract is one of the architectural elements of blockchain-based systems that is prone to multiple attacks, as demonstrated in Chapter 3. This chapter bridges an identified gap by proposing a debt-aware approach for assessing security architectural design issues in smart contracts at the early design stage. The results show that our assessment approach increases the visibility of security design issues. It also allows developers to concentrate on resolving these issues through technical debt impact analysis and prioritisation.

Most of the content in this chapter is based on our paper [5] that was published in the 2021 IEEE/ACM International Conference on Technical Debt.

## 4.1 Overview

A smart contract is a software deployed on a blockchain-based development platform to support, verify and impose a decentralised negotiation and implementation of digital agreements between un-trusted parties [356]. Despite the infancy of smart contracts, their emergence has disrupted several sectors, including cryptocurrencies, financial services, insurance, healthcare, and decentralised management, among others [132]. The widespread usage of this emerging technology has incentivised attackers to exploit its existing security and privacy challenges. Various malicious attacks have been accomplished due to deploying poorly designed or vulnerable smart contracts [60, 338].

Uploading smart contracts to the public blockchain is not free as a specific amount of money is required to be paid for each deployment process [392]. Moreover, different from traditional distributed software that can be patched when vulnerabilities are discovered, smart contracts are irreversible and immutable [225]. These properties mean that once a smart contract is deployed in the chain, it cannot be subsequently modified. Therefore, timely identification of vulnerability is crucial to saving business value [168, 167]. In practice, design vulnerabilities do not only originate from 50% of security issues [302] but they are also the most harmful [317] and difficult to identify [260, 388]. Securing smart contracts calls for approaches that can accelerate the identification of root causes of vulnerabilities in the design of such contracts.

The novel contribution of this chapter is a debt-aware approach for assessing security design vulnerabilities in smart contracts. The technical debt metaphor has proven to be effective to measure the impact of security weaknesses exploitation in terms of damage to business value [167], the ramifications of negative decisions over time, and locating the design root of security vulnerabilities [260]. Then, we leverage the metaphor to estimate the monetary cost of redeploying the patched version of the vulnerable contract and the

evolution of the debt interest linked to a design vulnerability. Our approach is based on both automated analysis tools and manual analysis to discover potential security vulnerabilities caused by design decisions. The evolution of the negative consequences of these security issues in smart contracts is estimated as an analogy with the concepts of debt principal and interest growth rate.

This research intends to answer the following research questions (RQ):

- **RQ1:** How to identify design vulnerabilities in smart contracts? What are the specific analysis techniques and tools?
- **RQ2:** How to quantify the impact of technical debts related to design vulnerabilities in smart contracts?

Although prior works have introduced the idea of technical debt in the context of software security [168, 167, 260, 291], to the best of our knowledge, our work represents a novel initiative in introducing the metaphor to discuss design vulnerabilities that lead to security issues in smart contracts. Moreover, although previous works have surveyed smart contract vulnerabilities [212, 152, 60], up to our knowledge, this research is among the initial efforts that focus on design vulnerabilities and map them to their related entries in the Common Weakness Enumeration (CWE) list, which is a catalog of common software and hardware weakness types [243]. Additionally, different from previous efforts that performed empirical evaluations of automated analysis tools to compare their real capabilities [101, 270, 202], this work characterises a set of automated analysis tools that can discover a group of design vulnerabilities pertinent to smart contracts.

The remainder of the chapter is organised as follows. Section 4.2 introduces necessary preliminaries of smart contracts and technical debt, while Section 4.3 provides a detailed overview of our debt-aware approach to assess security design issues in smart contracts. We

report on experiments guided by our approach in Section 4.4, followed by a discussion of our results in Section 4.5. The evaluation of the proposed approach is discussed in Section 4.6. Finally, related works are contrasted with ours in Section 4.7. Section 4.8 summarises the chapter.

## 4.2 Preliminaries

### 4.2.1 Ethereum Platform

Ethereum is the most popular general-purpose blockchain platform that enables developers to deploy smart contracts written using Turing-complete languages such as Solidity. Solidity is an object-oriented language designed specifically for writing blockchain contracts. The programmable contracts are then compiled into bytecode that executes on the Ethereum Virtual Machine (EVM). EVM executes typical cryptocurrency transactions and contracts bytecode, which are a special kind of transaction [365].

In practice, several properties distinguish smart contracts from regular computer programs and make the implications of deploying vulnerable contract more severe. First, the primary distinction is that contracts operate on a decentralised public blockchain network that makes the contract open for inspection, with the state of the contract transparent and traceable by everyone [356]. Second, contracts typically handle monetary transactions that can involve considerable amounts of Ether, an Ethereum cryptocurrency with a current market value equivalent to billions of dollars [60]. The combination of monetary value and public availability makes vulnerable smart contracts compelling targets for attackers. Third, smart contracts are irreversible and immutable. Consequently, vulnerable contracts cannot be patched after deployment to the blockchain [225]. The only way to fix the contract is

to deploy a new version with a repair code. However, the old version will remain in the blockchain. Fourth, deploying and executing smart contracts cost an amount of gas (the fuel of computation in Ethereum) [392].

To prevent the abuse of computational resources, in Ethereum, developers and users are required to pay a gas fee to deploy and execute contracts, respectively [365]. The concept of gas involves the following: (i) **gas cost**, which is a constant amount that determines the computational effort required to execute specific operations; (ii) **gas price**, which denotes the amount of Ether that users are required to pay for each unit of gas; it is dynamic, governed by Ethereum miners, and measured in Gwei (1 Gwei = 0.000000001 Ether); and (iii) **gas fee**, which is the incentive received by miners in exchange for the computational resources used to execute the operations of a contract and the building of blocks; gas fee is converted to Ether and paid to miners.

#### 4.2.2 The Common Weakness Scoring System

The Common Weakness Scoring System (CWSS™) [244] provides a quantitative mechanism to score unfixed CWEs found in software. The system supports prioritisation of weaknesses in terms of their potential negative consequences. The scoring formula is based on three metrics: (i) Base Finding (BF), which consists of several factors that estimate the level of technical impact once the weakness is successfully exploited, the accuracy of the finding, and the effectiveness of the existing control; (ii) Attack Surface, whose factors estimate how easy the attacker could exploit the weakness; (iii) Environmental (E), which involves factors that refer to a specific operational context such as the potential impact to the business and the likelihood of discovery. Each metric factor is assigned a numeric value that is calculated based on specific formulas, and the sub-score of each metric is multiplied with each other to generate the final CWSS score in a range between 0 and 100.

### 4.2.3 Technical Debt

Technical debt is a metaphor devised to capture how the value of software engineering decisions evolves over time [240]. Specifically, the metaphor supports the identification of the roots of a sub-optimal decision, the estimation of its value, and the monitoring of the environmental trade-offs in which the decision was made [193]. The metaphor also supports attributes that are intrinsic to debts in finance such as principal and interest [332]. In the metaphor, the principal represents the value of the gap between the ideal and the actual decision, whereas the interest represents the additional effort that needs to be incurred to pay back the principal.

Technical debt has been applied in architectural design to value the gap between the ideal and an actual decision [214], to identify the architectural root of an issue [176], and to manage the debt incurred by a decision [14], among others. Since some roots of security issues tend to be intertwined with architectural decisions, the metaphor has also been used to identify security vulnerabilities [260], prioritise the attention to security weaknesses [167], and estimate the consequences on business value if vulnerabilities are exploited [168].

## 4.3 Our Approach for Assessing Technical Debts

This Section presents our approach to assessing the security technical debts incurred in smart contracts design. The assessment involves detecting design issues and quantifying their consequences to security if remain unfixed.

We have defined the steps that support the creation of secure-by-design smart contracts:

1. Identify security design vulnerabilities in a smart contract.
  - (a) Run automated security analysis tools on the smart contract source code to obtain a list of potential vulnerabilities.
  - (b) Perform a manual analysis complementing the automated analysis to uncover any missed vulnerability.
  - (c) Determine the design vulnerabilities and map them to related weaknesses to highlight the root cause of the issues.
  - (d) Classify the design vulnerabilities per design flaw categories to determine the negative technical impact upon the contract.
2. Measure the ramifications of the identified design vulnerabilities.
  - (a) Estimate the monetary cost of technical debt principal by calculating the gas fee for redeploying a patched version of the vulnerable contract.
  - (b) For each vulnerability identified, estimate technical debt interest value by quantifying the negative security consequences and the interest growth rate over time.

Our approach is a debt-aware assessment approach, as it facilitates visualisation of the debt incurred by exploitable flaws created while designing smart contracts. Furthermore, by applying this approach, developers can be aware of the long-term consequences of security design issues. Subsequently, the developer can prioritise the debt based on both the monetary cost and the interest value associated with vulnerability violations.

### **4.3.1 Identification of Security Design vulnerabilities**

Building our assessment approach involves two steps: aggregating vulnerabilities caused by flaws in smart contracts design and selecting security analysis tools that assist the identifi-

cation process.

### **Mapping Design vulnerabilities to Security Design Weaknesses**

Security flaws coming from design decisions have been reported as the main cause of software security problems [260]. If these flaws remain unaddressed, these can be viewed as root cause of technical debt, with consequences observed through an accumulation of interests over time [168, 260]. Additionally, if security software engineers are aware of these issues but they do not fix them, these workarounds can be considered as (self-) admitted technical debts.

Although several efforts, from industry and academia, illustrate smart contract vulnerabilities, they have missed distinguishing between coding and architectural design vulnerabilities. Therefore, in this study, we present a set of vulnerabilities rooted in the architectural design of smart contracts and their related security weaknesses. The aggregated set is classified based on the security impact of such issues.

Security weaknesses are flaws in software that may lead to exploitable security vulnerabilities. Therefore, the identification of weaknesses can assist us in understanding the security problems and performing root cause analysis. Our study leverages Common Weakness Enumeration (CWE™) [243], which is a community-developed list of common security weaknesses that may appear in the architecture, design, or implementation of software. Thus, we aim to map vulnerabilities in contract design to security weaknesses in the CWE catalog that stands out regarding adoption and scope of coverage. Although CWE does not refer to any weaknesses particular to smart contracts, it depicts associated weaknesses at higher abstraction layers.

We followed a systematic procedure to collect and map each design vulnerability in contract design architecture to the corresponding weaknesses in CWE. This procedure



consisted of two steps.

First, we collected a list of security vulnerabilities from academic papers that surveyed existent vulnerabilities in smart contracts [152, 212, 60, 24]; we also considered Ethereum community [126], wiki [360], and developers' blogs [408, 288] that listed and explained exploitable flaws and anti-patterns not discussed in the literature. Specifically, we extracted information about the vulnerabilities, such as their descriptions, ways of exploitation, and preventive techniques that can be used to avoid them. The collected information assisted in knowing the negative impacts of these issues on the contracts and the cost of fixing them. In addition, the information also helped to determine the vulnerabilities that result from the flaws manifested at the contract's design stages. Second, we analysed each collected vulnerability to map it with the subset of weaknesses (438/1248) of CWE. This subset represents weaknesses that can be introduced during a design stage.

To reduce inherent biases in the mapping process, we separately worked over all the collected vulnerabilities. After completing the analysis, results were compared and double-checked with Smart Contract Weakness Classification (SWC) [315] Registry. This registry was established by a group of developers, auditors, and researchers at ConsenSys Diligence [66] that provides smart contract developers with a system analog to CWE. However, at the time of writing, only 20 architectural design security weaknesses are listed in the registry. Finally, a group discussion was conducted to resolve any disagreement.

We observed that the identified design vulnerabilities can be classified based on their impact into ten categories. Front-Running, Time Manipulation, Denial of Services (DoS), Broken Access Control, Arithmetic Issues, and Bad Randomness. These first six categories are also presented in Decentralised Application Security Project (DASP) [147] taxonomy. Other categories are Sensitive Data Exposure and Using Components with Known Vulnerabilities, which are the third and ninth categories, respectively in Open Web Application

Security Project (OWASP)-Top 10 Security Risks [127]. The two remaining categories are Improper Inheritance and Modularity Violation, which are the flaws that violate object-oriented design principles. Since Solidity is an objected oriented language, the contracts written by this language are also prone to these types of design flaws.

Table 4.2 describes each design flaws category, whereas Table 4.1 shows an example of mapping the design vulnerabilities classified under DoS to relevant CWEs. Our replication package, described in Subsection 4.3.3 carries full details of the mapping.

Our classification allows the designers to select a specific design flaws category and visualise all related vulnerabilities and weaknesses. It may also serve as a guide for architects to avoid common security architectural issues when creating smart contracts.

### **Selection of Security Analysis Tools**

The analysis of potential vulnerabilities in smart contracts needs to be performed before its deployment to the immutable environment of the blockchain, in which refactoring or updating the contract is costly and not trivial.

Recently, a growing number of security analysis tools have emerged to detect common vulnerabilities and bad practices in Ethereum smart contracts written in Solidity. Although these tools have been developed by different teams such as academic teams [334], community teams [225], and industrial teams [248], most of the existing tools are inaccurate in finding the security issues as they yield a considerable number of false positives [101, 270]. Furthermore, each tool only detects a limited scope of vulnerabilities. Consequently, the combination of several existing tools is essential to increase coverage and accuracy of vulnerability detection.

In this study, we selected a set of security analysis tools that help in identifying design vulnerabilities in smart contracts. To do that, we investigated the academic literature,

Table 4.1: An Example of Vulnerabilities Mapping for DoS to CWE Categorisation

Design Vulnerabilities	CWE	Design Weaknesses
Exception handling problem	CWE-703	Improper Check or Handling of Exceptional Conditions
Non-validated arguments	CWE-20	Improper Input Validation
DoS by external contract/Call	CWE-703	Improper Check or Handling of Exceptional Conditions
Calculates the upper bond of Gas	CWE-400	Uncontrolled Resource Consumption
Balance equality /Unexpected Ether balance	CWE-667	Improper Locking
Costly pattern/Costly loop	CWE-400	Uncontrolled Resource Consumption
Call to untrusted contract	CWE-807	Reliance on Untrusted Inputs in a Security Decision
Non-validated return value	CWE-20	Improper Input Validation
Exception Disorder	CWE-703	Improper Check or Handling of Exceptional Conditions
Reachable SELFDESTRUCT operation	CWE-284	Improper Access Control
Destroyable / Suicidal contract	CWE-267	Privilege Defined With Unsafe Actions
Unprotected simultaneous execution of sensitive tasks	CWE-667	Improper Locking

Table 4.2: Description of Design Flaws Categories

Study	Purpose	Focusing Area
Front-Running	DASP	A type of race condition where a malicious user can steal the solution and submit a transaction with a higher gas price to make their transaction assigned to the block before the victim.
Time Manipulation	DASP	The timestamp of the block is adjusted by malicious miners to their own advantage.
Denial of Services	DASP	The attacker can make the contract inoperable temporarily or permanently.
Arithmetic Issues	DASP	An arithmetic operation that reaches the max or min size of a type and presents incorrect results that compromise contract security and reliability.
Bad Randomness	DASP	The random number generator is written in a way that is predictable and exploitable.
Sensitive Data Exposure	OWSAP-10	The developer does not adequately protect critical information related to the contract and assumes that private type variables cannot be read.
Using Components with Known Vulnerabilities	OWSAP-10	Malicious or deficient components, such as libraries or off-chain data sources, where the developer unaware of all the running code.
Improper Inheritance	Object Oriented Design Flaws	When a contract inherits another contract, the presence of multiple variables with the same name in both contracts might lead to unintended effects.
Modularity Violation	Object Oriented Design Flaws	This consists of a tight coupling between contracts that makes them frequently change together.

searched the Internet, and scanned Github. The tools selected for our study were those matching the following criteria (C):

**C#1.** The tool is free, publicly available, and specific information about it can be found in at least one official source. This excludes tools such as Mythx [66] and DappGuard [69]. The former is not free and the latter could not be found.

**C#2.** The tool is up to date. This excludes deprecated tools such as Porosity [321], and tools that detect several outdated vulnerabilities, such as Oyente [225].

**C#3.** The tool can take source code of the contract as an input, which excludes tools such as Echidna [202] as it only takes EVM bytecode.

**C#4.** The tool identified at least one design vulnerability or bad practice related to design. This excludes tools such as Solgraph [287].

After exploring all the collected tools, we identified only nine tools that met the criteria. These tools are: Slither [121], SmartCheck [331], and Securify [338] which perform static analysis techniques. Mythril [249] which is a symbolic analysis tool and Manticore [248] which performs dynamic symbolic execution identification. Another tool is sFuzz [254] which applies a fuzzing testing technique. Solhint [281] and Ethlint [98] perform static analysis to identify security issues and bad practices. The final tool is Mythos which applies a combination of dynamic symbolic execution and fuzzing techniques.

### **4.3.2 Measurement of Negative Consequences of Design vulnerability in Smart Contracts**

Besides detecting vulnerabilities, it is also crucial to estimate the associated implications of taking a shortcut by not fixing those issues. To this end, we adopt the notions of principal and interest to quantify the debt cost and value related to each identified vulnerability in a smart contract. Quantifying the cost and value of the security debts aids developers to

apply a cost-effective technique and justifies the investment in fixing security problems.

### **Estimation of Gas Cost (Quantifying the Principal)**

Immutability in smart contracts restricts developers' ability to patch vulnerabilities after deploying the contract to the blockchain. The developer needs to upload a new version of the contract after fixing those vulnerabilities, and this requires additional gas consumption with the associated expenses. As a result, the need to refactor vulnerable deployed contracts has financial implications.

Our aim is to calculate the gas consumption fee for redeploying a contract to estimate technical debt principal (P). This estimation provides useful insights for developers regarding the cost of repairing vulnerabilities in the deployed contracts. We use the following formula:

$$P_{SecurityDebt} = Gas\_D(c) + Gas\_U(c) \quad (4.1)$$

where Gas\_D indicates the cost of gas required to deploy the repaired contract (c), and Gas\_U indicates the cost of the gas required to update the contract (c) for a given issue.

**Gas Cost Required to Redeploy Repaired Contract.** The cost of the gas required for deployment depends on the size of the smart contract. The number of functions (NoF) and lines of code (LoC) are both influencing factors. Each operation consumes a specific amount of gas based on its complexity and resource requirements. More complex operations, such as loops or heavy calculations, consume more gas. Similarly, when a smart contract interacts with other contracts on the blockchain, it incurs additional gas costs. This encourages developers to design contracts that minimise external interactions, reducing the risk of introducing vulnerabilities or performance bottlenecks due to excessive inter-contract calls.

The gas cost model provides cost predictability for users. Before executing a trans-

action, users can estimate the gas cost based on the anticipated complexity and external interactions. This discourages the execution of overly complex code, as users would need to pay more for these resource-intensive operations. This helps maintain network performance by preventing excessive computational demands.

The uploading cost fee is computed using  $gas\_price \times gas\_cost$ , where the former is the value of a unit of gas as specified by the market, and the latter is mostly determined by the summation of the following factors:

$G_{create}$ , 32000 gas, paid for a CREATE operation,

$G_{transaction}$ , 21000 gas, paid for every transaction,

$G_{codedeposit}$ ,  $200 \times |o|$  gas, paid per byte for a create operation,  $|o|$  amount of bytecode in the compiled contract,

*Execution costs*, gas paid for necessary computation processes.

The last factor refers to the costs associated with the part of the code that requires to be executed before the creation of the contract, such as initialising the state variables whose values are permanently stored in the contract storage, and executing a constructor. If the constructor requires a lot of computation to generate the bytecode, then there will be an extra expense. Appendix G in the Ethereum yellow paper [365] shows the costs, in gas, of several opcode operations.

**Gas Cost of Update Pattern.** Since redeployment results in a new contract with a new address, an updated pattern needs to be utilised to avoid using the vulnerable deprecated contract. The developer can use a self-destruct opcode, which costs  $G_{selfdestruct} = 5000$  gas, to destroy the contract. Another option is to upload the main contract with a proxy smart contract, which has a changeable variable that stores the main contract's address. Thus, once an updated contract version is released, the value of this version is updated.

### **Estimation of Security Consequences (Quantifying the Interest)**

In the security context, the interest value represents the undesirable effects that can result if the vulnerability is exploited. The longer the exploitable design flaw remains unaddressed in the deployed contract, the higher the chance the debt interest associated with this flaw grows. Accordingly, three factors are considered when estimating the accumulated interest: CWSS score, contract activity level, and contract lifespan.

**CWSS Score.** We adopted the CWSS to estimate the severity of identified weaknesses in a contract. The CWSS framework provides different scoring methods; in this study, we use the targeted method, which assesses individual design weakness. Multiple factors are used to quantify CWSS scores in smart contracts, such as mapping the vulnerabilities to the related CWEs and the proposed design flaws categories. This information allows the negative impact on the contract in the case of an attack to be estimated. The estimated score generated by CWSS allows for technical debt items to be visualised and those debts that lead to more severe consequences to be addressed. In practice, addressing contract security design issues early, in the pre-deployment stage, minimises debt.

The developer can use the OWASP method to accurately estimate the technical and business impact factors of CWSS, in case vulnerabilities are exploited. This method suggests dividing the technical impact into sub-factors aligned with the traditional security areas of concern: loss of confidentiality, integrity, availability, and accountability. Similarly, the method divides the business impact factor into sub-factors common to many businesses: financial damage, reputation damage, non-compliance, and privacy violation.

*Accumulated interest.* Debt interest increases when exploited contracts become extremely common like, for example, the hacked Parity Multi-Sig Wallet contract [60]. This was a smart contract for a multiple signature wallet that had a critical, exploitable vulnera-



bility, which allowed an attacker to steal millions of dollars. The debt interest also increases if the exploitation of vulnerabilities in the contract has irreparable consequences, as is the case with suicidal vulnerability. With this vulnerability, any arbitrary account can kill a contract, causing it to stop functioning and locking its ether [338]. The accumulation of interest associated with vulnerabilities in smart contracts is difficult to quantify; however, we find that the activity level and the lifespan of the contract are useful for estimating interest growth.

**Contract Activity Level (CAL).** CAL refers to the expected number of active users and the number of transactions that a contract is expected to deal with. This factor can be predicted from statistical information about contracts provided on websites, such as State of the DApps [82], which is a website with a curated list of smart contract-based applications. This website ranks the contracts and categorises them based on their activity level. Statistical research [263] shows that the top three high-activity contract categories in 2020 are games, currency exchanges, and gambling.

**Contract Lifespan (CLS).** CLS refers to the time, measured in days, between the contract's deployment and its last execution. Most smart contracts live between 2 and 800 days [222], and the average age of smart contracts is 295.6 days [263]. A contract can be short-lived, medium-lived, or long-lived [263, 222]. A contract is long-lived when it is expected to be executed for a long interval or even for as long as the network exists. Conversely, a short-lived contract is mostly designed to be executed for a limited time and then either the interaction with it ends or it is destroyed. This type of contract typically includes few operations, involves few fixed users, and has limited transactions. Thus, the security implications of breaching this contract are not significant compared to a long-lived, highly active contract in which there is interaction with thousands of customers and thousands of transactions are accepted over a long period of time.

Therefore, during the CLS period (i.e., before the contract reaches its maturity stage), the accumulated interest (AI) of the security technical debt can be estimated as follows:

$$AI_{SecurityDebt} = CWSScore \times CAL \times CLS \quad (4.2)$$

Nevertheless, we acknowledge that some issues may go beyond, and the interest can be adjusted accordingly. For this work, we focus on interest within the CLS period, as this enables action to be taken before contract redemption to avoid the accumulation of issues that are rooted in technical debt.

The developer can assign a point scale to each contract activity level category and lifespan category. The multiplication of the scores of the three factors determines the accumulated interest.

### 4.3.3 Replication Package for Replicability

The data source and replication package for our experiments are publicly available <sup>1</sup> to assist in verifiability and reproducibility of our results. The package includes: (i) the complete mapping of all aggregated design vulnerabilities and their rated CWE entries; (ii) the list of all founded analysis tools; (iii) the set of vulnerabilities claimed to be identified by each tool; (iv) details of inclusion criteria that were not met by each of the excluded tools; (v) the dataset used in the experiment; and (vi) the detailed results of each step of the approach.

## 4.4 Experimentation

To demonstrate our approach, the following example shows how the steps of our approach are performed to assess the potential security-related debts in smart contracts.

---

<sup>1</sup>[https://bitbucket.org/Smart\\_Contract/assessing-smart-contracts-security](https://bitbucket.org/Smart_Contract/assessing-smart-contracts-security)

### 4.4.1 Experiment Setup

We collected a dataset of 16 representative vulnerable smart contracts that are either actual contracts identified as vulnerable or explicitly programmed to demonstrate a specific vulnerability. This dataset comes from several publicly-available resources: (i) *GitHub* repositories such as SWC Registry, not-so-smart-contracts [39], and Solidity-Security [234]; (ii) *Ethernet* [326], an online game for smart contract attacking challenges; and (iii) *blog.positive* [311], which publishes articles discussing vulnerabilities in smart contracts.

We intentionally chose a dataset of known vulnerable contracts to examine the ability of the nine selected security analysis tools to identify vulnerabilities. Our selection of smart contracts enables a discussion of their source codes and permits the publication of our results for replication without users being involved in legal and privacy issues.

All the tools are installed and run on the same computer utilising Solidity compiler-`solc` (v 0.7.1), under Ubuntu 18.04.5 operating system. Only `sFuzz` provides a web-based interface for using the tool, then there was no need to install anything related to it.

### 4.4.2 Experimental Study

Figure 4.1 represents the overall procedure of our experiment.

**Step 1.a.** We first ran the automatic security tools against each contract in our dataset to identify the design vulnerabilities. As mentioned in the previous Section, different types of tools were selected to enhance the detection abilities of design vulnerabilities. We started the operation process by running the tools that perform static analysis techniques. After the static analysis has been completed, operating the tools that perform dynamic analysis techniques was the next step for obtaining deeper results. We finalised the analysis

Table 4.3: Experiment Dataset. For each Vulnerable Contract, We Provide Design Flaws Categories of its Flaws (DFC), Number of Design Vulnerabilities (Vulns), and Lines of Code (LOC).

Contracts	DFC	#Vulns	#LOC
FindThisHash	Front-Running	1	9
EtherLotto	Time Manipulation / Bad Randomness	1	20
Roulette	Time Manipulation	1	14
Lottopollo	Time Manipulation / Bad Randomness	1	24
DosAuction	DoS	1	13
SimpleToken	DoS / Access Control Broken	1	65
Etheraffle	DoS/ Bad Randomness	2	122
DosNumber	DoS	1	28
AccessControl	Access Control Broken	1	53
BlockdBuildDemo	Access Control Broken	3	62
FunctionTypes	Access Control Broken	2	18
OddEven	Sensitive Data Exposure	1	22
Transaction_malleability	Sensitive Data Exposure	1	77
Token	Arithmetic Issues	1	17
TokenSaleChallenge	Arithmetic Issues	1	20
CEOThrone	Improper Inheritance	1	21

by running the tool's applied fuzzing techniques.

**Step 1.b.** We conducted a manual analysis to complement the previous step as the selected tools could not detect all the known vulnerabilities in the dataset. We annotated the known vulnerabilities and detected several other flaws that were not mentioned in the online source with regard to the vulnerable contracts. We ended up with a set of possible vulnerabilities that was composed of flaws not related to the design.

**Step 1.c.&1.d.** We checked the aggregated set of design vulnerabilities and the aligned CWEs to only record the issues that belonged to that set. Each contract had one or more vulnerabilities mapped in at least one of the design flaw categories. There were 20 distinct vulnerabilities in our dataset. Table 4.3 shows each collected contract, its name, the categories of its flaws, the number of design vulnerabilities, and the lines of code.

**Step 2.a.** The cost fees required to deploy the patches contracts were calculated by Formula 4.1 to quantify the technical debt principal. A self-destruct updated pattern was considered when calculating the total gas cost. At the time of experimenting, the average gas price was 126 Gwei, and Ether's price was around \$500 US.

**Step 2.b.** We estimated the security risk severity scores related to identified vulnerabilities and weaknesses using the CWSS formula that explained in Subsection 4.2.2. We used the OWASP method to estimate the technical and business impact sub-factors. Additionally, our proposed categories of design flaws support determining the technical impact as each category shows the ramifications of the exploitation. In OWASP method, each sub-factor has several scored options. We selected one of the options and then averaged the scores for each factor, technical and business impacts, to decide their severity level.

There are other required factors to calculate the CWSS score, such as required privileges; authentication and access vector to perform the attack; and the acquired privilege

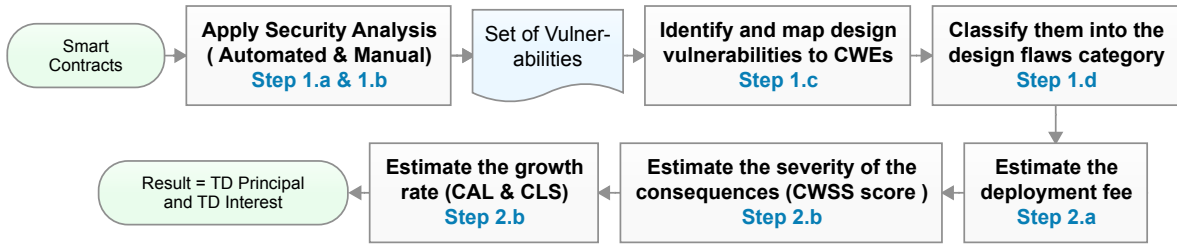


Figure 4.1: Experiment Execution Steps

after accomplishing it. These factors were determined based on the information collected about each vulnerability and the way of exploiting it. Additionally, the knowledge about the vulnerable contracts and security analysis performed assisted in determining factors such as finding confidence, the likelihood of discovery, and exploitation.

In addition to the CWSS score, the contract's activity level and lifespan factors have to be estimated to visualise the potential accumulated debt interest. A six-point scale is used to determine the activity level, where six refers to the top three highly active contract categories, and one refers to the lowest three categories. The ranking of the categories is informed by the State of the DApps website. Since most smart contracts live between 2 to 800 days, we thus divided the lifespan into three intervals and gave each interval a score as follows: (i) Short-lived, 1-266 days, 0.17 score; (ii) Medium-lived, 267- 533 days, 0.35 score; (iii) Long-lived, 534-800+ days, 0.5 score. The accumulated interest for each unfixed vulnerability was calculated by Formula 4.2. The value of the final score is between 0 and 300. Noticeably, we set those particular scores to clearly visualise how the interest could redouble. However, smart contract developers can customise the scores based on their context.

## 4.5 Results and Discussion

In this Section, the collected dataset of some represented vulnerable contracts is analysed and discussed with respect to our research questions. Additionally, we discuss the practical implications of our findings and we outline the potential threats to their validity.

### 4.5.1 Identification of Design Vulnerabilities (RQ1)

Locating design vulnerabilities in smart contracts source code is the main step in our assessment approach. Figure 4.2 presents the results of executing nine security analysis tools on 16 vulnerable contracts in order to locate their design vulnerabilities. It shows the percentage of vulnerabilities that each tool was able to detect per design flaws category.

Nine tools, in combination, were only able to locate 70% (14/20) of all the vulnerabilities. Most of the tools were generally able to detect a high percentage of vulnerabilities classified in the categories *Time Manipulation*, *Improper Inheritance*, and *Bad Randomness*. However, the tools underperformed when it came to identifying vulnerabilities of the categories *Front-Running* and *Sensitive Data Exposure*. The nine tools failed to detect four vulnerabilities linked to *Access Control Broken*, one vulnerability linked to *DoS*, and another linked to *Sensitive Data Exposure*. Throughout the analysis, we observed that the output of some tools was noisy, as they failed to disregard the false warnings and detected unreal flaws.

Figure 4.2 indicates that the tools demonstrate different abilities to locate design vulnerabilities. The tool Mythril surpassed all other tools as it was able to identify more vulnerabilities in the 16 contracts than any of the others. Securify is the only tool that detected the vulnerability, Transaction Ordering Dependency, linked to the *Front-Running*

*category*. Although the documentation of several tools such as Manticore claimed that they could detect this vulnerability, they failed to identify it. Similarly, Smartcheck is the only tool that identified the vulnerability, Unencrypted Private Data On-Chain, in the *Sensitive Data Exposure* category. However, the output of Smartcheck does not reveal a clear explanation of the identified flaws. Mythril and Mythos both provide informative output as they link the identified vulnerabilities to related SWC. Securify resulted in the most confusing output among all the tools because it showed a large number of false alarms.

Our study emphasises that conducting manual analysis to complement automated ones is essential to discover most vulnerabilities. This is because, as our analysis experiment shows, the scope of issues addressed by state-of-the-art tools is limited. Additionally, manual analysis assists in eliminating false positives that might be detected by automated tools. In fact, the accuracy of automated analysis tools needs to be increased by reducing the likelihood of false-positive alarms that confuse the developers and discourage them from leveraging the results. Even though an empirical study [101] claimed that combining different types of analysis tools yields more vulnerability coverage, the existing tools are not powerful enough to replace manual analysis. Bhardwaj et al. [34], who contrasted the outcomes of manual penetration testing with automated test scanners, further confirm this assertion. The study shows that manual vulnerabilities investigation outperformed the automated smart contract tools in terms of precision and soundness of vulnerabilities detected.

The manual analysis provides a deeper understanding of the smart contract's logic, interactions, and underlying business rules. Complex attacks often exploit nuanced vulnerabilities that automated tools might miss. Manual analysis can reveal hidden attacks by examining how different parts of the contract interact in intricate ways. Moreover, complex attacks often exploit zero-day vulnerabilities that have not been previously documented. Manual analysis can detect these vulnerabilities by assessing the contract's code, logic, and interactions, filling the gap left by automated tools' reliance on known signatures.



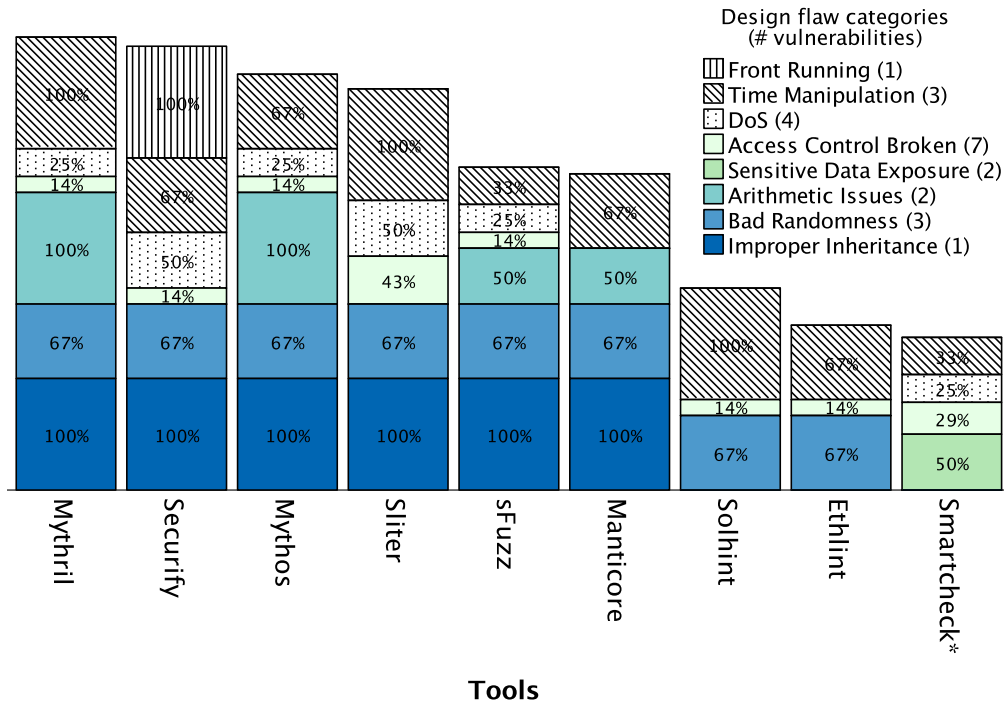
Table 4.4: Overall Severity Level

	Value		
Cost	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Manual analysis offers a critical layer of expertise that complements the capabilities of automated tools. It addresses the limitations of automated tools in detecting complex attacks. By combining automated tools and manual analysis, developers can achieve an effective approach to identifying and mitigating sophisticated security vulnerabilities. Therefore, taking a shortcut by deploying the contract to the public without performing both automated and manual inspection processes might lead to invisible exploitable flaws that are prone to attacks.

Through the analysis, all the detected flaws were found in our aggregated set of design vulnerabilities and their associated weaknesses. In contrast, some of these detected flaws, including no restricted write and no restricted transfer, did not exist in the SWC Registry which provides a set of classified issues that come up in smart contract development. Moreover, our proposed design flaw categories cover all the detected flaws, while the current DASP taxonomy is not extensive enough to include all types of security design issues that affect smart contracts. Unlike DASP, our classification includes vulnerabilities of the following categories: Sensitive Data Exposure, Using Components with Known Vulnerabilities, Improper Inheritance, and Modularity Violation.

Classifying vulnerabilities based on their impact and mapping them to a wider group of agreed-upon weaknesses (CWEs) come with several benefits: (i) it provides a common



\* Example: several vulnerabilities can be detected by the Smartcheck tool including 1/2 of Sensitive Data Exposure, 1/7 of Access Control Broken, 1/4 of DoS, 1/3 of Time manipulation, and 0 from the rest of the categories

Figure 4.2: Identified Vulnerabilities per Category by each Tool.

language for defining security architectural design issues in smart contracts; (ii) it offers a simple way to classify security issues in such contracts; and (iii) it generates useful insights into the root causes of vulnerabilities, and the negative impacts posed by the exploitable ones.

#### 4.5.2 Estimation of Negative Consequences (RQ2)

The second step in our approach is to estimate the ramification of deploying vulnerable contracts to the public blockchain in terms of debt principal and accumulated interest. Figure 4.3 shows the debt principal and interest of 10 vulnerable contracts from our dataset out of 16. The Figure represents the estimated monetary cost of refactoring each contract and the value of the accumulated interest in relation to the effects of the vulnerabilities over time if

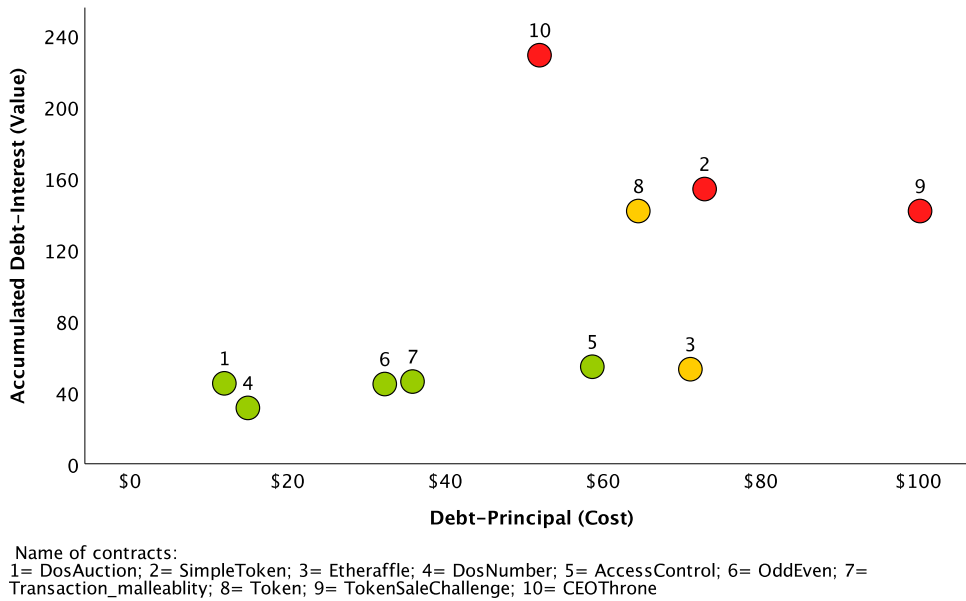


Figure 4.3: Principal and Interest of Ten Vulnerable Contracts.

they are left unfixed.

For the sake of visualising and discussing the data, we have categorised the coordinates of the (cost, value) points in the graph into three categories: high, medium, and low, depending on their severity. Given the score for values in this range [0,300], we defined  $(100 < \text{value} \leq 200)$  as medium, values greater than 200 as high, and values lower than or equal to 100 as low. As in our example the highest cost is \$100.25, we defined  $(\$33.33 < \text{cost} \leq \$66.66)$  as a medium, a cost greater than \$66.66 as high, and a cost lower than or equal to \$33.33 as low. The final degrees of severity are then derived from both the cost and the value using Table 4.4. As such, vulnerabilities in contracts 2, 9, and 10 falls into the high category; vulnerabilities in contracts 3 and 8 into the medium category; and those in contracts 1, 4, 5, 6, and 7 into the low category.

Estimating the cost and value related to security violations in smart contracts facilitates decision-making regarding which issues need more attention when seeking to reduce technical debt. For instance, a vulnerability in the CEOThrone contract is assumed to be

in the high category. The cost of gas required to fix the issue that has not been fixed before deployment is 897,200, which is equivalent to \$51 at the time of writing. The size of this contract is relatively small compared to other contracts in our dataset. There is only one operation in the constructor, and there are no state variables that need to be initialised.

However, the contract suffers from variable shadowing vulnerability, which occurs when a variable declared within a child contract has the same name as a variable declared in a parent contract. In the case of CEOThrone, this allows an unauthorised node to withdraw the balance of the contract, leading to a significant technical and business impact. The likelihood of discovering and exploiting this issue is also high since most of the analysis tools were able to detect it. The vulnerability in this contract allows any attacker to gain the required privilege to exploit it. CEOThrone is a contract created for a game and it can thus be assumed that the contract will be highly active and have a long lifespan. As such, we estimated the accumulated security debt interest for this contract at around 228.9, which is a high score compared with other contracts.

Beyond the results of our experiments, employing our assessment approach helps significantly reduce the cost of developing smart contracts, in particular in cases where a contract is big and, as such, has a potentially higher cost, or in the case of fixing an issue in a contract that requires modifying and redeploying other dependent contracts. We provided a quantitative way of indicating the relative costs and values of publicly deploying vulnerable contracts. Developers can thus make informative decisions before uploading a contract to the blockchain. Additionally, they can prioritise the resolution of issues based on the available quantitative information. Overall, the assessment approach presented here supports reducing the introduction of unintentional debt caused by unawareness of security issues rooted in smart contract design while increasing the visibility of such issues and helping manage the debt more strategically.

## 4.6 Evaluation

We conducted an online survey with smart contract experts to assess the usefulness and clarity of our proposed approach for assessing smart contracts' security technical debts. The experts were pragmatically choosing based on their expertise and experience that they stated on their *LinkedIn* profile. We selected experts who had more than two years of experience in developing smart contracts. We sent a link to the survey to 150 smart contact practitioners and we received 20 responses. The participants were not asked for any personally identifying information.

### 4.6.1 Survey Questions

Table 4.5 shows the eight questions we devised. The primary goal of the first question was to understand the specific experience of the participant. A high percentage of the participants are working as programmers, designers, and/or architects, while few participants are working as decision-makers and/or researchers. One participant is working as an auditor. The following questions (Q2 to Q7) were kept simple with the aim of encouraging participants to justify the reasons for their answers. The purpose of these questions was to confirm the benefits and clarity of our approach. The final question (Q8) was left open-ended in order to gather suggestions for enhancing the approach.

### 4.6.2 Key Findings

The following are the key findings from the survey:

1. *Understandability.* The steps of our assessment approach, according to all of the participants, are clear and understandable. One participant suggested assigning each of

the key steps an acronym to help people remember them.

2. *Learnability.* The majority of the participants agreed that the approach will be potentially easy to use. One of the participants made the point that, in the case of complex smart contracts, the approach's implementation might become extensive. Additionally, two participants also emphasised that some developers might need help learning how to manually analyse smart contracts and that they might need to have a basic understanding of each class of vulnerabilities beforehand. We acknowledge that some of the steps might require a learning curve. To use automated tools or to conduct manual analysis, for instance, some of the developers may require training. But after they get past the learning stage, developers will become accustomed to the strategy and be able to use it effortlessly each time they design smart contracts.
3. *Usefulness.* The participants affirmed that the proposed approach is useful for developing secure contracts. Three of the participants emphasised that early detection of design issues is necessary as the impact of such issues might be substantial.

The participants affirmed that the approach is also useful in assessing the security risk of smart contracts. Most participants also believed that the strategy helped make security issues more visible. One participant made the observation that the developers of the smart contract often fail to conduct a thorough examination of the contract's security; however, by adopting our steps, they will be aware of the potential security issues and their effects.

All the participants affirmed that following our approach assists in designing secure contracts compared with ad-hoc approach. One of the participants emphasised the importance of dealing with smart contract security in a deliberate and structured manner. Another participant added that our strategy aids in directing an in-depth post-developmental analysis of the contract.

Table 4.5: Survey Questions

ID	Questions
Q1	What best describes your main role in the area of smart contracts?
Q2	Are the steps of the proposed approach clear and understandable?
Q3	Is the proposed approach potentially easy to use?
Q4	Is the proposed approach useful to support designing and developing secure contracts?
Q5	Is the proposed approach potentially useful in assessing the security risk of smart contracts?
Q6	Is the proposed approach potentially helpful in increasing the visibility of the security design issues related to smart contracts?
Q7	Does the proposed approach add additional value in designing secure smart contracts compared with doing so in an ad hoc manner?
Q8	What are your suggestions to enhance the suitability of our approach?

4. *Potential improvements.* One participant suggested predetermining the contract's scope and then deciding on the depth of the necessary analysis in order to expedite the proposed approach. We agree that the type of contract application determines how serious the security implications are. Therefore, we will incorporate this suggestion into our current strategy. Another participant suggested broadening the approach and making it applicable to contracts that are not Ethereum-based. Even though step 2.a of our approach concentrates on Ethereum-specific tools, the other steps are transferable to other (non-Ethereum) blockchains that support smart contracts.

Appendix 3 shows the participants' answers in detail.

### 4.6.3 Threats to Validity

A potential threat to internal validity is related to the possibility of considering alternative security flaws scoring methods. However, unlike other scoring systems, the CWSS can be applied at the early stages of the development process. CWSS provides support if there is incomplete information, as most of its factors have values for uncertainty and flexibility, such as Unknown and Not applicable. Furthermore, CWSS facilitates a thorough estimation of the security consequences of uploading vulnerable contracts. It considers 16 factors compared to the only four factors in the risk rating of the OWASP method.

Another internal threat relates to the potential subjectivity of CWSS results. We mitigated this possible risk as follows: (1) we proposed categories of design flaws that enable precise determination of technical impact; (2) conducting the security analysis creates awareness of the ease and likelihood of discovering any flaws; (3) in the proposed estimation Formula 4.2, the CWSS score is accompanied with smart contract activity level and lifespan factors.

Another internal threat might be related to using a six-point scale approach to determine the activity level of the smart contract. However, we consider this approach as offering a balance between granularity and simplicity, facilitating more nuanced evaluations and allowing for clearer distinctions between contract activity levels. These features make it a useful tool for our assessment scenarios. Nevertheless, its effectiveness depends on the context and objectives of the evaluation. Considering the pros and cons is essential when deciding whether to use a six-point scale or another scale format to assess the security debt interest.

There might also be an internal threat related to not mentioning the cost of development efforts in the technical debt principal equation. However, the equation can be updated



if fixing the detected flaws leads to a total redesign of the smart contract application. The developer can then update the equation by adding a factor that refers to the cost of the development effort that is required to refactor and fix each identified vulnerability before the redeployment.

A potential threat to external validity relates to the fact that the contracts we have considered in our experiment are relatively small and simple (approximately 1.5KB) - the typical size of commonly used contracts in practice as contracts are recommended to be simple and not complex. Nevertheless, the same steps and analysis can scale up to larger ones, often not exceeding 24KB in practice. Referring to our 10 out of 16 selected contracts in our study, the estimated monetary costs required to pay for repairing them were not significant. However, the quantitative approach provided for calculating the debt principal is applicable to any size and type of smart contract. As mentioned earlier, the monetary cost is dynamic as it depends on the gas price and the Ether price, either of which may significantly increase. For instance, in January 2018 the price of Ether rose dramatically to \$1066 US. Hence, uploading even a small and simple contract might cost a significant amount of money.

A potential threat to construct validity is the completeness of the aggregated set of design vulnerabilities. We mitigated this risk by conducting a thorough inspection of the academic papers, the Ethereum community, Wiki pages, and developers' blogs. Nevertheless, the set of vulnerabilities supplied is continually evolving because of the high potential for the emergence of new exploitable security design flaws in smart contracts.

Another threat to construct validity is the number of experts who assess the clarity and usefulness of our approach. Despite the evaluation's limited scope, the participants are all experienced smart contract practitioners or researchers with more than two years of industry experience.

## 4.7 Related Work

We discuss closely related work, covering (i) smart contract vulnerabilities; (ii) empirical evaluations of automated analysis tools that are relevant to our work; and (iii) security technical debt.

**Smart Contract Vulnerabilities.** Previous studies have illustrated, discussed, or surveyed various security vulnerabilities in blockchain and smart contracts. Li et al. [212] reviewed the security vulnerabilities in blockchain, the corresponding attacks, and suggested several security solutions, without differentiating between Bitcoin and Ethereum. Similarly, the authors in [152], classified the security vulnerabilities based on blockchain generation. They also provided a detailed explanation of known vulnerabilities and subsequent potential attacks. Unlike these studies, we focus on design vulnerabilities in smart contracts run on Ethereum. Other studies such as that of Atzei et al. [24], discussed 12 known vulnerabilities in Ethereum smart contracts, and classified them into three categories based on the level where they presented: the Solidity level, the EVM level, and the blockchain level. Similarly, authors in [60] provided the same classification of vulnerabilities in Ethereum smart contracts; however, they provided a more comprehensive list. In contrast, we aggregated vulnerabilities caused by flaws in a contract's architectural design, mapped them to their related CWE entries, and classified them based on their impact.

**Automated Analysis Tools.** As several automated tools have recently emerged to analyse the security of smart contracts, empirical studies have been conducted to compare their real capabilities and the techniques used. Durieux et al. [101] carried out a systematic evaluation of several state-of-the-art automated tools and discussed their accuracy and efficiency. We also followed a systematic method when it came to selecting the nine tools that were used in the vulnerability identification process. But different from other studies, we identified and analysed a set of tools able to discover a subset of design vulnerabilities.

Parizi et al. [270] performed an assessment of four static analysis tools with regard to 10 vulnerable smart contracts. In contrast, in our analysis, besides using static analysis tools, we included tools that use dynamic or fuzzing techniques. Additionally, we analysed their ability to identify design vulnerabilities on 16 vulnerable contracts. Leid et al. [202] compared the effectiveness of symbolic and fuzzy testing tools by evaluating their effectiveness when analysing smart contracts. Despite a large number of available automated analysis tools, they only considered three tools in their assessment.

**Security Technical Debt.** The nature of work presented in this chapter is generally related to applying technical debt to raise the visibility of security risks, and the costly consequences of deploying vulnerable smart contracts. Despite the vast contributions on technical debt in software, only few studies have looked at security-related debts including [168] [167] [260] and [291]. In particular, Izurieta et al. [168] established an approach to prioritise security technical debt in a software system. This was related to CWEs entries by leveraging the CWSS scoring systems. Unlike their approach, our study proposes a mechanism to estimate the accumulated debt interest, not only by using the CWSS score but also by including the activity level and lifespan of the smart contract under consideration. Additionally, we quantify the debt principle of refactoring the vulnerable contract. In [167], authors mapped attack tactics to the related posed consequences of the exploitable vulnerabilities. This helps to prioritise vulnerabilities that require the most attention to reduce technical debt. In [260], the authors applied a preliminary experiment to show the correlation between technical debt and software vulnerabilities. A recent study [291], regarding security debt, discussed the concept of managing security risk by leveraging technical debt. They emphasised that the combination of software security engineering techniques and technical debt increases the security of software systems. By articulating security technical debts in smart contracts, our study advances previous security technical debt research.

## 4.8 Summary

In this chapter, we have presented a debt-aware approach for assessing security design vulnerabilities in smart contracts. We used nine state-of-the-art tools to widen the detection abilities of security design issues in smart contracts. We use the CWE catalogue when analysing the identified vulnerabilities and their weaknesses. We adopted the community-informed scoring mechanism to consider contract activity level and the contract lifespan. The combination helps security software engineers to estimate the accumulated debt interests related to design vulnerabilities in a contract. Debt principal was quantified by calculating the gas fee required to redeploy the patched version of a vulnerable contract. Experiment results demonstrated that our approach can allow developers to visualise and prioritise technical debts, rooted in unaddressed smart contract design vulnerabilities. The approach can increase the visibility of debts and their ramifications. We evaluated the proposed approach based on expert judgment regarding its usefulness and clarity in guiding the architectural design of secure smart contracts and recognising the consequences of potential security debts.

# Chapter Five

## Decision Support Model for Blockchain Oracle Platform Selection

In Chapter 4, we propose a debt-aware approach for assessing security architectural design issues in smart contracts at the early design stage. In this chapter, we assess security debts that might be shipped to smart contracts from external third parties such as off-chain data feeders, which are also known as blockchain oracles. As the number of oracle alternatives and their features increases, the secure decision-making process becomes increasingly complex. Thus, in this chapter, we provide a decision-support model for the oracle selection problem. The model assists architects to compare and assess various platforms precisely and make a secure, cost-effective and optimal decision.

### 5.1 Overview

Blockchain-based smart contracts run in an isolated environment, preventing data about the situation and events in the real world from being acquired from outside the blockchain system. Blockchains get their most desirable qualities, such as security and trustlessness, by being

purposefully isolated from external systems. Many smart contract applications, however, cannot function without access to off-chain data sources, such as insurance smart contracts, which rely on online data to make policy payout choices. A blockchain oracle is a component that allows blockchains and off-chain systems to communicate. This component collects and validates data from a variety of external sources and transmits it to the blockchain. As the data that oracles supply influences how smart contracts behave, oracles have a significant level of control over how the contracts are executed, which makes them less trustworthy as a decentralised network component [48].

Numerous blockchain oracle platforms have emerged recently, such as Provable Things [282] and Chainlink [107], and are utilised in a variety of smart contract applications. As the number of available oracle platforms and their assessment criteria continues to grow, the selection of oracle alternatives available increases complexity and makes taking wrong or sub-optimal decisions more likely [232, 113]. These decisions become costly to fix because switching from one oracle platform to another requires uploading the smart contracts to the public blockchain, which is not free. For each deployment process, a specific amount of money needs to be paid [225]. Moreover, incorrect or unjustified oracle selections result from a failure to properly assess the available oracle platforms and they may also inadvertently lead to hidden security risks [54].

This study provides a model to support decision-making when selecting blockchain oracles. Our decision model is built using the multi-criteria decision-making (MCDM) approach [337], which involves evaluating a collection of alternatives while considering a set of decision criteria. The model assists blockchain and security software engineers in selecting an optimal oracle alternative to integrate into blockchain-based software. This model will inform them of the existing collection of oracle platforms, the relevant decision criteria, and quality attributes they should consider when building this software category. The presented decision model includes reusable knowledge on various oracle platforms and features. This

knowledge can help blockchain software engineers and decision-makers understand the capabilities of oracle systems, the features, and the quality attributes supported by each oracle platform. Our model leverages the security technical debt metaphor to assess the ill consequences of sub-optimal selection that manifest as a debt that will accumulate interest over time [5]. Therefore, blockchain software engineers and decision-makers can assess each oracle alternative systematically and make better-informed decisions, considering the monetary cost of integrating and executing alternative options. This article's primary contributions are as follows:

- It introduces a blockchain oracle decision support model, which is among the initial efforts to guide blockchain software engineers and decision-makers with the systematic selection of a feasible, secure, and cost-effective oracle for specific blockchain applications or systems.
- It provides a compilation of information on the features and quality attributes of eight cutting-edge blockchain oracle platforms to inform oracle selection decision-making systematically.
- It leverages the security technical debt metaphor to quantify the security risks of possible attacks on the examined blockchain oracles, and assists in eliminating solutions that can manifest into potentially costly and risky security technical debts. The model uses a multi-objective optimisation valuation mechanism to assist in the selection of the best alternatives, considering the monetary costs and security technical debts.

Our study represents a novel initiative in proposing an oracle decision model that assists smart contract developers in analysing the security of each alternative oracle by using well-informed decision-making. Previous research investigated and analysed only a small subset of the features of blockchain oracles [48, 232]. Our contribution goes beyond existing

research to compile an extensive set of features of blockchain oracles, to describe how they can be mapped to quality attribute requirements of the blockchain-based system under development, and to calculate the monetary cost of integrating each of the available oracle options. In contrast to previous efforts that described several attacks aimed against blockchain oracles [113], this work categorises and evaluates the attacks based on their likelihood and potential consequences and considers the domains of smart contracts in representative applications.

The remainder of the chapter is structured as follows. Section 5.2 introduces the necessary preliminary explanations of blockchain oracles and technical debt, while Section 5.3 provides an overview of our research approach. Section 5.4 presents the decision model for the oracle platform selection problem. Section 5.5 demonstrates two smart contract applications that were analysed to evaluate the effectiveness of the decision model, followed by an evaluation and a discussion of our results in Section 5.6 and Section 5.7, respectively. Finally, Section 5.8 sets out the summary of our work.

## 5.2 Preliminaries

Smart contracts are self-executing computer programs that run automatically when certain criteria are satisfied. Many smart contract applications built on blockchain technology, such as Ethereum [365], need to interact with other external systems. Smart contracts may need data such as price feeds, weather data, or even the generation of random numbers. However, smart contracts cannot fetch the required data since it is not possible to access resources outside of the blockchain environment [221].

Oracles can obtain and validate external data for smart contracts by using methods such as web APIs or market data feeds. The oracle initially monitors the blockchain network for off-chain data requests from users or smart contracts. It then interacts with the data



source to find the required data and it provides these data, and any associated proofs to the smart contract on the blockchain for consumption [58]. Therefore, the execution of the smart contract relies on the data retrieved from the data feed. Using an oracle introduces a trusted third party to the decentralised, trustless blockchain environment. This leads to a conflict between third-party oracles and the trustless execution of smart contracts regarding security, authenticity, and trust.

Third-party or external dependency, which describes the use of any framework or software developed by external parties, is one of the causes that contribute to technical debt accumulation [236]. Technical debt is a metaphor devised to encapsulate how the value of software engineering decisions evolves. Evidence shows that a great deal of software depends on external parties to at least some extent, making it almost impossible to avoid incurring this debt [84]. This is because reliance on external sources might lead to invisible issues, such as security issues that might be transferred to the main product and make the entire system prone to attacks.

In blockchain oracle selection, technical debt can manifest because of compatibility issues between the chosen oracle platform and a specific application and/or the misconfiguration of the oracle, which can lead to security, availability, reliability, or performance issues. Technical debt can also be caused by selecting an oracle without proper justification, e.g., if the features and services provided might not fully meet the use case's requirements. These issues could require re-selection, reconfiguration, or redeployment, introducing additional costs and overheads.

In our model, we utilised the technical debt metaphor to assess the security issues that are rooted in the oracle platforms and can affect the whole smart contract application. Security technical debt has been shown to be an effective method for estimating the consequences to the business value of the vulnerabilities that are being exploited [167]. Steep

accumulation of interest can be avoided if the debt is analysed at an early design stage [5]. Leveraging technical debt in our model benefits decision-making as it quantifies the impact of security issues over time, encouraging the designers to slow down when selecting an oracle. This allows them to select a secure and optimal oracle for their smart contract applications.

### 5.3 Research approach

The number of smart contracts oracle platforms and their features has recently expanded, complicating the decision-making process when selecting them. Making optimal decisions calls for smart contract developers to recognise the existing collection of alternatives and understand their related features so that it is possible to compare and assess the various platforms. Multi-criteria decision-making (MCDM) techniques can be leveraged to rank and prioritise alternative options based on the required features and their relative importance. In general, MCDM techniques involve six steps [230] including (1) determining the objective; (2) determining various alternatives; (3) determining the criteria; (4) determining the weighting approach; (5) applying the aggregate approach; and (6) using the aggregate findings to make decisions.

MCDM processes can be executed using a variety of mathematical approaches, which are chosen based on the nature of the problem and the amount of complexity ascribed to the decision-making process. In this study, we extend the security technical debt equation proposed in our previous study [5] to estimate the security consequences and quantify the security debt interest. Our approach assists developers with recognising the security risks and uncertainties associated with each alternative and supports them in selecting a secure solution. In addition to security, we leverage Multi-Objective Optimisation (MOO) techniques to analyse the cost of deploying and executing a contract. These techniques produce

a Pareto front optimisation, in which all the solutions are Pareto optimal [57]. This suggests that no single solution is best for all purposes but rather a collection of alternatives with various trade-offs among the competing goals.

Our systematic decision model is grounded in a document analysis of an extensive review of the main sources of knowledge, i.e., the literature concerning smart contracts and oracles. To build the decision model, we examined oracle data feeds from multiple resources, including primary and secondary studies, white papers, and blogs. Expert interviews were another source of knowledge used to refine and validate the decision model. We interviewed five experts (three oracle co-founders and two domain experts). The interviews followed a semi-structured interview protocol, presented in Appendix 4, and lasted between 30 and 45 minutes. We demonstrate how the decision model can be applied using two exploratory theory-testing case studies. We support the decision model with a stability and sensitivity analysis, which checks the extent to which the final decision can be affected when altering critical factors.

## 5.4 MCDM For Blockchain Oracle Platform Selection

This section discusses the eight-step process used to formulate the blockchain oracle selection decision support model. Three of the steps (namely five, six, and seven) were based on a well-known decision support system proposed by Farshidi et al. [118]. Adapting these steps in our context is feasible as they are applied to a decision model related to the blockchain platform selection problem [119]. However, we contribute the other five steps. Detailed explanations of each step are provided in the following subsections. First, the process of extracting, organising, and arranging the information from various sources is explained in Subsection 5.4.1 Then in Subsection 5.4.2, we elaborate the aggregation equations that transform the

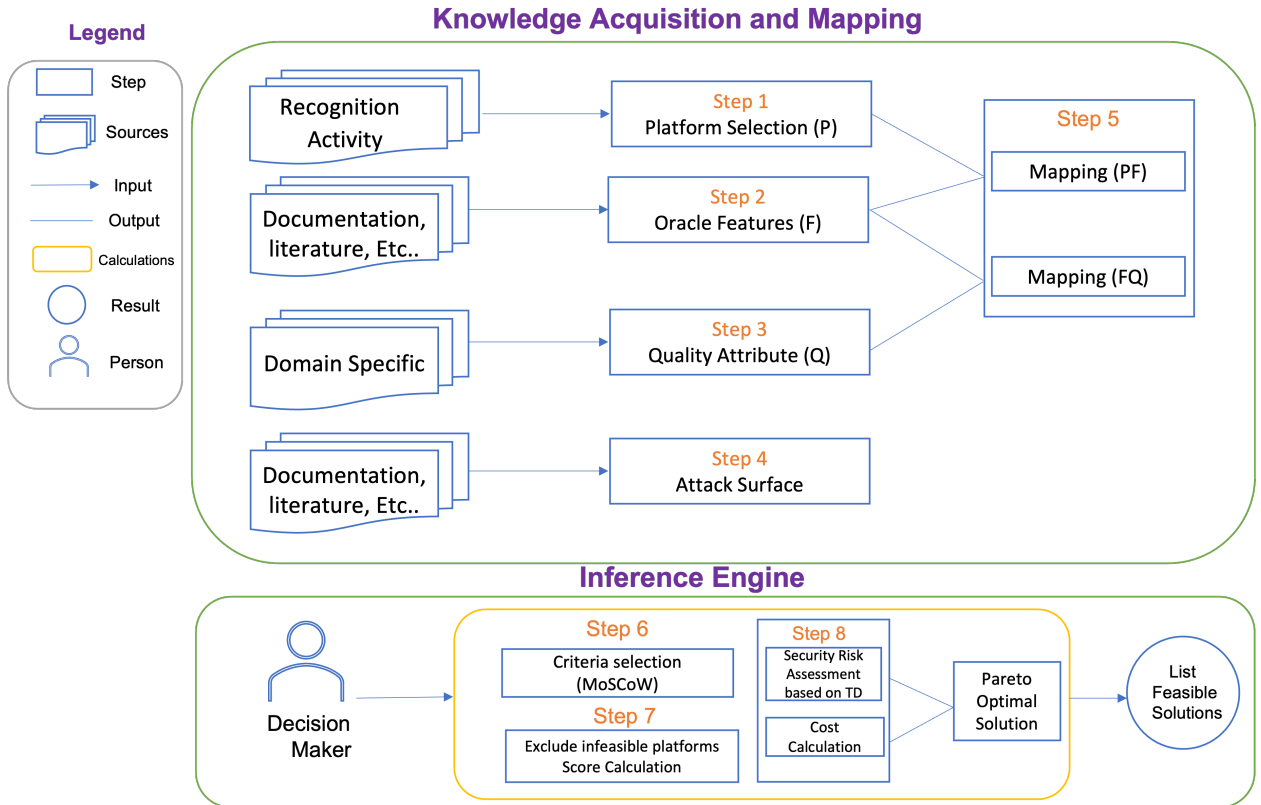


Figure 5.1: Decision Model Structure for Oracle Selection

criteria into a set of results to find the best-fit candidate oracle platforms. We also justify how the multi-objective optimisation technique can be adapted to find secure and cost-effective solutions. Figure 5.1 illustrates the main steps of our systematic model.

#### 5.4.1 Knowledge Acquisition and Mapping

Expert interviews, online documentation, and literature studies were the primary sources of information used to formulate the steps related to the knowledge acquisition and organisation phase. We investigated the literature using the following search query: '(Blockchain OR Smart Contract) AND (Oracle OR Oracles OR Decentralised Oracle OR Data feed)'. Due to the novelty of blockchain oracles and the industry's rapid growth and development, we

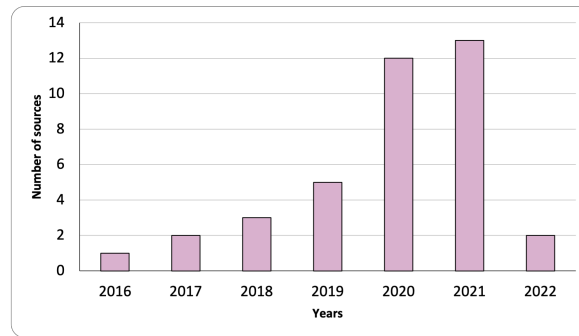


Figure 5.2: Distribution of Chosen Studies Throughout the Document Analysis Phase

could not find many academic articles investigating them. Therefore, we also considered grey literature as part of our knowledge-acquisition process. At the time of writing, around 45% of the results found are academic papers, 26% are blogs and forum articles, 18% are white papers and documentation of the platforms themselves, and 11% are collections of videos that demonstrate how some platforms are used or that present interviews with developers of some platforms explaining their features. Based on the selected sources of knowledge, we observed that the investigation of blockchain oracles has gained more attention in the past two years, as shown in Figure 5.2.

We implemented a document analysis procedure [70] for reviewing and examining the selected sources of knowledge in order to derive information and insights regarding the oracles. We created an extraction form that includes the following criteria to compile information comprehensibly: oracle platforms, oracle features, related quality attributes, security flaws, and potential attacks.

This section presents the five steps needed to acquire and organise the information.

**Step One (Platform Selection):** In addition to the extracted information from the document analysis phase, we inspected GitHub repositories to identify oracle alternatives. At the time of writing, we have identified 22 oracle platforms. All platforms and their sources are publicly available [4]. Popularity is a key factor that we used to measure the suitability

of these platforms for inclusion in our decision model. Thus, two metrics are used to assess the popularity of each oracle platform: *recognition* and *activity*.

*Recognition* refers to the degree of acceptance of the oracle platform by developers and others. The recognition indicators that we used are the number of forks, the total number of commits, and the number of stars in GitHub. GitHub forks denote the number of people who have participated (or want to participate) in the oracle; the total number of commits in GitHub denotes the development rate of the oracle; and the number of stars in GitHub denotes the number of developers who like the oracle.

*Activity* tracks the extent to which developers and others use a particular platform. The oracle is no longer popular when developers stop updating and enhancing it, or people stop talking about it. As an indicator of activity, we use the number of commits in GitHub in the previous month to express the development of the oracle platform since the previous month. Additionally, we use the Keywords Everywhere tool [178] to find the monthly search volume of the GitHub page and the main web page of the oracle platform if it exists. We have made the indicator values of each metric public for interested readers [4].

Since this study is primarily concerned with smart contracts that run on the Ethereum blockchain, we only considered the oracle platforms compatible with Ethereum. As a result, based on *recognition*, *activity metrics*, and *compatibility with Ethereum*, we ended up with eight oracle platforms which are: (1) Provable Things [282], (2) Town Crier[397], (3) Chainlink [107], (4) Witnet [274], (5) BandChain [26], (6) Paralink [268], (7) Tellor [328], and (8) iExec [164].

**Step Two (Features Identification):** We performed an iterative content analysis method [46] to organise the information extracted from the interviews and the document analysis into categories related to oracle features. These categories continued to evolve as new concepts were identified in the sources. The insights from the sources allowed us

to identify 37 features, which were then categorised into 13 categories. Our sources and extraction forms are publicly available [4]. Even though our model includes a large number of features - 37 features - it allows decision-makers to adjust the selection to their own needs, allowing analysts to focus on the most important features that have the potential to meet the quality attributes of their blockchain-based application.

We briefly explain each category and its related features to prevent semantic mismatches throughout the oracle platform selection process.

1. **Type of Oracle** refers to the type of entities that provide data from an external source to a smart contract. An oracle can be either centralised, semi-decentralised, or fully-decentralised. A centralised oracle relies on a central authority with complete control over the data being provided to the smart contracts. A semi-decentralised oracle relies on a decentralised network of nodes, or a predetermined number of nodes in some oracle, to provide data. Still, they also rely on a central authority or centralised infrastructure. A fully-decentralised oracle relies on an undetermined number of decentralised nodes to provide data, and it does not depend on any central authority to control or manipulate the data.
2. **Type of Data Source** refers to the data sources that oracles use to collect the data they send to smart contracts. A single-source oracle uses just one data source, while a multi-source oracle uses numerous data sources.
3. **Data Validation Mechanism** refers to the process through which oracles verify that the data submitted to the contract is accurate. Some oracles utilise a consensus-based solution for data validation. Other platforms rely on a third-party approach and assume the data source is trustworthy. A trusted execution environment is also used by a set of oracles to ensure that data is processed and protected in a secure environment.
4. **Integration Methods** refer to the mechanisms for connecting an oracle to a blockchain

to provide data. Delivering the data to the contract can be completely off-chain, on-chain, or hybrid.

5. **Encryption Methods** are used to ensure confidentiality when data is sent from external data sources to the oracles.
6. **Data Feeders Selection Method** ensures that the correct data is sent to the blockchain by only including legitimate data feeders and excluding malicious ones. This technique can be collateral-based (e.g., staking) or reputation-based. In the reputation-based approach, a reputation contract is generated to track the accuracy of the data provided by each feeder. This helps in picking the best data feeders from all those available.
7. **Aggregation Mechanism** combines several data inputs into a single output. The output quality is determined by the data feeders chosen and by the aggregate method employed. Mean, median, and mode are the three most important aggregation statistics. Weights can be used in the computation process to enhance the quality of simple statistics such as the median and mode.
8. **Dispute Resolution** is designed to ensure the quality of the result and to provide stakeholders with an opportunity to correct potential errors. Several resolution mechanisms are available such as staking, voting and statistical approaches. Staking-based resolution incentivises the discovery of incorrect outcomes. In the case of a voting-based resolution, any token holder can vote on disputed data, and the decision is not limited to the data feeders. In a statistical approach, the median or the mode of the values is calculated from a set of values proposed by different feeders. After the data has been aggregated, statistical methods can be applied automatically and almost instantaneously. Voting, on the other hand, involves human judgement, which may result in more accurate results in complex situations.
9. **Incorrect data** can be either corrected or reverted. The new, uncontested value will



be reflected in the corrected data. Reversion means the result will be nullified, and the system will have to start over to obtain a new outcome.

10. **Incentive Schemes** are intended to encourage feeder nodes to be truthful and provide correct results to maximise their profits.
11. **Punishment Methods**, on the other hand, apply appropriate punishments to feeders if the data they provide is shown to be incorrect. These may include banning, slashing, or suffering a reputation loss.
12. **Native Tokens** are included with most oracle platforms. For example, Tellor has the TRB token, whereas other solutions, such as Provable, have not introduced a new token.
13. **Using a Native Token** in some oracle platforms such as Chainlink requires users to use their token to request external data, while in the case of other platforms such as Witnet, users are allowed to use Ether for the requesting transaction.

**Step Three (Attack Surface):** In addition to the information acquired through the document analysis and the extracted information from the expert interviews, we use the categories that we developed in Ahmadjee et. al [6] in which attacks that targeted centralised and decentralised blockchain oracles were identified and classified. We use this classification to categorise the attack surfaces. We have identified 12 distinct attacks that might target the oracle. Table 5.1 offers a brief explanation of each attack, and shows which type of oracle each of them targets. A set of attacks, such as front-running attacks [54], that target both centralised and decentralised oracles. Other attacks, such as Sybil attacks [266], are only applicable when a decentralised oracle is in use, while other attacks, such as flash loan attacks [54] and Software Guard Extensions (SGX) attacks [400], target centralised oracles. A 'flash loan' is a transaction in which a borrower takes out a loan without collateral, uses

the money to complete specific tasks, and then repays the initial loan at the end of the transaction. Otherwise, the transaction fails if the original loan is not repaid at the end of the transaction.

It is important to note that some of the attacks, such as MITM attacks and data manipulation attacks, might also target decentralised blockchain oracles. However, the decentralised nature of this type of oracle offers significant advantages in mitigating the impact of these attacks and making them less effective. Decentralised oracles often rely on consensus mechanisms, where multiple nodes must agree on the validity of data before it is accepted by the smart contract. If one node attempts to manipulate data, it is likely to be flagged as inconsistent with the data from other reputable sources. This distributed consensus makes it extremely difficult for an attacker to manipulate data without controlling a significant portion of the oracle network, making the attack less effective. Decentralised oracles also often aggregate data from multiple sources to provide a more accurate and reliable result. This aggregation process further reduces the impact of a MITM attack, as the manipulated data from one source would be diluted by the data from other trustworthy sources. Therefore, due to the inherent security features and architecture of decentralised oracles, which provide several layers of protection against these types of attacks, decentralised oracles are not considered prone to them in Table 5.1.

**Step Four (Quality Attributes):** As this study intends to assist in selecting secure and cost-efficient oracle platforms, we identified a set of security quality attributes that relate to blockchain oracles using the following sources: (i) the extracted form that was derived in the document analysis phase, where several quality attributes related to the blockchain oracle were determined; (ii) the NISTIR 8202 report [380] that presents information about blockchain technology; and (iii) the ISO/TR 23455:2019 document [2] that provides a detailed overview about smart contracts. The following security attributes were found in these sources: confidentiality, integrity, availability, non-repudiation, authenticity,

and transparency. In addition to the security attributes, other criteria, such as cost efficiency, latency, and accessibility are also considered. The oracles' features were analysed based on their impact on the quality attributes that had been determined.

**Step Five (Mapping):** This step concerns performing the mapping between the set of Features (F) and the set of Platforms (P), and between the set of Qualities (Q) and the set of Features (F) during the knowledge acquisition and organisation stage. To apply the mapping, let  $P = \{p1, p2, \dots, p|Platforms|\}$  be a set of blockchain oracle platforms in the market (i.e., Chainlink and BandChain) and  $F = \{f1, f2, \dots, f|Features|\}$  be a set of features (i.e., Integration Methods and Incentive Schemes) of the oracle platforms. Each  $p \in P$  supports a subset of features in the set F. The main objective is to select the most secure and cost-effective oracle platform p that supports required oracle features.

According to the argument put forward by Farshidi et al. [118], the main sources of knowledge in this step could be the documentation of alternatives, literature studies, and experts' knowledge. Thus, we determined the (PF) mapping between the sets of Platforms and Features based on the information extracted from the sources of knowledge and expert interviews. PF mapping is accomplished using a Boolean adjacency matrix ( $Feature \times Platform \rightarrow \{0, 1\}$ ).  $PF(p, f) = 0$  means that the oracle platform p does not support the oracle feature f, whereas  $PF(p, f) = 1$  means that the platform employs the feature. For example, Provable, Town Crier, and Chainlink support the Encryption Method feature as they can apply asymmetric encryption to the transmitted data, while the other platforms do not. Figure 5.3.a represents a sample of PF mapping. Appendix 5 demonstrates the full mapping.

The (FQ) mapping between the set of Features and the set of Qualities is determined based on the extracted information form derived from the selected sources of knowledge, expert interviews, and the reviewers' expertise. Their expertise is assessed in the fields of

Boolean Oracle Criteria and Platforms	Provable	Town-Crier	Chainlink	Witnet	Tellor	Paralink	BandChain	iExec
<b>Type of Data Feeder</b>								
Centralised	1	1	0	0	0	0	0	0
Semi-decentralised	0	0	1	1	0	1	1	0
Fully-Decentralised	0	0	0	0	1	0	0	1
<b>Type of Data Source</b>								
Single	1	1	1	1	1	1	1	1
Multiple	0	1	1	1	1	1	1	1
<b>Encryption Method</b>								
Symmetric cryptography	0	0	0	0	0	0	0	0
Asymmetric cryptography	1	1	1	0	0	0	0	1

a. PF Mapping

Boolean Oracle Criteria and Attributes	Confidentiality	Integrity	Availability	Non-repudiation	Authenticity	Trustlessness	Transparency	Low latency	Low cost	Low transactional financial Cost	Accessibility
<b>Type of Data Feeder</b>											
Centralised	0	0	0	0	0	0	0	1	0	0	0
Semi-decentralised	0	0	1	0	0	0	0	0	0	0	0
Fully-Decentralised	0	1	1	0	0	1	0	0	0	0	0
<b>Type of Data Source</b>											
Single	0	0	0	0	0	0	0	1	0	0	0
Multiple	0	1	1	0	0	1	0	0	0	0	0
<b>Integration Methods</b>											
On-chain	0	1	1	0	0	0	1	0	0	0	0
Off-chain	0	0	0	0	0	0	0	0	1	0	0
Hybrid	0	1	0	0	0	1	1	0	1	0	0

b. FQ Mapping

Figure 5.3: A Sample of the Mapping Between: **a.** the Boolean Oracle Features and Platforms and **b.** the Boolean Oracle Features and Quality Attributes

software architecture evaluation, quality attribute and trade-off analysis, blockchain, and security. A Boolean adjacency matrix is used in FQ mapping ( $Quality \times Feature \rightarrow Boolean$ ) which means that each feature either provides or does not provide a specific quality attribute. To reduce inherent biases in the mapping process, there are two reviewers that worked separately to map the qualities with the related features. The first reviewer is a PhD student with expertise in the area of cybersecurity and software engineering. The other is a university professor with extensive industrial experience (11 years) who works in the area of cloud software engineering. Using Equation 5.1, we calculated Cohen's Kappa ( $k$ ) [320], which is a statistical approach to assess the agreement between two reviewers deciding on the same issue, as follows:

$$k = (P_o - P_e)/(1 - P_e), \quad (5.1)$$

where  $P_o$  is the probability of the observed agreement, and  $P_e$  is the probability of random agreement. Based on our two different mappings, we got a  $P_o$  of 0.96, and a  $P_e$  of 0.76 which produced a  $k$  of 0.83 indicating near-perfect agreement. Finally, three reviewers, including a professor who is an expert in distributed and autonomous software engineering

discussed the overall mapping and resolved any disagreements. Another professor, who created several highly influential methods and tools for architecture analysis used in the software industry worldwide provided feedback. We have used Cohen's Kappa to check for disagreements and discrepancies in scores to elevate subjectivity and strive for consistency. The FQ mapping does not affect the final results, but they do assist in the qualitative assessment and profiling of the alternative solutions under consideration.

Figure 5.3.b shows a sample of a final FQ mapping. Zero has been assigned if the oracle feature is prone to security threats that would likely have an effect on the quality attribute(s) that are of interest. Otherwise, a value of one has been assigned if the likelihood of threat is not evidenced in the review of the source of knowledge or further discussions on the case. For instance, a centralised oracle introduces a single point of failure threat as it is prone to denial of service attacks, which might affect oracle availability [113]. A decentralised oracle resolves these issues as it contains several redundant oracle servers, which leads to better availability. Additionally, as the Figure shows, the features provided by oracles can support developers in acquiring many quality attributes that are of interest to them. For example, the on-chain integration method, as an oracle feature, supports several quality attributes such as integrity, availability, and transparency. Appendix 5 demonstrates the full mapping.

### 5.4.2 Inference Engine

This Section describes an inference engine that makes a secure and cost-effective optimal decision by computing the score of the feasible oracle platforms and excluding the infeasible ones; it ultimately recommends a ranked shortlist of feasible options. Steps six, seven, and eight are involved with the inference engine.

**Step Six (Weighting Method):** This step aims to specify the importance of

Table 5.1: Explanation of Attacks that Target Centralised (C) and/or Decentralised (DC) Oracle Types

<b>Attack</b>	<b>Explanation</b>	<b>Oracle Type</b>
Flash Loan Attack	An exploitation of a smart contract's security in which an attacker borrows a large sum of money without requiring collateral and then manipulates a crypto asset's price on one market before immediately reselling it on another.	C
Man-in-the-middle	A malicious actor intercepts the communication between the oracles and the contract and alters or falsifies the data.	C
Denial of Service	Attackers flood the data source with requests to slow it down or they prevent other users from using oracle's server by congesting the whole network.	C
SGX Attack	Executed by extracting attestation keys signed by Intel from SGX's private quoting enclave. The attacker can impersonate real Intel machines by signing arbitrary SGX attestation quotes.	C
Data Manipulation	When the data that the oracle collects has been tampered with at the data source.	C
Front Running	The transparency of data collected by oracles is used for personal gain.	C and DC
Malfunctions	Circumstances in which oracles offer skewed data despite the source being reliable and trustworthy.	C and DC
Sybil Attack	The Sybil node manipulates the decentralised oracle platforms to obtain the most votes by replicating their votes to gain a higher weight when compared to others.	DC

*Continued on next page*

Table 5.1 Explanation of Attacks (*Continued from previous page*)

<b>Attack</b>	<b>Explanation</b>	<b>Oracle Type</b>
Mirroring	To assure the lowest cost of data gathering and the best possibility of selling the data to the client platform, the Sybil node gathers the data once and then distributes it to other nodes under its control.	DC
Freeloading	When a malicious entity duplicates data retrieved by another entity without making an effort to obtain the data itself.	DC
Spam Attack	An attack that drains the data feeder's balance with high gas fees.	DC
Sidechain Oracle Attack	An attacker can take control of oracle nodes in a sidechain to manipulate the data the oracles report and to change the outcome of smart contracts on the sidechain.	DC

each feature. Similar to Farshidi et al. [118], the MoSCoW [67] prioritisation technique is utilised to assign priority weights (W) to the oracle features. We employed this technique due to its simplicity, suitability for medium-to-large features and lack of complexity [179] [341]. WMoSCoW refers to four categories: wMust\_have, wShould\_have, wCould\_have, wWon't\_have where  $\forall w \in WMoSCoW; 3 \geq w \geq 0$ . We assigned a specific score to each category to be able to rank platforms based on numerical values. Thus, must-have and won't-have oracle features are given priority weights of 3 and 0, respectively, and act as hard constraints, while the features with should-have and could-have priorities are weighted 2 and 1, respectively, and act as soft constraints. A potential oracle platform must support all oracle features with must-have priorities, but not the features with won't-have priorities. For example, if the decision-maker assigns must-have priority weight to the decentralisation feature, would mean that Provable Things and Town Crier, as centralised oracles, would be excluded from the ranked shortlist of potential platforms.

**Step Seven (Score Calculation):** The possible oracle alternatives are ranked by the inference engine based on their estimated scores using Equations 5.2 and 5.3.

- The Weight Sum Model (WSM) [385] equation is applied to each supported oracle alternative to prioritise the oracle platforms based on the selected features. It is formulated as follows:

$$\sum_{i=1}^n W_i * Score_{PF(i)}, \quad (5.2)$$

where  $n$  denotes the number of features,  $W_i$  represents the weight assigned to the feature, and  $Score_{PF(i)}$  represents the Boolean score of mapping the Features and Platforms set (PF).

*Note:* If the resulting ranked list is large, the decision-maker can select certain threshold platforms and then apply the equation 5.3 to them.



**Step Eight (STDI Values and Monetary Cost Estimation):**

- The Security Technical Debts Interest (STDI) equation, proposed by Ahmadjee et. al [5], is adapted to estimate the security consequences of each oracle alternative. It is formulated as follows:

$$STDI = CL * CAL * SR, \quad (5.3)$$

where **Contract Lifespan (CL)** refers to the time, measured in days, between the contract's deployment and its last execution. The followings are the durations and corresponding weights: (i) Short-lived, 1-266 days, 0.17 score; (ii) Medium-lived, 267-533 days, 0.35 score; (iii) Long-lived, 534-800+ days, 0.5 scores.

**Contract Activity Level (CAL)** refers to the expected number of active users and the number of transactions that a contract is expected to deal with. A six-point scale is used to determine the activity level.

The **Security Risk Value (SR)** is calculated as follows:

$$SR = Likelihood * TI * BI, \quad (5.4)$$

where **Likelihood** is the mean of the attack likelihood factors, based on the OWASP [267] risk rating methodology.

**Technical Impact (TI)** is the mean of the TI factors, based on the OWASP [267] risk rating methodology.

**Business Impact (BI)** is the mean of the BI factors, based on the OWASP [267] risk rating methodology.

*Note:* In the context of security, the debt interest value describes the negative consequences of exploiting a vulnerability. The longer the exploitable design flaw goes unresolved

in the deployed contract, the higher the risk of interest accruing as a result of the flaw.

We propose a public framework that uses the aforementioned equations to help decision-makers quantify, estimate, and prioritise oracle alternatives.

Finally, by adapting multi-objective optimisation to apply cost-value trade-offs, the inference engine offers secure and cost-effective optimal solutions. The application of multi-objective optimisation techniques supports the selection of a set of secure and cost-effective solutions rather than one solution, as there is no unique option that can be the best fit with respect to security and low-cost objectives. The identified set of results is known as a Pareto-optimal solution [59]. Simple non-dominated sorting algorithms can be utilised to implement a vector ranking method that detects and indicates the elitism of each solution within a set of alternatives and returns a set of non-dominated solutions.

To clarify the application of the MOO technique and the related algorithms, we executed price feed smart contracts that fetch the Ethereum price from external sources and feed it to the contracts using several blockchain oracle platforms. This implementation allows us to recognise and compare the costs of deployment transactions and calling oracle transactions. The costs are calculated by adding the contract's deployment cost to the cost of a transaction that requests external data from an oracle. The cost required for deployment depends on the size of the smart contract and the part of the code that needs to be executed before the creation of the contract. The more complex the contract is, the higher the required cost. Interested readers can find a detailed explanation of the deployment cost in [5].

The cost of the oracle transaction mostly depends on the data validation mechanism's features and the number of data feeders participating in the validation process. Using a decentralised oracle requires two transactions to request data from an external source: the first transaction is responsible for sending tokens to the contract to be able to successfully implement the second transaction, which is requesting the price through the oracle. Table

5.2 shows the cost prices, in ETH, that are required to integrate each oracle alternative. For the sake of visualising and discussing the data, we have converted the costs to dollars after adding the cost of deployment to the cost of requesting data for each oracle alternative.

The values are the results of the STDI equation (Equation 5.3) for each oracle platform. We estimated the STDI of each attack identified in step three, and the results are presented in Table 5.3. The likelihood and the impact factors were estimated based on the smart contract context and on the facts presented in the sources of knowledge that we used in previous steps. If the attacks have been successful, the corresponding likelihood estimation scores are higher than for unsuccessful attacks. Also, the quantified impact scores are higher if the attacks lead to catastrophic consequences. For instance, in November 2020, Cheese Bank was attacked for \$3.3 million as a result of flash-loan attacks [273]. Other attacks such as mirroring [329] and freeloading [329] have not yet been successful. However, if these attacks do take place, the integrity of the entire system might be affected. We assume that the price feed contract is long-lived and that the contract is highly active. The explanations and scores for each factor are presented in detail in our public framework [4].

We added estimated values for the attacks that targeted centralised oracles and the values for the attacks that targeted decentralised oracles. For Chainlink, in addition to the total of STDI values for the decentralised oracle, we added the estimated value of SGX attacks because Chainlink supports SGX in its implementation. Sidechain oracle attacks mostly target BandChain as it uses the sidechain mechanism to provide cross-chain compatibility. Thus, for BandChain the estimated value of sidechain attacks was added to the total of STDI values for the decentralised oracle. The costs in dollars and STDI values of six blockchain oracle platforms are presented in Table 5.4.

To find the Pareto-optimal solutions, we need to select the solutions with minimum cost and minimum STDI value. If we compare Provable and Tellor, we notice that while

Table 5.2: Deployment and Transactional (tx.) Cost of Integrating each Oracle Platform

Oracle Platforms	Deployment Cost	Tx Execution Cost
Provable	0.00161819 ETH	0.000299591 ETH
ChainLink	0.001632422 ETH	0.000306035 ETH
Witnet	0.003937 ETH	0.0075315 ETH
Tellor	0.002345111 ETH	0.00045238 ETH
BandChain	0.0052231 ETH	0.000400999 ETH
iExec	0.0072231 ETH	0.000372546 ETH

Provable is better from the cost perspective, Tellor is better from the security angle. However, if we compare Tellor with Witnet and iExec, we notice that Tellor is equal to them in STDI values, but better than both of them in terms of cost. Tellor is better than BandChain in terms of both security and cost. Hence, Tellor dominates Witnet, BandChain, and iExec. Similarly, Provable dominates Chainlink as it is better in terms of both security and cost. Therefore, Tellor and Provable form a non-dominated set, and they are Pareto-optimal solutions in the price feed smart contract example.

The inference engine finally offers a shortlist of feasible oracle platforms based on selected features, security assessment, and monetary cost analysis. However, decision-makers can apply MOO analysis to perform further investigations, such as performance vs cost, to find the best fitting blockchain platform for their application.

## 5.5 Application of the model using two case studies

Two case studies of non-trivial scale were selected. The first describes the case of dynamic legal agreements [293], which are smart contracts that can adapt and respond to unpredictable events and which were used during the COVID era (described in Subsection 5.5.1).

Table 5.3: STDI Quantification of Attacks for Price Feeds Smart Contract

Type of Oracle	Attacks	Likelihood	TI	BI	CAL	CLS	STDI
Centralised	Data Manipulation	0.58125	0.475	0.45	5	0.5	1.344140625
	Flash Loan Attacks	0.46875	0.475	0.4875	5	0.5	1.127929688
	Man-in-the-middle	0.3375	0.65	0.43125	5	0.5	0.912304688
	Denial of Service	0.3375	0.625	0.45	5	0.5	0.90703125
	Front Running	0.46875	0.375	0.35625	5	0.5	0.856933594
	Malfunctions	0.525	0.225	0.4125	5	0.5	0.83671875
	SGX Attacks	0.39375	0.6	0.39375	5	0.5	0.978222656
Dcentralised	Denial of Service	0.28125	0.475	0.13125	5	0.5	0.426269531
	Front Running	0.46875	0.425	0.35625	5	0.5	0.915527344
	Sybil Attacks	0.28125	0.425	0.4125	5	0.5	0.588867188
	Mirroring	0.28125	0.475	0.4125	5	0.5	0.588867188
	Freeloading	0.28125	0.475	0.4125	5	0.5	0.624023438
	Malfunctions	0.525	0.325	0.4125	5	0.5	0.96796875
	Spam attacks	0.4875	0.525	0.3	5	0.5	1.00546875
	Sidechain oracle attack	0.5438	0.7	0.3938	5	0.5	1.4868164

Table 5.4: Final Cost and Security Value of Oracle Platforms

Oracle Platforms	Cost	Value (STDI)
Provable	\$8.37	5.941113281
ChainLink	\$8.44	6.09521484375
Witnet	\$45.87	5.1169921875
Tellor	\$11.19	5.1169921875
BandChain	\$22.49	6.603808594
iExec	\$30.38	5.1169921875

The second case concerns decentralised auction applications [264] that capture the interactions between auctioneers and bidders (described in Subsection 5.5.2). Both cases leverage blockchain oracles, where the decision on selecting oracles is of paramount importance for dependable operations. In our analyses of these cases, we use the guidelines recommended by Per and Martin [296] and the framework proposed by Ebneyamini et al. [105] to ensure clarity in the exposition and potential for replication and reproducibility, as described below:

1. **Objectives of the case studies:** Our objective for presenting the case studies is to show how the blockchain oracle decision model can be applied in practice and to examine the applicability of the model in more than one case study. This allows us to ensure the usefulness of the proposed decision model for the oracle selection problem.
2. **Reasons for the use of case studies:** We intentionally selected two smart contract projects as case studies as both will need to employ oracles that are suitable and effective fits for their applications. Even though the first case study employed an initial oracle platform, as the authors stated in the documentation, they are looking to integrate another oracle platform to improve the application. The second case study's developers, however, have not decided which oracle platform to employ. Therefore, applying our decision model could assist in the selection of a feasible, secure, and cost-effective oracle platform for both applications based on the features that the developers state in the documentation.
3. **Methods of gathering data:** We searched for multiple sources such as documents, white papers, academic papers, and repositories. These data sources allowed us to learn which oracle features are desirable and which specifications are required for each case study application. This is known as the third-degree method with regards to data collection [296], where available data is analysed and compiled independently without direct interaction with subjects. For the first case study, we were able to find several

sources for data collection [293, 349, 350, 348], while for the second, we were only able to find one source [264].

4. **Data analysis:** Before we started the analysis process, we compiled the available data and prepared a template in an excel sheet where the oracle features could be filled in based on the extracted data. The analysis was done by one reviewer and reviewed by the other reviewers. For both applications, we assume the blockchain is the best fitting technology because we are only interested in the selection and integration of oracle platforms. The oracle feature requirements were specified for both applications according to the MoSCoW priorities as shown in Table 5.5. Then, the inference engine recommended feasible solutions based on feature priorities and Pareto-optimal solutions. Finally, we compared our decision model outcomes with outcomes using ad-hoc methods.

The remainder of this Section describes the applications in both case studies and discusses the outcome of applying the decision model.

### 5.5.1 Application one: Dynamic Legal Agreements (DLA)

A dynamic legal agreement [293] is a smart contract that can adapt and respond to an unpredictable event with severe consequences, such as the COVID-19 pandemic. The contract needs to employ an oracle to fetch the required data regarding the event, such as the rates of virus infection and transmission, from external sources and feed it to the contract. Therefore, the agreement can be adapted based on the provided data. An example might include a service level agreement relying on the supply of goods being limited due to the closure of a production line. This application is implemented using the Provable oracle platform. However, as mentioned above, the developers are looking to integrate another oracle plat-

form to enhance the application. Thus, applying our decision model assists in the selection of a feasible, secure, and cost-effective oracle platform for the DLA contract based on the features that the authors state in the documentation.

The DLA application was developed by the CTO-in-Residence at the UCL School of Management and a business analyst, who collaboratively contributed to providing contractual certainty in an uncertain time. They wrote a discussion paper [293] that was published by the UCL Centre for Blockchain Technologies and participated in a hackathon [350, 349] where all their code was made public in online repositories [348].

The following are some of the expected application functionalities and related oracle feature requirements extracted from the available sources. The MoSCoW technique is utilised to assign priority weights (W) to the desirable oracle features.

1. Since the developers of the DLA application wanted to update the application and employ multiple entities to provide data, integrity, trustlessness and availability of the fetched data needs to be ensured. A *fully-decentralised oracle* is a suitable type, and it is considered a must-have feature. A *semi-decentralised oracle* is prioritised as could-have, while won't-have priority weight is assigned to a *centralised oracle*.
2. The developers aim to get the most accurate data for the contracts to use. Integrity, therefore, must be ensured. This type of application requires retrieved data to be trustless and available as the data needs to be sourced from multiple trusted sources. Therefore, *multiple sources* need to be leveraged and prioritised as a must-have. A *single source* is considered a could-have feature.
3. The developers' intention is to avoid centralisation in the updated application to better meet qualities like security, availability, and fault tolerance. Hence, a *trusted third-party* is considered a won't-have feature as it introduces centralisation to the application. A *consensus-based solution* is a must-have feature as it is more applicable to decentralised applications. A *trusted execution environment* can be utilised if the data needs to be processed



in a protected environment. This feature is therefore considered a could-have.

4. The developers aim to provide a means of transparency in the application. Thus, *on-chain* integration or *hybrid* integration are the suitable integration methods and are prioritised as should-have, while off-chain integration is prioritised as won't-have.

5. The application does not have strict requirements for data confidentiality and sensitivity. Therefore, both *symmetric* and *asymmetric encryption* methods are prioritised as could-have oracle features.

6. Since the integrity and correction of the data are important in DLA applications, *staking* and *reputation* can be leveraged as selection methods for data feeders, and they are both considered must-have features. The remaining features, such as *PoW*, *PoCo*, *random*, and *pseudo-random*, can be considered should-have features as they will increase the integrity of the data even if they lead to transaction latency because the speed of transactions in the DLA application is not a crucial issue.

7. The *median aggregation mechanism* is a should-have feature in the DLA application as most of the required data is numerical, such as the number of infections. However, some non-numerical data, such as time periods for restrictions to be started, might also be required, and *mode* is more suitable for use on non-numerical data. Therefore, applying the *statistical approaches* and letting the data requesters select a suitable method based on the nature of the external data is more appropriate. Thus, *statistical approaches* are also a should-have feature, and the rest of the aggregation methods are considered could-have features.

8. To ensure the quality of the result, *dispute resolution* based on *staking* is a more feasible solution as it enforces the feeders to behave honestly. Otherwise, they could suffer economic loss by having their stake slashed. Therefore, *staking* and *slashing* are should-have features as dispute resolution and punishment methods, respectively. Additionally, a *statistical approach* can be employed in combination with staking by slashing the collateral of the feeder if the data deviates from the median by some threshold. Data feeders can be punished not only by losing their collateral tokens, but also by reducing their reputation rank on the rep-

utation registry. The loss of tokens is an immediate penalty, but the loss of reputation may influence future revenue. As a result, the *statistical approach* and *reputation loss* are could-have features for dispute resolution and punishment, respectively. *Banding* is considered a could-have feature.

9. Since the speed of the transaction is not crucial in the DLA application and the integrity of the data is important, *correction* is a should-have feature for dispute data. Data correction demands the submission of new data, followed by a decision taken on all the options. A could-have priority is assigned to the *revision* feature.

10. A *reward incentive scheme* is a should-have feature as all decentralised oracle platforms use rewards as an incentive to encourage data feeders to be honest. Similarly, the *existence of a native token* is a should-have feature as each decentralised oracle has its own token.

11. The DLA application has no strict requirements regarding which token is used when requesting external data. Thus, a could-have priority weight is assigned to both features (required and not required).

### 5.5.2 Application two: Decentralised Auctions (DA)

This decentralised auction application [264] is a blockchain-based solution that uses an Ethereum smart contract, a decentralised storage system, and oracles to capture the interactions between auctioneers and bidders. The oracle acts as an external timer that indicates the bidding start and end times of the contract. Even though the developers of the DA application have discussed and analysed it in detail in their paper, they have not specified which oracle platform will be integrated into their application. Our blockchain oracle selection decision model can overcome guesses and ad hoc practices; should it be used in similar contexts, it can provide systematic means for informing the selection of more secure and cost-effective oracle alternatives that can be integrated into the Ethereum DA application.

The auction application project was proposed by five researchers — three professors and two research associates — whose work was supported by the Research Center for Digital Supply Chain and Operations Management at Khalifa University. Although we were only able to find their published academic paper [264] regarding the auction application, we were able to extract the main oracle requirements from the paper. However, we would have preferred to have had access to multiple sources of data as this would have been more useful.

The criteria and the required oracle features that were stated by the developers of the DA application in their paper are listed below. The MoSCoW technique is utilised to assign priority weights (W) to the oracle features.

1. The main objective of the DA developers was to avoid placing trust in a single source to avert having a single point of failure. Applying a centralised oracle would reintroduce this issue to a decentralised environment. Therefore, a *fully-decentralised* oracle is a suitable type, and it is considered a must-have feature, while a *semi-decentralised* oracle and *centralised* are considered could-have and won't-have features, respectively.
2. The DA application also tries to avoid using a *single source* for its time data to increase availability, integrity, and trustlessness. Thus, *multiple sources* need to be leveraged and prioritised as a must-have. A *single source* is considered a could-have feature.
3. Since third party intermediaries need to be eliminated, a decentralised environment needs to be established in the DA application. Therefore, a *trusted third-party* is considered a won't-have feature and a *consensus-based* solution is a must-have feature as it is more appropriate for decentralised applications. A *trusted execution environment* is a could-have feature as it could be useful if the data needs to be processed in a protected environment.
4. Enhancing transparency and tractability of online auctions are the main objectives of developing the DA application. Thus, *on-chain* integration or *hybrid* integration are the suitable integration methods and are prioritised as should-haves, while *off-chain* integration is prioritised as won't-have.

5. The application does not have strict requirements for data confidentiality and sensitivity. Therefore, both *symmetric* and *asymmetric encryption* methods are prioritised as could-have oracle features.
6. Since the developers of DA intend to address the core security concerns related to data integrity in online auctions, both *staking* and *reputation* are considered must-have features that can be leveraged as selection methods for data feeders. Although applying the remaining features, such as *PoW*, *PoCo*, *random*, and *pseudo-random*, increases the integrity of retrieved data, it also increases transaction latency which the DA application requires to be low. Thus, a could-have priority is assigned to these features.
7. In the DA application, time is calculated using the Unix epoch time in Solidity, which considers time as a number. Thus, the *median* aggregation mechanism is a should-have feature in the DA application as it is more suitable for numerical values. The rest of the mechanisms are considered could have features as they might be applied if time were to be treated as a string.
8. The DA developers refer to the necessity of employing an oracle platform that rewards truth-reporting nodes while punishing malicious nodes in the network. Therefore, a *rewarded incentive scheme* is a should-have feature. *Slashing* and *reputation loss* punishment methods are also should-have features, while *banding* is a could-have feature.
9. Since transaction speeds are crucial in the DA application, *data reversion* is a should-have feature for dispute data as it is faster and less complex than *data correction*, which is considered a could-have feature.
10. To ensure the quality of the result, *dispute resolution* based on *staking* is a more feasible solution as it enforces the feeders to behave honestly. Otherwise, they could suffer economic loss by having their stake slashed. Therefore, *staking* is a should-have feature for dispute resolution. Additionally, a *statistical approach* can be employed in combination with staking by slashing the collateral of the feeder if the data deviates from the median by some threshold. Thus, the statistical approach is a could-have feature for dispute resolution.

11. The *existence of a native token* is a should-have feature, as each decentralised oracle has its own token.
12. The DLA application has no strict requirements regarding which token should be used when requesting external data. Thus, a could-have priority weight is assigned to both features (required and not required).

### 5.5.3 Results and Analysis

The blockchain oracle features were specified according to the MoSCoW priorities. Then, the inference engine of the oracle decision support model identified optimal solutions for each application. Table 5.6 shows the oracle alternatives along with their calculated scores based on WSM. The Table also indicates the Pareto-optimal solutions for each application. Although the estimation of STDI factors was based on the attack information extracted in Step 4, certain factors, including some security risk factors, Contract Lifespan and Contract Activity Level, might need to be updated based on the application context. STDI calculations for each application are presented in detail in our public framework [4]. The monetary cost of integration for each oracle is demonstrated in Table 5.3.

Based on the oracle features that we extracted from the DLA documentation, the oracle decision support model assigned the highest score to the Tellor oracle platform. Tellor supports all the must-have, and most of the should-have and could-have, oracle features. Moreover, Tellor is one of the Pareto-optimal solutions as it is more secure and provides a better cost-benefit than any of the other decentralised solutions. The second most feasible oracle platform is iExec, which has higher scores than the BandChain, Chainlink, and Paralink platforms. The Witnet platform is excluded as it does not support one of the must-have features, which is the staking method for data feeder selection. Provable and Town Crier are also excluded as they do not support more than one of the must-have features, such as

Table 5.5: Entire List of Oracle Features of the Two Case Studies

MoSCoW	Dynamic Legal Agreements	Decentralized Auctions
Must-have	Fully-decentralised oracle Multiple sources Consensus-based solution Stacking and reputation	Fully-decentralised oracle Multiple sources Consensus-based solution Stacking and reputation
Should-have	On-chain Hybrid Median aggregation mechanism Statistical measure Stacking dispute resolution Slashing punishment Data correction Rewarded incentive scheme PoW, PoCo, random, and pseudo-random Existence of a native token	On-chain Hybrid Median aggregation mechanism Rewarded incentive scheme Stacking dispute resolution Slashing punishment Data reversion Reputation loss Existence of a native token
Could-have	Single data source Semi-decentralised oracle Trusted execution environment Symmetric and asymmetric encryption Mode aggregation mechanism Mean aggregation mechanism Voting aggregation mechanism Reputation loss Banding	Single data source Semi-decentralised oracle Trusted execution environment Symmetric and asymmetric encryption PoW, PoCo, random, and pseudo-random Mode aggregation mechanism Mean aggregation mechanism Voting aggregation mechanism Statistical aggregation mechanism

*Continued on next page*

Table 5.5 Entire List of Oracle Features of the Two Case Studies (*Continued from previous page*)

MoSCoW	Dynamic Legal Agreements	Decentralized Auctions
	Statistical measure for dispute resolution Data reversion Using of a native token Using Ether	Banding Data correction Statistical measure for dispute resolution Using of a native token Using Ether
Won't-have	Centralised oracle Trusted third-party Off-chain	Centralised oracle Trusted third-party Off-chain

decentralisation and consensus data validation methods.

Regarding the Decentralised Auctions application, the oracle decision support model also assigned the highest scores to the iExec, Tellor, and BandChain oracle platforms. They support all the must-have, and most of the should-have and could-have, oracle features. However, Tellor is better than both BandChain and iExec as it provides a better cost-benefit than all other decentralised solutions. The fourth most feasible oracle platform is Chainlink, which has higher scores than Paralink platforms, which ranked as the fifth most feasible solution. The remaining three platforms, Witnet, Provable, and Town Crier, are excluded as they do not support one or more of the must-have features.

The inference engine concludes that Tellor is a feasible oracle platform for both applications, which means that this platform at least supports all oracle features with must-have priority and does not support the features with won't-have priority. Additionally, it provides better security and cost-benefit than the other platforms. The developers of both applications had not considered Tellor as a potential feasible oracle platform for their applications.

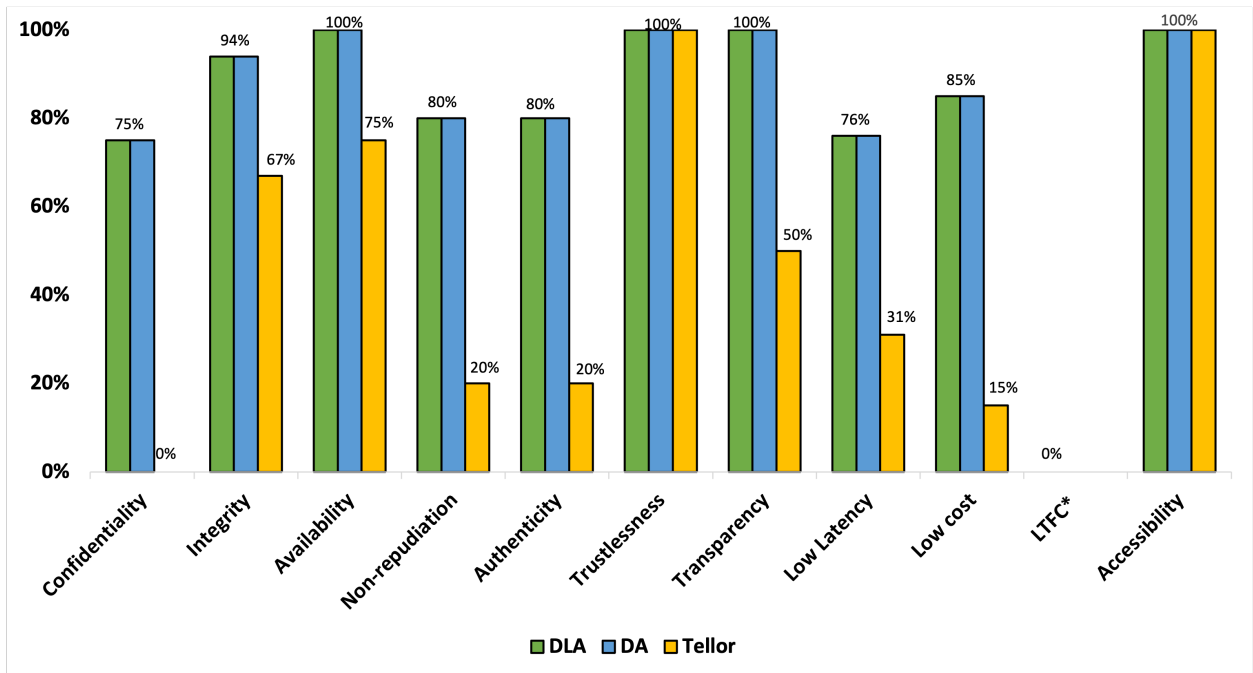


Figure 5.4: The Percentages of the Q that are Desired for each Application and the Percentages that Tellor Provides

DLA employed Provable in the current version and is considering employing Chainlink for the updated version as stated in the documentation. Although the DA developers have not integrated any oracle yet, they considered Provable, Chainlink, and Witnet as candidate platforms to be implemented in the DA application.

Figure 5.4 shows the percentages of the quality attributes based on the selected oracle features that need to be implemented in each application. Additionally, the Figure demonstrates the percentages of the quality attributes that the Tellor platform provides. Availability and integrity are the main objectives that both applications need to meet. Tellor provides high availability and integrity with 75% and 67%, respectively. Transparency is also a primary feature required in both applications and Tellor provides a reasonable percentage of 50%. Tellor affords a high percentage for trustlessness, which is one of the main aspects that needs to be provided in the DA application. Confidentiality is not provided by



Table 5.6: Inference Engine Outcomes based on Features' Priorities and Pareto-Optimal Solutions (POS)

Oracle Platforms	WSM (DLA)	POS (DLA)	WSM (DA)	POS (DA)
Tellor	<b>34</b>		<b>31</b>	
iExec	<b>33</b>		<b>32</b>	
BandChain	<b>31</b>		<b>30</b>	
Chainlink	29	Chainlink	29	Provable
Paralink	27		28	
Witnet	29	Tellor	28	Tellor
Town-Crier	8		8	
Provable	4		4	

Tellor. However, both applications do not have strict requirements for data confidentiality and sensitivity. Providing a low cost solution is the objective of the DA application, and based on our experiments, Tellor provides lower costs than most of the other decentralised alternatives. Moreover, Tellor is resistant to MITM attacks and data manipulation, which are the main concerns stated in the DA documentation. Even though Tellor is better than other platforms in terms of security, cost, and most of the other attributes, Tellor is slow as it provides a low latency of 31% because the dispute resolution step may take a long time to resolve. Therefore, developers might need to make a further trade-off analysis to compare and investigate the applicability of other platforms. The analysis can be done using our decision model framework.

Table 5.7: Sensitivity Analysis of WSM Results of DLA Application

Features	Main Scenario			Scenario 1			Scenario 2		
Data Feeders Selection Method	Weight	Oracle	Score	Weight	Oracle	Score	Weight	Oracle	Score
Stacking	<b>3</b>	Tellor	34	<b>2</b>	Tellor	31	<b>2</b>	Tellor	35
Reputation	<b>3</b>	iExec	33	<b>2</b>	iExec	29	<b>2</b>	iExec	33
PoW	<b>2</b>	BandChain	31	<b>1</b>	BandChain	28	<b>3</b>	BandChain	30
PoCo	<b>2</b>	Chainlink	29	<b>1</b>	Chainlink	27	<b>3</b>	Witnet	30
Random	<b>2</b>	Witnet	29	<b>1</b>	Witnet	26	<b>3</b>	Chainlink	27
Pseudo-random	<b>2</b>	Provable	4	<b>1</b>	Provable	4	<b>3</b>	Provable	4

## 5.6 Evaluation

In this Section, we evaluate our decision model and appraise its reliability and usefulness. We report on the sensitivity analysis, discuss the added value of our method when compared to ad-hoc selection practices, and explore how we go beyond existing work in providing an explicit oracle selection method.

### 5.6.1 Sensitivity Analysis

The sensitivity analysis [41] is important in terms of evaluating the quality and stability of our decision model and its recommendations when faced with uncertainties and changes to input values. It assists in showing the extent to which the estimations and the rankings of the results continue to be valid despite changes to inputs. For example, the values might be altered due to the uncertainty of the inputs or to test the design space. We have assessed the sensitivity of the blockchain oracle decision model by observing that alterations in the weight values in the WSM equation can be used as a means of detecting the consequences of changes to the ranking of oracle alternatives. In addition, we altered the values of some factors in the STDI equation to see if the results would change depending on the severity of the attacks.

The key to sensitivity analysis is the ability to identify the most significant assumptions that affect the output, i.e., which input variables have the strongest impact on the target variable. We considered altering the weights of those oracle categories that involve the largest number of features, which in our model is the *Data Feeders Selection Method*. The new scores were then recalculated according to the updated group weights. We applied the sensitivity analysis to the DLA application example. Table 5.7 illustrates the results of the analysis.

Sensitivity analysis is applied here by changing the priority weighting according to the MoSCoW [67] scheme using alternative scenarios. We first considered the consequences of decreasing the weight parameter by one (Scenario 1), such as altering the should-have (2) features to be could-haves (1), and then increasing the weight by one (Scenario 2), such as altering the should-have (2) features to become must-haves (3). We noticed that the final score changed for each alternative, which consequently slightly changed the ranking. However, compared with the other alternatives, Tellor still has the highest ranking. The model is sensitive when it comes to hard constraints (must-have and won't-have weights) because it affects the list of platforms included and excluded. For example, assigning a should-have weight instead of a must-have weight to the staking feature leads to Witnet being included in the suggested platforms, even though that platform should be excluded as it does not support one of the must-have features.

A One-Factor-At-a-Time (OFAT) [347] approach has been applied to evaluate the sensitivity of STDI estimation outputs against the assigned set of factor values. Five alternative scenarios (Scenario 2 to Scenario 6) were constructed, as shown in Table 5.8, to apply the analysis to. These scenarios are effective in revealing the sensitivity of the STDI estimations. In particular, we modified the values of the technical impact factors because they have a higher impact on the final estimation score compared with the two other factors, namely the likelihood and the business impact. We slightly changed the range of one technical impact factor each time by increasing or decreasing one of the values. For example, we changed the 'loss of availability' technical impact factor for flash loan attacks in Scenario 4, from 1 (minimal secondary services interrupted) to 5 (minimal primary services interrupted), which is the subsequent score values based on the OWASP [267] risk rating methodology. Then the final STDI result was recalculated based on the updated score. This analysis indicated that the STDI estimation is stable, as the final estimations of the attacks' severity range ( $1.00 \leq \text{STDI} < 1.5$ ) did not change. Additionally, we simultaneously modified the values of the

technical impact factors in Scenario 6 to observe the changes in the final estimation values. As with OFAT, the final estimations of the attacks' severity range remained unchanged.

### 5.6.2 Comparison with Previous Work

This section considers the academic papers we selected in the knowledge acquisition and organisation phase. We also applied a snowball method to ensure that all studies related to our work are included. This section's objective is to compare our decision model's novelty and effectiveness against existing work related to blockchain oracles. We compared our work to the academic literature in terms of the number of oracle features discussed in *step 2* and quality aspects covered in *step 3*, as well as the type of methods that have been used.

Table 5.9 illustrates the oracle features that are covered by each study. As shown in the Table, the majority of studies agreed concerning certain essential features such as the *type of data feeder*, *type of data source*, and *data validation mechanism*. Some features, such as *integration methods* and *encryption methods*, are only discussed in a few studies. Only one study [113], analysed *aggregation mechanisms*, *dispute resolution*, *incorrect data*, *punishment methods*, and several methods of *data feeder selection*. Our model has compiled additional features that had not been covered by any other work, including *incentive schemes* and the *existence of and use of native tokens*. These features allow decision-makers to anticipate the monetary cost of oracle integration. Additionally, only our model involves *random* and *pseudo-random* techniques as features concerning data feeder selection methods. Considering these features at the decision-making stage assists in recognising the extent to which the chosen oracle is secure [25]. Our model considers more features than other works that have discussed blockchain oracles. Even though there is no agreement on the number of features that should be involved in the decision-making process, enlarging the set of features can provide more meaningful comparisons between oracle alternatives, which may lead to the

selection of a more optimal solution. Although our model is more comprehensive regarding the number of features, it gives decision-makers a degree of flexibility when it comes to customising the selection so that analysts can focus on the key features relevant to their situation.

Unlike previous works, we have linked each oracle feature to corresponding quality attributes, as illustrated in Figure ?? and Appendix 5. Mammadzada et al. [232] discussed the confidentiality and integrity provided by some oracle features. Al-Breiki et al. [48] compared the integrity, confidentiality, authenticity, and accessibility of the oracle solutions. Lo et al. [221] investigated the reliability of seven oracles in terms of their probability of failure. Xu et al. [373] considered availability and transparency when discussing oracle features. However, none of these studies discussed the security implications of the analysed oracle platforms. Some studies explored and analysed the potential security vulnerabilities and the types of attacks that target blockchain oracles. Eskandari et al. [113] analysed oracle design options, explored potential system vulnerabilities, discussed cyberattacks, and showed some attack mitigation measures. They also mentioned examples of some real attacks that had severe consequences. Pasdar et al. [271] analysed Sybil attacks that mostly target decentralised oracles. Even though the studies discussed some attacks that targeted oracles, they failed to quantify the consequences of the security risks. In our model, we classify and assess the possible attacks based on their likelihood, possible impacts, and type of smart contract. Moreover, we are the only study that has discussed and compared the monetary cost of each oracle alternative.

To the best of our knowledge, there is a dearth of academic studies that provide a systematic approach to comparison and evaluation of oracle alternatives and which assist smart contract developers in selecting feasible solutions for their applications. Mammadzada et al. [232] conducted a systematic literature review to formulate an oracle framework that defines what blockchain oracles are, and how they relate to blockchain applications. Al-

Breiki et al. [48] investigated the trustworthiness of the multiple oracles and their system design, benefits, and drawbacks. A fault tree analysis was utilised by Lo et al. [221] to study oracle systems' dependability and to analyse their designs. Eskandari et al. [113] conducted a systematisation of knowledge study to present a classification of existing oracle implementations. Pasdar et al. [271] provided a comprehensive review and categorised oracles into two main groups: voting-based and reputation-based. Xu et al. [373] briefly discussed oracle problems by providing a simple decision model for connecting blockchain systems with the external world.

Even though the methods applied in previous works might implicitly assist in the oracle selection process, we are not aware of any systematic method that provides a step-by-step approach when it comes to the selection of a suitable oracle platform. In the following Subsection 5.6.3 we have provided analysis of the value added by our systematic model when compared to prevalent ad-hoc methods of selection.

### 5.6.3 Comparison with Ad-hoc Methods

Our comparison is based on the fact that our systematic model obliges decision-makers to consider an application's required features, cost, and security attributes. However, decision-makers can opt not to examine any of these aspects when picking an oracle platform on an ad-hoc basis. Indeed, ignoring any of these aspects can lead to a different, potentially sub-optimal, outcome. Table 5.10 shows the results of our comparison.

By applying all the steps of our model for the DA application, we found that iExec had the highest WSM scores followed by Tellor. However, Tellor was better in terms of cost. If the cost attribute is not considered, iExec might be employed despite being more expensive than Tellor. In contrast, if cost is the only factor in the decision process, Provable

Table 5.8: Sensitivity Analysis of STDI Results

Attack		Technical Impact Factors							Final Result
Flash Loan Attacks	Loss of Confidentiality	Score	Loss of Integrity	Score	Loss of Availability	Score	Loss of Accountability	Score	STDI
<b>Scenario 1</b> (main)	Minimal non-sensitive data disclosed	2	All data totally corrupt	9	Minimal secondary services interrupted	1	Possibly traceable	7	1.1279297
<b>Scenario 2</b>	Minimal critical data disclosed	6	All data totally corrupt	9	Minimal secondary services interrupted	1	Possibly traceable	7	1.12451172
<b>Scenario 3</b>	Minimal non-sensitive data disclosed	2	Extensive seriously corrupt data	7	Minimal secondary services interrupted	1	Possibly traceable	7	1.0693359
<b>Scenario 4</b>	Minimal non-sensitive data disclosed	2	All data totally corrupt	9	Minimal primary services interrupted	5	Possibly traceable	7	1.12451172
<b>Scenario 5</b>	Minimal critical data disclosed	2	All data totally corrupt	9	Minimal secondary services interrupted	1	Completely anonymous	9	1.1865234
<b>Scenario 6</b>	Minimal critical data disclosed	6	Extensive seriously corrupt data	7	Minimal primary services interrupted	5	Completely anonymous	9	1.3623046875

would be the best choice, followed by Chainlink. However, neither platform is optimal for DA applications, as Provable has several unsuitable features, and Chainlink has a high STDI value. Finally, if the security attribute is the only consideration, Witnet, Tellor, and iExec would be the best options. However, not considering the required features of the application and the cost of employing the platform might lead to the selection of a sub-optimal platform for an application such as Witnet. That platform is unsuitable for DA applications, as it is the most expensive platform and does not provide features that must be applied in DA applications.



Table 5.9: Our Work in Comparison to Previous Works

Oracle Features	[232]	[48]	[221]	[113]	[271]	[373]	This work
Type of Data Feeder	×	✓	×	✓	✓	✓	✓
Type of Data Source	✓	×	✓	✓	✓	×	✓
Data Validation Mechanism	✓	✓	✓	×	✓	×	✓
Integration Methods	✓	✓	×	×	×	×	✓
Encryption Method	✓	✓	×	×	×	×	✓
Aggregation Mechanism	×	×	×	✓	×	×	✓
Dispute Resolution	×	×	×	✓	×	×	✓
Incorrect data	×	×	×	✓	×	×	✓
Incentive scheme	×	×	×	×	×	×	✓
Punishment methods	×	×	×	✓	×	×	✓
Native Token (NT)	×	×	×	×	×	×	✓
Using NT	×	×	×	×	×	×	✓
<b>Data Feeders Selection Method</b>							
Stacking	×	×	×	✓	×	×	✓
Reputation	×	×	×	✓	✓	×	✓
PoW	×	×	×	✓	×	×	✓
PoCo	×	×	×	×	×	×	✓
Random	×	×	×	×	×	×	✓
Pseudo-random	×	×	×	×	×	×	✓

Table 5.10: Comparison of our Model and Ad-hoc Methods

<b>Our model (all steps)</b>		<b>Cost only</b>	<b>STDI only</b>	<b>WSM only</b>
<b>WSM</b>	<b>POS</b>			
<b>iExec</b>		<b>Provable</b>	<b>Witnet</b>	<b>iExec</b>
<b>Tellor</b>		<b>Chainlink</b>	<b>Tellor</b>	<b>Tellor</b>
BandChain	Provable	Tellor	<b>iExec</b>	BandChain
Chainlink		BandChain	Provable	Chainlink
Witnet	Tellor	iExec	Chainlink	Witnet
Provable		Witnet	BandChain	Provable

## 5.7 Discussion

The existence of multiple blockchain oracle platforms with varying features and criteria makes it difficult for decision-makers to select the best platform for their applications. Based on expert interviews, blockchain oracle platforms are currently chosen in an ad hoc manner, based on their reputation, or based on the wisdom of the crowd. Our decision model can overcome ad-hoc practices and assist smart contract developers in the systematic selection of feasible, more secure, and cost-effective oracle solutions that the developers may not be able to discern without the support of the proposed model. The importance of our decision model was also emphasised by blockchain oracle experts, who affirmed the usefulness of our model in facilitating transparent oracle selections by making oracle alternatives and their associated features and qualities more explicit. The model can be applied at the architectural design stage, where architects and blockchain software engineers have to integrate oracle solutions into the software being designed, by probing for features that best meet the quality attributes of interest for the said application. The systematic steps that we provide as part of our model serve this objective.

One of the distinctive features of our model is that it provides cost analysis, covering the deployment and execution of the oracles under investigation. These are particularly important factors to consider when integrating oracles with Ethereum smart contracts because the cost can differ among platforms. For example, using the Provable platform comes with a different cost compared to using Chainlink. The observation is also evidenced by previous studies, which indicate that different oracle features can lead to different costs [48, 156]. Additionally, one study [271] emphasised the need for further research that investigates the operating costs of blockchain oracles. Unlike previous studies, our study implements price feed contracts to compare and analyse the exact costs of each oracle.

Another distinctive feature of our model is the provision of a security technical debt assessment for the selection. Our technical debt analysis considers inherent issues in the oracle that can make the solution more susceptible to attacks that target blockchain oracle platforms. The evolution of oracles has led to an increase in attacks which demands a greater focus on security analysis and assessment [113, 54]. Using our model, developers can thus make informed decisions before integrating oracle platforms into the smart contract. Applying a security debt assessment step helps to reduce the introduction of unintentional debt caused by unawareness of external security issues related to oracle platforms, while increasing the visibility of such issues and helping manage the debt more strategically. Furthermore, smart contract developers can avoid debt that might accumulate as a result of incorrect oracle selection decisions that would lead to re-selection, re-development, and redeployment of the contract.

Our decision model indicates that there is no unique optimal solution that can be the best fit for all quality attributes. This should come as no surprise. The model suggests more than one feasible oracle platform based on the required features selected by the developer, the security assessment, and the cost analysis. Mapping quality attributes to each oracle feature and each oracle platform permits the identification of quality attribute trade-offs

related to the chosen features and suggested oracle solutions. This assists in choosing among the oracle alternatives while bearing in mind the desired quality attributes.

Systematic and fine-grained analysis like our decision support model can uncover issues that might be ignored when taking an ad-hoc approach or even when consulting with experts. Without the use of approaches such as our model, a sub-optimal design decision might be made. Furthermore, choosing a more optimal solution reduces the overhead of redesigning and updating the smart contract, which therefore reduces the overall cost of the contract's development.

The application of our model can benefit both uninformed and knowledgeable decision-makers. The model provides information on oracle platforms to assist inexperienced decision-makers while also providing experienced ones with a systematic model. An investigation into our model by industry would likely contribute towards a fuller understanding of the costs and benefits that its introduction at the decision-making stage would provide. According to the interview and survey by Zou et al. [409], there is a lack of standardised knowledge and useful guidance for developing secure smart contracts, and our model aims to fill that gap. Practitioners also refer to the need to define criteria and features for selecting the most appropriate smart contract components [276].

### 5.7.1 Threats to Validity

Based on Wohlin et al. [362] and Runeson and Höst [296], we have considered the following potential threats to validity: internal validity, conclusion validity, construct validity, external validity, and reliability.

A potential threat to internal validity is the completeness of the knowledge extraction steps. We mitigated this risk by conducting a thorough inspection of multiple sources,

including the online documentation for each platform, industry white papers, academic papers, and blogs. Furthermore, we refined the extracted information by conducting interviews with Oracle co-founders and experts. The landscape of oracle features and cyberattacks is continuously evolving and completeness cannot be guaranteed. Nevertheless, our work provides comprehensive and detailed coverage that goes beyond existing works, as discussed in Subsection 5.6.2. New attacks can always emerge and attackers may exploit hidden security flaws in oracles; the visibility of these attacks can then become noticeable over time. However, our decision model is adaptable and flexible to evolve and cope with new additions and changes. Researchers can add more oracle platforms, oracle features, and/or possible oracle attacks to the decision model by systematically following the five steps of knowledge acquisition and organisation that we presented.

A potential threat to conclusion validity is related to the need to involve decision-makers for blockchain oracle applications in the examination of the applicability of the decision model. However, we mitigated this threat by using public and well-documented applications for the decision model application.

There is a construct threat regarding the possibility of inaccuracy when assigning scores to the impact and likelihood factors. However, the sensitivity analysis we conducted demonstrates the stability of the STDI estimation approach. Furthermore, our framework is public [4] and allows decision-makers to change the scores based on their context. Another threat might be related to employing the MoSCoW prioritisation technique rather than another technique. Based on Farshidi et al. [118, 119], who evaluated their decision models using multiple experts and according to the comparative analysis conducted by [179], and [341], the MoSCoW technique is simple, suitable for large-medium features, easy to use, and imbues users with confidence when used. For these reasons, we chose to employ the MoSCoW technique in our model. Nevertheless, researchers can use other types of prioritisation techniques to define their feature requirements. Regarding the data collection in terms of the

case studies, there is a potential threat related to misinterpretation of the oracle features required by each application. To mitigate this, two reviewers independently reviewed the extracted features. Whenever there was a disagreement, all reviewers discussed the matter until a conclusion was reached.

There is an external threat related to the applicability of our decision model to other types of blockchain oracle platforms, i.e., not just Ethereum oracle platforms. Ethereum is, however, the most used platform for developing decentralised applications. Nevertheless, the decision model presented here can be updated to involve more types of oracle platforms without changing the main steps. Moreover, we applied the model in two domains, insurance, and auction application domains, to prove that our decision model can be generalised to other contexts.

To strengthen the reliability of our study, all the data sources and information related to knowledge acquisition and mapping have been made available. We have also been open and explicit about the calculation process used in the inference engine. Additionally, the data sources for the case study applications have been given, and the templates for the extracted features have been made available to the public [4]. This enables easy replication of the case study analysis with a high likelihood of achieving the same outcomes.

## 5.8 Summary

In this chapter, we have presented a decision support model for blockchain oracle platform selection. The selection process was modeled as an MCDM problem, with a set of solutions being evaluated and a set of choice criteria being considered. Our model provides knowledge about blockchain oracle platforms to assist inexperienced decision-makers while providing knowledgeable decision-makers with a sound decision model. In addition, our model: (1)

prioritises possible oracle solutions; (2) assesses the security of each solution using technical debt; and (3) finds Pareto-optimal solutions in terms of security and cost-effectiveness. We conducted two case studies to evaluate the usefulness and effectiveness of our decision support model. Based on the developer's required criteria, security evaluation, and cost analysis, the model provides more than one suitable oracle platform to choose from. We assessed our decision model's reliability and applicability using a sensitivity analysis and described the increased benefit of our technique when compared to previous work and ad-hoc selection procedures.

# Chapter Six

## Reflection and Appraisal

### 6.1 Overview

The purpose of this chapter is to review the research questions from Chapter 1 and see how this thesis resolved them. This chapter also explains how each contribution was evaluated.

### 6.2 Analysis of the Research Questions

This Section discusses how far previous chapters resolved the four research questions.

**RQ1: a) What are the common security architectural design approaches used when architecting blockchain-based systems and smart contracts? b) What are the existing frameworks, models, and methodologies for security risk assessment in blockchain-based systems and smart contracts?**

In Chapter 2, we conducted SLR to investigate existing blockchain architectural design approaches for security and to understand current methodologies to assess security risk in blockchain-based systems and smart contracts. The review led us to identify four commonly



used techniques that support the secure architectural design of blockchain-based systems: (i) decision models; (ii) taxonomies; (iii) design patterns; and (iv) guidelines. Additionally, we found that some of the selected studies contribute to blockchain risk identification methods, risk analysis, and risk calculation methods.

Based on the review results, we found a lack of systematic security architectural-centric approaches, security standards, and complete security risk assessment methodologies that provide the identification and quantification of security risks related to blockchain systems and smart contracts' architectural design decisions. Additionally, there is a need to investigate the security implications of design decisions regarding various aspects of smart contracts, such as programming languages and off-chain integration. Moreover, we found that systematising the process of smart contract security from the early design stage is an essential step toward designing secure smart contracts. Therefore, drawing on these pending challenges, we derived the following needs that steered investigations into the thesis: (i) a systematic approach to assist architects in making secure architecture design and configuration decisions for blockchain-based systems; (ii) an approach for assessing smart contracts security risks; and (iii) a decision support model for blockchain oracle platform selection.

**RQ2. a) What are the architecturally significant design decisions in blockchain-based systems? b) How can potential threats and attacks be traced to blockchain architectural decisions and to which components?**

In Chapter 3, we devised a taxonomy that defines, illustrates, and classifies the key architectural decisions regarding blockchain-based systems. This taxonomy is the result of an approach partially guided by an SLR. The findings of the review indicate that the dimensions of key architectural decisions are: (i) blockchain access type; (ii) data storage and transaction computation; (iii) consensus mechanism; (iv) block configuration; (v) key management; (vi) cryptographic primitives; (vii) chain structure; (viii) node architecture;

and (ix) smart contract.

We provided a mapping approach that associates architectural decisions of blockchain-based systems with potential security attacks and threats. We searched for seminal surveys, reviews, articles, and reports on this topic to help us identify commonly documented blockchain attacks and threats. We used MITRE's attack tactic [245] categories and Microsoft STRIDE threat modelling [199] to systematically classify a collected set of threats and their associated attacks. The classification allows for identifying potential attacks and threats in blockchain-based systems.

Mapping the proposed taxonomy with its security implications enables software architects to fully understand the impact and scope of the security challenges of the architecture of blockchain systems. Threat implications are directly related to the likelihood and impact of potential attacks on the entire system. This method provides a foundation for assessing potential security risks across all system dimensions.

**RQ3. a) How to identify design vulnerabilities in smart contracts? What are the specific analysis techniques and tools? b) How to quantify the impact of technical debts related to design vulnerabilities in smart contracts?**

In Chapter 4, we developed a debt-aware approach for assessing security design vulnerabilities in smart contracts. This approach utilises both automated analysis tools and manual analysis to identify potential security vulnerabilities caused by design decisions of smart contracts. We mapped the smart contracts design vulnerabilities and the related weaknesses to highlight the root cause of the issues. We classified the design vulnerabilities based on their impact, including front-running, time manipulation, and denial of services. We estimated the evolution of the negative consequences of the identified security issues in smart contracts as an analogy with the concepts of technical debt principal and interest growth rate. In particular, the approach leveraged the technical debt metaphor to estimate

the monetary cost of redeploying the patched version of the vulnerable contract and the evolution of the debt interest linked to a design vulnerability.

Experiment results demonstrated that our approach enables developers to visualise and prioritise technical debts caused by unaddressed smart contract design vulnerabilities. The approach increases the visibility of debts and their ramifications. Moreover, smart contract experts confirmed the usefulness and clarity of this approach in guiding the architectural design of secure smart contracts and recognising the consequences of potential security debts.

**RQ4. How can we advance the oracle selection problem by designing decision support models that assist in systematically selecting secure and cost-effective oracle platforms feasible for decentralised applications?**

Chapter 5 presented a decision support model for blockchain oracle platform selection. The selection process was modelled as an MCDM problem [337], which evaluates a collection of oracle alternatives while considering a set of decision criteria and quality attributes for building this software category. Our model employs the MoSCoW [67] prioritisation technique to assign the priority weights to each oracle feature. The model also leverages the security technical debt metaphor to assess the ill consequences of sub-optimal selection that can manifest into debt and accumulate interest on that debt [5]. In addition to the security debt estimation, the model also estimates the monetary cost of integrating each oracle alternative into a smart contract. The multi-objective optimisation technique is applied to choose a set of secure and cost-effective solutions. The oracle decision model finally offers a short-ranked list of feasible oracle platforms based on selected features, security assessment, and monetary cost analysis.

The oracle selection decision model overcomes ad hoc-practices and assists smart contract developers in the systematic selection of feasible, more secure, and cost-effective

oracle solutions that the developers may not have without the support of the proposed model. A guided decision model reveals issues that might otherwise go unnoticed if done haphazardly, reduces decision-making efforts, and provides a cost-effective solution.

## 6.3 Reflection on the Evaluation

In this work, we follow the DESMET evaluation method [187] to select suitable methods for evaluating our approaches. We employed hybrid evaluation methods by considering various ways to evaluate our techniques, including three case studies (electronic medical record (EMR) blockchain-based systems [99], a dynamic legal agreement (DLA) decentralised application [293], and a decentralised auctions (DA) application [264]), experiments related to assessing security debts of smart contracts, and experts' opinions in the form of surveys and interviews. Using various evaluation methods assists in mitigating the potential risk of incorrect conclusions [187].

### 6.3.1 Evaluation Criteria

In this Subsection, we reflect on our evaluation of this work by adopting the evaluation strategy and criteria used by Kitchenham et al. [187] to evaluate DESMET methods, as follows:

**Basic Validation** - According to Kitchenham et al. [187], this validation assesses the quality of the approach's documentation through several subfeatures. We found the following subfeatures suitable to our context:

- **Completeness** - Even though we cannot guarantee the completeness of the taxonomy

of blockchain architectural design decisions presented in Chapter 3, we performed an iterative content analysis by following the SLR methodology to systematically review the literature and mitigate the risk of missing critical dimensions. Nevertheless, our taxonomy is adaptable and flexible to evolve and cope with new additions and changes. Additionally, we thoroughly inspected the academic papers, the Ethereum community, Wiki pages, and developers' blogs to aggregate the set of design vulnerabilities that we provided in Chapter 4. This set is continually evolving because of the high potential for the emergence of new exploitable security design flaws in smart contracts. However, the same provided steps apply to identifying and classifying new issues. In Chapter 5, we implemented a document analysis procedure [70] for examining the selected online documentation and literature studies to formulate the knowledge acquisition and organisational steps for the proposed blockchain oracle decision model. Moreover, we consulted domain experts and oracle platforms co-founders to confirm the completeness of the extracted information. The participants affirmed that the decision model includes the key features and is general enough to assist in selecting an oracle. Nevertheless, the decision model can acquire additional oracle platforms, features, and possible attacks by systematically following the proposed steps of knowledge acquisition and organisation.

- **Organisation** - We presented our approaches systematically with well-defined steps and sub-steps. We structured and organised the information in tables and templates. For instance, tables presented the mapping of architectural design decisions with security attacks and threats. Blockchain oracle features appeared in a template that allows practitioners to assign priority to each feature based on the requirements of their decentralised applications. The source of knowledge and the methods that we used to formulate the approaches and demonstrate them are publicly available to facilitate tracing the provided information, replicating the demonstrations, and maintaining the

transparency of our studies.

**Use Validation** - As Kitchenham et al. [187] stated, this validation evaluates the quality of the techniques. We assess our approaches through the following subfeatures:

- **Ease of Implementation** - Some blockchain and smart contract software engineers require a learning curve to become familiar enough with our proposed methods to use them effectively. Learning time is necessary to comprehend some of the principles of the proposed approaches. Some practitioners, for instance, might need a deeper understanding of some security issues to assess their ramifications on blockchain systems or smart contracts. We mitigate this concern by explaining most security issues, their categories, and classifications. Additionally, some developers require time to become proficient in using our approaches. We provided clear guidelines and explications of the proposed steps to simplify their implementation.
- **Applicability** - We demonstrated and proved the applicability of our approaches through case studies and experiments. In Chapter 3, we demonstrated the mapping of the proposed taxonomy with the related security ramifications through the EMR blockchain-based system proposed by Dubovitskaya et al. [99] in collaboration with the Stony Brook University Hospital. We demonstrated the architectural decisions for the EMR system using our taxonomy. We analysed the alternative architectural choices regarding key management dimensions as we found that the currently implemented choice is sub-optimal and showed how it might affect the security of the EMR project. We used the EMR system as a case study since we found that the applicability of the taxonomy and mapping approach is observable in this type of project. Nevertheless, our future study will demonstrate the utility of the taxonomy and mapping approach to various blockchain systems to prove its applicability to any such system. In Chapter 4, we demonstrated the applicability of the smart contracts security technical debt

assessment approach through quantitative experiments since the results can be clearly quantifiable and directly measurable. The experiments worked with a dataset of 16 representatively vulnerable smart contracts. In Chapter 5, we presented two case studies—DLA application and DA application—to demonstrate the applicability of the blockchain oracle decision model. We selected these applications as case studies since the developers of the first DLA plan to change the current oracle platform to improve the application security, while the developers of the DA have not yet decided which oracle platform to employ. Thus, the proposed decision model can assist developers of both case studies in selecting suitable, secure, and cost-effective oracle platforms for their applications.

**Gain Validation** - Based on Kitchenham et al. [187], gain validation assesses the benefits delivered by the approaches through the multiple subfeatures. Our context considers the following subfeatures:

- **Usefulness** - The approaches proposed in this work provided multiple benefits for designing and evaluating secure blockchain systems and smart contracts: (i) they supported early interventions of the blockchain security issues, starting from the inception and design stages; (ii) they assisted software engineers to recognise the root causes of security issues and understand the impact and scope of the challenges associated with them; (iii) they helped developers to manage security issues more strategically; (iv) they narrowed the number of blockchain architectural options to make decision-making easier; (v) they guided architects to make justifiable design decisions that would result in more secure implementations and reduce security complications. For instance, in Chapter 5, our decision model selected Tellor above the other option as the first choice for integration into the DA application because it is more secure, affordable, and suited to that application. However, as was seen in the chapter, failing to employ

a systematic decision-making process could result in the integration of a less-than-ideal alternative, which could be detrimental to the application's efficiency or security. Indeed, the experts we have consulted through the survey and interviews have confirmed these benefits.

- **Clarity** - The provided approaches were systematic and comprised distinct steps. We thoroughly explained each step to be clear enough to render a transparent decision-making process for the architectural components of blockchain systems and smart contracts, with clear rationale, criteria, and steps to inform the decision as judged by experts. Additionally, based on the experts' judgments, the structured manner for assessing the security of smart contracts helped increase the visibility of the underlying security debts associated with the design of smart contracts.
- **Comparison with alternative approaches** - Our investigation showed that only a few studies [119, 124, 376, 369, 336] provided systematic approaches with clear steps for architecting and designing blockchain-based systems. However, the main focus of these studies was not security. Most current practices of designing blockchain and smart contract systems are ad hoc. When we compared each proposed technique to the existing methods, we observed that our approaches were distinct as they offered systematic steps to guide software engineers in making informed and secure design decisions regarding blockchain systems and smart contracts. For example, to the best of our knowledge and based on the opinions of blockchain oracle experts, blockchain oracle platforms are chosen ad hoc, by their reputation, or based on the wisdom of the crowd. Additionally, we formulated our approaches based on a thorough investigation of the current literature and experts' knowledge. Moreover, this is among the initial efforts that introduce the metaphor of technical debt to analyse and assess security issues in smart contracts. The detailed comparison appears in Chapters 3, 4, and 5.
- **Cost-effectiveness** - Following our approaches when architecting blockchain and



smart contracts applications would reduce the overall development cost since they assist in making secure and optimal decisions in the first place, which reduces the overhead of redesigning and updating applications due to sub-optimal decisions. Although there might be an overhead or cost related to tools installation, developers' training, or work-process restructuring, the adding values of employing the approaches extend their potential overhead, especially in the case of a sensitive application where making ill-informed decisions might lead to severe security implications or wrong selection of the architectural components. Nevertheless, the steps of our approaches are flexible and refinable based on the developers' requirements and application context.

# Chapter Seven

## Conclusion Remarks and Future Work

This chapter highlights our contributions and outlines directions for future research.

### 7.1 Contributions

This thesis aims to provide systematic approaches that guide software engineers to build secure blockchain-based systems and smart contracts. To achieve this aim, we have addressed the following:

- **A systematic literature review (SLR) of existing architectural design approaches for building secure blockchain systems and smart contracts.** Chapter 2 analysed academic publications and grey literature to determine the current methods for architecturally designing secure blockchain systems and smart contracts. We provided a classification of existing publications that provide secure architectural design approaches, and we also categorised the publications that support blockchain security risk assessment methods. The review led us to determine gaps and opportunities for further research related to the inadequacies of the current methods in providing

a clear guide to building secure blockchain and smart contract systems.

- **A taxonomy of blockchain architecture design decisions and their security attacks and threats [6].** In Chapter 3, we conducted a review partially guided by the SLR method to derive a taxonomy of commonly used architecture design decisions in blockchain-based systems. We mapped each decision to potential security attacks and their posed threats. MITRE's [245] attack tactic categories and Microsoft STRIDE [330] threat modelling classify threats and their associated attacks to identify potential security issues in blockchain-based systems. Our mapping approach aims to guide architects to make justifiable design decisions that result in more secure implementations. The demonstration showed how this approach assists blockchain systems architects in recognising the impact and scope of the potential security challenges associated with suboptimal design decisions.
- **A technical debt-aware approach to designing secure smart contracts [5].** In Chapter 4, we provided an assessment approach that allows developers to recognise the consequences of deploying vulnerable contracts. Our assessment approach involves two steps: (i) identification of design vulnerabilities using security analysis techniques and (ii) an estimate of the ramifications of the identified vulnerabilities leveraging the technical debt metaphor, its principal, and interest. We use a dataset of vulnerable contracts to demonstrate the applicability of our approach. The results showed that our assessment increases the visibility of security design issues. It also allows developers to concentrate on resolving smart contract vulnerabilities through technical debt impact analysis and prioritisation. We evaluated the proposed approach based on experts' judgment. The experts affirmed that the steps of our technique are clear and understandable, and it helps develop secure smart contracts.
- **A decision support model for blockchain oracle platform selection.** In Chapter 5, we provided a decision support model for the oracle selection problem. As the

number of oracle alternatives and their features increases, the decision-making process becomes increasingly complex. The model supports smart contract decision-makers in selecting a secure, cost-effective, and feasible oracle platform for their applications. We interviewed oracle co-founders and smart contracts experts to refine the knowledge base of our decision model and confirm its usefulness. We used two real-world smart contract application case studies to evaluate the decision model. Based on the developer's required criteria, security assessment, and cost analysis, the model prioritises and suggests more than one possible oracle platform. A guided decision model can reveal issues that might go unnoticed if done haphazardly, reduce decision-making efforts, and provide a cost-effective solution.

## 7.2 Future Work

### 7.2.1 Enhancing Current Security Analysis Tools

According to our examination of the available security analysis tools in Chapter 4, they only provide a limited list of security issues. One possible future direction is to enhance security tools by using artificial intelligence (AI), which makes the tools adaptable and able to evolve in response to new and emerging attack techniques. AI-enhanced tools leverage machine learning and adaptive algorithms to learn from new data and experiences, enabling them to improve their detection capabilities over time. These tools can integrate data from various sources, such as the collection of attacks and threats that we classified in Chapter 3, the design vulnerabilities that we classified in Chapter 4, and public sources reporting new vulnerabilities. By assimilating diverse data, they stay informed about the latest attack trends. Machine learning algorithms process and analyse a multitude of features and patterns within smart contracts. As new types of vulnerabilities emerge, the algorithms can adapt

to identify these patterns, even if they were previously unknown. Dynamic updates can also incorporate behavioural analysis, where tools identify patterns of behaviour that might signal new attack vectors. By learning from these behaviours, the tools can predict and prevent attacks that were previously unseen.

Another possible future direction is to enhance the output of the tools and make them more meaningful by, for example, integrating our assessment approaches into these tools to complement the current analysis and improve the effectiveness of their outcomes. They will not only list the vulnerabilities but will also map them to their corresponding weaknesses, providing an assessment and severity level of the potential security debts.

We observed that none of the current tools identifies vulnerabilities brought on by the injected data from the oracle. An oracle that contains outdated or malicious data could significantly impact the entire contract. As a result, a tool that can spot security flaws in oracle integration with smart contracts is necessary. This tool must verify the call function to ensure only the oracle can make the call to avoid malicious parties supplying false information. The tool could also check the assigned access control while requesting and receiving data from an oracle. Moreover, we found a need for a general tool that can work for different blockchain platforms since multiple tools that detect a small set of issues and only work on a specific platform complicate security analysis.

### **7.2.2 Extending the Approaches Proposed**

We can expand the application of the proposed approaches in this work to involve platforms other than Ethereum, like Hyperledger Fabric. Although our techniques apply to other blockchain platforms, some steps require updating or extending. Chapter 4 presented a systematic procedure to collect and map each design vulnerability in smart contract de-

sign architecture to the corresponding weaknesses in CWE. This procedure can collect and map design issues in chaincode, a smart contract that runs over the Hyperledger blockchain. Researchers should investigate security analysis tools that detect flaws with chaincodes and apply the established criteria in the chapter to select suitable tools. The same remaining steps can assess security debt identified in the chaincodes. In Chapter 5, most of the oracle alternatives in our decision model can only integrate with Ethereum contracts. An interesting future research direction would be to add other oracle platforms that integrate with Hyperledger to the list of alternatives. This extension might expand the list of features and attacks. However, the rest of the model will not need to be updated.

### 7.2.3 Fully Automating the Approaches Proposed

Fully automating the approaches proposed in this work would be an interesting direction for future work to promote quicker and more efficient analysis. Automated support will ensure consistency in the analysis and allow security software engineers to quickly estimate the implications of applying alternative architectural design components and security options when designing blockchain systems and smart contracts. One possible project is to integrate the blockchain oracle decision model proposed in Chapter 5 into a decision model studio<sup>1</sup> provided by a decision support system (DSS) proposed by Farshidi et al. [119]. The oracle decision model needs to be uploaded to the knowledge base of the DSS to facilitate the decision-making process. The DSS offers a venue for discussion that empowers decision-makers at software-producing companies to reach a consensus. Moreover, the DSS can be utilised throughout the whole lifecycle and adjust its suggestions in response to changing requirements.

Another possible direction is to enhance decision support models using AI and ma-

---

<sup>1</sup><https://dss-mcdm.com/>

chine learning, as they offer numerous opportunities to provide more accurate, timely, and personalised recommendations. AI-enhanced models can simulate different scenarios based on historical data and user input. This allows decision-makers to evaluate the potential outcomes of various decisions, enabling strategic planning. Decision support models can also continuously learn from user feedback and the outcomes of decisions. This feedback loop helps refine recommendations over time, making the system more accurate and aligned with user needs. Moreover, natural language processing can be leveraged to enable decision support models to understand and process natural language queries. Decision-makers can interact with the system using everyday language, making it more accessible and user-friendly.

#### **7.2.4 Addressing Security and Architectural Dimension Limitations.**

Based on the findings of Chapter 3, we have identified several research areas that require more investigation.

**Architectural Dimension Limitations.** There is a lack of significant research on aspects of node architecture, as described in the taxonomy. Specifically, there is still a need to investigate and design more effective architectural solutions that allow nodes to receive and store transactional data more efficiently and securely. Another architectural dimension that requires more research is block configuration. Not enough research has investigated the limitations and security challenges related to block size and its propagation, and how these decisions would affect the latency and the throughput of the blockchain network. The throughput of the blockchain network needs to be improved to be applicable in current production environments. At present, in most blockchain platforms, a transaction takes minutes to be confirmed. Thus, improving the confirmation latency to seconds, while still preserving security, is a key challenge.

**Security Limitations.** Even though blockchain has been considered as a solution to tackle

DDoS attacks because of its decentralisation and distributed properties, it can be seen from our mapping that several blockchain system architectural dimensions are still vulnerable to DDoS attacks. Introducing centralised components, which become a single point of failure, into the blockchain system, makes the system highly prone to DDoS attacks. These attacks prevent the system from delivering the required services to the users. Therefore, techniques that can improve security against this type of attack need to be explored. Public-key algorithms are prone to quantum attacks which might easily break transaction signatures. Few studies have investigated this situation and suggested alternative anti-quantum algorithms. However, there is a lack of studies that have analysed the alternative solutions as a means of enhancing the cryptographic algorithms applied in blockchain and selecting the optimal one in terms of the security attributes required by the application domain.



# Appendix One

## Quality Assessment of Selected Studies

Table A illustrates the quality assessment scores of each included study in a systematic literature review presented in Chapter 2.

Table A.1: Quality Assessment

Study	Rationality			Rigor			Credibility		Total
	Empirical	Context	Objectives	Method	Data	Analysis	Results	Limitations	
[119]	1	1	1	1	1	1	1	1	8
[373]	1	1	1	1	0	1	1	1	7
[124]	1	1	1	1	1	1	1	0	7
[376]	1	1	1	1	0	1	1	0	6
[369]	1	1	1	1	0	1	1	0	6
[378]	1	1	1	1	0	1	1	0	6
[318]	0	1	1	1	0	1	0	0	4
[303]	1	1	1	1	0	1	0	0	5
[336]	1	0	1	1	0	1	1	1	6
[51]	1	1	1	1	0	1	1	0	6
[289]	0	0	1	1	1	1	0	0	4
[220]	0	1	1	1	0	1	0	0	4
[382]	1	1	1	1	1	1	1	1	8
[374]	0	1	1	1	0	1	1	0	5
[159]	1	1	1	1	1	1	1	1	8
[154]	1	1	1	1	0	1	1	0	6
[398]	0	1	1	1	0	1	1	0	5
[34]	1	1	1	1	1	1	1	0	7
[8]	1	1	1	1	0	1	1	0	6
[308]	1	1	1	1	1	1	1	1	8
[9]	1	1	1	1	0	1	1	0	6
[183]	1	1	1	1	0	1	0	0	5
[247]	0	1	1	1	0	1	1	0	5
[364]	1	1	1	1	1	1	1	0	7
[228]	0	1	1	1	0	1	0	0	4
[128]	0	1	1	0	0	1	0	0	3
[44]	1	1	1	1	1	1	1	1	8

## Appendix Two

# Mapping Selected Publications with the Taxonomy

A representative selection of studies that belong to each dimension of architectural design decision of blockchain-based systems can be found in Table B.1. It shows that some studies contribute to more than one dimensions.

Table B.1: Mapping Selected Publications with the Taxonomy

Dimensions of Architectural Decisions		Studies
<b>Access Type</b>	Public	[231, 207, 197, 204, 368]
	Private	[231, 207, 197, 20, 368]
	Consortium	[231, 207, 20, 94] [283, 383, 368, 367, 90]
<b>Storage and Computation</b>	On-Chain	[377, 104, 383, 12, 144, 173]
	Off-Chain	[377, 104, 383, 372, 401, 405, 375, 246]
<b>Consensus Mechanism</b>	Proof-based (PoX)	[231, 207, 383, 180, 30, 209, 102, 62, 137, 115, 266, 241]
	Voting-based	[231, 283, 407, 323, 56, 33, 198]
<b>Block Configuration</b>	Block Size	[375, 137, 96]
	Block Confirmation	[405, 137, 383, 205, 96]
	Block Propagation	[137, 115, 86]
<b>Key Management</b>	Single Key	[319, 52, 96]
	Multi-Signature	[135, 141, 319]
	Threshold-Signature	[142, 135, 141, 319]
<b>Cryptographic Primitives</b>	Essential Primitive	[355, 129, 344, 206, 305]
	Optional Primitive	[142, 355, 304, 171, 305, 190, 146, 136]
<b>Chain Structure</b>	Single Chain	[405, 205, 266, 286, 316]

*Continued on next page*

Table B.1 Mapping Selected Publications with the Taxonomy (*Continued from previous page*)


Dimensions of Architectural Decisions		Studies
	Multiple Chains	[405, 210, 204, 366, 12, 170, 78]
Node Architecture	Light Node	[72, 12, 3]
	Full Node	[72, 125]
Smart Contract	Platform	[20, 345, 372, 119, 79]
	Developing Languages	[345, 364, 219, 262, 190]
	External Oracle	[377, 190, 79, 221, 364, 232, 219, 246]

# Appendix Three






## Survey Questioners and Responses

In this appendix, we present the responses of each participant to the survey questionnaires that we distributed to the smart contract experts to evaluate our approach to assessing smart contract security issues presented in Chapter 4. We also present the ethical approval confirmation to conduct this survey.

## Survey about Evaluating our Approach for Assessing Smart Contract Security Issues

1. Do you agree to give us your opinion about our proposed approach?				
Answer Choices			Response Percent	Response Total
1	Yes		100.00%	27
2	No		0.00%	0
			answered	27
			skipped	0

### Questionnaires

2. What best describes your main role in the area of smart contracts?				
Answer Choices			Response Percent	Response Total
1	Decision maker		21.05%	4
2	Designer		36.84%	7
3	Architect		26.32%	5
4	Programmer		78.95%	15
5	Researcher		21.05%	4
			answered	19
			skipped	8
Other: (2)				
1	08/11/2022 16:23 PM ID: 203389038	I work on SC design and implementation (programming) together with colleagues - we share responsibility.		
2	09/11/2022 14:13 PM ID: 203473925	Auditor		

3. Are the steps of the proposed approach to smart contracts security assessment clear and understandable?				
Answer Choices			Response Percent	Response Total
1	Yes		100.00%	19
2	No		0.00%	0



3. Are the steps of the proposed approach to smart contracts security assessment clear and understandable?			
		answered	19
		skipped	8
Comments: Why? (1)			
1	01/12/2022 09:57 AM ID: 205680184	Reasonably so. It would be nice if there is a quicker way to capture it. For instance, if there is an acronym one can use that reflects the various steps?	


4. Is the proposed approach to smart-contract security assessment potentially easy to use?			
Answer Choices		Response Percent	Response Total
1	Yes		94.44% 17
2	No		5.56% 1
		answered	18
		skipped	9
Comments: Why? (2)			
1	09/11/2022 22:28 PM ID: 203534872	Not always because with complex smart contracts, the process implementation will become too extensive	
2	01/12/2022 09:57 AM ID: 205680184	There will be challenges because it needs tool support, so such tool support should be easy. There also is a learning curve to understand various classes of vulnerabilities--how does such learning take place? Should there be training of developers?	

5. Is the proposed approach to smart contract security assessment useful to support designing and developing secure contracts?			
Answer Choices		Response Percent	Response Total
1	Yes		100.00% 18
2	No		0.00% 0
		answered	18
		skipped	9
Comments: Why? (2)			
1	09/11/2022 22:28 PM ID: 203534872	yes more extensive assessment of smart contracts will not help in early detection but also impact of the vulnerability in accordance with specific use-case scope	
2	01/12/2022 09:57 AM ID: 205680184	Something like this is very necessary, since the impact of security-related software bugs can be substantial.	



6. Is the proposed approach to smart contract security assessment potentially useful in assessing the security risk of smart contracts?				
Answer Choices			Response Percent	Response Total
1	Yes		100.00%	18
2	No		0.00%	0
			answered	18
			skipped	9

7. Is the proposed approach to smart contract security assessment potentially helpful in increasing the visibility of the security design issue related to smart contracts?				
Answer Choices			Response Percent	Response Total
1	Yes		94.44%	17
2	No		5.56%	1
			answered	18
			skipped	9
Comments: Why? (1)				
1	01/12/2022 09:57 AM ID: 205680184	I think this aspect is important: software engineers should deal with smart contract security in a deliberate and structured manner.		

8. Does the proposed approach to smart contract security assessment add additional value in designing secure smart contracts compared with doing so in an ad hoc manner?				
Answer Choices			Response Percent	Response Total
1	Yes		100.00%	18
2	No		0.00%	0
			answered	18
			skipped	9
Comments: Why? (2)				
1	09/11/2022 22:28 PM ID: 203534872	Yes for sure as it will help in set rules of deep 'post dev' analysis of contracts		
2	01/12/2022 09:57 AM ID: 205680184	Similar comment to 7		

9. What are your suggestions to enhance the suitability of our approach?				
Answer Choices			Response Percent	Response Total
1	Open-Ended Question		100.00%	19
1	05/11/2022 18:28 PM ID: 203170434	I think tools like Trail of bits Manticore are helpful in formal verification of contracts. Program exploration and state assertions. Also I think automated tools are helpful but cannot test the intended design errors of how a contract is intended to be used.		
2	06/11/2022 15:23 PM ID: 203194037	Its well and good		
3	06/11/2022 16:37 PM ID: 203196376	No suggestions		
4	06/11/2022 18:46 PM ID: 203199785	In general I do agree with your approach. However I find the full approach to be more suitable for a code review style of security verification.  For example when designing/developing new contracts, the original devs should certainly perform 1a and 1b but will normally not perform steps 1c, 1d and step 2.  Also one should always be wary from over relying on automated tools. Some may consider these as a security seal and be less thorough when performing 1b.  Indeed the real difficult part is on how to do 1b. This is where I believe all devs need help.		
5	07/11/2022 16:28 PM ID: 203272264	Encouraging a deep understanding of each vulnerability category (Time manipulation, reentrancy etc), enables for efficient human audit after the results of the automated analysis tools. Focus on smart contract behaviour and functionality being as intended by the management, is an easily overlooked factor when we predominantly focus on the procedural exploitability of the code. Yet from my experience, logical errors and malicious developer-implemented functions are common and overlooked.		
6	08/11/2022 16:23 PM ID: 203389038	A couple of topics come to mind that might be worth exploring further.  The approach you've described is in some ways similar to the process used during a contract audit. Is it expected that such an approach would be used by developers themselves instead of an audit, or in addition to an audit carried out by a third party?  Several of the steps mention Ethereum-specific tools and concepts (Mythos, Solhint, gas) - is the approach generally targeted at Ethereum, or can/will it be extended to other (non-EVM) blockchains that support smart contracts?  Good work! :-)		
7	09/11/2022 08:20 AM ID: 203431544	Asking more specific questions		
8	09/11/2022 09:49 AM ID: 203439303	Add formal verification methods and tools like Certora.		
9	09/11/2022 14:13 PM ID: 203473925	I would suggest increasing the amount of work done in the identification phase. Even suggested tools are good, most of the vulnerabilities found in Smart Contracts are related to business logic. I think writing formal specifications and invariants for the system is a way for improving identification phase		

10	09/11/2022 22:28 PM ID: 203534872	Since most contracts are custom build for the specific use case, predetermining the contract use case and scope will help in determining the depth of analysis required, thus helping in speeding up the proposed process	
11	01/12/2022 09:57 AM ID: 205680184	1. Aim to simplify yet further. 2. Try to be creative in terms of how to capture your approach in some that people can remember, eg an acronym that represents the main steps. 3. Address the learning curve aspect, how can engineers learn the necessary things to easily do the structured approach?	
12	04/12/2022 20:31 PM ID: 205888320	They are clear	
13	04/12/2022 22:42 PM ID: 205888680	None	
14	06/12/2022 10:44 PM ID: 205888772	Make the steps automated to become easy to apply.	
15	07/12/2022 20:46 PM ID: 205888838	This kind of method is helpful especially for beginner and non-experts smart contract developers.	
16	10/12/2022 20:48 PM ID: 205888878	The approach is easy to follow. I suggested to posted in a blog so developers can follow it because not all developers read research papers.	
17	13/12/2022 20:52 PM ID: 205888944	I have no suggestions	
18	15/12/2022 11:56 PM ID: 205889146	None...	
19	16/12/2022 09:02 PM ID: 205889338	Good work	
		answered	19
		skipped	8



UNIVERSITY OF  
BIRMINGHAM

Dear Rami Bahsoon ,

**RE:** Security Assessment Approaches for Blockchain-based Systems

**Application for Ethical Review:** ERN\_2022-0558

Your project has been considered in line with the University of Birmingham's research ethics processes and on the basis of the information you have provided, it is understood that while your project does involve human participants, the project raises no substantial research ethics issues and therefore no further ethics review is required

Any adverse events occurring during the study should be promptly brought to the Committee's attention by the Principal Investigator and may necessitate further ethical review.

Please ensure that the relevant requirements within the University's Code of Practice for Research and the information and guidance provided on the University's ethics webpages (available at <https://intranet.birmingham.ac.uk/finance/accounting/Research-Support-Group/Research-Ethics/Links-and-Resources.aspx> ) are adhered to.

Please be aware that whilst Health and Safety (H&S) issues may be considered during the ethical review process, you are still required to follow the University's guidance on H&S and to ensure that H&S risk assessments have been carried out as appropriate. For further information about this, please contact your School H&S representative or the University's H&S Unit at [healthandsafety@contacts.bham.ac.uk](mailto:healthandsafety@contacts.bham.ac.uk).

Kind regards,

The Co-Chairs of the Science, Technology, Engineering and Mathematics Committee

E-mail: [ethics-queries@contacts.bham.ac.uk](mailto:ethics-queries@contacts.bham.ac.uk)

# Appendix Four

## Interviewee Structure

In this appendix, we present the semi-structured interview script that we followed when conducting the interview with blockchain oracle experts to refine and validate the blockchain oracle decision model represented in Chapter 5. The ethical approval confirmation to conduct the interviews was presented in Appendix 3.

## Interview Script

### Introduction

I really appreciate your taking the time to meet with me today. Let me briefly explain to you our project, its aim, and the purpose of the interview. We are developing a blockchain oracle decision support model to assist smart contract developers and decision-makers in selecting suitable, secure, and cost-effective oracle platforms. During the interview, I would like to ask you several questions about blockchain oracle characteristics, then we will discuss your experience of developing and/or integrating oracle into smart contracts. Your answers will help us in refining and validate the knowledge base of our decision model.

### Interview Questions

#### Demographics

When it comes to blockchain oracles and smart contracts, how many years of experience do you have?

#### Open-Ended Questions

Q1: Which blockchain oracle platforms are you familiar with?

Q2: From your experience, how do you think smart contract developers are selecting the oracle platform to integrate it into their applications?

Q3: From your perspective, which features should be supported by blockchain oracle?

Q4: When people attempt to select oracle, what are the features that they should look at?

Q5: Based on your knowledge, what are the current oracle platforms that support your stated features?

Q6: Based on your knowledge, what are the quality attributes provided by the stated features?

Q7: Which sort of security threats you can imagine that target such type of blockchain oracle?

Q8: Do you think there is a need for a decision model to assist the decision-makers in selecting a suitable oracle?

Q9: Do you think employing a decision model makes the features of oracle visible to the decision-makers and increases transparency?

We would like to thank you for the discussion and for answering our questions. We are working to formulate the decision model, and we will send you the final version.

## Appendix Five

# Linking Oracle Features with Platforms and Quality Attributes

A complete mapping between: **a.** the Boolean oracle features and platforms and **b.** the Boolean oracle features and quality attributes

# Linking Oracle Features with Platforms and Quality Attributes

Boolean Oracle Criteria and Platforms	Provable	Town-Crier	Chainlink	Witnet	Tellor	Paralink	BandChain	iExec
<b>Type of Data Feeder</b>								
Centralised	1	1	0	0	0	0	0	0
Semi-decentralised	0	0	1	1	0	1	1	0
Fully-Decentralised	0	0	0	0	1	0	0	1
<b>Type of Data Source</b>								
Single	1	1	1	1	1	1	1	1
Multiple	0	1	1	1	1	1	1	1
<b>Data Validation Mechanism</b>								
Trusted third party	1	0	0	0	0	0	0	0
Trusted Execution Environment	0	1	1	0	0	0	0	1
Consensus	0	0	1	1	1	1	1	1
<b>Integration Methods</b>								
On-chain	0	0	1	1	1	1	1	0
Off-chain	1	1	0	0	0	0	0	0
Hybrid	0	0	0	0	0	0	0	1
<b>Encryption Method</b>								
Symmetric cryptography	0	0	0	0	0	0	0	0
Asymmetric cryptography	1	1	1	0	0	0	0	1
<b>Data Feeders Selection Method</b>								
Stacking	0	0	1	0	1	1	1	1
Reputation	0	0	1	1	0	1	1	1
PoW	0	0	0	1	1	0	0	0
PoCo	0	0	0	0	0	0	0	1
Random	0	0	0	0	1	0	1	1
Pseudo-random	0	0	0	1	0	0	0	0
<b>Aggregation Mechanism</b>								
Statistical Measure	0	0	1	0	0	0	1	0
Mean	0	0	0	1	0	1	0	0
Median	0	0	0	1	1	1	0	0
Mode	0	0	0	1	0	0	0	0
Voting	0	0	0	0	0	0	0	1
<b>Dispute Resolution</b>								
Voting	0	0	0	0	1	0	0	0
Stacking	0	0	0	0	1	0	1	0
Statistical Measure	0	0	1	0	0	0	1	0
<b>Incorrect data</b>								
Corrected	0	0	0	0	1	0	0	0
Reverted	0	0	0	0	0	0	0	0
<b>Incentive scheme</b>								
Reward	0	0	1	1	1	1	1	1
No reward	1	1	0	0	0	0	0	0
<b>punishment methods</b>								
Slash	0	0	1	1	1	1	1	1
Ban	0	0	0	0	1	0	0	0
Reputation Loss	0	0	1	1	0	1	1	1
<b>Native Token (NT)</b>								
Exist	0	0	1	1	1	1	1	1
Not exist	1	1	0	0	0	0	0	0
<b>Using NT for requesting</b>								
Required	0	0	1	0	1	1	1	1
Not-required	1	1	0	1	0	0	0	0

a. PF Mapping

Boolean Oracle Criteria and Attributes	Confidentiality	Integrity	Availability	Non-repudiation	Authenticity	Trustlessness	Transparency	Low latency	Low cost	Low transactional financial Cost	Accessibility
<b>Type of Data Feeder</b>											
Centralised	0	0	0	0	0	0	0	1	0	0	0
Semi-decentralised	0	0	1	0	0	0	0	0	0	0	0
Fully-Decentralised	0	1	1	0	0	1	0	0	0	0	0
<b>Type of Data Source</b>											
Single	0	0	0	0	0	0	0	1	0	0	0
Multiple	0	1	1	0	0	1	0	0	0	0	0
<b>Data Validation Mechanism</b>											
Trusted third party	1	1	0	1	1	0	0	1	0	0	0
Trusted Execution Environment	1	1	0	1	1	0	0	1	0	0	0
Consensus	0	1	0	1	1	1	0	0	0	0	0
<b>Integration Methods</b>											
On-chain	0	1	1	0	0	0	1	0	0	0	0
Off-chain	0	0	0	0	0	0	0	1	0	0	0
Hybrid	0	1	0	0	0	1	1	0	1	0	0
<b>Encryption Method</b>											
Symmetric cryptography	1	0	0	0	0	0	0	0	0	0	0
Asymmetric cryptography	1	0	0	0	0	0	0	0	0	0	0
<b>Data Feeders Selection Method</b>											
Stacking	0	1	0	0	0	0	0	0	0	0	0
Reputation	0	1	0	0	0	0	0	0	0	0	0
PoW	0	1	0	0	0	0	0	0	0	0	0
PoCo	0	1	0	0	0	0	0	0	0	0	0
Random	0	1	0	1	1	0	0	0	0	0	0
Pseudo-random	0	1	0	1	1	0	0	0	0	0	0
<b>Aggregation Mechanism</b>											
Statistical Measure	0	0	0	0	0	0	0	1	1	0	0
Mean	0	0	0	0	0	0	0	1	1	0	0
Median	0	0	0	0	0	0	0	1	1	0	0
Mode	0	0	0	0	0	0	0	1	1	0	0
Voting	0	0	0	0	0	0	0	0	0	0	0
<b>Dispute Resolution</b>											
Voting	0	0	0	0	0	0	0	0	0	0	0
Stacking	0	1	0	0	0	0	0	0	0	0	0
Statistical Measure	0	0	0	0	0	0	0	1	1	0	0
<b>Incorrect data</b>											
Corrected	0	0	0	0	0	0	0	0	0	0	0
Reverted	0	0	0	0	0	0	0	1	0	0	0
<b>Incentive scheme</b>											
Reward	0	1	0	0	0	0	0	0	0	0	0
No reward	0	0	0	0	0	0	0	0	0	0	1
<b>punishment methods</b>											
Slash	0	1	0	0	0	0	0	1	0	0	0
Ban	0	1	0	0	0	0	0	1	0	0	0
Reputation Loss	0	1	0	0	0	0	0	1	0	0	0
<b>Native Token (NT)</b>											
Exist	0	0	0	0	0	0	0	0	0	0	0
Not exist	0	0	0	0	0	0	0	0	0	0	0
<b>Using NT for requesting</b>											
Required	0	0	0	0	0	0	0	0	0	0	0
Not-required	0	0	0	0	0	0	0	0	0	0	1

b. FQ Mapping



# References

- [1] 27001academy. *ISO 27001 Risk Assessment, Treatment, & Management: The Complete Guide*. Mar. 2022. URL: <https://advisera.com/27001academy/iso-27001-risk-assessment-treatment-management/>.
- [2] ISO/TC 307. *Blockchain and distributed ledger technologies — Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems*. ISO. Mar. 2019. URL: <https://www.iso.org/standard/75624.html>.
- [3] Ryosuke Abe, Shigeya Suzuki, and Jun Murai. “Mitigating bitcoin node storage size by DHT”. In: *Proceedings of the 2018 Asian Internet Engineering Conference*. New York, NY, USA: ACM, 2018, pp. 17–23.
- [4] S. Ahmadjee. *Oracle DSM*. bitbucket. Mar. 2022. URL: [https://bitbucket.org/Smart\\_Contract/oracle\\_dsm/src/master/](https://bitbucket.org/Smart_Contract/oracle_dsm/src/master/).
- [5] S. Ahmadjee, C. Mera-Gomez, and R. Bahsoon. “Assessing Smart Contracts Security Technical Debts”. In: *2021 2021 IEEE/ACM International Conference on Technical Debt (TechDebt) (TechDebt)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2021, pp. 6–15.
- [6] Sabreen Ahmadjee et al. “A Study on Blockchain Architecture Design Decisions and Their Security Attacks and Threats”. In: *ACM Trans. Softw. Eng. Methodol.* 31.2 (Apr. 2022). ISSN: 1049-331X.

- 
- [7] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. “Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams”. In: *IEEE Transactions on Dependable and Secure Computing* 15.5 (2016), pp. 840–852.
- [8] Maitha Al Ketbi et al. “Establishing a Security Control Framework for Blockchain Technology”. In: *Interdisciplinary Journal of Information, Knowledge, and Management* 16 (2021), p. 307.
- [9] Ranwa Al Mallah, David López, and Bilal Farooq. “Cyber-Security Risk Assessment Framework for Blockchains in Smart Mobility”. In: *IEEE Open Journal of Intelligent Transportation Systems* 2 (2021), pp. 294–311.
- [10] Maher Alharby and Aad Van Moorsel. “Blockchain-based smart contracts: A systematic mapping study”. In: *arXiv preprint arXiv:1710.06372* (2017).
- [11] Muhammad Salek Ali et al. “Applications of blockchains in the Internet of Things: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1676–1717.
- [12] Muneeb Ali et al. “Blockstack: A global naming and storage system secured by blockchains”. In: *Proceedings of the 2016 Annual Technical Conference*. Denver, CO: USENIX, 2016, pp. 181–194.
- [13] Mohamed Almorsy, John Grundy, and Amani S Ibrahim. “Automated software architecture security risk analysis using formalized signatures”. In: *2013 35th International Conference on Software Engineering (ICSE)*. San Francisco, CA, USA: IEEE, 2013, pp. 662–671.
- [14] Esra Alzaghoul and Rami Bahsoon. “CloudMTD: Using real options to manage technical debt in cloud-based service selection”. In: *2013 4th International Workshop on Managing Technical Debt (MTD)*. IEEE, 2013, pp. 55–62.

- 
- [15] Esra Alzaghoul and Rami Bahsoon. “Economics-driven approach for managing technical debt in cloud-based architectures”. In: *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2013, pp. 239–242.
- [16] Sidney Amani et al. “Towards verifying ethereum smart contract bytecode in Isabelle/HOL”. In: *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM, 2018, pp. 66–77.
- [17] Areti Ampatzoglou et al. “The financial aspect of managing technical debt: A systematic literature review”. In: *Information and Software Technology* 64 (2015), pp. 52–73.
- [18] Nitish Andola et al. “Vulnerabilities on hyperledger fabric”. In: *Pervasive and Mobile Computing* 59 (2019), p. 101050.
- [19] Elli Androulaki et al. “Evaluating user privacy in bitcoin”. In: *Proceedings of the 2013 International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2013, pp. 34–51.
- [20] Elli Androulaki et al. “Hyperledger fabric: a distributed operating system for permissioned blockchains”. In: *Proceedings of the Thirteenth EuroSys Conference*. New York, NY, USA: ACM, 2018, p. 30.
- [21] AA Andryukhin. “Phishing attacks and preventions in blockchain based projects”. In: *2019 International Conference on Engineering Technologies and Computer Science (EnT)*. Moscow, Russia: IEEE, 2019, pp. 15–19.
- [22] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. “Hijacking bitcoin: Routing attacks on cryptocurrencies”. In: *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 375–392.

- 
- [23] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. “A survey of attacks on ethereum smart contracts”. In: *Principles of Security and Trust*. Berlin, Heidelberg: Springer, 2017, pp. 164–186.
- [24] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. “A survey of attacks on Ethereum smart contracts.” In: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 1007.
- [25] band. *Band Protocol*. band. Mar. 2021. URL: <https://docs.bandchain.org/whitepaper/system-overview.html>.
- [26] BandChain. *BandChain Whitepaper*. Band. Mar. 2020. URL: <https://docs.bandchain.org/whitepaper/system-overview.html>.
- [27] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [28] Juan Benet. “Ipfes-content addressed, versioned, p2p file system”. In: *arXiv preprint arXiv:1407.3561* 92 (2014), pp. 399–406.
- [29] Fabrice Benhamouda, Shai Halevi, and Tzipora Tracy Halevi. “Supporting private data on Hyperledger Fabric with secure multiparty computation”. In: *IBM Journal of Research and Development* (2019).
- [30] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. “Cryptocurrencies without proof of work”. In: *Proceedings of the 2016 International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2016, pp. 142–157.
- [31] Iddo Bentov et al. “Proof of Activity: Extending Bitcoin’s Proof of Work via Proof of Stake.” In: *IACR Cryptology ePrint Archive 2014* (2014), p. 452.
- [32] Terese Besker, Antonio Martini, and Jan Bosch. “Managing architectural technical debt: A unified model and systematic literature review”. In: *Journal of Systems and Software* 135 (2018), pp. 1–16.

- 
- [33] Alysson Bessani, João Sousa, and Eduardo EP Alchieri. “State machine replication for the masses with BFT-SMaRt”. In: *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Atlanta, GA, USA: IEEE, 2014, pp. 355–362.
- [34] Akashdeep Bhardwaj et al. “Penetration testing framework for smart contract blockchain”. In: *Peer-to-Peer Networking and Applications* 14.5 (2021), pp. 2635–2650.
- [35] Karthikeyan Bhargavan et al. “Formal verification of smart contracts: Short paper”. In: *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*. ACM, 2016, pp. 91–96.
- [36] Karthikeyan Bhargavan et al. “Formal verification of smart contracts: Short paper”. In: *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*. ACM, 2016, pp. 91–96.
- [37] Alex Biryukov and Dmitry Khovratovich. “Equihash: Asymmetric proof-of-work based on the generalized birthday problem”. In: *Ledger* 2 (2017), pp. 1–30.
- [38] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. “Deanonymisation of clients in Bitcoin P2P network”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2014, pp. 15–29.
- [39] Trail of Bits. *(Not So) Smart Contracts*. 2018. URL: <https://github.com/crytic/not-so-smart-contracts>.
- [40] Joseph Bonneau et al. “Sok: Research perspectives and challenges for bitcoin and cryptocurrencies”. In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy (SP)*. San Jose, CA, USA: IEEE, 2015, pp. 104–121.

- 
- [41] Emanuele Borgonovo et al. “Sensitivity analysis”. In: *An Introduction for the Management Scientist. International Series in Operations Research and Management Science*. Cham, Switzerland: Springer (2017).
- [42] Amiangshu Bosu et al. “Understanding the motivations, challenges and needs of Blockchain software developers: a survey”. In: *Empirical Software Engineering* 24.4 (2019), pp. 2636–2673.
- [43] Bounty. *Ethereum Bounty Program*. bitbucket. 2019. URL: <https://bounty.ethereum.org/>.
- [44] Sarah Bouraga. “A taxonomy of blockchain consensus protocols: A survey and classification framework”. In: *Expert Systems with Applications* 168 (2021), p. 114384. ISSN: 0957-4174.
- [45] Sarah Bouraga. “A taxonomy of blockchain consensus protocols: A survey and classification framework”. In: *Expert Systems with Applications* 168 (2021), p. 114384.
- [46] Glenn A Bowen. “Document analysis as a qualitative research method”. In: *Qualitative research journal* (2009).
- [47] Lorenz Breidenbach. *An In-Depth Look at the Parity Multisig Bug*. 2017. URL: <https://bit.ly/3sTxkMd>.
- [48] Hamda Al-Breiki et al. “Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges”. In: *IEEE Access* 8 (2020), pp. 85675–85685.
- [49] Lexi Brent et al. “Vandal: A scalable security analysis framework for smart contracts”. In: *arXiv preprint arXiv:1809.03981* (2018).
- [50] Sotirios Brotsis et al. “On the Security and Privacy of Hyperledger Fabric: Challenges and Open Issues”. In: *2020 IEEE World Congress on Services (SERVICES)*. Beijing, China: IEEE, 2020, pp. 197–204.

- 
- [51] Sotirios Brotsis et al. “On the suitability of blockchain platforms for IoT applications: Architectures, security, privacy, and performance”. In: *Computer Networks* 191 (2021), p. 108005. ISSN: 1389-1286.
- [52] Thanh Bui et al. “Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet”. In: *Proceedings of the 12th European Workshop on Systems Security*. New York, NY, USA: ACM, 2019, p. 3.
- [53] Vitalik Buterin. *Thinking About Smart Contract Security*. 2016. URL: <https://bit.ly/3BkiMIn>.
- [54] Giulio Caldarelli and Joshua Ellul. “The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach”. In: *Applied Sciences* 11.16 (2021). ISSN: 2076-3417.
- [55] Fran Casino, Thomas K Dasaklis, and Constantinos Patsakis. “A systematic literature review of blockchain-based applications: current status, classification and open issues”. In: *Telematics and Informatics* 36 (2018). ISSN: 0736-5853.
- [56] Miguel Castro, Barbara Liskov, et al. “Practical Byzantine fault tolerance”. In: *Proceedings of the 1999 OSDI*. usenix, 1999.
- [57] Yair Censor. “Pareto optimality in multiobjective problems”. In: *Applied Mathematics and Optimization* 4.1 (1977), pp. 41–59.
- [58] Chainlink. *What Is the Blockchain Oracle Problem?* Chainlink. Mar. 2020. URL: <https://blog.chain.link/what-is-the-blockchain-oracle-problem>.
- [59] Kuang-Hua Chang. “Chapter 19 - Multiobjective Optimization and Advanced Topics”. In: *e-Design*. Ed. by Kuang-Hua Chang. Boston: Academic Press, 2015, pp. 1105–1173. ISBN: 978-0-12-382038-9.
- [60] Huashan Chen et al. “A survey on ethereum systems security: Vulnerabilities, attacks, and defenses”. In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–43.

- 
- [61] Jiachi Chen et al. “DEFECTCHECKER: Automated Smart Contract Defect Detection by Analyzing EVM Bytecode”. In: *IEEE Transactions on Software Engineering* 48 (2021), pp. 1–1.
- [62] Lin Chen et al. “On security analysis of proof-of-elapsed-time (poet)”. In: *Proceedings of the 2017 International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Cham: Springer, 2017, pp. 282–297.
- [63] Ting Chen et al. “TokenScope: Automatically Detecting Inconsistent Behaviors of Cryptocurrency Tokens in Ethereum”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 1503–1520.
- [64] Jieren Cheng et al. “A survey of security threats and defense on Blockchain”. In: *Multimedia Tools and Applications* 80.20 (2021), pp. 30623–30652.
- [65] Catalin Cimpanu. *DNS hijacks at two cryptocurrency sites*. 2021. URL: <https://bit.ly/3gGtO33>.
- [66] CONSENSYS. *mythx*. 2020. URL: <https://mythx.io/>.
- [67] DSDM Consortium. *The DSDM Agile Project Framework*. Agile Business Consortium. Dec. 2014. URL: [https://www.agilebusiness.org/page/ProjectFramework\\_10\\_MoSCoWPrioritisation](https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation).
- [68] Mauro Conti et al. “A survey on security and privacy issues of bitcoin”. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3416–3452.
- [69] Thomas Cook, Alex Latham, and Jae Hyung Lee. “Dappguard: Active monitoring and defense for solidity smart contracts”. In: *Retrieved July 18 (2017)*, p. 2018.
- [70] Juliet Corbin and Anselm Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.



- 
- [71] Corda. *An Open Source Blockchain Platform for Businesses / Corda*. 2019. URL: <https://www.corda.net>.
- [72] Leonardo da Costa et al. “Securing light clients in blockchain with DLCP”. In: *International Journal of Network Management* 29.3 (2019), e2055.
- [73] Michael Crosby et al. “Blockchain technology: Beyond bitcoin”. In: *Applied Innovation* 2.6-10 (2016), p. 71.
- [74] CSA. *Over 200 Documented Blockchain Attacks, Vulnerabilities and Weaknesses*. 2020. URL: <https://bit.ly/3kuMgwx>.
- [75] Ward Cunningham. “The WyCash portfolio management system”. In: *ACM SIG-PLAN OOPS Messenger* 4.2 (1993), pp. 29–30.
- [76] European Union Agency for Cybersecurity. *Distributed Ledger Technology & Cybersecurity - Improving information security in the financial sector*. 2017. URL: <https://www.enisa.europa.eu/publications/blockchain-security>.
- [77] Mingjun Dai et al. “A low storage room requirement framework for distributed ledger in blockchain”. In: *IEEE Access* 6 (2018), pp. 22970–22975.
- [78] Hung Dang et al. “Towards scaling blockchain systems via sharding”. In: *Proceedings of the 2019 international conference on management of data*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 123–140.
- [79] Florian Daniel and Luca Guida. “A service-oriented perspective on blockchain smart contracts”. In: *IEEE Internet Computing* 23.1 (2019), pp. 46–53.
- [80] DanielCawrey. *37 Coins Plans Worldwide Bitcoin Access With SMS-Based Wallet*. 2014. URL: <https://bit.ly/3kzmQ0Q>.
- [81] Chris Dannen. *Introducing Ethereum and Solidity*. Vol. 1. Springer, 2017.
- [82] State of the DApps. *state of the dapps*. 2020. URL: <https://www.blockdata.tech/>.

- 
- [83] Dipankar Dasgupta, John M Shrein, and Kishor Datta Gupta. “A survey of blockchain from security perspective”. In: *Journal of Banking and Financial Technology* 3.1 (2019), pp. 1–17.
- [84] Saulo S. de Toledo, Antonio Martini, and Dag I.K. Sjøberg. “Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study”. In: *Journal of Systems and Software* 177 (2021), p. 110968. ISSN: 0164-1212.
- [85] Christian Decker and Roger Wattenhofer. “Information propagation in the bitcoin network”. In: *Proceedings of the 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013, pp. 1–10.
- [86] Christian Decker and Roger Wattenhofer. “Information propagation in the bitcoin network”. In: *Proceedings of the 2013 IEEE P2P*. Trento, Italy: IEEE, 2013, pp. 1–10.
- [87] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. “A Survey on Long-Range Attacks for Proof of Stake Protocols”. In: *IEEE Access* 7 (2019), pp. 28712–28725.
- [88] Deloitte. *Risk Advisory*. 2022. URL: [https://www2.deloitte.com/uk/en/services/risk-advisory.html?icid=top\\_risk-advisory](https://www2.deloitte.com/uk/en/services/risk-advisory.html?icid=top_risk-advisory).
- [89] Qi Deng and Shaobo Ji. “A review of design science research in information systems: concept, process, outcome, and evaluation”. In: *Pacific Asia journal of the association for information systems* 10.1 (2018), p. 2.
- [90] Harsh Desai, Murat Kantarcioglu, and Lalana Kagal. “A hybrid blockchain architecture for privacy-enabled and accountable auctions”. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 34–43.
- [91] Giuseppe Destefanis et al. “Smart contracts vulnerabilities: a call for blockchain software engineering?” In: *Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2018, pp. 19–25.

- 
- [92] Thomas Dickerson et al. “Adding concurrency to smart contracts”. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017, pp. 303–312.
- [93] digitalshadows. *Cryptocurrency Attacks To Be Aware Of In 2021*. 2021. URL: <https://bit.ly/2WyC3XG>.
- [94] Sheng Ding et al. “A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT”. In: *IEEE Access* 7 (2019), pp. 38431–38441.
- [95] Tien Tuan Anh Dinh et al. “Blockbench: A framework for analyzing private blockchains”. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017, pp. 1085–1100.
- [96] Ali Dorri et al. “Blockchain: A distributed solution to automotive security and privacy”. In: *IEEE Communications Magazine* 55.12 (2017), pp. 119–125.
- [97] Nusi Drljevic, Daniel Arias Aranda, and Vladimir Stantchev. “Perspectives on risks and standards that affect the requirements engineering of blockchain technology”. In: *Computer Standards & Interfaces* 69 (2020), p. 103409. ISSN: 0920-5489.
- [98] Raghav Dua. *Protofire*. 2019. URL: <https://github.com/duaraghav8/Ethlint>.
- [99] Alevtina Dubovitskaya et al. “Secure and trustable electronic medical records sharing using blockchain”. In: *Proceedings of the 2017 AMIA Annual Symposium Proceedings*. Vol. 2017. American Medical Informatics Association, 2017, p. 650.
- [100] Evan Duffield and Kyle Hagan. “Darkcoin:Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system”. In: *bitpaper.info* (2014).
- [101] Thomas Durieux et al. “Empirical review of automated analysis tools on 47,587 Ethereum smart contracts”. In: *Proceedings of the ACM/IEEE 42nd International*

- 
- Conference on Software Engineering*. Seoul South Korea: ACM/IEEE, 2020, pp. 530–541.
- [102] Stefan Dziembowski et al. “Proofs of space”. In: *Proceedings of the 2015 Annual Cryptology Conference*. Springer, 2015, pp. 585–605.
- [103] Dennis de Vries Eamonn Maguire Kiran Nagaraj. *Realizing blockchain’s potential*. 2018. URL: <https://assets.kpmg/content/dam/kpmg/xx/pdf/2018/09/realizing-blockchains-potential.pdf>.
- [104] Jacob Eberhardt and Stefan Tai. “On or off the blockchain? Insights on off-chaining computation and data”. In: *Proceedings of the European Conference on Service-Oriented and Cloud Computing*. Cham: Springer, 2017, pp. 3–15.
- [105] Shiva Ebneyamini and Mohammad Reza Sadeghi Moghadam. “Toward Developing a Framework for Conducting Case Study Research”. In: *International Journal of Qualitative Methods* 17.1 (2018), p. 1609406918817954.
- [106] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. “Impact of Man-In-The-Middle Attacks on Ethereum”. In: *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. Salvador, Brazil: IEEE, 2018, pp. 11–20.
- [107] Steve Ellis, Ari Juels, and Sergey Nazarov. *Chainlink: A decentralized oracle network*. 2018. URL: <https://research.chain.link/whitepaper-v1.pdf>.
- [108] Joshua Ellul and Gordon J Pace. “Runtime verification of ethereum smart contracts”. In: *Proceedings of the 2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 158–163.
- [109] E English, AD Kim, and M Nonaka. *Advancing Blockchain Cybersecurity: Technical and Policy Considerations for the Financial Services Industry*. 2018. URL: <https://rb.gy/jv8rra>.

- 
- [110] Neil A Ernst et al. “Measure it? manage it? ignore it? software practitioners and technical debt”. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 50–60.
- [111] Neil A Ernst et al. “What to Fix? Distinguishing between design and non-design rules in automated tools”. In: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE. Gothenburg, Sweden: IEEE, 2017, pp. 165–168.
- [112] Shayan Eskandari et al. “A first look at browser-based Cryptojacking”. In: *Proceedings of the 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2018, pp. 58–66.
- [113] Shayan Eskandari et al. “SoK: Oracles from the Ground Truth to Market Manipulation”. In: *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 127–141.
- [114] Ittay Eyal and Emin Gün Sirer. “Majority is not enough: Bitcoin mining is vulnerable”. In: *Communications of the ACM* 61.7 (2018), pp. 95–102.
- [115] Ittay Eyal et al. “Bitcoin-ng: A scalable blockchain protocol”. In: *Proceedings of the 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. Santa Clara, CA: USENIX, 2016, pp. 45–59.
- [116] Weidong Fang et al. “Digital signature scheme for information non-repudiation in blockchain: a state of the art review”. In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), pp. 1–15.
- [117] Kurt Fanning and David P Centers. “Blockchain and its coming impact on financial services”. In: *Journal of Corporate Accounting & Finance* 27.5 (2016), pp. 53–57.
- [118] Siamak Farshidi et al. “A decision support system for software technology selection”. In: *Journal of Decision Systems* 27.sup1 (2018), pp. 98–110.

- 
- [119] Siamak Farshidi et al. “Decision Support for Blockchain Platform Selection: Three Industry Case Studies”. In: *IEEE Transactions on Engineering Management* 67.4 (2020), pp. 1109–1128.
- [120] Amir Feder et al. “The impact of DDoS and other security shocks on Bitcoin currency exchanges: Evidence from Mt. Gox”. In: *Journal of Cybersecurity* 3.2 (2018), pp. 137–144.
- [121] J. Feist, G. Grieco, and A. Groce. “Slither: A Static Analysis Framework for Smart Contracts”. In: *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. Montreal, QC, Canada, Canada: ACM/IEEE, 2019, pp. 8–15.
- [122] Qi Feng et al. “A survey on privacy protection in blockchain system”. In: *Journal of Network and Computer Applications* 126.2 (2018). ISSN: 1084-8045.
- [123] Mohamed Amine Ferrag et al. “Blockchain technologies for the internet of things: Research issues and challenges”. In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 2188–2204.
- [124] Ernestas Filatovas et al. “A MCDM-based framework for blockchain consensus protocol selection”. In: *Expert Systems with Applications* 204 (2022), p. 117609. ISSN: 0957-4174.
- [125] Martin Florian et al. “Erasing data from blockchain nodes”. In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. Stockholm, Sweden: IEEE, 2019, pp. 367–376.
- [126] forum openzeppelin forum. *Discuss about smart contract security patterns, vulnerabilities, hacks and best practices*. openzeppelin, Nov. 2020. URL: <https://forum.openzeppelin.com/c/security/25>.

- 
- [127] OWASP Foundation. *Open Web Application Security Project*. 2022. URL: <https://owasp.org/www-project-top-ten/>.
- [128] Xiang Fu, Huaimin Wang, and Peichang Shi. “A survey of Blockchain consensus algorithms: mechanism, design and applications”. In: *Science China Information Sciences* 64.2 (2021), pp. 1–15.
- [129] Yu-Long Gao et al. “A secure cryptocurrency scheme based on post-quantum blockchain”. In: *IEEE Access* 6 (2018), pp. 27205–27213.
- [130] Weichao Gao, William G Hatcher, and Wei Yu. “A survey of blockchain: techniques, applications, and challenges”. In: *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN)*. Hangzhou, China: IEEE, 2018, pp. 1–11.
- [131] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. “The bitcoin backbone protocol: Analysis and applications”. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2015, pp. 281–310.
- [132] Valentina Gatteschi et al. “Blockchain and smart contracts for insurance: Is the technology mature enough?” In: *Future Internet* 10.2 (2018), p. 20.
- [133] Peter Gaži, Aggelos Kiayias, and Alexander Russell. “Stake-bleeding attacks on proof-of-stake blockchains”. In: *Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 85–92.
- [134] Rosario Gennaro and Steven Goldfeder. “Fast multiparty threshold ecdsa with fast trustless setup”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1179–1194.
- [135] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. “Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security”. In: *Proceedings of the 2016*

- 
- International Conference on Applied Cryptography and Network Security*. Springer, 2016, pp. 156–174.
- [136] Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*. Vol. 20. 09. Stanford University Stanford, 2009.
- [137] Arthur Gervais et al. “On the security and performance of proof of work blockchains”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2016, pp. 3–16.
- [138] Ilias Giechaskiel, Cas Cremers, and Kasper Bonne Rasmussen. “On Bitcoin Security in the Presence of Broken Crypto Primitives.” In: *IACR Cryptology ePrint Archive 2016* (2016), p. 167.
- [139] Github. *GitHub - 1522402210/BlockChain-Security-List: BlockChain-Security-List*. 2018. URL: <https://bit.ly/3mM8SeR>.
- [140] Bill Gleim. *General Philosophy - Ethereum Smart Contract Best Practices*. 2017. URL: <https://bit.ly/3DpudAy>.
- [141] Steven Goldfeder et al. *Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme*. 2015. URL: <https://tinyurl.com/3p2p2s85>.
- [142] Steven Goldfeder et al. *Securing bitcoin wallets via threshold signatures*. 2014. URL: <https://tinyurl.com/k2cj4ee2>.
- [143] Rishab Goyal and Vipul Goyal. “Overcoming cryptographic impossibility results using blockchains”. In: *Proceedings of the 2017 Theory of Cryptography Conference*. Springer, 2017, pp. 529–561.
- [144] Matthew Green and Ian Miers. “Bolt: Anonymous payment channels for decentralized currencies”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 473–489.



- 
- [145] Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind. “Foundations and tools for the static analysis of ethereum smart contracts”. In: *International Conference on Computer Aided Verification*. Springer, 2018, pp. 51–78.
- [146] Jens Groth. “Short pairing-based non-interactive zero-knowledge arguments”. In: *Proceedings of the 2010 International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 321–340.
- [147] NCC Group. *Decentralized Application Security Project*. 2018. URL: <https://dasp.co/>.
- [148] Huaqun Guo and Xingjie Yu. “A survey on blockchain technology and its security”. In: *Blockchain: Research and Applications 3.2* (2022), p. 100067.
- [149] Empire Hacking. *Echidna: A Fast Smart Contract Fuzzer*. 2018. URL: <https://github.com/crytic/echidna>.
- [150] Alireza Toroghi Haghighat and Mehdi Shajari. “Block withholding game among bitcoin mining pools”. In: *Future Generation Computer Systems 97* (2019), pp. 482–491.
- [151] John H Hartman, Ian Murdock, and Tammo Spalink. “The Swarm scalable storage system”. In: *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*. IEEE, 1999, pp. 74–81.
- [152] Huru Hasanova et al. “A survey on blockchain cybersecurity vulnerabilities and possible countermeasures”. In: *International Journal of Network Management 29.2* (2019), e2060.
- [153] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. “Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions”. In: *Future Generation Computer Systems 97* (2019), pp. 512–529.
- [154] Cédric Hebert and Francesco Di Cerbo. “Secure blockchain in the enterprise: A methodology”. In: *Pervasive and Mobile Computing 59* (2019), p. 101038.

- 
- [155] Ethan Heilman et al. “Eclipse Attacks on Bitcoin’s Peer-to-Peer Network”. In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 129–144.
- [156] Jonathan Heiss, Jacob Eberhardt, and Stefan Tai. “From Oracles to Trustworthy Data On-Chaining Systems”. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. Atlanta, GA, USA: IEEE, 2019, pp. 496–503.
- [157] Everett Hildenbrandt et al. “Kevm: A complete formal semantics of the ethereum virtual machine”. In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE. 2018, pp. 204–217.
- [158] Yoichi Hirai. *Formal Verification of Deed Contract in Ethereum Name Service*. Tech. rep. 2016. URL: <https://bit.ly/3t7DHMd>.
- [159] Ivan Homoliak et al. “The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses”. In: *IEEE Communications Surveys & Tutorials* 23.1 (2020), pp. 341–390.
- [160] Yao-Chieh Hu et al. “Hierarchical interactions between ethereum smart contracts across testnets”. In: *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*. ACM, 2018, pp. 7–12.
- [161] Xin Huang et al. “Synthesizing Qualitative Research in Software Engineering: A Critical Review”. In: *Proceedings of the 40th International Conference on Software Engineering*. ICSE ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 1207–1218.
- [162] Pete Humiston. *Smart Contract Attacks [Part 2] - Ponzi Games Gone Wrong - By*. 2018. URL: <https://bit.ly/2WqVcew>.
- [163] IEC. *Information technology - Server management command line protocol*. ISO. Mar. 2011. URL: <https://www.iso.org/standard/53458.html>.

- 
- [164] iExec. *iExec Technical Documentation*. iExec. Mar. 2022. URL: <https://docs.iex.ec/>.
- [165] German Federal Office for Information Security: Bonn. *Towards Secure Blockchains*. 2019. URL: [https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Secure\\_Blockchain.html](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Secure_Blockchain.html).
- [166] KPMG International. *Securing the chain*. Tech. rep. 2017. URL: <https://vz.to/3kuHXS0>.
- [167] Clemente Izurieta and Mary Prouty. “Leveraging secdevops to tackle the technical debt associated with cybersecurity attack tactics”. In: *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE. 2019, pp. 33–37.
- [168] Clemente Izurieta et al. “A position study to investigate technical debt associated with security weaknesses”. In: *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 2018, pp. 138–142.
- [169] Bo Jiang, Ye Liu, and WK Chan. “Contractfuzzer: Fuzzing smart contracts for vulnerability detection”. In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018, pp. 259–269.
- [170] Hai Jin, Xiaohai Dai, and Jiang Xiao. “Towards a novel architecture for enabling interoperability amongst multiple blockchains”. In: *Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1203–1211.
- [171] Aram Jivanyan. “Lelantus: Towards Confidentiality and Anonymity of Blockchain Transactions from Standard Assumptions.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 373.
- [172] Aditya Joshi. *Modifying the Batch Size in Hyperledger Fabric v2.2*. 2020. URL: <https://bit.ly/3ve5zyM>.

- 
- [173] Maxim Jourenko et al. “SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 352.
- [174] JWWeatherman. *bitcoin security threat model*. 2018. URL: <https://bit.ly/2Y0xJkx>.
- [175] Satyanarayan Kar et al. *Risk Analysis of Blockchain Application for Aerospace Records Management*. Tech. rep. SAE Technical Paper, 2019.
- [176] Rick Kazman et al. “A case study in locating the architectural roots of technical debt”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. IEEE. 2015, pp. 179–188.
- [177] Staffs Keele et al. *Guidelines for performing systematic literature reviews in software engineering*. Tech. rep. Technical report, ver. 2.3 ebse technical report. ebse, 2007.
- [178] keywordseverywhere. *keywords every where*. keywordseverywhere. Mar. 2022. URL: <https://keywordseverywhere.com/>.
- [179] Javed Ali Khan et al. “Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique.” In: *International Journal of Modern Education & Computer Science* 7.11 (2015).
- [180] Aggelos Kiayias et al. “Ouroboros: A provably secure proof-of-stake blockchain protocol”. In: *Proceedings of the 2017 Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.
- [181] Lucianna Kiffer, Dave Levin, and Alan Mislove. “Stick a fork in it: Analyzing the Ethereum network partition”. In: *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 94–100.
- [182] Evgeniy O Kiktenko et al. “Quantum-secured blockchain”. In: *Quantum Science and Technology* 3.3 (2018), p. 035004.

- 
- [183] Chang Yeon Kim and Kyungho Lee. “Risk management to cryptocurrency exchange and investors guidelines to prevent potential threats”. In: *2018 international conference on platform technology and service (PlatCon)*. IEEE, 2018, pp. 1–6.
- [184] Eamonn Maguire Kiran Nagaraj. *Securing the chain*. 2017. URL: <https://assets.kpmg/content/dam/kpmg/xx/pdf/2017/05/securing-the-chain.pdf>.
- [185] KirstenS. *Category: Attack - OWASP*. 2010. URL: <https://bit.ly/3yukAwX>.
- [186] Barbara Kitchenham. “Procedures for performing systematic reviews”. In: *Keele, UK, Keele University* 33.2004 (2004), pp. 1–26.
- [187] Barbara Kitchenham, Stephen Linkman, and David Law. “DESMET: A method for evaluating software engineering methods and tools”. In: *Keele University* (1996).
- [188] Tim Klinger et al. “An enterprise perspective on technical debt”. In: *Proceedings of the 2nd Workshop on managing technical debt*. ACM, 2011, pp. 35–38.
- [189] Lukas König et al. “Comparing blockchain standards and recommendations”. In: *Future Internet* 12.12 (2020), p. 222.
- [190] Ahmed Kosba et al. “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts”. In: *Proceedings of the 2016 IEEE Symposium on Security and Privacy*. IEEE, 2016, pp. 839–858.
- [191] KPMG. *KPMG*. 2022. URL: <https://home.kpmg/xx/en/home.html>.
- [192] Mark Kreitz. “Security by Design in Software Engineering”. In: *SIGSOFT Softw. Eng. Notes* 44.3 (Oct. 2020), p. 23.
- [193] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. “Technical debt: From metaphor to theory and practice”. In: *Ieee software* 29.6 (2012), pp. 18–21.
- [194] Nir Kshetri. “Can blockchain strengthen the internet of things?” In: *IT professional* 19.4 (2017), pp. 68–72.

- 
- [195] Satpal Singh Kushwaha et al. “Ethereum Smart Contract Analysis Tools: A Systematic Review”. In: *IEEE Access* 10 (2022), pp. 57037–57062.
- [196] Alexandr Kuznetsov et al. “Performance of hash algorithms on gpus for use in blockchain”. In: *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*. IEEE, 2019, pp. 166–170.
- [197] Roy Lai and David LEE Kuo Chuen. “Blockchain—from public to private”. In: *Handbook of Blockchain, Digital Finance, and Inclusion, Volume 2*. Elsevier, 2018, pp. 145–177.
- [198] Leslie Lamport et al. “Paxos made simple”. In: *ACM Sigact News* 32.4 (2001), pp. 18–25.
- [199] David LeBlanc and Michael Howard. *Writing secure code*. Pearson Education, 2002.
- [200] Ledger. *e-commerce and marketing data breach*. 2020. URL: <https://bit.ly/2WI33EC>.
- [201] Sangsup Lee et al. “Who spent my {EOS}? on the (in) security of resource management of eos. io”. In: *13th {USENIX} Workshop on Offensive Technologies ({WOOT} 19)*. 2019.
- [202] Alexander Leid, Brink van der Merwe, and Willem Visser. “Testing Ethereum Smart Contracts: A Comparison of Symbolic Analysis and Fuzz Testing Tools”. In: *Conference of the South African Institute of Computer Scientists and Information Technologists 2020*. 2020, pp. 35–43.
- [203] Jiewu Leng et al. “Blockchain Security: A Survey of Techniques and Research Directions”. In: *IEEE Transactions on Services Computing* 15.4 (2022), pp. 2490–2510.
- [204] Kaijun Leng et al. “Research on agricultural supply chain system with double chain architecture based on blockchain technology”. In: *Future Generation Computer Systems* 86 (2018), pp. 641–649.

- 
- [205] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. “Inclusive block chain protocols”. In: *Proceedings of the 2015 International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.
- [206] Chao-Yang Li et al. “A New Lattice-Based Signature Scheme in Post-Quantum Blockchain Network”. In: *IEEE Access* 7 (2018), pp. 2026–2033.
- [207] Daming Li et al. “Information security model of block chain based on intrusion sensing in the IoT environment”. In: *Cluster Computing* 22.1 (2019), pp. 451–468.
- [208] Kai Li et al. “As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service”. In: *NDSS*. NDSS, 2021.
- [209] Wenting Li et al. “Securing proof-of-stake blockchain protocols”. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 297–315.
- [210] Wenting Li et al. “Towards scalable and private industrial blockchains”. In: *Proceedings of the 2017 ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 9–14.
- [211] Xiaoqi Li et al. “A survey on the security of blockchain systems”. In: *Future Generation Computer Systems* 107.4 (2017).
- [212] Xiaoqi Li et al. “A survey on the security of blockchain systems”. In: *Future Generation Computer Systems* 107 (2020), pp. 841–853.
- [213] Zengyang Li, Paris Avgeriou, and Peng Liang. “A systematic mapping study on technical debt and its management”. In: *Journal of Systems and Software* 101 (2015), pp. 193–220.
- [214] Zengyang Li, Peng Liang, and Paris Avgeriou. “Architectural debt management in value-oriented architecting”. In: *Economics-Driven Software Architecture*. Elsevier, 2014, pp. 183–204.

- 
- [215] Chao Lin et al. “BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0”. In: *Journal of Network and Computer Applications* 116 (2018), pp. 42–52.
- [216] Fei Lin and Minqian Qiang. “The Challenges of Existence, Status, and Value for Improving Blockchain”. In: *IEEE Access* 7 (2018), pp. 7747–7758.
- [217] Iuon-Chang Lin and Tzu-Chun Liao. “A Survey of Blockchain Security Issues and Challenges.” In: *IJ Network Security* 19.5 (2017), pp. 653–659.
- [218] Jing Liu and Zhentian Liu. “A survey on security verification of blockchain smart contracts”. In: *IEEE Access* 7 (2019), pp. 77894–77904.
- [219] Yue Liu et al. “Applying design patterns in smart contracts”. In: *Proceedings of the 2018 International Conference on Blockchain*. Springer, 2018, pp. 92–106.
- [220] Yue Liu et al. “Design patterns for blockchain-based self-sovereign identity”. In: *Proceedings of the European Conference on Pattern Languages of Programs 2020*. 2020, pp. 1–14.
- [221] Sin Kuang Lo et al. “Reliability analysis for blockchain oracles”. In: *Computers & Electrical Engineering* 83 (2020), p. 106582.
- [222] Matthias Lohr and Sven Peldszus. “Maintenance of Long-Living Smart Contracts.” In: *Software Engineering (Workshops)*. Innsbruck, Österreich: ceur-ws, 2020.
- [223] Yang Lu. “The blockchain: State-of-the-art and research challenges”. In: *Journal of Industrial Information Integration* 15 (2019). ISSN: 2452-414X.
- [224] Christian Lundkvist et al. “Uport: A platform for self-sovereign identity”. In: *short-url.at/ktS08* (2017).
- [225] Loi Luu et al. “Making smart contracts smarter”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. Vienna Austria: ACM, 2016, pp. 254–269.



- 
- [226] Loi Luu et al. “Making smart contracts smarter”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 254–269.
- [227] Loi Luu et al. “On power splitting games in distributed computation: The case of bitcoin pooled mining”. In: *Proceedings of the 2015 IEEE 28th Computer Security Foundations Symposium*. IEEE, 2015, pp. 397–411.
- [228] Beverley Mackenzie, Robert Ian Ferguson, and Xavier Bellekens. “An assessment of blockchain consensus protocols for the Internet of Things”. In: *2018 International Conference on Internet of Things, Embedded Systems and Communications (IIN-TEC)*. IEEE. 2018, pp. 183–190.
- [229] Daniel Macrinici, Cristian Cartoceanu, and Shang Gao. “Smart contract applications within blockchain technology: A systematic mapping study”. In: *Telematics and Informatics* 35 (8 2018).
- [230] Mrinmoy Majumder. “Multi criteria decision making”. In: *Impact of urbanization on water shortage in face of climatic aberrations*. Springer, 2015, pp. 35–47.
- [231] Imran Makhdoom et al. “Blockchain’s adoption in IoT: The challenges, and a way forward”. In: *Journal of Network and Computer Applications* 125 (2018).
- [232] Kamran Mammadzada et al. “Blockchain oracles: A framework for blockchain-based applications”. In: *International Conference on Business Process Management*. Springer, 2020, pp. 19–34.
- [233] Christopher Mann and Daniel Loebenberger. “Two-factor authentication for the Bitcoin protocol”. In: *International Journal of Information Security* 16.2 (2017), pp. 213–226.
- [234] Adrian Manning. *solidity-security-blog*. 2019. URL: <https://github.com/sigp/solidity-security-blog>.

- 
- [235] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. “Low-Resource Eclipse Attacks on Ethereum’s Peer-to-Peer Network.” In: *IACR Cryptology ePrint Archive 2018* (2018), p. 236.
- [236] Antonio Martini, Jan Bosch, and Michel Chaudron. “Architecture Technical Debt: Understanding Causes and a Qualitative Model”. In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. Verona, Italy: IEEE, 2014, pp. 85–92.
- [237] Hartwig Mayer. “ECDSA security in bitcoin and ethereum: a research survey”. In: *CoinFabrik*, June 28 (2016).
- [238] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. “Refund attacks on Bitcoin’s payment protocol”. In: *Proceedings of the 2016 International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 581–599.
- [239] Alexander Mense and Markus Flatscher. “Security Vulnerabilities in Ethereum Smart Contracts”. In: *Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services*. ACM, 2018, pp. 375–380.
- [240] Carlos Mera-Gómez et al. “A Multi-Agent Elasticity Management Based On Multi-Tenant Debt Exchanges”. In: *Proceedings of the 12th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2018)*. IEEE. 2018.
- [241] Andrew Miller et al. “Permacoin: Repurposing bitcoin work for data preservation”. In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 475–490.
- [242] Mitar Milutinovic et al. “Proof of luck: An efficient blockchain consensus protocol”. In: *proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM, 2016, p. 2.
- [243] MITRE. *Common Weakness Enumeration*. 2006. URL: <https://cwe.mitre.org/>.

- 
- [244] MITRE. *Common Weakness Scoring System*. 2014. URL: <https://cwe.mitre.org/cwss/>.
- [245] mitre. *MITRE ATT&CK*. 2018. URL: <https://attack.mitre.org/>.
- [246] Carlos Molina-Jimenez et al. “Implementation of smart contracts using hybrid architectures with on and off-blockchain components”. In: *Proceedings of the 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*. IEEE, 2018, pp. 83–90.
- [247] Giacomo Morganti, Enrico Schiavone, and Andrea Bondavalli. “Risk assessment of blockchain technology”. In: *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*. IEEE. 2018, pp. 87–96.
- [248] Mark Mossberg et al. “Manticore: A user-friendly symbolic execution framework for binaries and smart contracts”. In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. San Diego, California, United States: IEEE, 2019, pp. 1186–1189.
- [249] Bernhard Mueller. “Smashing ethereum smart contracts for fun and real profit”. In: *HITB SECCONF Amsterdam 9* (2018), p. 54.
- [250] Francis Jee Nadia Hewett Sumedha Deshmukh. *Cybersecurity*. 2020. URL: <https://widgets.weforum.org/blockchain-toolkit/cybersecurity>.
- [251] Satoshi Nakamoto et al. “Bitcoin: A peer-to-peer electronic cash system”. In: (2008). URL: <https://rb.gy/lk0e98>.
- [252] Christopher Natoli and Vincent Gramoli. “The blockchain anomaly”. In: *Proceedings of the 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2016, pp. 310–317.
- [253] Giang-Truong Nguyen and Kyungbaek Kim. “A Survey about Consensus Algorithms Used in Blockchain.” In: *Journal of Information processing systems* 14.1 (2018).

- 
- [254] Tai D. Nguyen et al. “SFuzz: An Efficient Adaptive Fuzzer for Solidity Smart Contracts”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 778–788. ISBN: 9781450371216.
- [255] Robert C Nickerson, Upkar Varshney, and Jan Muntermann. “A method for taxonomy development and its application in information systems”. In: *European Journal of Information Systems* 22.3 (2013), pp. 336–359.
- [256] Ivica Nikolić et al. “Finding the greedy, prodigal, and suicidal contracts at scale”. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 2018, pp. 653–663.
- [257] nist. *STANDARDS & MEASUREMENTS*. 2022. URL: <https://www.nist.gov/>.
- [258] nodejs. *Vulnerabilities / Node.js*. 2019. URL: <https://bit.ly/2Y0wGB7>.
- [259] Shen Noether. “Ring Signature Confidential Transactions for Monero.” In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1098.
- [260] Robert L Nord et al. “Can knowledge of technical debt help identify software vulnerabilities?” In: *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*. 2016.
- [261] Organisation internationale de normalisation. *Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): System and Software Quality Models*. ISO/IEC, 2011.
- [262] Russell O’Connor. “Simplicity: A new language for blockchains”. In: *Proceedings of the 2017 Workshop on Programming Languages and Analysis for Security*. ACM, 2017, pp. 107–120.

- 
- [263] Gustavo A Oliva, Ahmed E Hassan, and Zhen Ming Jack Jiang. “An exploratory study of smart contracts in the Ethereum blockchain platform”. In: *Empirical Software Engineering* (2020), pp. 1–41.
- [264] Ilhaam A. Omar et al. “Implementing decentralized auctions using blockchain smart contracts”. In: *Technological Forecasting and Social Change* 168 (2021), p. 120786. ISSN: 0040-1625.
- [265] orbit. *orbit-db: Peer-to-Peer Databases for the Decentralized Web*. 2017. URL: <https://bit.ly/3zwVcYI>.
- [266] Pim Otte, Martijn de Vos, and Johan Pouwelse. “TrustChain: A Sybil-resistant scalable blockchain”. In: *Future Generation Computer Systems* 107 (2017).
- [267] owasp. *OWASP Risk Rating Methodology*. owasp. Mar. 2019. URL: [https://owasp.org/www-community/OWASP\\_Risk\\_Rating\\_Methodology](https://owasp.org/www-community/OWASP_Risk_Rating_Methodology).
- [268] Paralink. *Paralink Network*. Paralink. Dec. 2021. URL: <https://cutt.ly/0DPtJY4>.
- [269] Reza M Parizi, Ali Dehghantanha, et al. “Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security”. In: *Proceedings of the 2018 International Conference on Blockchain*. Springer, 2018, pp. 75–91.
- [270] Reza M. Parizi et al. “Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains”. In: *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. CASCON ’18. Markham, Ontario, Canada: IBM Corp., 2018, pp. 103–113.
- [271] Amirmohammad Pasdar, Zhongli Dong, and Young Choon Lee. “Blockchain Oracle Design Patterns”. In: *arXiv preprint arXiv:2106.09349* (2021).
- [272] Celeste Lyn Paul. “A modified Delphi approach to a new card sorting methodology”. In: *Journal of Usability studies* 4.1 (2008), pp. 7–30.

- 
- [273] PeckShield. *Cheese Bank Incident: Root Cause Analysis*. medium. Dec. 2020. URL: [shorturl.at/zDX38](https://shorturl.at/zDX38).
- [274] Adán Sánchez de Pedro, Daniele Levi, and Luis Iván Cuende. “Witnet: A decentralized oracle network protocol”. In: *arXiv preprint arXiv:1711.09756* (2017).
- [275] Ken Peffers et al. “A design science research methodology for information systems research”. In: *Journal of management information systems* 24.3 (2007), pp. 45–77.
- [276] Simone Porru et al. “Blockchain-oriented software engineering: challenges and new directions”. In: *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. Buenos Aires, Argentina: IEEE, 2017, pp. 169–171.
- [277] portswigger. *Latest cryptocurrency security news*. 2021. URL: <https://bit.ly/38nQL6h>.
- [278] Purathani Praitheeshan et al. “Security analysis methods on Ethereum smart contract vulnerabilities: A survey”. In: *arXiv preprint arXiv:1908.08605* (2019).
- [279] Abhishek Biswas Prakash Santhana. *Blockchain risk management Risk functions need to play an active role in shaping blockchain strategy*. 2017. URL: <https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-fsi-blockchain-risk-management.pdf>.
- [280] ProtocolLabs. *Filecoin: A Decentralized Storage Network*. 2017. URL: <https://filecoin.io/filecoin.pdf>.
- [281] Protofire. *solhint*. 2020. URL: <https://github.com/protofire/solhint>.
- [282] Provable. *Provable Documentation*. 2019. URL: <https://docs.provable.xyz/>.
- [283] Rui Qiao et al. “Optimization of dynamic data traceability mechanism in Internet of Things based on consortium blockchain”. In: *International Journal of Distributed Sensor Networks* 14.12 (2018), p. 1550147718819072.

- 
- [284] Kaihua Qin et al. “Attacking the DeFi ecosystem with flash loans for fun and profit”. In: *arXiv preprint arXiv:2003.03810* (2020).
- [285] Samuel T Redwine Jr and William E Riddle. “Software technology maturation”. In: *Proceedings of the 8th international conference on Software engineering*. IEEE Computer Society Press, 1985, pp. 189–200.
- [286] Emanuel Regnath and Sebastian Steinhorst. “LeapChain: efficient blockchain verification for embedded IoT”. In: *Proceedings of the 2018 International Conference on Computer-Aided Design*. New York, NY, USA: ACM, 2018, p. 74.
- [287] Raine Revere. *solgraph*. 2015. URL: <https://github.com/raineorshine/solgraph>.
- [288] Yos Riady. *Common Smart Contract Vulnerabilities and How To Mitigate Them*. Yos Riady, Oct. 2018. URL: <https://yos.io/2018/10/20/smart-contract-vulnerabilities-and-how-to-mitigate-them/>.
- [289] Richard Richard et al. “Smart Contract Development Model and the Future of Blockchain Technology”. In: *2020 the 3rd International Conference on Blockchain Technology and Applications*. 2020, pp. 34–39.
- [290] Eric Dashofy Richard N. Taylor Nenad Medvidovic. *Software Architecture: Foundations, Theory, and Practice*. Addison-Wesley Professional, 2009.
- [291] Kalle Rindell, Karin Bernsmed, and Martin Gilje Jaatun. “Managing Security in Software: Or: How I Learned to Stop Worrying and Manage the Security Technical Debt”. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ARES ’19. Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019.
- [292] Kalle Rindell and Johannes Holvitie. “Security risk assessment and management as technical debt”. In: *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE. 2019, pp. 1–8.

- 
- [293] Niall Roche and Alastair P Moore. “Oraclised Data Schemas: Improving contractual Certainty in uncertain Times”. PhD thesis. London University; UCL Centre for Blockchain Technologies, 2020.
- [294] Sara Rouhani and Ralph Deters. “Security, performance, and applications of smart contracts: A systematic survey”. In: *IEEE Access* 7 (2019), pp. 50759–50779.
- [295] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. “Coinshuffle: Practical decentralized coin mixing for bitcoin”. In: *Proceedings of the 2014 European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.
- [296] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14.2 (2009), pp. 131–164.
- [297] Jungwoo Ryoo, Rick Kazman, and Priya Anand. “Architectural analysis for security”. In: *IEEE Security & Privacy* 13.6 (2015), pp. 52–59.
- [298] Muhammad Saad et al. “Countering selfish mining in blockchains”. In: *Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 360–364.
- [299] Muhammad Saad et al. “Exploring the attack surface of blockchain: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* 22.3 (2020), pp. 1977–2008.
- [300] Muhammad Saad et al. “Exploring the attack surface of blockchain: A systematic overview”. In: *arXiv preprint arXiv:1904.03487* (2019).
- [301] K. Salah et al. “Blockchain for AI: Review and Open Research Challenges”. In: *IEEE Access* 7 (2019), pp. 10127–10149.
- [302] Joanna CS Santos, Katy Tarrit, and Mehdi Mirakhorli. “A catalog of security architecture weaknesses”. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017, pp. 220–223.



- 
- [303] Gohar Sargsyan et al. “Blockchain security by design framework for trust and adoption in IoT environment”. In: *2019 IEEE world congress on services (SERVICES)*. Vol. 2642. IEEE. 2019, pp. 15–20.
- [304] Eli Ben Sasson et al. “Zerocash: Decentralized anonymous payments from bitcoin”. In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE, 2014, pp. 459–474.
- [305] Masashi Sato and Shin’ichiro Matsuo. “Long-term public blockchain: Resilience against compromise of underlying cryptography”. In: *Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN)*. Vancouver, BC, Canada: IEEE, 2017, pp. 1–8.
- [306] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. “Smart Contract: Attacks and Protections”. In: *IEEE Access* 8 (2020), pp. 24416–24427.
- [307] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. “Smart contract: Attacks and protections”. In: *IEEE Access* 8 (2020), pp. 24416–24427.
- [308] Vincent Schlatt et al. “Attacking the trust machine: Developing an information systems research agenda for blockchain cybersecurity”. In: *International Journal of Information Management* (2022), p. 102470. ISSN: 0268-4012.
- [309] Franklin Schrans, Susan Eisenbach, and Sophia Drossopoulou. “Writing safe smart contracts in flint”. In: *Proceedings of 2nd International Conference on Art, Science, and Engineering of Programming*. ACM, 2018, pp. 218–219.
- [310] David Schwartz, Noah Youngs, Arthur Britto, et al. “The ripple protocol consensus algorithm”. In: *Ripple Labs Inc White Paper* 5 (2014), p. 8.
- [311] ICO Security. *blog.positive*. 2017. URL: <https://blog.positive.com/>.
- [312] Graham Sh. “Hyperledger Fabric version: 1.1 Assessment Technical Report”. In: *Hyperledger Fabric* (2017).

- 
- [313] Mary Shaw. “What makes good research in software engineering?” In: *International Journal on Software Tools for Technology Transfer* 4.1 (2002), pp. 1–7.
- [314] slowmist. *EOS DApp total loss money by hacked is about*. 2021. URL: <https://bit.ly/3ynTKX1>.
- [315] SmartContractSecurity. *SWC Registry*. 2020. URL: <https://swcregistry.io/>.
- [316] Yonatan Sompolinsky and Aviv Zohar. “Secure high-rate transaction processing in bitcoin”. In: *Proceedings of the 2015 International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.
- [317] Leonardo Sousa et al. “Identifying design problems in the source code: A grounded theory”. In: *Proceedings of the 40th International Conference on Software Engineering*. 2018, pp. 921–931.
- [318] Mirko Staderini, Enrico Schiavone, and Andrea Bondavalli. “A requirements-driven methodology for the proper selection and configuration of blockchains”. In: *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2018, pp. 201–206.
- [319] C Stathakopoulous and Christian Cachin. “Threshold signatures for blockchain systems”. In: *Swiss Federal Institute of Technology* (2017). URL: <https://rb.gy/rqgvk3>.
- [320] Stephanie. *What is Cohen’s Kappa Statistic?* Statistics How To. May 2014. URL: <https://www.statisticshowto.com/cohens-kappa-statistic/>.
- [321] Matt Suiche. *porosity*. 2017. URL: <https://github.com/comaeio/porosity>.
- [322] Harish Sukhwani et al. “Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)”. In: *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2018, pp. 1–8.
- [323] Harish Sukhwani et al. “Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)”. In: *Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017, pp. 253–255.

- 
- [324] He Sun et al. “Multi-blockchain model for central bank digital currency”. In: *Proceedings of the 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. IEEE, 2017, pp. 360–367.
- [325] Don Tapscott and Alex Tapscott. “How blockchain will change addresss”. In: *MIT Sloan Management Review* 58.2 (2017), p. 10.
- [326] openzeppelin team. *ethernaut*. 2019. URL: <https://ethernaut.openzeppelin.com/>.
- [327] American Council for Technology-Industry Advisory Council. *Blockchain Playbook Online - beta*. 2021. URL: <https://rb.gy/g5ci8e>.
- [328] Tellor. *Tellor: A decentralized Oracle*. Tellor. Mar. 2021. URL: <https://docs.tellor.io/tellor/>.
- [329] Julien Thevenard. *Decentralised Oracles: a comprehensive overview*. medium. Dec. 2019. URL: [shorturl.at/mxFL7](https://shorturl.at/mxFL7).
- [330] ThomasKur. *Threats and Countermeasures / Microsoft Docs*. 2018. URL: <https://bit.ly/38i3Pu6>.
- [331] Sergei Tikhomirov et al. “Smartcheck: Static analysis of ethereum smart contracts”. In: *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*. Gothenburg Sweden: ACM/IEEE, 2018, pp. 9–16.
- [332] Edith Tom, AybüKe Aurum, and Richard Vidgen. “An exploration of technical debt”. In: *Journal of Systems and Software* 86.6 (2013), pp. 1498–1516.
- [333] Edith Tom, AybüKe Aurum, and Richard Vidgen. “An exploration of technical debt”. In: *Journal of Systems and Software* 86.6 (2013), pp. 1498–1516.
- [334] Christof Ferreira Torres, Julian Schütte, and Radu State. “Osiris: Hunting for integer bugs in ethereum smart contracts”. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. San Juan PR USA: ACM, 2018, pp. 664–676.

- 
- [335] Muoi Tran et al. “A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network”. In: *Proceedings of IEEE Symposium on Security and Privacy (IEEE S&P)*. San Francisco, CA, USA, 2020.
- [336] Nguyen Khoi Tran and M Ali Babar. “Anatomy, concept, and design space of blockchain networks”. In: *2020 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2020, pp. 125–134.
- [337] Evangelos Triantaphyllou et al. “Multi-criteria decision making: an operations research approach”. In: *Encyclopedia of electrical and electronics engineering* 15.1998 (1998), pp. 175–186.
- [338] Petar Tsankov et al. “Securify: Practical Security Analysis of Smart Contracts”. In: *18 Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 67–82.
- [339] Petar Tsankov et al. “Securify: Practical security analysis of smart contracts”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 67–82.
- [340] Petar Tsankov et al. “Securify: Practical security analysis of smart contracts”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 67–82.
- [341] Hanny Tufail et al. “Towards the selection of Optimum Requirements Prioritization Technique: A Comparative Analysis”. In: *2019 5th International Conference on Information Management (ICIM)*. 2019, pp. 227–231.
- [342] Muhammad Usman et al. “Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method”. In: *Information and Software Technology* 85 (2017), pp. 43–59.

- 
- [343] Anna Vacca et al. “A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges”. In: *Journal of Systems and Software* 174 (2021), p. 110891.
- [344] Dejan Vujičić, Dijana Jagodić, and Siniša Randić. “Blockchain technology, bitcoin, and Ethereum: A brief overview”. In: *Proceedings of the 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. IEEE, 2018, pp. 1–6.
- [345] Marko Vukolić. “Rethinking permissioned blockchains”. In: *Proceedings of the 2017 ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. New York, NY, USA: ACM, 2017, pp. 3–7.
- [346] Vyper. *Vyper*. 2019. URL: <https://bit.ly/38mnVmY>.
- [347] Z Wahid and N Nadir. “Improvement of one factor at a time through design of experiments”. In: *World Applied Sciences Journal* 21.1 (2013), pp. 56–61.
- [348] Niall Roche Walter Hernandez. *An Oracle to allow Pandemic-aware Policies*. bitbucket. Mar. 2020. URL: <https://github.com/niallroche/covidhack-oracle-provable>.
- [349] Niall Roche Walter Hernandez. *An Oracle to allow Pandemic-aware Policies*. bitbucket. Mar. 2020. URL: <https://devpost.com/software/covidhack-oracle-provable>.
- [350] Niall Roche Walter Hernandez. *Oracle Data Lexicon – Bringing contractual certainty in uncertain time*. bitbucket. Mar. 2020. URL: <https://challenge.globallegalhackathon.com/gallery/5ec84aef202da60044c03d6b>.
- [351] Zhiyuan Wan, Xin Xia, and Ahmed E Hassan. “What do programmers discuss about blockchain?” In: *IEEE Transactions on Software Engineering* 07 (2021), pp. 1331–1349.
- [352] Zhiyuan Wan et al. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1410–1422.

- 
- [353] Zhiyuan Wan et al. “Smart Contract Security: A Practitioners’ Perspective”. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 2021, pp. 1410–1422.
- [354] Hai Wang et al. “An Overview of Blockchain Security Analysis”. In: *Proceedings of the 2018 China Cyber Security Annual Conference*. Springer, 2018, pp. 55–72.
- [355] Licheng Wang et al. “Cryptographic primitives in blockchains”. In: *Journal of Network and Computer Applications* 127 (2019), pp. 43–58.
- [356] Shuai Wang et al. “Blockchain-enabled smart contracts: architecture, applications, and future trends”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.11 (2019), pp. 2266–2277.
- [357] Wenbo Wang et al. “A survey on consensus mechanisms and mining strategy management in blockchain networks”. In: *IEEE Access* 7 (2019), pp. 22328–22370.
- [358] Yunsen Wang and Alexander Kogan. “Designing confidentiality-preserving Blockchain-based transaction processing systems”. In: *International Journal of Accounting Information Systems* 30 (2018), pp. 1–18.
- [359] Jon Whittle, Duminda Wijesekera, and Mark Hartong. “Executable misuse cases for modeling security concerns”. In: *Proceedings of the 30th international conference on Software engineering*. ACM, 2008, pp. 121–130.
- [360] github wiki. *List of Security Vulnerabilities*. github, Apr. 2019. URL: <https://github.com/runtimeverification/verified-smart-contracts/wiki/List-of-Security-Vulnerabilities>.
- [361] Claes Wohlin. “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE ’14. London, England, United Kingdom: Association for Computing Machinery, 2014.

- 
- [362] Claes Wohlin et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [363] Maximilian Wohrer and Uwe Zdun. “Smart contracts: Security patterns in the ethereum ecosystem and solidity”. In: *Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2018, pp. 2–8.
- [364] Maximilian Wohrer and Uwe Zdun. “Smart contracts: security patterns in the ethereum ecosystem and solidity”. In: *Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. Campobasso, Italy: IEEE, 2018, pp. 2–8.
- [365] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper 151.2014* (2014), pp. 1–32.
- [366] Jing Wu et al. “A New Sustainable Interchain Design on Transport Layer for Blockchain”. In: *Proceedings of the 2018 International Conference on Smart Blockchain*. Springer, 2018, pp. 12–21.
- [367] Lijun Wu et al. “Democratic centralism: a hybrid blockchain architecture and its applications in energy internet”. In: *2017 IEEE International Conference on Energy Internet (ICEI)*. IEEE, 2017, pp. 176–181.
- [368] Karl Wüst and Arthur Gervais. “Do you need a Blockchain?” In: *Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology*. IEEE, 2018, pp. 45–54.
- [369] Karl Wüst and Arthur Gervais. “Do you need a blockchain?” In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. 2018, pp. 45–54.
- [370] Valentin Wüstholtz and Maria Christakis. “Learning inputs in greybox fuzzing”. In: *arXiv preprint arXiv:1807.07875* (2018).
- [371] Jennifer J Xu. “Are blockchains immune to all malicious attacks?” In: *Financial Innovation* 2.1 (2016), p. 25.

- 
- [372] Quanqing Xu et al. “Building an Ethereum and IPFS-Based Decentralized Social Network System”. In: *Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 1–6.
- [373] Xiwei Xu et al. “A decision model for choosing patterns in blockchain-based applications”. In: *2021 IEEE 18th International Conference on Software Architecture (ICSA)*. IEEE, 2021, pp. 47–57.
- [374] Xiwei Xu et al. “A Pattern Collection for Blockchain-Based Applications”. In: *Proceedings of the 23rd European Conference on Pattern Languages of Programs*. EuroPLoP ’18. Irsee, Germany: Association for Computing Machinery, 2018.
- [375] Xiwei Xu et al. “A pattern collection for blockchain-based applications”. In: *Proceedings of the 23rd European Conference on Pattern Languages of Programs*. ACM, 2018, p. 3.
- [376] Xiwei Xu et al. “A taxonomy of blockchain-based systems for architecture design”. In: *Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 243–252.
- [377] Xiwei Xu et al. “Designing blockchain-based applications a case study for imported product traceability”. In: *Future Generation Computer Systems* 92 (2019), pp. 399–406.
- [378] Xiwei Xu et al. “The blockchain as a software connector”. In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 182–191.
- [379] Xiwei Xu et al. “The blockchain as a software connector”. In: *Proceedings of the 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 182–191.



- 
- [380] Dylan Yaga et al. “Blockchain technology overview”. In: *arXiv preprint arXiv:1906.11078* (2019).
- [381] Kazuhiro Yamashita et al. “Potential risks of hyperledger fabric smart contracts”. In: *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2019, pp. 1–10.
- [382] Wendy Yáñez et al. “Architecting Internet of Things Systems with Blockchain: A Catalog of Tactics”. In: 30.3 (Apr. 2021).
- [383] Jinhong Yang et al. “Proof-of-Familiarity: A Privacy-Preserved Blockchain Scheme for Collaborative Medical Decision-Making”. In: *Applied Sciences* 9.7 (2019), p. 1370.
- [384] Lanxin Yang et al. “Quality Assessment in Systematic Literature Reviews: A Software Engineering Perspective”. In: *Information and Software Technology* 130 (2021), p. 106397.
- [385] Xin-She Yang. “Chapter 14 - Multi-Objective Optimization”. In: *Nature-Inspired Optimization Algorithms*. Ed. by Xin-She Yang. Oxford: Elsevier, 2014, pp. 197–211.
- [386] Zhe Yang et al. “Blockchain-based decentralized trust management in vehicular networks”. In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 1495–1505.
- [387] Zheng Yang and Hang Lei. “Formal process virtual machine for smart contracts verification”. In: *arXiv preprint arXiv:1805.00808* (2018).
- [388] Udi Yavo. *Design Vulnerabilities: They Hide and You can't Catch Them*. infosecurity, June 2016. URL: <https://www.infosecurity-magazine.com/opinions/design-vulnerabilities-hide-you/>.
- [389] Congcong Ye et al. “Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting”. In: *Proceedings of the 2018 5th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2018, pp. 15–24.

- 
- [390] Wei Yin et al. “An anti-quantum transaction authentication approach in blockchain”. In: *IEEE Access* 6 (2018), pp. 5393–5401.
- [391] Jesse Yli-Huumo et al. “Where is current research on blockchain technology?—a systematic review”. In: *PloS one* 11.10 (2016), e0163477.
- [392] Xiao Liang Yu et al. “Smart Contract Repair”. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29.4 (2020), pp. 1–32.
- [393] Yong Yuan and Fei-Yue Wang. “Blockchain and cryptocurrencies: Model, techniques, and applications”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48.9 (2018), pp. 1421–1428.
- [394] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. “Rapidchain: Scaling blockchain via full sharding”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 931–948.
- [395] Martin Zeilinger. “Digital art as ‘monetised graphics’: Enforcing intellectual property on the blockchain”. In: *Philosophy & Technology* 31.1 (2018), pp. 15–41.
- [396] Fan Zhang. *Liberating web data using DECO, a privacy-preserving oracle protocol*. 2019. URL: <https://bit.ly/2UXD0IA>.
- [397] Fan Zhang et al. “Town Crier: An Authenticated Data Feed for Smart Contracts”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 270–282.
- [398] Peiyun Zhang and Mengchu Zhou. “Security and trust in blockchains: Architecture, key technologies, and open issues”. In: *IEEE Transactions on Computational Social Systems* 7.3 (2020), pp. 790–801.
- [399] Rui Zhang, Rui Xue, and Ling Liu. “Security and Privacy on Blockchain”. In: *arXiv preprint arXiv:1903.07602* (2019).

- 
- [400] Yahui Zhang et al. “Survey of Attacks and Defenses against SGX”. In: *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*. Chongqing, China: IEEE, 2020, pp. 1492–1496.
- [401] Qihong Zheng et al. “An innovative IPFS-based storage model for blockchain”. In: *Proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2018, pp. 704–708.
- [402] Zibin Zheng et al. “An overview of blockchain technology: Architecture, consensus, and future trends”. In: *Proceedings of the 6th International Congress on Big Data*. IEEE, 2017, pp. 557–564.
- [403] Zibin Zheng et al. “Blockchain challenges and opportunities: A survey”. In: *Work Pap.-2016* (2016). URL: <https://rb.gy/btg0vl>.
- [404] Zibin Zheng et al. “Blockchain challenges and opportunities: A survey”. In: *International Journal of Web and Grid Services* 14.4 (2018), pp. 352–375.
- [405] Lin Zhong et al. “A secure versatile light payment system based on blockchain”. In: *Future Generation Computer Systems* 93 (2019), pp. 327–337.
- [406] Tong Zhou, Xiaofeng Li, and He Zhao. “DLattice: A Permission-Less Blockchain Based on DPoS-BA-DAG Consensus for Data Tokenization”. In: *IEEE Access* 7 (2019), pp. 39273–39287.
- [407] Liehuang Zhu et al. “Controllable and trustworthy blockchain-based cloud data management”. In: *Future Generation Computer Systems* 91 (2019), pp. 527–535.
- [408] Kaden Zipfel. *New Smart Contract Weakness: Hash Collisions With Multiple Variable Length Arguments*. Medium, Jan. 2019. URL: <https://medium.com/swlh/new-smart-contract-weakness-hash-collisions-with-multiple-variable-length-arguments-dc7b9c84e493>.

- [409] Weiqin Zou et al. “Smart Contract Development: Challenges and Opportunities”. In: *IEEE Transactions on Software Engineering* 47.10 (2021), pp. 2084–2106.
- [410] Weiqin Zou et al. “Smart contract development: Challenges and opportunities”. In: *IEEE Transactions on Software Engineering* 47.10 (2019).