DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
36

MUBASHAR IQBAL

Reference Framework for Managing Security Risks Using Blockchain

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
36

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
36

MUBASHAR IQBAL

Reference Framework for Managing Security Risks Using Blockchain

UNIVERSITY OF TARTU
Press
1632

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on August 25, 2022 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof. PhD. Raimundas Matulevičius
Institute of Computer Science
University of Tartu, Tartu, Estonia

Opponents

Prof. PhD. Agnes Koschmider
Institutes of the Faculty of Engineering
Department of Computer Science
Kiel University, Kiel, Germany

Assoc. Prof. PhD. Hans Weigand
Tilburg School of Economics and Management (TiSEM)
Tilburg University, Tilburg, Netherlands

The public defense will take place on September 30, 2022 at 12:15 in Narva mnt 18-1021, and also via Zoom.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9916-27-007-3 (print)
ISBN 978-9916-27-008-0 (PDF)

University of Tartu Press
`http://www.tyk.ee/`

To my family and friends

ABSTRACT

Security risk management (SRM) is concerned with lowering the risk of intentional unauthorized harm to valuable assets to that level that is acceptable to the system's stakeholders by preventing and reacting to malicious harm, misuse, threats, and risks. Managing security risks includes identifying, analyzing, and treating security threats to maintain the system's confidentiality, integrity, and availability. Various programs (e.g., OWASP), threat models (e.g., STRIDE), security risk management models (e.g., ISSRM), and regulations (e.g., GDPR) exist to communicate and reduce the security threats to build secure software. However, the security threats continuously evolve because the traditional technology infrastructure does not implement security measures by design. Despite the increasing number of security incidents, many organizations do not prioritize software security. Blockchain infrastructure is a befitting solution to mitigate traditional applications' security threats (e.g., data tampering, data theft, single-point failure). However, there exist no unified and formal knowledge models that can support the SRM and a clear explanation of how the blockchain-based applications can mitigate the security threats of traditional applications.

Moreover, the blockchain domain is constantly evolving, providing new techniques and often interchangeable design concepts, resulting in conceptual ambiguity and confusion in treating the security threats effectively. Although blockchain-based applications are considered less vulnerable, they did not become the silver bullet for securing against different security threats (e.g., Sybil attack, Double-spending). Similarly, to perform the SRM of blockchain-based applications, unified and formal knowledge models are required. Overall, this thesis addresses the problem of traditional applications' SRM using blockchain as a countermeasure solution and the SRM of blockchain-based applications.

We start by surveying how blockchain mitigates the security threats of traditional applications, where the security threats of blockchain are looked at from two distinct perspectives. For example, the security threats of traditional applications that can be mitigated by using blockchain, and the security threats of blockchain-based applications. We derive the components of traditional applications and consolidate the identified security threats, including blockchain components. The outcome is a blockchain-based reference model (BbRM) that adheres to the SRM domain model. However, the BbRM depicts the static knowledge and inefficient instantiation of the SRM. To overcome these gaps, we present an upper-level reference ontology (ULRO) as a foundation ontology by using the SRM domain model. The ULRO provides semantic interoperability, general concepts of the SRM common to all domains, and enables a common foundation for ontologies in the information security domain. The ULRO presents a structural representation that can support the dynamic knowledge encoding and instantiation with information security knowledge for the SRM of domain-specific applications.

We provide two instantiations of the ULRO. The first instantiation of the ULRO

includes permissioned blockchain components using the Corda as a permissioned blockchain and the financial case related to the capital market's post-trade matching and confirmation. The second instantiation of the ULRO includes the permissionless blockchains components and the healthcare case. Both ontology representations help in the SRM of traditional applications by combining the blockchain as a countermeasure solution and the SRM of blockchain-based applications. Furthermore, we built a web-based ontology parsing tool, OwlParser. OwlParser leverages the developed ontology representations and provides a web-based explorer feature that allows one to dive into the SRM concepts and assist the users in finding and navigating through the concepts that fit to their needs.

The contributions of this thesis adhere to fundamental concepts of the SRM domain model. We use the Protégé to build ontology representations and then publish them to the MMISM and the GitHub public repository. OwlParser's source code is also accessible on GitHub. We evaluated the contributions of this thesis by employing various methods. Contributions resulted in an ontology-based security reference framework for managing the security risks using blockchain. The framework establishes common ground and systematic understanding that can assist researchers, developers, practitioners, and other stakeholders in explaining the SRM of traditional and blockchain-based applications. The framework is dynamic, supports the iterative process of SRM, and can be updated continuously when new security threats, vulnerabilities, or countermeasures emerge. Moreover, it can communicate the security needs and assist in the SRM of blockchain-based applications. Ultimately, the reference framework can potentially lessen the security threats of traditional and blockchain-based applications.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SRM	Security risk management
ISSRM	Information system security risk management
SLR	Systematic literature review
IoT	Internet of things
P2P	Peer-to-peer network
ULRO	Upper-level reference ontology
BbRM	Blockchain-based reference model
GDPR	General data protection regulation
STRIDE	Spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege
OWASP	Open web application security project
DAO	Decentralized autonomous organization
HLF	Hyperledger fabric
ETH	Ether
BTC	Bitcoin
PoW	Proof of work
DPoW	Deplayed proof of work
PoS	Proof of stake
DPoS	Delegated proof of stake
PBFT	Practical byzantine fault tolerance
PoR	Proof of reputation
PoA	Proof of authority
EOA	Externally owned account
CA	Contract account
JVM	Java virtual machine
RPC	Remote procedure call
CorDapp	Corda-based application
HTTP	Hypertext transfer protocol
TLS	Transport layer security
TCP	Transmission control protocol
MACs	Message authentication codes
PKI	Public key infrastructure
SPARQL	SPARQL Protocol and RDF Query Language
RDF	Resource description framework
OWL	Web ontology language
OwlParser	Web ontology language parser
CPNs	Colored petri nets

THAs	Traditional healthcare applications
BbHAs	Blockchain-based healthcare applications
EHR	Electronic healthcare records
PHR	Patient healthcare records
CordaSecOnt	Corda-based security ontology
HealthOnt	Healthcare security ontology
XML	Extensible markup language
XSD	XML Schema Definition
CNAM	Caisse Nationale d'Assurance Maladie
BPPHA	Back-pain patients' healthcare application
DC	Dublin core
DL	Description logic
TBox	Terminological component
ABox	Assertion component
MMISW	Marine metadata interoperability semantic framework
DoS	Denial of service
MitM	Man in the middle
BGP	Border gateway protocol
NIST	National institute of standards and technology
HSM	Hardware security module
QCT	Quantum computing threat
IRI	Internationalized resource identifier
TPS	Transactions per second
WSNs	Wireless sensor networks
DHT	Distributed hash table
TEE	Trusted execution environment
ZKP	Zero-knowledge proof
UFO	Unified foundational ontology

1. INTRODUCTION

Centralized technology infrastructure is vulnerable to a variety of security threats (e.g., data tampering, data theft, single-point failure) that could negate the confidentiality, integrity, and availability of the system [1, 2, 3]. The financial industry, for example, ranks second in terms of data breaches since its infrastructure is constantly challenged by various security threats [4]. The security threats harm the business processes and valuable assets, e.g., hackers exploited SWIFT credentials to steal $81 million from the Bangladesh bank [5], and the data breaches that each cost an average $4.24 million [6, 7]. Similarly, the studies [8, 9] investigated the security threats of healthcare applications and identified that the centralized infrastructure is incompatible and does not provide security measures by design. Digitization, the internet of things, and smart devices in healthcare generate massive electronic health records, empowering patients and the healthcare sector [10, 11]. The medical data is confidential and plays an essential role in a patient's health diagnosis and treatments to reduce medical mistakes [12]. Therefore, such data heightens the concerns about securing it from security threats [13].

Blockchain-based applications are appearing to address such security challenges, improve data integrity, and restructure the transaction process to be distributed, decentralized, and irreversible. Blockchain is a distributed, decentralized, and immutable ledger technology that operates over a peer-to-peer (P2P) network. Blockchain is making inroads in various domains (e.g., finance, healthcare, supply chain) [10] with the promise to overcome the security challenges of traditional infrastructure. For example, the study [14] discusses a blockchain-based healthcare application to protect medical data from tampering, theft, and unauthorized access. Wang et al. [15] focus on the tamper-proof property of blockchain to implement a blockchain-based authentication mechanism for securing single-factor authentication in the industrial internet of things (IoT). The success of blockchain is contingent on distributed, decentralized, verifiable, and tamper-proof characteristics. Blockchain-based applications are considered to be less vulnerable because of a distributed network, decentralized consensus, immutable ledger, and cryptography. However, blockchain-based applications have not been proven to be unreservedly secure because various security threats are appearing within them [16]. For example, the Monero blockchain experienced the Sybil attack where the attacker[1] creates numerous fake distributed nodes trying to get undue influence on the blockchain network [17], and BTC gold and Ethereum classic blockchains suffered from the Double-spending [18, 19].

We will use the term traditional applications as a more specific one for applications relying on centralized infrastructure. This thesis addresses the problem of traditional applications' security risk management (SRM) using blockchain as a countermeasure solution and the SRM of blockchain-based applications. Contri-

[1] In this thesis, a general term "attacker" is used instead of the more specific "threat agent".

butions to resolving these problems resulted in an ontology-based security reference framework for managing the security risks of traditional applications using blockchain. The framework also supports the SRM of blockchain-based applications. The framework establishes common ground and systematic interpretation that can assist developers, researchers, practitioners, and other associated stakeholders in explaining how blockchain can mitigate the security threats, what security threats may appear within blockchain-based applications, and what security countermeasures should be implemented.

1.1. Problem Statement

Various programs (e.g., OWASP [20]), threat models (e.g., STRIDE [21]), security risk management models (e.g., ISSRM [22, 23]), and regulations (e.g., GDPR [24]) exist to communicate and reduce the security threats in order to develop secure software. However, the security threats continuously evolve because the traditional technology infrastructure does not implement security measures by design. For example, the financial and healthcare sectors are the leading targets for the attackers [8, 4]. Furthermore, despite the increasing number of security incidents, many organizations do not prioritize software security. Blockchain infrastructure is a befitting solution to mitigate traditional applications' security threats. For instance, data tampering is one of the main security concerns of data-centric applications, which is attempted to be mitigated using blockchain-based applications [16]. However, there exist no unified and formal knowledge models that can support the SRM and a clear explanation of how blockchain-based applications mitigate traditional applications' security threats. Moreover, the blockchain domain is constantly evolving, providing new techniques and design concepts that are often interchangeable, resulting in conceptual ambiguity and confusion in treating the security threats effectively [14, 25].

Like traditional applications, security plays an important role in guaranteeing the acceptability of blockchain-based applications. Although blockchain-based applications are less vulnerable, they did not become the silver bullet for securing against different security threats. Also, the involvement of the monetary assets raises security concerns because the attackers can steal the monetary assets and damage the system. For example, such a situation occurred in the reentrancy attack on the Ethereum-based decentralized autonomous organization (DAO) smart contracts, where the attacker gained control on $60 million Ethers [26, 27]. To realize the true potential of blockchain-based applications, the first step is to understand the associated security threats and vulnerabilities. The attacker can exploit these threats and vulnerabilities and affect valuable assets. The security issues arise from the wrong security decisions, incomplete knowledge, or misunderstanding of the security needs of the software. For example, the security threats that appear (e.g., Sybil and Double-spending, etc.) [16] within the blockchain-based applications are debatable. However, the same problem exists in managing

the security risks of blockchain-based applications because there is no unified and comprehensive blockchain-based security reference framework to evaluate the security of blockchain-based applications systematically. The blockchain-based security reference framework can establish a common ground and systematic interpretation for professionals and researchers regarding the security of blockchain-based applications. In addition, there exist a few studies reporting on the security challenges of blockchain platforms [28, 29], but they do not focus on the security and the SRM of the blockchain-based applications. We formulate the following research objectives to assist in overcoming the aforementioned problems.

- ***Research objective 1:*** To perform the security risk analysis of traditional applications to identify the security threats that can be mitigated by using blockchain-based applications. This research objective aims to show blockchain as a countermeasure solution.
- ***Research objective 2:*** To perform the security risk analysis of blockchain-based applications to identify the security threats that may arise in them, including the countermeasures to mitigate them. This research objective states that blockchain is prone to security threats; consequently, security aspects in blockchain-based applications should be properly assessed.
- ***Research objective 3:*** To provide a blockchain-based security reference framework that encapsulates the security threats mitigated in traditional applications using blockchain, as well as the security threats that occur in the blockchain-based applications. The reference framework shall provide a unified and comprehensive information security knowledge for managing the security risks of traditional and blockchain-based applications.

The blockchain-based security reference framework can enable a systematic evaluation, and establish a common ground and systematic interpretation for the developers, practitioners, and researchers regarding the security of traditional and blockchain-based applications. Moreover, it can communicate security requirements to technical experts effectively and efficiently. The reference framework is required for the SRM of blockchain-based applications to identify the security threats and their impacts timely. Ultimately, the reference framework can potentially lessen the security threats of traditional and blockchain-based applications.

1.2. Research Questions

To address the research problem identified in the previous section and to achieve the research objectives, we establish the main research question: How can a reference framework be developed for managing the security risks of traditional and blockchain-based applications? To produce the expected outcomes, the main research question is further divided into four following research questions (RQ):

RQ1: What is the current state of the art for performing the SRM of traditional applications utilizing blockchain and the SRM of blockchain-based applications?

Answering RQ1 involves:

- Context and assets identification of traditional and blockchain-based applications using the results of a systematic literature review (SLR);
- Blockchain-based reference model (BbRM) that adheres to the concepts of the SRM domain model;
- Evaluation of BbRM using the data tampering threat of traditional applications;
- Evaluation of BbRM by exploring the Sybil and Double-spending threats of blockchain-based applications;

RQ2: How can SRM domain model abstraction be used to build an ontology-based reference framework?

Answering RQ2 involves:

- The upper-level reference ontology (ULRO) as a foundation ontology using the SRM domain model;
- Smart healthcare application use case for instantiating the ULRO using the data tampering and Sybil attack;
- Expert evaluation to assess the suitability and correctness of the ULRO;
- Colored Petri nets-based evaluation of the ULRO considering the blockchain as a countermeasure solution;

RQ3: How might permissioned blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissioned blockchain-based applications?

Answering RQ3 involves:

- Security risk analysis of traditional capital market's post-trade matching and confirmation process;
- Corda-based permissioned blockchain to mitigate the security threats of traditional capital market's post-trade matching and confirmation process;
- Security risk analysis of Corda-based permissioned blockchain;
- Corda security ontology, an instantiation of the ULRO using the components of Corda-based permissioned blockchain;
- Corda security ontology evaluation using the automated tools, qualitative assessment criteria, and tasks-based evaluation;

RQ4: How might permissionless blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissionless blockchain-based applications?

Answering RQ4 involves:

- Context and assets identification of traditional healthcare applications;
- Security risk analysis of traditional healthcare applications using permissionless blockchains;
- Security risk analysis of permissionless blockchain-based healthcare applications;
- Health security ontology, an instantiation of the ULRO using the components of permissionless blockchains;
- Back-pain patients' healthcare application case to evaluate the healthcare security ontology;

1.3. Research Approach

To answer **RQ1**, we explore blockchain from two different perspectives to comprehend how blockchain-based applications can mitigate the security threats of traditional applications? First, we identify the security threats of traditional applications mitigated by introducing blockchain-based applications. Second, we report the security threats of blockchain-based applications. Following the analytic method, we analyze the results and construct the BbRM using the concepts of the SRM domain model. We present the architecture highlighting the main components of traditional applications that is extended with the components of blockchain to construct the BbRM. Further, we have extended the BbRM to incorporate the security threats of blockchain-based applications and their countermeasures. To evaluate the BbRM, we investigate the data tampering of traditional applications and provide the traditional, Ethereum, and Hyperledger Fabric-based countermeasures. In addition, we evaluate the BbRM utilizing the Sybil and Double-spending threats associated with blockchain-based applications.

To answer **RQ2**, we use the results of the first contribution and the SRM domain model to build the ULRO as a foundation ontology, including the fundamental constructs of the SRM. We use the smart healthcare application case for instantiating the ULRO using the data tampering and Sybil attack. To assess the suitability and correctness of the ULRO, first, we conduct an expert evaluation. Second, we perform the colored Petri nets-based evaluation, considering the blockchain as a countermeasure solution and to know the feasibility while incorporating the ontology during the system design.

We provide the two instantiations of the ULRO as a proof of concept. First, we instantiate the ULRO using the permissioned blockchain components, and it answers **RQ3**. We perform the security risk analysis of the traditional case of the financial industry, the capital market's post-trade matching and confirmation. We use Corda-based permissioned blockchain to mitigate security threats of the centralized infrastructure-based capital market's post-trade matching and confirmation. Further, we perform the security risk analysis of the Corda-based capi-

tal market's post-trade matching and confirmation. We encoded the components of the Corda-based permissioned blockchain and present the Corda security ontology. Corda security ontology is evaluated using automated tools, qualitative assessment criteria, and tasks-based evaluation.

Second, we instantiate the ULRO to permissionless blockchain components, and it answers **RQ4**. We perform the literature review for defining the traditional healthcare applications' context and assets. Later, we perform a security risk analysis of traditional healthcare applications using permissionless blockchains and the security risk analysis of permissionless blockchain-based healthcare applications. We encode the components of permissionless blockchains and present the Healthcare security ontology. Employing the automated tools, qualitative assessment criteria, and tasks-based evaluation, we also take a back-pain patients' healthcare application to evaluate the healthcare security ontology.

We build a web-based ontology parsing tool, OwlParser, to leverage the developed ontologies without acquiring knowledge of ontologies, an editing tool, or querying the concepts through SPARQL queries. To assess the usability of the OwlParser, we conduct a survey with software developers, information security specialists, blockchain, and ontology experts.

1.4. Contributions

This thesis provides incremental contributions to the main research question by answering the above-mentioned four research questions.

Contribution 1: *Blockchain-based reference model for the SRM of traditional and blockchain-based applications.* This contribution addresses the **RQ1**. In the SLR, we investigate the security threats of traditional and blockchain-based applications. The findings contain a list of security threats of traditional applications that are addressed by blockchain-based applications, and a list of security threats that may appear in blockchain-based applications. Using these results, we build the BbRM that adheres to the concepts of the SRM domain model. We present the evaluation of BbRM utilizing its components to investigate data tampering, Sybil attack, and double-spending in detail. The evaluation shows that the BbRM can support the SRM of traditional and blockchain-based applications.

Contribution 2: *Ontology-based reference framework for managing security risks*. This contribution responds to the **RQ2**. We build an upper-level reference ontology as a foundation ontology using the SRM domain model to encode information security concepts. The ULRO provides semantic interoperability, general concepts common to all domains, and enables a common foundation for ontologies in the information security domain. The ULRO is an ontology-based structural representation that can support the dynamic knowledge encoding and instantiation with information security knowledge for the SRM of domain-specific appli-

cations. Experts' evaluation assesses the correctness of the ULRO, and colored Petri nets-based evaluation shows the applicability.

Contribution 3: *Permissioned blockchain-based security ontology*. Blockchain is ready to revolutionize the financial industry, and the blockchain-based Corda platform provides suitable technological infrastructure. This contribution addresses the research question **RQ3** by presenting a permissioned blockchain-based security ontology (CordaSecOnt). We perform the security risk analysis of the capital market's post-trade matching and confirmation where the Corda platform is used as a countermeasure solution. We also look into what security threats might arise in Corda-based post-trade matching and confirmation and what countermeasures are available. The CordaSecOnt provides classifications of assets, security criteria, threats, vulnerabilities, and countermeasures. The evaluation of CordaSecOnt determines the correctness of encoded concepts and organizational fitness. Overall, the CordaSecOnt may improve the SRM of the financial industry.

Contribution 4: *Permissionless blockchain-based security ontology*. The healthcare industry is digitizing its operations and generating huge amounts of medical data to make prompt and informed decisions in patients' health diagnosis and care. This contribution addresses the research question **RQ4** where we present permissionless blockchain-based healthcare security ontology (HealthOnt). HealthOnt is an ontological representation of healthcare applications' security combined with blockchain. HealthOnt provides the classifications of assets, security criteria, threats, vulnerabilities, and countermeasures. Moreover, the HealthOnt encodes the security threats that may appear in blockchain-based healthcare applications and their countermeasures. We evaluate the ontology by performing the SRM of a back-pain patient's healthcare application case. The results show that HealthOnt offers coherent and formal information models that delineate blockchain as a countermeasure solution for the SRM of traditional healthcare applications.

Contribution 5: *Web-based ontology parsing tool, OwlParser*. In previous contributions, we outline blockchain-based security ontology representations in finance (e.g., CordaSecOnt) and healthcare (e.g., HealthOnt). OwlParser leverages the developed ontologies without acquiring knowledge of ontologies, an editing tool, or querying the concepts through SPARQL queries. OwlParser provides a web-based explorer feature that allows one to dive into the SRM concepts and assist the practitioners in finding and navigating through the concepts that fit their needs. The evaluation of OwlParser yielded positive findings since the tool allows easy examination and navigation through the encoded concepts.

1.5. Thesis Roadmap

We organize this manuscript into eight chapters. The current chapter (i.e., chapter 1) outlines the introduction, problem statement and objectives, research approach, and contributions. The rest of the thesis is structured as follows.

Chapter 2 overviews the blockchain and blockchain platforms, outlines the security risk management domain model, the ontology development method, and presents a systematic literature review. Chapter 3 discusses the thesis's first contribution, and this chapter's outcome is a blockchain-based reference model. Chapter 4 focuses on the second contribution, where we build the upper-level reference ontology as an ontology-based reference framework for managing security threats. Chapter 5 presents the third contribution, and it comprises the security risk analysis of the capital market's post-trade matching and confirmation, including the development of CordaSecOnt, and its evaluation. Chapter 6 is a fourth contribution of the thesis that describes the security risk analysis of traditional and blockchain-based healthcare applications, the development of HealthOnt, and its evaluation. Chapter 7 confers the web-based ontology parsing tool as a fifth contribution to the thesis. Chapter 8 concludes the thesis and highlights the future research directions that have evolved as a result of this thesis's contributions.

2. BACKGROUND

In this chapter, we discuss the key concepts that are used throughout the thesis. For example, Section 2.1 provides an overview of blockchain, blockchain types, and blockchain platforms. Section 2.2 describes the security risk management domain model and the fundamental concepts for building the ontology representations. Section 2.3 discusses the ontology development approach and tools. In Section 2.4, we undertake a literature study to identify and synthesize relevant blockchain research to comprehend the present state-of-the-art security risk management combining blockchain.

2.1. Blockchain

Blockchain is an append-only decentralized distributed ledger technology that operates over a P2P network. Blockchain eliminates trusted intermediaries in a transactional process and records transactions in a distributed ledger. The ledger is immutable and updates every time over a P2P network when a new block populates. The network nodes establish a chain of blocks (Fig. 1) where each block connects to a previous block by a unique cryptographic hash. The first block in a blockchain is a genesis block, and every block has a header and body (Table 1). Block header includes a unique block hash, a previous block hash, Merkle root, block version, nonce, timestamp, and difficulty target. Block body contains a valid list of hashed and ordered transactions as a Merkle tree.

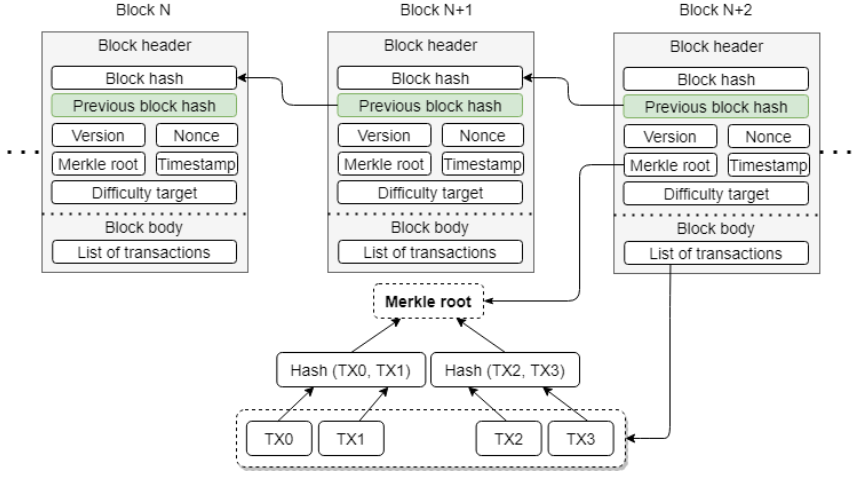

Figure 1. Blockchain and block structure

Table 2 provides a comparison of blockchain platforms. Blockchain platforms can be classified as permissionless (e.g., Bitcoin, Ethereum) or permissioned (e.g., Hyperledger Fabric (HLF), Corda) [30]. In permissionless blockchains, anybody around the world can join the network, and no central or designated authority

Table 1. Blockchain block format

Block element	Detail
Block hash	A unique cryptographic hash of a block.
Previous block hash	A unique cryptographic hash of a previous block.
Version	Indicates the version of the block.
Nonce	A unique random counter.
Merkle root	The hash of Merkle tree.
Timestamp	The creation time of the block.
Difficulty target	Indicates a mining difficulty in PoW-based consensus.
Transactions	A list of transactions stored in Merkle tree format.

manages their participation. The network participants do not require permission to participate in the consensus or execute a transaction. Also, the transactions are publicly visible to everyone. In contrast, in permissioned blockchains, only pre-verified nodes can join the network, the access control layer controls the network participants' operations, and transaction visibility is restricted [31]. Permissioned blockchains are arising in businesses where privacy is of utmost importance.

Table 2. Comparison of blockchain platforms

	Bitcoin	**Ethereum**	**HLF**	**Corda**	**EoS**	**Ripple**	**Stellar**
Type	Permissionless	Permissionless	Permissioned	Permissioned	Permissioned	Permissioned	Both
Consensus	PoW	PoW, *PoS*	PBFT, CFT	Validity, Uniqueness	DPoS	Probabilistic voting	Stellar protocol
Participation	Open	Open	Member-only	Member-only	Member-only	Member-only	Open, Member-only
Read/Write operation	Any participant	Any participant	Verified member	Verified member	Verified member	Verified member	Any participant, Verified member
Smart contract	Yes	Yes	Yes	Yes	Yes	No	Yes
SC Language	Script	Solidity	GO, Java	Kotlin	C++	- - -	Node JS
Cryptocurrency	BTC	ETH	- - -	- - -	EOS	XRP	XLM
Throughput	7 TPS	15 TPS	3500+ TPS	1678 TPS	4000+ TPS	1500+ TPS	1000+ TPS
Scalable	Difficult	Difficult	Moderate	Moderate	Moderate	Moderate	Moderate
Nodes capacity	Unlimited	Unlimited	Moderate	Limited	Unlimited	Limited	Moderate
Confidentiality	No	No	Yes	Yes	Yes	Yes	Yes
Transparency	High	High	Moderate	Low	Moderate	Low	Moderate
Open source	Yes	Yes	Yes	Yes	Yes	No	Yes
Governance	- - -	Ethereum developers	Linux foundation	R3 consortium	EOSIO core arbitration	Ripple labs	Stellar foundation
Industry focus	Cryptocurrency	Enterprise	Enterprise	Financial	Enterprise	Financial	Financial
Adoption	High	High	High	High	Moderate	Moderate	Low

Blockchain uses a decentralized consensus mechanism to maintain the state and immutability of the ledger. Permissionless blockchains (e.g., Bitcoin and Ethereum) use Proof of Work (PoW) consensus. PoW is a computational rich energy-waste consensus strategy where nodes, called miners, define the ledger's state by solving the complex cryptographic puzzle. In contrast, Proof of Stake (PoS) is an energy-efficient consensus [32] where miners become validators [33] and lock a certain amount of cryptocurrency to participate in the consensus. The blockchain platforms NXT and PeerCoin use PoS consensus, and Ethereum is moving to PoS consensus. HLF uses practical byzantine fault tolerance (PBFT) consensus that tolerates Byzantine faults (e.g., malicious nodes) [34, 35, 36]. PBFT is an energy-efficient and effective consensus mechanism for high-throughput transactions. There exist other consensus mechanisms, for example, Delegated Proof of Stake (DPoS), Proof of Authority (PoA), Proof of Reputation (PoR), and new ones are appearing. These consensus mechanisms are fault-tolerant and achieve mutual agreement on a single state of the ledger [18].

The advent of smart contracts introduces value exchange in P2P and decentralized manners. A smart contract is an autonomous computer program [33] written in a Turing-complete programming language [37] (e.g., Solidity, Go, Java, NodeJs, C++). The smart contract constitutes a digital contract in blockchain to store data and execute automatically [38] when certain conditions meet. For example, the Ethereum platform provides Solidity programming language to write smart contracts and to build decentralized applications (dApps) [33]. In HLF, the smart contract is known as the chaincode and performs actions according to the coded terms in a contract. Similarly, other blockchain platforms (e.g., Corda, EoS, Stellar) have smart contracts to reach contractual agreements in a digital realm [26]. In addition, permissionless blockchains have limited transaction throughput, are difficult to scale, and have low transaction confidentiality. Permissioned blockchains, on the other hand, offer fast throughput and improved transaction confidentiality but have poor to moderate transparency.

2.1.1. Ethereum Blockchain

Ethereum is an example of a permissionless blockchain, and it is open-source. Unlike Bitcoin, Ethereum is not only limited to cryptocurrency. Ethereum introduces smart contracts, one of the essential concepts in blockchain, to enable the development of decentralized applications across a wide variety of domains. Ethereum offers a Solidity programming language for creating smart contracts that run on the Ethereum virtual machine, a stack-based processing engine. Ethereum has its native cryptocurrency, Ether (ETH). Every transaction on the Ethereum blockchain requires a gas fee, which may be paid in either ETH or Gwei (Gwei is the smallest denominator of ETH). Ethereum blockchain commonly recognizes four types of nodes [39]. For example, the archive nodes have the data since the genesis block. The full nodes receive copies of transactions and keep the current state of the blockchain. Light nodes, on the other hand, do not retain the whole state of the blockchain, which is advantageous for limited memory and computing devices; nonetheless, light nodes rely on full nodes. The miner nodes run a PoW consensus to verify the transactions and add them to the blocks. Ethereum blockchain manages externally owned accounts (EOA) and contract accounts (CA). EOA accounts are owned by people holding ETH balances and controlled by a private key. When a new account is created, for example, a pair of public and private keys are produced where the private key is kept safe by the owner while the public key is used to complete transactions (e.g., receive funds). CA accounts hold a smart contract code that executes when it receives a transaction from an EOA because CA cannot initiate transactions independently.

Figure 2 presents the workflow of Ethereum to process a transaction from one user to another. EOA initiates the transaction containing the gas price, gas limit, receiver account address, the ETH amount, and the nonce (it ensures that transaction data output is unique). The transaction is signed by the private key of the

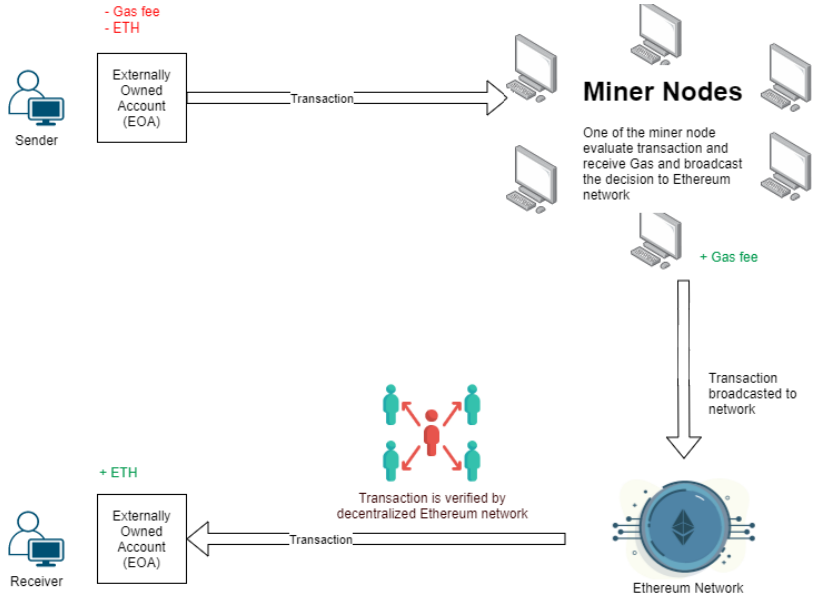

Figure 2. Workflow of Ethereum transaction from one user to another, adapted from [40]

initiator and submitted to the validator nodes to validate that the initiator account signs this transaction. Miner nodes receive a transaction and start performing the PoW consensus. The miner node that finds the block first receives the gas fee as an incentive, and the transaction is then added to the block. Then, the decision broadcasts to the whole Ethereum network to prevent the evaluation of the transaction by other miner nodes. Finally, the receiver gets the sent ETH, and the initiator node syncs the transaction with the local copy of the blockchain.

2.1.2. Hyperledger Fabric Blockchain

Hyperledger Fabric is an example of a permissioned blockchain introduced by IBM and the Linux Foundation. HLF introduces the permissioned settings and built-in access control mechanism to prevent unwanted users from accessing and joining the network. HLF has three types of peer nodes. For example, the endorsing peer nodes are responsible for creating the transaction proposal in a single chaincode container based on the chaincode results. Chaincode installation is compulsory on these nodes. Committing peer nodes hold full ledger and do not invoke any chaincode function. Ordering peer nodes contain the entire ledger transaction history, including valid and invalid transactions. Other types of nodes only hold a valid transaction history. Moreover, these nodes are responsible for receiving the endorsed transaction proposal and sending it to all other nodes to validate that transaction and update their Ledgers.

HLF follows an execute-order-validate algorithm that combines the concepts of channels, organizations, membership service providers, and ordering services. All these components allow multiple organizations to have their smart contracts

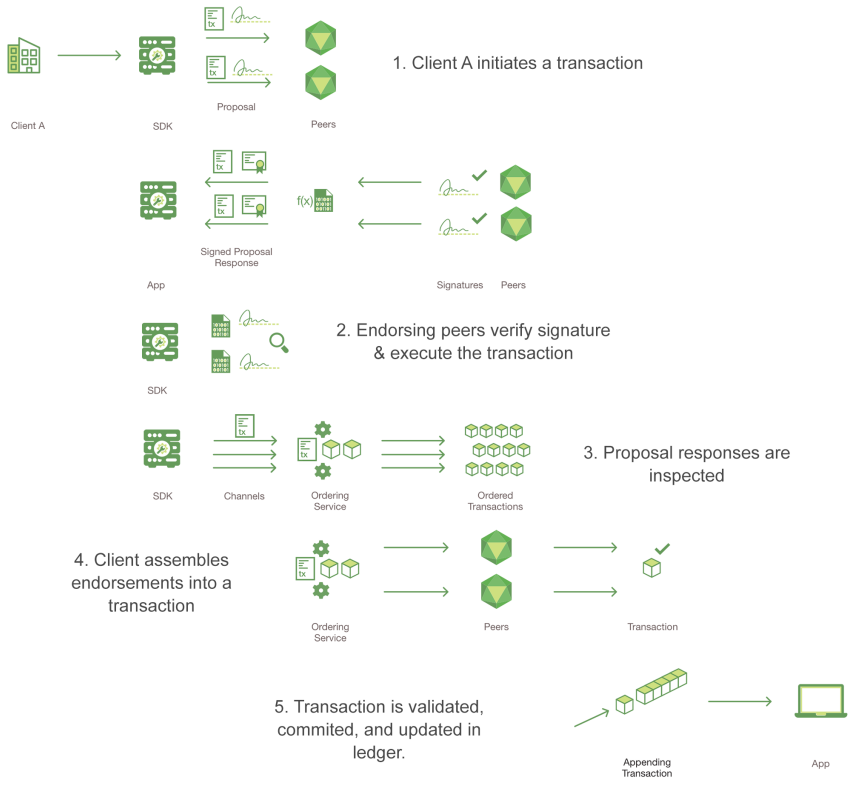

Figure 3. Workflow of HLF transaction from one user to another, adapted from [40]

and make their transactions without interrupting the other organizations' operations. Channel is a logical unit to form a cluster of peer nodes to perform transactions regardless of letting know other peers who belong to different channels. Organizations are the members of the network who have one or more peer nodes within the network where multiple organizations interact to fulfill their common business needs. A membership service provider is a certificate manager implemented as a certificate authority for the enrollment of users to the network, invokes the transactions, and ensures the secure connection between the users and the components of the blockchain network. The ordering service creates a package of transactions and transfers between different peer nodes and ensures the transaction is delivered over the network.

2.1.3. Corda Blockchain

Corda is an open-source enterprise permissioned blockchain-based platform, and CorDapp is a Corda-based decentralized application. Corda focuses on bringing privacy, transparency, and security to financial operations between different cross-organizational parties. The Corda platform uses *(i)* validity and *(ii)* unique-

ness consensus [41, 42]. The validity consensus ensures the correctness of input states, output states, and required signatures in a transaction. The uniqueness consensus checks if the transaction is not already consumed (e.g., protection against Double-spending) [41, 42]). In Corda, peers communicate point-to-point (Fig. 4). For example, the transaction specifies a list of message recipients that sign and verify the transaction. The flow mechanism supports multi-step coordination and verification of transaction messages. Corda network peers communicate back and forth before a transaction commit. Transaction flows validate the sequence of steps that make a transaction legitimate and commit on the ledger replicating over different peers on the P2P network. In Fig. 4, Alice initiates the transaction flow and creates a transaction. Alice signs the newly created transaction and sends it to Bob. Alice's flow is suspended and check-pointed in this stage, and now Bob's transaction flow begins. Bob inspects and verifies the transaction upon receiving it from Alice. Then, Bob signs a transaction and commits. The transaction is now signed by both parties (Alice and Bob), and the system sends it back to Alice to verify the signature of Bob. After verifying Bob's signature, Alice commits the transaction that fulfills the criteria of a successful transaction process and completes the flow. It is an example of a two parties transaction process, but Corda flows can involve multiple parties, and multiple signatures [43].

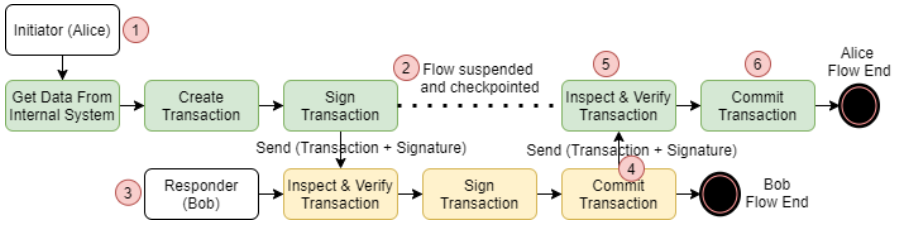

Figure 4. Workflow of Corda transaction from one user to another

A Corda node is a Java virtual machine (JVM) run-time environment. It has a unique identity on the network and hosts Corda services and CorDapps. CorDapp is installed at the level of the individual node, rather than on the network itself [44]. Corda uses an asynchronous protocol (e.g., AMQP/TLS) [44] when nodes communicate with each other and HTTP communication for registering a Corda node. Corda platform combines various components to provide infrastructure for CorDapp [45, 44]. Such as a persistence layer for storing data, for example, in an SQL-based database and a network interface for interacting with other nodes. In Corda, client applications use RPC calls via the RPC interface to communicate with Corda nodes. A service hub allows the node's flows to call upon the node's other services, which are node utilities. CorDapp interface and provider for extending the node by installing CorDapp. Corda vault is a database that relies on java database connectivity from the Corda node. Transaction storage is based on a key-value store for attachments, transactions, and serialized state machines. Flow state machine manager that manages the operation of flow state machines. Flows are routines to run and update the ledger for the node. States are the facts to reach

an agreement. Identity and key management manages various supported identities and generates keys to sign transactions. The scheduler schedules the operations for future points in time. A network map is a searchable phone book of nodes on the network. Notary obtains authorized signatures and messaging components for providing the interface with other nodes.

2.2. Security Risk Management

The primary goal of this research is to provide a comprehensive reference framework for pursuing the SRM of information systems combining blockchain. Security is described as the prevention, detection, and response against malicious harm to a valuable asset [46]. In accordance, the SRM is concerned with lowering the risk of intentional unauthorized harm to valuable assets to that level acceptable to the system's stakeholders by preventing and reacting to malicious harm, misuse, threats, and security risks [47]. In this work, we follow the SRM domain model (Fig. 5) [22, 23] that enables a systematic approach for analyzing the security of information systems and provides the key concepts and relationships supporting the definition of security and SRM. The SRM domain model addresses the SRM process at three different conceptual levels, for example, in the perspective of assets-, risk-, and risk treatment-related concepts.

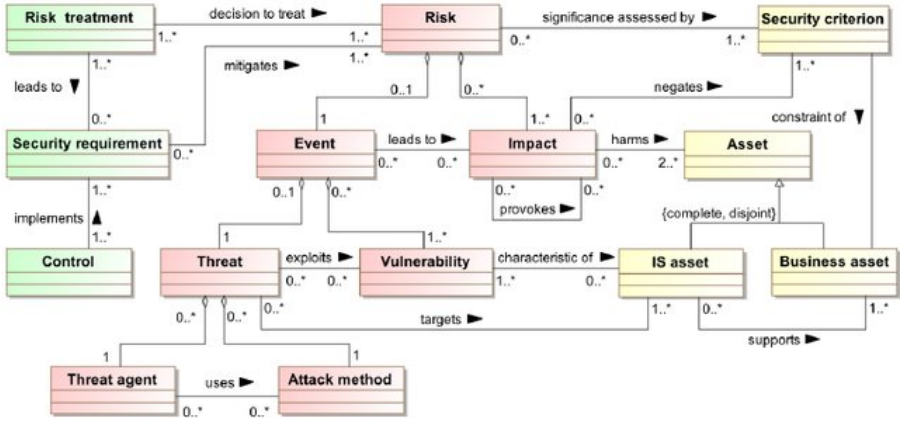

Figure 5. Security risk management domain model, adapted from [22, 23]

The assets can be classified as system or business assets and play a vital role in accomplishing an organization's objectives. The business assets are items of value and describe the information, processes, capabilities, skills essential to the business, and the system assets support the business assets [23]. Security criteria (C - Confidentiality, I - Integrity, and A - Availability) are business assets' constraints or properties because security criteria distinguish the security needs of business assets. In risk-related concepts, the risk is a combination of risk event and impact. The risk event constitutes the threat and one or more vulnerabilities that negatively impact one or more assets by harming them. Vulnerability is a characteristic of a

system that depicts its weakness or flaw. The threat agent uses an attack method and triggers the threat that targets the one or more system assets by exploiting their vulnerabilities. The risk treatment-related concepts present decisions to treat the security threats by defining the security requirements. Security requirements are implemented as the security controls, e.g., countermeasures implement the requirements that mitigate the security risk.

Table 3. Fundamental concepts of the SRM domain model

Level	Concept	Relation	Detail
Asset	System asset	supports	System asset supports business asset.
	Business asset	hasConstraint	Business asset has security criteria constraint.
	Security criteria	constraintOf	Security criteria is a constraint of business assets.
Risk	Threat	targets	Threat targets one or more system assets
		exploits	Threat exploits zero to several vulnerabilities.
	Vulnerability	characteristicOf	Vulnerability is a characteristic of one or more system assets that exposes their weakness.
		harms	Vulnerability harms one or more assets.
		negates	Vulnerability negates the defined security criteria of business assets.
Treatment	Countermeasure	mitigates	Countermeasure mitigates one or more vulnerabilities.

The SRM domain model explains the relationships among assets, risks, and risk treatment-related concepts that establish a process (Fig. 6) combining different activities to perform the SRM. For example, to build an ontology-based reference framework for the SRM, we explore what assets to secure and determine the security objective in the context of confidentiality, integrity, and availability. We identify the threats that exploit various vulnerabilities to harm the assets and negate the security criteria of the business assets. We define the risk treatment decisions to treat the threats by incorporating the countermeasures. The process of SRM is not static; therefore, the process model of the SRM enables it to perform several iterations to reach an acceptable level of each risk [23].

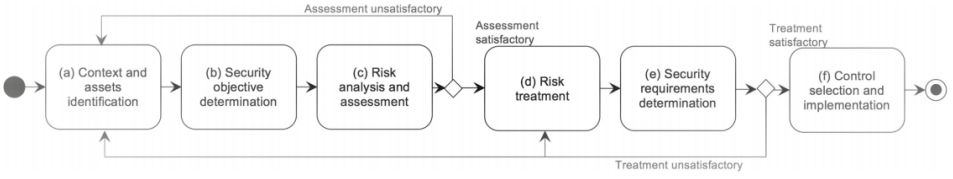

Figure 6. Process model of SRM domain model, adapted from [22, 23]

The SRM domain model is developed through a survey of security-related standards, and SRM standards and methods. We use the SRM domain model because among other SRM approaches [48], the SRM domain model fulfills the criteria of ISO/IEC 27001 information security standard. In addition, the SRM domain model supports the fundamental concepts of SRM and explores three aspects (e.g., assets-, risk-, and risk treatment-related) during the early phases of information system development. Thus, the SRM domain model provides a framework, conceptual layered structure, and fundamental concepts of the SRM intended to serve as a foundation for building our ontology-based reference model for performing the SRM of traditional and blockchain-based applications.

2.3. Ontology

Ontology is a formal specification of a shared conceptualization that deals with the nature of existence [49, 50]. According to the Oxford dictionary, ontology combines a list of concepts and categories in a subject area that shows the relationships between them [51]. An ontology brings generalization to a specific domain and establishes an exchange of information [52]. Moreover, the ontology structure area of interest [53] and elaborates the meaning of concepts along with their relationships to overcome the consequences of a misunderstanding that can be time-consuming and costly. For example, these studies [14, 54, 25] mentioned the conceptual ambiguities and semantic gaps about the blockchain as a countermeasure solution for treating the security threats of traditional healthcare applications. According to Fenz et al. [55], the ontology-based information security models can provide a unified and formal knowledge base that is a fundamental requirement for supporting the SRM. In this scenario, the ontology-based representation combining blockchain as a countermeasure solution can solve these issues. In addition, the same ontology can provide the unified and formal knowledge base for the SRM of blockchain-based applications. Noy and McGuinness [52] provide reasons for why and when to construct an ontology. The reasons focus on the practical significance to clarify if the ontology is practicable in the real world. We use the same reasons (Table 4) to explain why we should build an ontology-based reference framework to perform the SRM of traditional and blockchain-based applications.

Table 4. Reasons to build an ontology

General reasons to build ontology	**Our perspective**
To share a common understanding of the structure of information among people or software agents.	To share a common understanding related to the SRM of traditional applications using blockchain as a countermeasure solution, and the knowledge belonging to the SRM of blockchain-based applications among the security experts and practitioners.
To enable reuse of domain knowledge.	To enable the reuse of information security domain knowledge to construct the unified and formal knowledge models for the SRM of traditional and blockchain-based applications.
To make domain assumptions explicit.	To make explicit specifications and explanations about the characteristics of blockchain for the new users, security experts, and practitioners. The ontology also makes possible an easy change when new concepts emerge.
To separate domain knowledge from operational knowledge.	To describe information security knowledge from its ontology components according to a required specification.
To analyze domain knowledge.	Formalizing information security into an ontological knowledge domain that permits security experts to analyze the security of traditional and blockchain-based applications.

Our perspectives (Table 4) to build an ontology deliver the practical contribution of the ontology-based reference framework by describing the security knowledge and SRM of traditional and blockchain-based applications. The mentioned reasons are subsequently validated based on the aggregated results of this research in Section 8. Similarly, the theoretical contribution is essential for scientific understanding of a given topic of interest and states phenomena as they are [56]. The

theoretical contribution concerns understanding blockchain as a countermeasure solution for mitigating the security threats of traditional applications. For instance, to perform the SRM of traditional applications, blockchain-based countermeasure constructs have not been thoroughly investigated previously [16]. Additionally, this work articulates the underlying process that enables the SRM of blockchain-based applications to describe the security threats that appear in them and their possible countermeasures. Moreover, the present work objectifies overcoming the conceptual ambiguity and confusion in treating the security threats that emerge due to constantly evolving techniques and design concepts within the blockchain that are often interchangeable [14, 25]. So, the findings reinforce the necessity to develop the ontology-based reference framework that acknowledges the fundamental concepts of SRM, blockchain as a countermeasure solution, and the SRM of blockchain-based applications. Hence, the key theoretical contribution of our work is the development of an ontology-based reference framework that can be used for the SRM of traditional and blockchain-based applications.

We use OWL (Web Ontology Language) to build the ontology-based reference framework. OWL is based on description logic (DL) and W3C (WWW Consortium) standards. OWL is a semantic web language to illustrate rich and complex knowledge about things, groups of things, and relations between things [57]. OWL combines concepts (e.g., classes/subclasses) within a domain (e.g., information security domain), object properties (e.g., relationships of concepts), data properties, restrictions, individuals, and inference that is an automatic procedure to compute conclusions based on evidence and reasoning within an ontology.

DL deals with formal knowledge representation and provides a logical formalism for ontology. DL illustrates the fundamental modeling concept relating to roles and concepts [58]. DL-based knowledge includes two components: *(i)* terminological component (TBox), and *(ii)* assertion component (ABox) [59]. TBox is a conceptualization associated with a set of facts (e.g., ABox) within a knowledge domain. In Protégé, the DL query tab is a plug-in feature to search knowledge from an ontology when it is consistent and the reasoner is working. OWL supports a resource descriptive framework (RDF) to define metadata models [60]. For example, the RDF is a language that allows encoding, exchange, and reuse of structured metadata [61] to build a readable semantic infrastructure [60]. We use SPARQL (SPARQL Protocol and RDF Query Language) as a semantic query language to retrieve and manipulate domain knowledge that is mapped in RDF format [62]. SPARQL is a W3C recommended query language to write semantic queries and use RDF middle-ware to get results from an ontology. For example, the following SPARQL query gets the system assets from the ontology.

```
SELECT DISTINCT ?SystemAsset
    WHERE { ?SystemAsset rdfs:subClassOf SystemAsset }
```

We construct our ontology using Protégé ontology editor[1]. Protégé is a free,

[1] https://protege.stanford.edu

open-source, and most adapted ontology editor [63] that was proposed by Stanford University in California. Protégé provides a graphical user interface, lets users modify ontologies in OWL, and allows description logic classifiers and reasoners (e.g., pellet reasoner[2]) to maintain consistency inside the ontology. Furthermore, Protégé has several visualization tools, including OWLViz[3], which we used to create the "is-a" tree view hierarchies, and OntoGraf[4] for interactively navigating the relationships of the concepts. We utilize the ontology construction method (Fig. 7) proposed by Uschold and Gruninger [64] that provides the outline for developing and evaluating an ontology. The method has five distinctive stages: *(i)* Identify purpose and scope, *(ii)* Building ontology, which includes capture, coding, and integrating phases, *(iii)* evaluation, *(iv)* documentation, and *(v)* guidelines.

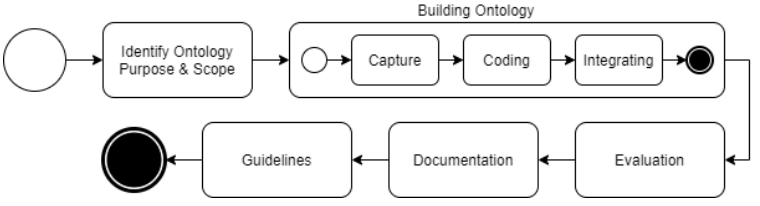

Figure 7. Ontology construction method

2.3.1. Ontology Scope

We begin the ontology development by defining its purpose and scope to clarify its intended uses. The purpose of the ontology is to provide unified and formal knowledge models, clear understanding, and communication of information security knowledge to support the SRM of traditional and blockchain-based applications. The detailed purpose of an ontology is discussed in Chapter 1. We follow the instructions of Noy and McGuinness [52] to define the scope of our ontology representations by answering the following questions:

Q#1: What is the domain that the ontology will cover? The first question helps to determine the domain of our ontology. For example, the domain is information security of traditional and blockchain-based applications. We utilize the SRM domain model to elicit the concepts and relationships of the ontology (Table 3).

Q#2: For what we are going to use the ontology? The ontology would help to perform the SRM from two different perspectives. Firstly, the security threats are mitigated by using blockchain-based applications. Secondly, the security threats that may appear within blockchain-based applications.

[2]https://www.w3.org/2001/sw/wiki/Pellet
[3]https://protegewiki.stanford.edu/wiki/OWLViz
[4]https://protegewiki.stanford.edu/wiki/OntoGraf

Q#3: What types of questions the ontology should provide answers to? To determine the scope of our ontology, we create a list of competence questions the ontology should answer. These questions help determine whether the ontology has enough knowledge to address these inquiries or whether more information is required. For example, the ontology should answer the following questions during the SRM process.

- What assets to secure?
- What system assets support business assets?
- What security threats are mitigated by using blockchain-based applications?
- What security threats appear within blockchain-based applications?
- What are the vulnerabilities?
- What security controls are in place to mitigate the vulnerabilities?

Q#4: Who will use and maintain the ontology? The ontology will be used and maintained by the security professionals, developers, practitioners, or anybody interested in learning the SRM of traditional and blockchain-based applications.

These questions serve as the starting point to identify the scope of our ontology. To further refine the scope of our ontology, we utilize the concepts and relationships from the SRM domain model (Table 3). These concepts are associated with the assets-, risks-, and risks treatment. We only chose the concepts related to the threat and vulnerability from the risk-related concepts, and countermeasure from the risk treatment-related concepts. It also allows us to narrow the scope of our ontology, making it more understandable to a broader audience.

2.3.2. Ontology Building

In building an ontology, we first create an upper-level reference ontology that serves as an ontology-based reference framework. The framework provides a foundational structure for encoding information security concepts in accordance with the fundamental constructs of SRM. Later, we propose two ontology representations (e.g., financial and healthcare domains) as proofs of concept. The framework establishes a common foundation for ontology representations, semantic interoperability, and generic concepts in the information security domain. We capture the domain information (e.g., concepts and relationships) for both ontology representations and classify it into taxonomic structures by extending the ontology-based reference framework. The built classifications refine the concepts related to system assets, business assets, security criteria, security threats, vulnerabilities, countermeasures, and improve the technical vocabulary of concepts. We code the concepts and relationships using the Protégé to formalize the domain knowledge in our ontology representations.

2.3.3. Ontology Evaluation

An ontology's scientific significance is enhanced through effective assessment. We assess the constructed ontology-based reference framework and ontology representations using various methods—such as expert discussion, CPNs-based evaluation, automated tools, qualitative assessment, tasks-based evaluation, and use case evaluation. For instance, we reach the domain experts to verify the upper-level reference ontology to ensure whether we are building the ontology right and fulfilling our requirements. Second, we use the CPNs models and streams to validate whether we are building the right ontology based on the requirements. Furthermore, we used the automated tools to check the consistency and coded concepts' correctness, qualitative assessment to check the ontology representations model the real-world domain knowledge for which the ontology was created. Tasks-based evaluation to check whether the encoded concepts and relationships have been encoded as intended and resulted in certain outputs, and use case evaluation to check the applicability.

2.3.4. Ontology Documentation and Guidelines

Inadequate documentation of ontology hurdles in effective knowledge sharing and understanding [65]. To overcome this issue, we document the concepts, such as classes, sub-classes, and relationships that clarify what ontology is about and interpret the meaning of ontological claims. We use the Protégé annotation properties to document the terms we used to build an ontology. The guidelines include the resources to educate users unfamiliar with OWL or OWL-based tools. Our ontology is created by using the Protégé ontology editor, and the ontology is publicly available on GitHub and accessible online via the marine metadata interoperability semantic framework (MMISW) repository. We use the OntoGraf to generate classifications graphs and pellet reasoner to validate the consistency of our ontology. To use the ontology, install Protégé, load an ontology, and retrieve information from it using the SPARQL queries.

2.4. Systematic Literature Review

Since the advent of Bitcoin, blockchain is growing and capturing the interest of various sectors to transform their business processes. Blockchain technology promises to overcome security challenges, enhance data integrity, and transform the transacting process into a decentralized, transparent, and immutable manner. Blockchain-based applications are considered less vulnerable because of decentralized consensus, immutable ledger, and cryptography. However, various security threats and vulnerabilities are reported within blockchain-based applications. Moreover, the involvement of monetary assets raises security concerns, mainly when the attacker may steal the assets or damage the system. For example, in a decentralized autonomous organization (DAO), the attacker exploited the reen-

trancy vulnerability in Ethereum smart contract and gained control of $60 million Ethers. Therefore, security becomes an important factor and plays a vital role in guaranteeing blockchain acceptability. Hence, the security challenges are debatable, and there is no comprehensive overview of the security threats which can potentially be mitigated using the blockchain or appear within blockchain-based applications. However, there exist a few studies reporting on the security challenges in the blockchain platforms [26, 66], but there is still a lack of focus on the blockchain-based application's security.

Here, we conduct a systematic literature review (SLR) following the guidelines of Kitchenham et al. [67]. The outcomes of this SRL are *(i)* a list of security threats mitigated using the blockchain-based applications, *(ii)* security threats that appear in blockchain-based applications, *(iii)* a list of countermeasures, and 4) an overview of the prominent research domains which are nourishing by the blockchain. The results of this study could be seen as a preliminary checklist of security threats when implementing blockchain-based applications.

2.4.1. Review Method

To achieve the objectives of the SLR, we follow the SLR guidelines of Kitchenham et al. [67] and accumulate the four research questions. For example, *(i)* what are the domains where blockchain is applied? *(ii)* What security threats are mitigated using the blockchain? *(iii)* What security threats appear within blockchain-based applications? and *(iv)* what are the countermeasures to mitigate the security threats? We use the primary search, backward and forward tracing techniques [68, 69] to collect the relevant studies. For instance, we perform a primary search based on search queries to identify an initial set of papers and a secondary search employing backward and forward tracing. The search queries (including alternative terms and synonyms) are formulated to gather literature studies discussing blockchain, blockchain-based applications, and security aspects. The selection of electronic databases and literature search is carried out by consulting with experts in software security. Literature studies are collected from the ACM digital library, IEEE Xplore, ScienceDirect, SpringerLink, Scopus, and Web of Science.

Search queries: *(("permissionless" OR "permissioned") AND ("blockchain" OR "blockchain-based" OR "decentralized") AND ("applications" OR "systems" OR "services")) AND ("security" OR "security threats" OR "security risks" OR "security risk assessment" OR "security risk management"))*

We run the search queries on the selected electronic databases and collected the literature studies. We also included non-academic research (e.g., gray literature). We applied exclusion (EC) and inclusion (IC) criteria to identify only the relevant papers. For example, the papers that were duplicates, not in English, shorter than five pages, inaccessible (via university subscriptions or internet search), or pub-

lished before 2008 were excluded (EC1 & EC2). We included the paper within the blockchain domain, covering the security aspects of blockchain and providing the blockchain-based countermeasures (IC1). To identify the security threats of blockchain-based applications and their potential countermeasures, we search for papers that discuss the security threats of blockchain-based applications and countermeasures to mitigate them (IC2).

Table 5. Inclusion and exclusion criteria

Inclusion criteria	
IC1	Papers discuss the security threats of traditional applications and blockchain-based countermeasures to mitigate them.
IC2	Papers discuss the security threats of blockchain-based applications and countermeasures to mitigate them.
Exclusion criteria	
EC1	Papers published before 2008 and not available freely
EC2	Papers shorter than five pages and not written in English

The studies were selected after reading the paper title, abstract, introduction, and conclusion sections. We follow the quality guidelines of [67] and based on the research scope of our study, we have assessed the quality of studies using the following questions:

- Are the goals and purpose of a study clearly stated?
- Is the study describing security threats on blockchain-based applications?
- Is the study providing the countermeasures to mitigate security threats?
- Is the study answered the defined research questions?
- How well the research results are presented?

The answers to these questions are scored as: 1=fully satisfied, 0.5=partially satisfied, 0=not satisfied. The studies with 2.5 or more points are included.

Table 6. Literature studies

Database	**Total**	**Exclusion**	**Inclusion**
ACM	24	13	11
IEEE	37	10	27
ScienceDirect	24	17	7
SpringerLink	30	15	15
Scopus	46	27	19
Total	161	82	79

Table 6 presents the screening results. Initially, a total of 161 studies were collected. Later, 82 studies were excluded by applying the inclusion, exclusion, and quality assessment criteria. Finally, 79 studies remained. We utilize the data extraction form (Table 7) to systematically extract the relevant data from the chosen literature studies and arrange the data in a way that would aid in the assessment of literature study quality, data synthesis, and conclusion-making. The extracted information mainly outlines the study identification, research problem, application domain, blockchain attributes, assets, security threats, and countermeasures. One sample of the data extraction procedure utilizing the data extraction form is shown in Table 7. In the data synthesis stage, we observe that Ethereum-based applications are gaining popularity among others. Also, the permissioned blockchain

platforms (e.g., HLF and customized permissioned (CP)) are arising because they support various industry-based use cases beyond cryptocurrencies. Also, we identify various customized permissionless (CPL) blockchain platforms to achieve customized tasks, fulfill specific business needs, and overcome the limitations of other blockchain platforms. The term generic refers to the studies where the blockchain type and platform are not explicitly mentioned.

Table 7. Data extraction form and extracted data sample

Study information	
Title	A Privacy-Preserving Voting Protocol on Blockchain
Context	Proposed privacy-preserving voting protocol on Hyperledger Fabric. The paper presents the communication flow among blockchain network peers, smart contract, and ledger of the voting protocol. The proposed voting protocol can protect the voters' privacy and enables detection and correction against cheating without any trusted party.
Source, year	IEEE, 2018
Author	W. Zhang et al.,
Threat information	
Domain	Voting
Blockchain type	Permissioned
Blockchain platform	Hyperledger Fabric
Threat (domain)	Data Tampering *(Mitigated)*
Threat reference	STRIDE threat model
Vulnerability	Dependency on third parties and weak controls for cheating detection and correction.
System assets Business assets	Database, Data operation, Voters, Voting Voting data, Voters data
Negated security criteria	Integrity
Consequence	Election frauds, negated voting system and result's integrity.
Risk-treatment decision	Risk reduction
Countermeasure	Decentralized and distributed voting and tallying service. Blockchain enabled cryptography-based votes verification. Permission settings of permissioned blockchain.

2.4.2. Applications Domains

Table 8 presents the quantity of research areas using blockchain. We categorize the results in application domains and technology solutions based on the type of blockchain and blockchain platform. The results showing the healthcare and financial domains are the most studied application domains using blockchain-based solutions. Moreover, the practitioners use blockchain as a technology solution where blockchain is implemented as a security layer. The results indicate that overall, Ethereum is a widely used permissionless blockchain for building decentralized applications and is famous in healthcare applications. In contrast, permissioned blockchains are arising in financial applications.

2.4.3. Security Threats Mitigated

Security threats result in harm to the system, and its components [70]. Table 9 presents the common security threats of traditional applications, and the researchers are utilizing the blockchain to overcome them. For example, data tampering is

Table 8. Research areas based on different blockchain platforms

	Permissionless			Permissioned			
	Bitcoin	**Ethereum**	**CPL**	**HLF**	**CP**	**Generic**	**Total**
Applications domains where blockchain is used							
Healthcare *(HC)*	0	5	1	2	4	1	13
Financial *(FI)*	2	1	1	2	5	0	11
Resource monitoring and digital rights management *(RM)*	1	3	2	0	2	1	9
Smart vehicles *(SV)*	1	0	1	1	3	0	6
Voting *(VO)*	1	1	0	2	0	1	5
Technology solutions where blockchain is used							
Security layer *(SL)*	6	7	1	0	1	0	15
Internet of things *(IoT)*	2	2	1	2	2	0	9
Total	13	19	7	9	17	3	68

mitigated in healthcare applications by using blockchain-based applications. In addition to these security threats, other threats found once or twice in the literature studies are side-channel attacks, impersonation attacks, phishing, and social engineering threats, password attacks, cache poisoning, arbitrary attacks, dropping attacks, appending attacks, authentication attacks, signature forgery attacks, keyword guess attacks, chosen message attacks, audit server attacks, inference attacks, binding attacks, and Bleichenbach-style attacks.

Table 9. Security threats mitigated using blockchain

	Permissionless			Permissioned			
	Bitcoin	**Ethereum**	**CPL**	**HFL**	**CP**	**Generic**	**Total**
Data tampering	7	11	4	7	9	1	39
DoS/DDoS attack	7	7	5	3	2	1	25
MitM attack	3	6	2	2	0	1	14
Identity theft/hijacking	1	0	3	0	0	1	5
Spoofing attack	2	0	1	0	1	0	4
Other threats	6	4	2	1	2	2	17
Total	26	27	17	13	14	6	104

In Table 10, we encompass the security threats along with the applications where the blockchain is used to mitigate frequently occurring security threats of traditional applications. The security threats are mitigated in the domain of healthcare, and finance, followed by resource monitoring, digital rights management applications, smart vehicles, and voting applications. Also, the blockchain is presented as a technology solution where researchers incorporated blockchain as a security layer to protect against the various security threats of traditional applications. Furthermore, a blockchain technology solution for IoT-based applications is rapidly increasing because it provides integrity, reliability, and security [20] and these are important for IoT-based solutions to reach high requirements of security.

2.4.4. Security Threats Appeared

Table 11 represents the security threats that appear within the blockchain-based applications where the Sybil and Double-spending attacks are the common security threats. The results reveal that security threats appear more frequently in permissionless blockchains than in permissioned blockchains. Apart from the secu-

Table 10. Applications where security threats are mitigated using blockchain

	Applications					Technology			
	HC	**FI**	**RM**	**SV**	**VO**	**SL**	**IoT**	**Other**	**Total**
Data tampering	8	6	5	4	3	2	5	6	39
DoS/DDoS attack	0	1	5	3	1	7	3	5	25
MitM attack	1	1	4	1	1	2	2	2	14
Identity theft/hijacking	0	1	2	0	0	0	1	1	5
Spoofing attack	0	0	0	0	1	0	1	2	4
Other threats	2	1	0	0	1	5	5	3	17
Total	11	10	16	8	7	16	17	19	104

rity threats of Table 11, other security threats which are appeared once or twice in the blockchain-based applications are eclipse attacks, block withholding attacks, 25% attacks, stake grinding attacks, block discarding attacks, a difficulty raising attacks, pool-hopping attacks, node masquerading attacks, blockchain timestamp attacks, balance attacks, signature forgery attacks, confidentiality attacks, private keys compromise, overspending attacks, collusion attacks, and illegal activities. In Table 12 we encompass the security threats along with the blockchain-based applications research areas to show which security threats are more frequently occurring on different blockchain-based applications. The security threats are frequently exposed in the resource monitoring and digital rights management applications, followed by the security layer and financial applications. Overall, the results show that the blockchain-based applications do not become the silver bullet with respect to securing the systems against different security threats. The attacker can exploit these security threats, and affect the valuable assets, and services of blockchain-based applications.

Table 11. Security threats appear within blockchain-based applications

	Permissionless			Permissioned			
	Bitcoin	**Ethereum**	**CPL**	**HLF**	**CP**	**Generic**	**Total**
Sybil attack	5	1	1	1	1	1	10
Double-spending	4	1	2	0	0	1	8
51% attack	3	3	1	0	0	1	8
Deanonymization	2	1	3	0	0	1	7
Replay attack	2	4	1	0	0	0	7
Quantum comp. threat	0	1	1	2	0	1	5
Selfish mining attack	1	0	2	1	0	0	4
SC reentrancy attack	0	2	0	0	0	1	3
Other threats	6	1	6	3	1	3	20
Total	23	14	17	7	2	9	72

2.4.5. Related Work

There exist a few surveys which consider blockchain platforms' security threats. For instance, Li et al. [66] overview the security aspects of the blockchain platforms, collect the security attacks, and summarize the security enhancements. Another related work [26] that is conducted on Ethereum smart contracts security and reports on the major security attacks and presents a taxonomy of common programming pitfalls which can result in different vulnerabilities. This study focuses on the security risks in the Ethereum-based smart contracts; further investigation

Table 12. Blockchain-based applications domains where security threats appear

	Applications					Technology			
	HC	**FI**	**RM**	**SV**	**VO**	**SL**	**IoT**	**Other**	**Total**
Sybil attack	1	1	1	1	2	1	1	2	10
Double-spending	0	2	2	0	0	2	0	2	8
51% attack	0	0	4	0	1	1	0	2	8
Deanonymization	0	1	2	1	1	1	1	0	7
Replay attack	0	1	2	0	0	4	0	0	7
Quantum comp. threat	1	0	0	0	0	2	0	2	5
Selfish mining	0	1	1	0	0	2	0	0	4
SC reentrancy	0	0	0	0	0	3	0	0	3
Other threats	0	5	11	0	0	2	1	1	20
Total	2	11	23	2	4	18	3	9	72

is required to explore possible security threats in smart contracts-based blockchain applications and their viable countermeasures.

The main attributes of blockchain are integrity, reliability, and security [71] which are important and required aspects in the IoT systems. The conventional approaches and reference frameworks of IoT networks are still unable to fulfill the security requirements [20]. Minhaj et al. [20] survey major security issues of IoT and discuss different countermeasures along with the blockchain-based solution. However, the survey is generic in the IoT domain and does not detail the security challenges belonging to any particular blockchain-based IoT applications. Our study reviews the different blockchain-based IoT applications and discusses their security threats and potential countermeasures. More specifically, in our SLR, we consider the security threats that are mitigated by using the blockchain, security threats of the blockchain-based applications, and countermeasures.

2.4.6. Discussion

In this SLR, we discuss how blockchain-based applications handle security threats of traditional applications and what security threats are reported in blockchain-based applications. Also, we highlight the application domains where blockchain is employed. For instance, the results indicate that Ethereum is a widely used permissionless blockchain for building decentralized applications in healthcare applications, whereas permissioned blockchains are arising in financial applications. Our findings may be used as a preliminary checklist to help developers make decisions when creating blockchain-based apps.

Our current SLR study has a few limitations. For example, the majority of blockchain-based applications we analyzed are in the prototype phase. Therefore, we cannot claim the illustrations of different security threats are entirely justifiable. Since the area of decentralized applications is constantly developing, not all potential security threats are being investigated. Therefore, a variety of security threats may likely appear in the future. This study discovered that many security threats and their countermeasures are either obscure to understand or lack a practical application. Overcoming these limitations can contribute to explaining the blockchain-based applications' security aspects more in-depth.

3. REFERENCE MODEL FOR SECURITY RISK MANAGEMENT

In this chapter, we answer the research question RQ1: What is the current state of the art for performing the SRM of traditional applications utilizing blockchain and the SRM of blockchain-based applications? The outcome of this chapter is a blockchain-based reference model (BbRM) that adheres to the SRM domain model. To support the construction of BbRM, we present the architecture highlighting the main components of traditional applications and a conceptual model for the SRM. The BbRM concentrates on the SRM of both traditional and blockchain-based applications. Initially, the BbRM presents the SRM of traditional applications, including the security threats that can be mitigated by employing blockchain technology. Further, we have extended the BbRM to incorporate the security threats that may appear in blockchain-based applications. Also, this chapter shows the implementation of BbRM. For example, we use the BbRM to explore and analyze the data tampering, Sybil attack, and Double-spending.

The remainder of the chapter is structured as follows: Section 3.1 builds the BbRM upon the results of a systematic literature review. The BbRM enables the SRM of both traditional and blockchain-based applications. Section 3.2 is the implementation of BbRM where we further explore the data tampering threat and demonstrate how traditional-, Ethereum- and HLF-based countermeasures can mitigate it. We compare the benefits and drawbacks of each approach. Section 3.3 & 3.4 also illustrate the implementation of BbRM where we explore the Sybil attack and Double-spending in detail. Both sections bring the comprehensive SRM framework based on the BbRM for Sybil attack and Double-spending. Section 3.5 evaluates the SRM framework by employing it on two different blockchain-based healthcare applications. Section 3.6 discusses the related work. Section 3.7 is the discussion, and Section 3.8 concludes this chapter.

3.1. Blockchain-based Reference Model (BbRM)

Nowadays, information systems comprise different components that collect, store, and process data to provide information and the promised services. These components are inter-connected and communicate with each other upon the request of the user to meet business and user requirements [72]. The architecture (Fig. 8) of traditional applications is a structural categorization of various components that are inter-connected and communicate with each other. For example, the user interacts with the application through a device to utilize the service. The service performs data operations to accomplish the requested task and activity. Data operation checks the access right rule, and if the user has valid access rights, then data operation can access the data object. The data object contains different methods (e.g., generate, store, view, share, and interpret data) and returns based on the

requested data operation type. The database also follows the access right rule, and if valid access rights, the database provides data to the data object. The database keeps the data (e.g., relational, non-relation) and computed data (e.g., reports, media). The application implements a communication mechanism using communication protocols to maintain secure data and request flow. The infrastructure includes the network that defines the routing, server (e.g., web, cloud), database server, and logs (e.g., error, access, action logs).

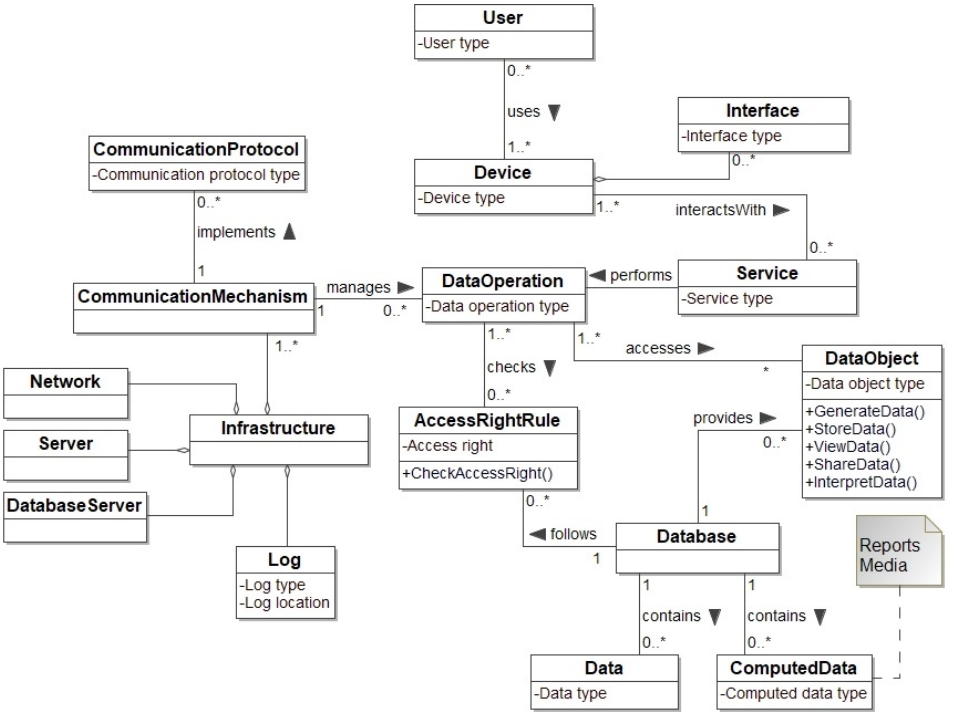

Figure 8. Architecture of traditional applications

We present a conceptual model (Fig. 9) that uses the components of the SRM domain model (Fig. 5) and the SRM process (Fig. 6) to explain the fundamental concepts of security risk assessment. The conceptual model satisfies the requirements of the ISO/IEC 27001 information security standard [48] and SRM standards [23]. As stated in a conceptual model (Fig. 9), the SRM process consists of six stages, e.g., it presents the *(i)* context and assets identification, *(ii)* security objective determination, *(iii)* risk analysis and assessment, *(iv)* risk treatment, *(v)* security requirements determination, and *(vi)* control selection and implementation. The SRM process begins by characterizing the system context and identifying its assets. The assets are classified as system assets and business assets. Then, the process allows determining the security objective of the business assets to define the security needs in terms of confidentiality, integrity, and availability. Next, we perform the security risk analysis and assessment to identify the threats that exploit various vulnerabilities that harm the assets and negate the security

criteria of the business assets. After the risk assessment, the risk treatment decisions are taken (e.g., risk avoidance, reduction, transferring, and acceptance), and in our case, we focus only on the risk reduction-based treatments. To mitigate the security threats, the process allows eliciting the security requirements implemented as controls (e.g., system-specific countermeasures [23]). The dotted concepts (e.g., threat agent, attack method, and security requirement) are not pursued further in this study. In accordance with the conceptual model, we present the context for SRM of traditional applications using blockchain (Table 13). The context refines and illustrates the security threats, what system assets to secure, business assets, their security criteria, and blockchain-based countermeasures to implement as controls.

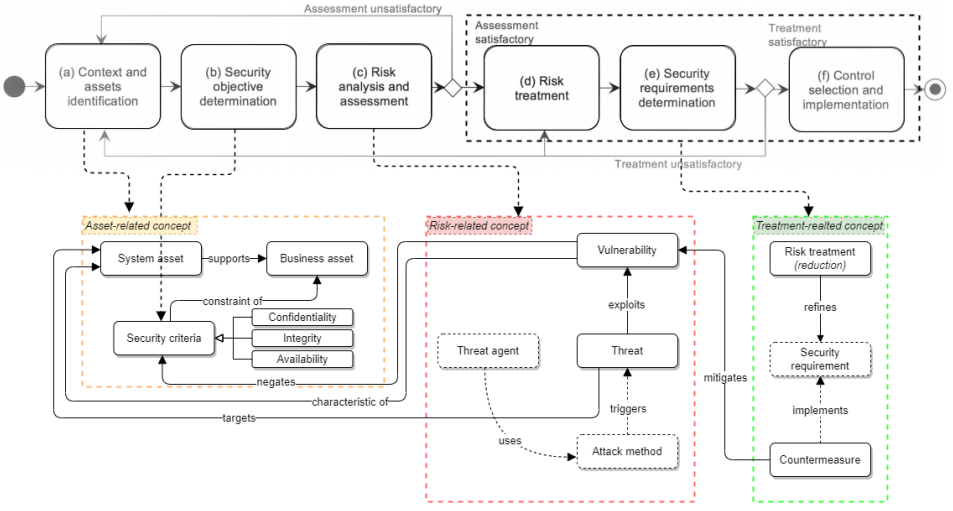

Figure 9. Conceptual model generalizing the SRM domain model and the SRM process

As previously mentioned, we utilize the SRM domain model and the SRM process as a basis to establish the conceptual model and to ensure that the conceptual model is built right (Fig. 10). As an input, we chose the architecture of traditional applications (Fig. 8) and the most common security threats based on the SLR results that frequently arise in traditional applications (Section 2.4) and can be mitigated using the blockchain-based application. The conceptual model plays an important role in the development of the reference model for security risk management of traditional applications using the blockchain, thus, fulfilling the fundamental objectives of the conceptual model [73]. For instance, the conceptual model provides a point of reference to interpret the SRM domain model constructs and to extract systems' specifications for the SRM of traditional and blockchain-based applications. The conceptual model improves comprehension of the representative system and allows for the effective exchange of system specifics among stakeholders. It also serves to document the system by addressing the design and knowledge problems for future reference [74].

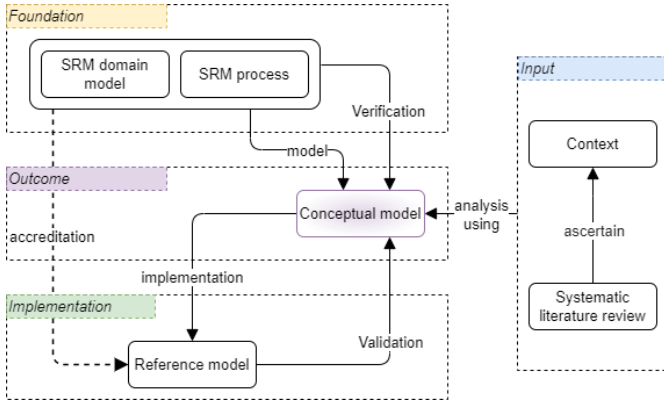

Figure 10. Role of conceptual model for building the blockchain-based reference model

The BbRM (Fig. 11) is presented as an implementation of the conceptual model and validates that whether we built the right conceptual model. The BbRM presents blockchain as a countermeasure solution where blockchain provides various security controls by design that help to mitigate the security threats of traditional applications. We use the «SystemAsset» stereotype in BbRM to define the system asset explicitly. The system assets play an important role in providing functionality and services to users. At this point, we forgo the vulnerability analysis and assume, in a manner conforming with the SRM domain model, that security threats exploit weaknesses of vulnerable system assets highlighted in the BbRM. We use the «BcCountermeasure» stereotype for the blockchain-based countermeasures that protect system assets against potential security threats.

Table 13. Context for SRM of traditional applications using blockchain

Threat	**System asset**	**Business asset**	**Blockchain**
Data Tampering	Database, Data operations, Access rights	Data (I), Computed data (I), Data object (I)	- Immutable ledger - Consensus mechanism - Decentralized access control
DDoS	Network, Server, Database server	Service (A), Database (A)	- P2P network - Decentralized
MitM	Communication mechanism, Communication protocol	Data (C, I), Communication (I), Data destination (I, A)	- P2P network - Distributed nodes - Encrypted communication
Repudiation	Access rights, Data operations, Logs	Data (I), Service (I)	- Immutable ledger - Provenance - Decentralized access control
Identity spoofing	User, Device, Communication mechanism	User identity (C), Data (C), Service (A)	- P2P network - Digital signatures - Decentralized access control

Here, we briefly explain the identified security threats of traditional applications and how they are mitigated using blockchain. For instance, **data tampering** exploits the weaknesses of vulnerable system assets (device, data object, data repository, and logging) and negates the security criteria (integrity) of business assets. Blockchain supports append-only immutable ledger, which ensures tamper-proof recording of transactions. Once the record is stored in the ledger, it cannot

be altered easily [75]. The immutable ledger brings data security and proof that data has not been changed or altered. Blockchain utilizes a consensus mechanism to ensure that data correctness is properly maintained [8]. Blockchain is decentralized in nature and supports decentralized access controls written in smart contracts that are immutable and not controlled by a centralized authority [76]. These characteristics redefine the entire data process within the blockchain to make data more secure and integral. The data tampering is further explored in Section 3.2 by using the BbRM as an input. **Denial of service (DoS)** combines DoS and DDoS (Distributed Denial of service), impeding legitimate requests by flooding the system with malicious traffic. DoS attack targets a server, network, and database to negate the security criteria (availability) of business assets. Blockchain works on a P2P-based distributed and decentralized network where thousands of nodes are interconnected and do not rely on traditional centralized client-server architecture [77]. Also, if the attacker triggers a DoS attack, he should attack each node individually, making this uneconomical attack [34]. Thus, there exists no central server for flooding with malicious traffic to deny the services to users.

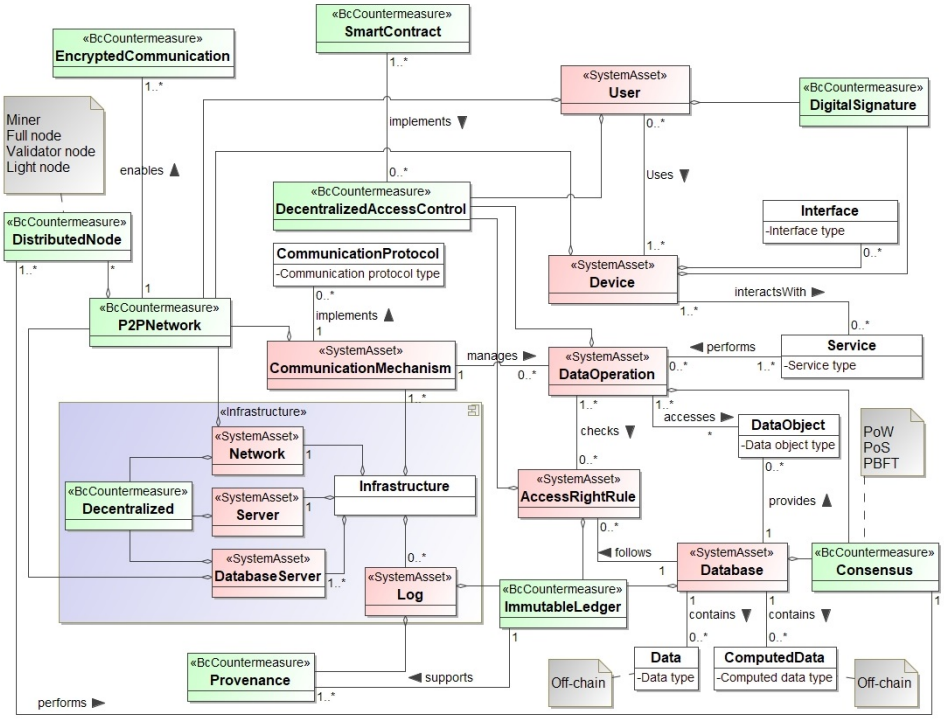

Figure 11. Blockchain-based reference model for SRM of traditional applications

In **man-in-the-middle (MitM)**, the attacker inserts himself in-between the legitimate users' communication to sniff information and data hijacking. MitM targets the communication mechanism and protocol to negate the security criteria (confidentiality, integrity, availability) of business assets. In a blockchain P2P network, the distributed nodes are interconnected, where each node acts as

a client and server [78]. Therefore, the blockchain-based application does not require middleware to enable the communication channel between two parties. Blockchain enables encrypted communication and data storage in the form of hashes confirmed by a consensus mechanism [8, 75]. **Repudiation** targets the access rights, data operations, logs, and negates the security criteria (integrity) of business assets. The immutable ledger records each action with a timestamp that supports the provenance [79]. The provenance provides seamless auditing and tracing of when and who did what. Blockchain-based decentralized access control validates the ownership to access and perform a specific action on data [80]. In **identity spoofing**, the attacker hijacks the credentials of legitimate users and sends spoofed messages that appear to come from a legit user. This threat targets the user, device, and communication mechanism and negates the security criteria (confidentiality, availability) of business assets. Blockchain operates on a P2P network, uses digital signatures for communicating [81], and the decentralized access control employs robust authentication and authorization processes [82]. Such blockchain-enabled characteristics overcome the identity spoofing threat.

Like traditional applications, security plays a vital role in guaranteeing the acceptability of blockchain-based applications. Although blockchain-based applications are considered to be less vulnerable, they did not become the silver bullet concerning securing against different security threats. We extended the BbRM (Fig. 12) with the security threats that may appear in blockchain-based applications and illustrated the context for the SRM of blockchain-based applications (Table 14) that defines what system assets to secure, business assets, their security criteria, and the countermeasures. Similarly, we select the most common security threats based on the results of SLR (Section 2.4). In the extended BbRM, we use the «Countermeasure» stereotype to highlight the countermeasures applied to vulnerable system assets to make them secure against security threats. These countermeasures are not entirely blockchain-based, but they represent a variety of domain-specific controls that can be incorporated into blockchain-based applications.

Table 14. Context for SRM of blockchain-based applications

Threat	System asset	Business asset	Countermeasure
Sybil attack	P2P network, Distributed nodes, Mining protocol	Consensus (I), Nodes identity (I), Transaction (A)	- Network joining fee - Monitor nodes - Nodes authentication
Double-spending	Consensus, Block confirmations	Transaction (I), Digital asset (I)	- Pluggable consensus - Increase confirmed blocks
51% attack	Computing power, Consensus	Immutable ledger (I), Digital asset (I)	- Monitor computing power - Transaction fee
Deanonymization	Immutable ledger	Transaction (C), User identify (C)	- Fresh keys for each transaction - Mixing techniques - Zero-knowledge proofs
Replay attack	Message protocol, Digital asset	Immutable ledger (I), Transaction (A)	- Strong repeat protection - Opt-in repeat protection - Lock digital assets

Here, we briefly explain the identified security threats of blockchain-based applications and how they are mitigated. For instance, **Sybil attack** is a P2P network attack where the attacker creates fake distributed nodes to gain a considerable influence on the network [83]. Using the Sybil nodes, the attacker can halt the mining process and transaction confirmation. Sybil attack targets P2P network, nodes, mining protocol and negates the security criteria (integrity, availability) of business assets. Sybil attacks are difficult to prevent, but some preventive measures exist to limit them. For example, use the network joining fee to make it expensive for the attacker to create fake identities. Also, monitor nodes' behavior on the network, and perform nodes authentication before joining the network [84]. The Sybil attack is further explored in Section 3.3. **Double-spending** is a data consistency attack, and it happens when spending the same digital money (or digital asset) twice [85]. In general, Double-spending is a technique to deceive someone about the state of a transaction. Double-spending exploits the weaknesses of the consensus mechanism and blocks confirmations to negate the security criteria (integrity) of business assets. Use pluggable consensus based on the business models, and requirements of the blockchain system [29]. Also, increase the number of confirmed blocks to wait for more confirmations before accepting the transaction [35]. Double-spending is further explored in Section 3.4.

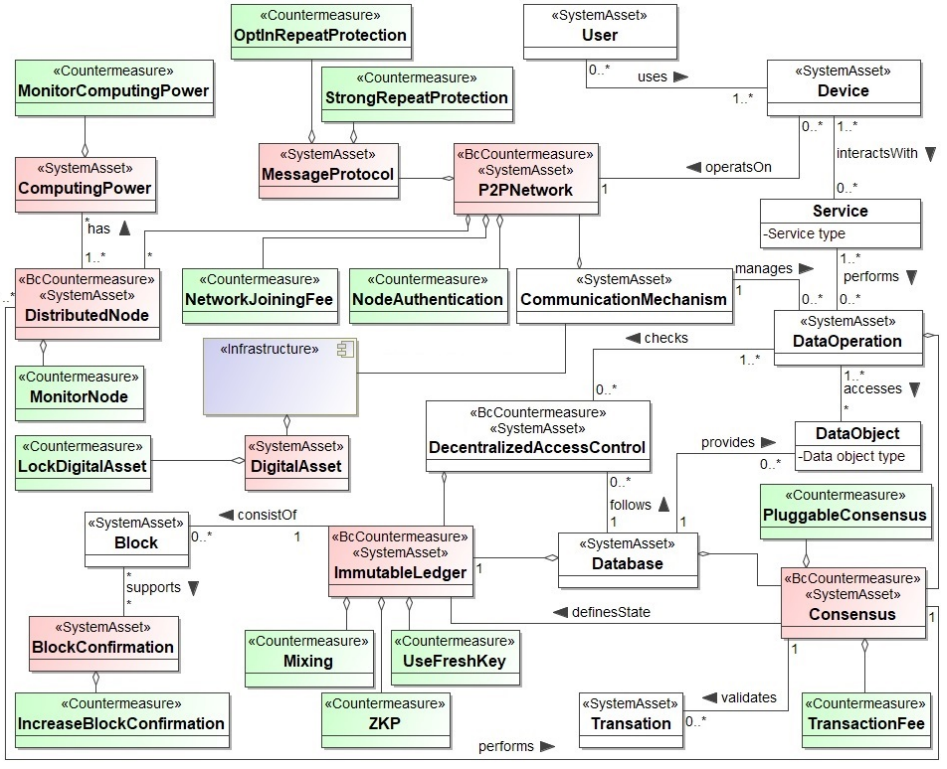

Figure 12. Blockchain-based reference model for SRM of blockchain-based applications

51% attack is a hash rate-based attack when the attacker gains 51 or more computing power in the blockchain network [86]. The attacker aims to double-spend or change the defined rules in the blockchain network to gain substantial monetary benefits. 51% attack targets the computing power, and consensus mechanism and negates the security criteria (integrity) of business assets. To protect against a 51% attack, use a power monitoring tool to monitor the computing power of nodes continuously and restrict when reaching a certain amount of computing power [87]. Moreover, incorporate transaction fees as an incentive mechanism to keep nodes honest in a blockchain network [86]. In **deanonymization attack**, the attacker links transactions to run a combined analysis to trace a user or company behind each transaction [88]. Deanonymization attack targets immutable ledger where the transactions are publicly available and negates the security criteria (confidentiality) of business assets. When initiating a transaction, use a new key (private and public) every time to overcome deanonymization attacks [88]. Various blockchain systems started using mixers as a service to overcome the deanonymization attack by obfuscating the transaction, and transaction flow [89].

Replay attack can happen during the blockchain hard fork, where the attacker can convince the forked network that the transaction processed on one ledger is valid on the second ledger [90]. Replay attack targets P2P network-based message protocol and digital assets and negates the security criteria (integrity, availability) of business assets. To overcome replay attacks, implement security protocols [90] such as strong repeat protection that adds a special marker in the transaction to make it invalid on a new ledger. Opt-in repeat protection does the same job but requires manual changes from the user if they wish to upgrade to a new system after the hard fork. Locking digital assets before the hard fork to a certain number of blocks can prevent a replay attack [90].

BbRM enables a go-to strategy for identifying potential security threats and their controls when they arise. BbRM also gives the visibility to make informed decisions on the SRM of traditional applications using the blockchain and the SRM of blockchain-based applications. Following this, we examine how the proposed BbRM may assist the SRM of traditional and blockchain-based applications? Particularly, we focus on the data tampering threat in traditional applications and how blockchain might operate as a countermeasure solution. In a similar context, to realize the true potential of blockchain-based applications, the first step is to explore and clarify the security threats that may appear. We extended our analysis to look at the Sybil attack and Double-spending in blockchain-based applications and their potential countermeasures. These security threats (i.e., data tampering, Sybil attack, Double-spending) are the most concerning which are observed in our SLR (Section 2.4). Therefore, we decided to examine them further in detail.

3.2. Data Tampering

Blockchain technology is emerging in different domains to overcome various security challenges. One is data tampering which involves the malicious modification of data by an unauthorized user. Data tampering is the leading security concern, which developers attempt to mitigate by using blockchain-based applications [78, 16, 91]. Mainly, data exists in two states; either in transit or stored. In both cases, tampering with the data can disrupt business operations and in a worst-case scenario, can put people's lives at risk, e.g., tampering with healthcare data [92]. Therefore, data becomes one of the most valuable assets in an organization. We use the newly built BbRM to explore the data tampering further to explain how it can be mitigated using blockchain-based applications. According to the BbRM, we describe the context and assets to secure, and vulnerabilities associated with the data tampering. Furthermore, we present the traditional-, Ethereum-, and HLF-based countermeasures to mitigate the data tampering. The results of this section can be considered when evaluating the software design from the perspective of data tampering to produce secure software using blockchain.

3.2.1. Context and Assets Identification

Table 15 shows the system assets to secure, the business assets, and their security criteria. For example, the system assets (e.g., database, data operation) support the business assets (e.g., patient and healthcare data), and (C - Confidentiality, I - Integrity, A - Availability) are the constraints of business assets.

Table 15. Assets to secure against data tampering

Paper	System asset	Business asset
[78, 93, 94]	*Database* (Patient data, healthcare data), *Data operation* (Store data, view data)	Patient data (I), Healthcare data (I)
[95]	*Database* (User preferences data), *Access right rule* (Access rights), *Data operation* (Store data)	User preferences (I), Privacy policies (I)
[77]	*Communication mechanism* (Response message), *Access right rule* (Access rights), *Data operation* (Store data)	Authentication (A), Response message (I)
[87]	*Database* (Resource consumption), *Data operation* (Store data, share data, interpret data)	Resource consumption data (I)
[75, 96]	*Database* (Voting data, voters data, voting result), *Data operation* (Store data, share data, interpret data)	Voting data (I), Voters data (C, I)
[97]	*Database* (Bidding data, bid-offer data), *Data operation* (Store data, interpret data)	Bidding data (I), Bid-offer data (I)
[91]	*Database* (Video recordings, CCTV settings), *Data operation* (Generate data, store data, share data)	Video recordings (I), CCTV settings (I)
[98]	*Database* (Drugs data, Supply chain data), *Data operation* (Store data, view data)	Drug certificate (I), Drug trail (I)

The assets are organized by using the BbRM from the literature studies used for SLR. We present the architecture (Fig. 13), which is an abstraction of the system assets and depicts the traditional application components. The architecture characterizes the components at four different layers that support the categorization of assets. The User Layer exposes the different users who can interact with the application via the Interface Layer that presents various interfaces of the application.

The user interacts with the services through their specific interface. The Service Layer consists of different services which execute upon the user's requested task. The Data Storage Layer shows the database that consists of data (e.g., relational, non-relation), computed data (e.g., reports, media), and access rights data.

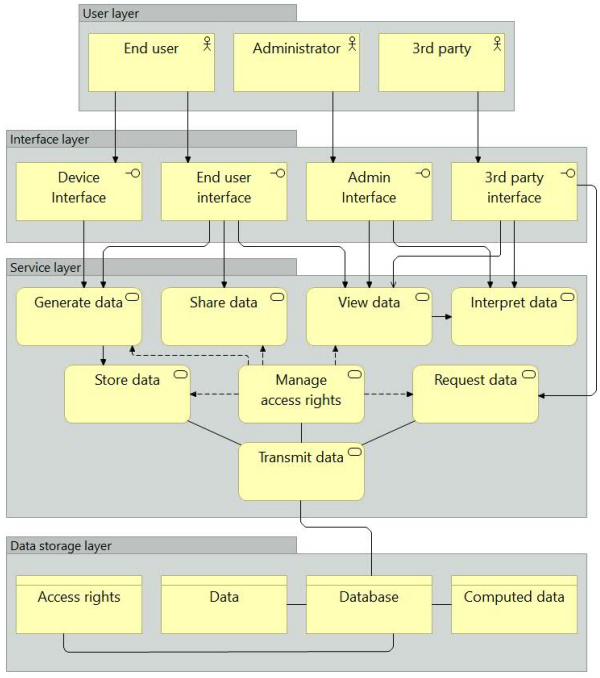

Figure 13. Assets architecture of traditional applications

We determined the most prevalent vulnerabilities (Table 16) from the literature studies that can lead to data tampering. We map the vulnerabilities on the assets architecture (Fig. 13) to visualize what are the vulnerable system assets (Fig. 14). The vulnerabilities connected to the system assets depict the weaknesses that allow the attacker to exploit them and negate the security criteria of business assets. For example, the Store data service is vulnerable because of error-prone data validation (V#1), lack of auditing and logging (V#2), and the insufficient use of cryptography (V#3). The Manage access rights service is vulnerable because of centralized or poorly implemented access control (V#4). Similarly, the Request data service is vulnerable due to an error-prone data validation (V#1), and the Data storage is vulnerable due to the insufficient use of cryptography (V#3) and weak communication mechanism (V#5).

3.2.2. Mitigation of Data Tampering

In this digital era, businesses regularly handle large volumes of data to make more informed decisions in their business processes. That data is critical, and prevention against data tampering has become necessary. According to Cypress Data

Table 16. Vulnerabilities that can cause the data tampering

Id	Vulnerability	Detail
V#1	Error-prone data validation [93, 94, 99]	Failure to adequately validate and verify data at the application's input and output points.
V#2	No auditing and logging [77]	No adequate system in place to tell when and who made modifications to the data.
V#3	Insufficient cryptography [8, 91]	Lack of cryptography or insecure usage of cryptography (e.g., weak encryption algorithms, flaws in encryption process) let the attacker modify the original message and pass it.
V#4	Centralized access control [8, 100, 95]	Lack of visibility and control into access management, for example, who is accessing data and making changes.
V#5	Weak communication mechanism [101, 102]	Weaken the data link layer security that let the attacker to perform data modification.

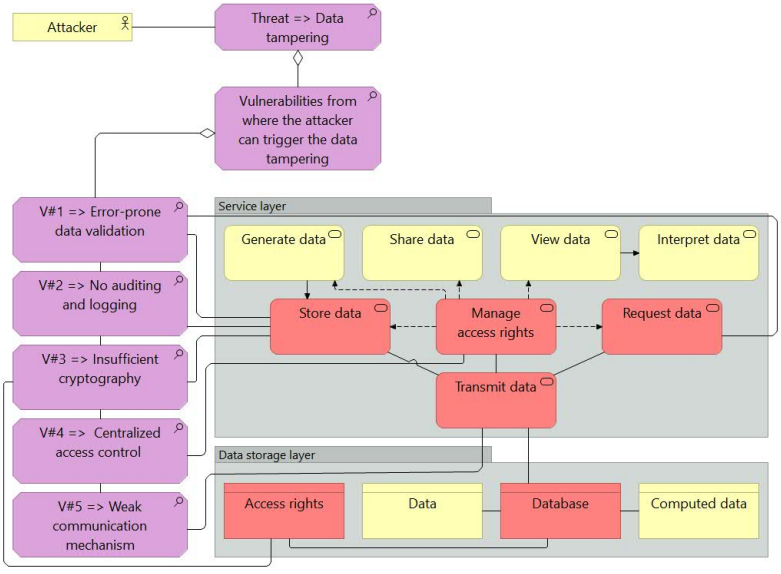

Figure 14. Architecture of data tampering to illustrate the vulnerable system assets

Defense [103], the system should contain two important aspects to work against data tampering. First, the system should support a tamper-proof lock to make data impervious to tampering. Second, the system should immediately detect and inform if any data tampering happens. To address the mitigation of data tampering, we present (*i*) traditional-, (*ii*) Ethereum-, and (*iii*) HLF-based countermeasures.

Traditional countermeasures: The traditional countermeasures (TC) (Table 17) are determined from the OWASP and STRIDE [104, 21], which has the corresponding set of countermeasures to reduce the data tampering threat. For example, the countermeasure (TC#1) employs the Store data and Request data to mitigate the vulnerability related to the error-prone data validation (V#1). Message authentication codes (MACs) are widely used to maintain data integrity. MACs verify that data is genuine, comes from a legitimate user, and has not been altered in transit. The countermeasure (TC#2) regarding the audit trails helps to mitigate

the no auditing and logging vulnerability (V#2) of Store data. The audit trails log the auditable events and allow the traceability of sensitive operations performed.

Table 17. Traditional countermeasures to mitigate data tampering

Id	Countermeasure	Mitigate
TC#1	Message authentication code	V#1
TC#2	Audit trails	V#2
TC#3	Cryptography and digital signatures	V#3
TC#4	Authorization and access control	V#4
TC#5	Secure communication with protocols	V#5

The countermeasure (TC#3) highlights the use of standard and approved cryptography and digital signatures schemes to overcome the insufficient cryptography vulnerability (V#3) of Store data, Transmit data, and Database. For example, when storing data in a database or delivering credentials and other sensitive data over the network, use robust encryption techniques. The countermeasure (TC#4) mitigates the poorly implemented or centralized access control (V#4). Centralized access controls are not tamper-proof and managed by the designated authority, which unfolds the traditional applications' weaknesses. However, the V#4 can be mitigated by limiting the access to specified operations using strong resource authorization and role-based access controls (RBAC) that should follow the least privileges principle. The countermeasure (TC#5) helps to mitigate the weak communication vulnerability (V#5). For instance, use tamper-resistant protocols, firewalls, and communication using transport layer security (TLS).

Ethereum-based countermeasures: Ethereum is a permissionless blockchain that provides an immutable distributed ledger, which ensures tamper-proof recording of transactions [93]. By design, Ethereum-based applications provide a number of approaches that help to assure data integrity. The Ethereum-based countermeasures (EC) (Table 18) were gathered from the literature studies that developed the Ethereum-based application to mitigate data tampering.

Table 18. Ethereum-based countermeasures to mitigate data tampering

Id	Countermeasure	Mitigate
EC#1	Decentralized data validation [77]	V#1
EC#2	Immutable ledger [93, 95, 96, 98]	V#2
EC#3	Strong cryptography for data storage and transmission [8, 91]	V#3
EC#4	Decentralized access control [95, 97]	V#4
EC#5	P2P network, decentralized data validation [77, 105]	V#5

The countermeasure (EC#1) mitigates the vulnerability related to the error-prone data validation (V#1) by providing decentralized data validation. On the Ethereum platform, the data validation performs by the decentralized distributed nodes. The blockchain's immutable ledger mitigates no auditing and logging vulnerability (V#2). The immutable ledger records each action performed over time and supports the provenance characteristic that enables a transparent, secure, and accurate data audit trail. The countermeasure (EC#3) mitigates the insufficient

cryptography vulnerability (V#3). Ethereum relies on cryptography to secure information exchange between parties and stores only encrypted data or the hashes of the data. The countermeasure (EC#4) mitigates the poorly implemented or centralized access control vulnerability (V#4). In an Ethereum-based application, a smart contract allows the implementation of decentralized access control that is tamper-proof, and no centralized designated authority manages it. The weak communication mechanism vulnerability (V#5) is controlled by the P2P network (EC#5) where distributed nodes act as a client and server. No middleware is required to enable communication between distributed nodes. Also, the invalid transmitted data is discarded during data validation and consensus stages (EC#5).

HLF-based countermeasures: As compared to the Ethereum platform, HLF is an example of a permissioned blockchain. HLF solves performance, scalability, and privacy issues by permissioned mode and fine-grained access control. Likewise, HLF-based decentralized applications introduce several techniques by design to mitigate data tampering. The HLF-based countermeasures (HC) (Table 19) are collected from the literature studies that built the HLF-based applications to mitigate data tampering. The HLF-based countermeasures are similar to Ethereum-based countermeasures because both follow the blockchain primitives. However, unlike Ethereum, HLF introduces the permissioned nodes with certain permissions that perform data validation and consensus. Moreover, HLF provides built-in decentralized access control.

Table 19. HLF-based countermeasures to mitigate data tampering

Id	Countermeasure	Mitigate
HC#1	Decentralized data validation [94, 98]	V#1
HC#2	Immutable ledger [78]	V#3
HC#3	Implement strong cryptography schemes for storage and transmission of data [75, 91, 98]	V#3
HC#4	Permission settings and decentralized access control [94]	V#4
HC#5	P2P network, decentralized data validation [34]	V#5

The countermeasure (HC#1) mitigates the vulnerability related to error-prone data validation (V#1). No auditing and logging vulnerability (V#2) is mitigated by the immutable ledger (HC#2) that supports the provenance and secure audit trails. The countermeasure (HC#3) mitigates the insufficient cryptography vulnerability (V#3) and HC#4 mitigates the poorly implemented or centralized access control vulnerability (V#4). HLF includes pre-verified nodes and assigns them certain permissions to allow access only to specific operations and data. Permission settings are an extra layer on the decentralized access control. Similar to the Ethereum case, the weak communication mechanism vulnerability (V#5) is mitigated by the P2P network, decentralized data validation, and consensus.

3.2.3. Discussion

In this section, we compare the approaches mentioned above by their respective countermeasures (Table 20). For example, to mitigate V#1, the traditional ap-

plications implement data validation using MACs that are vulnerable to multiple forgery attacks [106], and differential attack [107]. Another issue is that MACs do not protect against intentional changes in the message [108]. The attacker, for example, can modify the message and calculate a new checksum. The checksum algorithm will only detect randomly damaged parts of the message but not intentional modifications. The Ethereum-based application performs decentralized data validation by unverified data validator nodes. Data validator nodes validate the data and record it in a tamper-proof immutable ledger if valid. Similarly, HLF performs decentralized data validation by pre-verified nodes. As mentioned above HLF is a permissioned blockchain, and the participant nodes are verified. These mitigation techniques have benefits and limitations against one another. For instance, the traditional applications perform faster data validation but lack the full control [109] over data integrity. As the defined data validation techniques could be error-prone [106, 107]. Ethereum provides a transparent platform to define data validation rules agreed upon by other decentralized distributed nodes. All the nodes on the blockchain should follow the data validation rules [110] to validate the data before writing it into the ledger. Also, blockchain is an append-only structure, and users can only add data but can not modify or delete them [111]. Hence, this process reduces human error. But PoW is an energy-waste consensus mechanism, takes time for validation, and also pays an administration fee to the miners for performing this activity. These limitations are overcome by HLF which does not require PoW or administration fee. It uses the PBFT consensus for data validation. By the nature of HLF, it leverages the benefits of the permissionless blockchain (e.g., Ethereum) as well as provides faster, inexpensive, efficient, and privacy-oriented decentralized data validation [94]. However, permissioned blockchains require some authority to give permissions to network nodes, which contradicts the blockchain's dis-intermediation premise.

Table 20. Comparison of different solutions which mitigate data tampering

	Traditional	Ethereum	HLF
V#1	Message authentication code	Decentralized data validation by unverified nodes	Decentralized data validation by verified nodes
V#2	Audit trails	Tamper-proof immutable and distributed ledger	Tamper-proof immutable and distributed ledger
V#3	Cryptography and digital signatures	Default implementation of cryptography and digital signatures	Default implementation of cryptography and digital signatures
V#4	Authorization and access control	Smart contract-based decentralized access control	Permission settings and decentralized access control
V#5	Secure communication with protocols	P2P network, decentralized data validation and consensus	P2P network, decentralized data validation and consensus

To mitigate V#2, traditional applications optionally implement the functionality for keeping the audit trails (e.g., logs). Audit trails provide transparency and proof of records' integrity and accuracy. It also protects sensitive data from intentional misuse or harm from involving parties in the business process. The audit trails in the traditional applications are weak, centralized, and subject to attacks [112]. Also, the control remains to a designated authority and does not

provide transparent traceability and trust-able proof of audit trail integrity. In contrast, blockchain-based applications manage records in an immutable ledger, which provides tamper-proof transparent audit trails with backward traceability (e.g., provenance) [109]. In Ethereum, whenever a new transaction occurs, it appends on the ledger and replicates among nodes over a P2P network. Similarly, HLF provides the immutable ledger, and rich traceability of transactions [78].

The third vulnerability (V#3) is mitigated by incorporating cryptography and digital signatures. The traditional application integrates cryptography to save and transmit data securely. Again, it lacks control over data. Since the centralized authority is responsible for the administration of the database, and if the security is compromised, the attacker can steal, modify, or remove the data. It does not matter if the data is stored in an encrypted format or not. These attacks are common in traditional centralized applications [113]. Ethereum and HLF allow to save encrypted data on the ledger, so it becomes possible for a client node to encrypt the data before submitting it. The records are difficult to modify or delete because of the consensus mechanism and ledger redundancy among nodes over the P2P network and an append-only structure of the blockchain. As Ethereum is a permissionless blockchain and anyone can read the data from the ledger, it is possible for the attacker to trigger a deanonymization attack. In contrast, HLF overcomes this limitation by verifying nodes and permissioned settings of the ledger.

The fourth vulnerability (V#4) is mitigated by implementing authorization and access control. It is a security control [114] to check who can access the system and data. In traditional applications, authorization and access control settings could be tampered with because of centralized storage, designated authority managing the access control, and weak auditing. Ethereum-based access control settings are hard to tamper with because the nodes validate them. Also, the settings are distributed among nodes, making it impossible for the attacker to change them on all the nodes. Ethereum requires an extra effort to implement access control using smart contracts. In HLF, only pre-verified nodes can participate in the network and network operations, and it provides built-in fine-grained access control to share specific access rights among various nodes.

The last vulnerability (V#5) is related to the weak communication mechanism. In traditional applications, it is mitigated by providing secure communication with protocols that ensure transit data integrity. The implementation of communication protocols can be broken [115]. In this case, the attacker can intercept data transmission and modify the data. Ethereum overcomes this issue by leveraging the P2P network; also, all the communication on a P2P network is encrypted. In the Ethereum-based P2P network, nodes can send and receive data directly from each other and also behave both as a client and server [116] at the same time. In Ethereum, the valid transaction is usually signed before submitting but the associated data is not encrypted by default [117]. In this case, the client node encrypts the transaction data and then submits it to the network [105]. The acting server node knows that the transacting data is correct and valid because of the validation

and consensus process [109]. If the attacker tampers with the data, it will not be validated during the validation and consensus process. Similarly, HLF operates on a P2P network, provides encrypted data communication, and validates transit data.

Even though implementing the OWASP and STRIDE-based countermeasures to protect against data tampering, the traditional approaches lack full control over the data security. The attacker can get access to the database and can tamper with or delete it. The attacker can encrypt the database to trigger a ransomware attack, send the malicious code to tamper with the record, and leaves no traces because of the weak audit trails. Also, the issues related to weak authorization, centralized access control, and weak cryptography. These are only a few challenges from a long list that are counter by the traditional applications. Here come the blockchain-based applications, which record each transaction in a tamper-proof, immutable, and distributed ledger. Blockchain supports an append-only ledger and saves every transaction with a unique cryptography hash. The consensus mechanism and validator nodes validate incoming data before storing it on the ledger. The ledger provides transparency and audibility and ensures that the records on the ledger are accurate and unaltered. However, blockchain-based applications are not a silver bullet since numerous security vulnerabilities may appear, which we will examine in-depth in the next sections. This work has a few limitations. For example, the current approach has a limited number of studies that address the mitigation of data tampering by comparing it to the existing ones. In general, the blockchain looks promising from the perspective of application security, but it is still in its infancy. There are not many blockchain-based applications in production to assess the security and countermeasures on a larger scope. Overcoming these limitations can bring richer insights and enhancement in the results.

3.3. Sybil Attack

Sybil attack is well-known in the context of P2P networks [83], where the attacker can forge or create numerous fake identities to gain a considerable influence on the network. The P2P networks do not rely on a central trusted party chain of trust to verify the identity of each participant node; also relatively cheap to generate identities that treat equally on the network [118, 84]. Algorithm 1 explains the procedure of a Sybil attack that undermines a P2P network [83]. The Sybil attack combines the honest *(H)*, Sybil *(S)*, and attacker *(A)* nodes. To initiate the attack, the attacker creates numerous Sybil nodes and connects with the honest nodes that disconnect genuine connections of honest nodes with other honest nodes on the P2P network. The attacker takes control over the P2P network when he gains a disproportionately large influence on the network *(Δ)*.

Eventually, the attacker uses Sybil nodes to trigger various threats that subvert the reputation system of a P2P network. For example, the attacker uses Sybil identities to pollute the BitTorrent distributed hash table (DHT) routing to trig-

Algorithm 1: Sybil attack

```
H ← Honest nodes;
S ← Sybil nodes;
A ← Attacker node;
while (true) do
    A Creates S;
    A Connects S with H;
    A Gains fraction of the system ← Δ;
    if (Δ == true) then
        A Uses attack method;
        A Triggers threat;
        A Damages reputation system;
    end
end
```

ger delays, connection slot consumption, invalid file contents downloads, and bandwidth exhaustion [119]. Similarly, the attacker can use the clone node attack to target static wireless sensor networks (WSNs) [120] by taking control of *node n* and replicating it throughout the network. On a successful clone node attack, the attacker can initiate a Sybil attack, selective forwarding attacks, incorrect data injection, protocol interruptions, and traffic jams [120]. Sybil attacks are hard to prevent [121]. However, there exist preventive measures to increase protection against Sybil attacks. The authors [119] present a reputation-based scheme 'GOLF' to identify Sybil nodes on BitTorrent DHT based on patterns in IP addresses. Numan et al. [120] assemble the cloned node detection schemes along with their drawbacks and challenges. Furthermore, the study [121] categorized three different mechanisms to defend P2P systems against Sybil attacks: *(i)* Trusted certification (e.g., centralized or distributed certification using cryptographic primitives), *(ii)* resources testing (e.g., IP testing, network coordinates, requiring clients to solve puzzles), and *(iii)* social network techniques (e.g., SybilGuard, SybilLimit, SybilInfer, vote aggregation, and GateKeeper).

Blockchain systems run over the P2P network; therefore, it is possible to run multiple fake nodes [122]. Additionally, blockchain systems include valuable digital assets that motivate the attackers to execute this attack. Once fake identities gain recognition in the blockchain system, the attacker interrupts the flow of information, out-votes (or blocks) the honest nodes, and refuses to receive or transmit information [29, 84]. The threat spectrum is increasing on blockchain systems, and the attacker has different motives to carry out this attack. This section discusses Sybil attack-based threats (Table 22) in detail.

3.3.1. Break Consensus Protocol

Shard-based blockchain systems (e.g., Elastico [123]) process transactions in parallel to overcome transaction throughput and network scalability limitations of PoW. These systems are vulnerable to Sybil-based break consensus protocol (BCP) attack [124] when the attacker uses the Sybil nodes to disrupt the shard-based consensus process. Shard-based consensus protocol uses PoW to create verifiable node IDs to participate in the consensus process. A valid node in the network could act as the attacker and generate Sybil IDs in target shard(s). The Sybil IDs enable the attacker to break the intra-shard consensus protocol and prevent the insertion of transactions in a blockchain. When using PoW to generate verifiable node IDs, the existing shard-based protocol assumes that the network nodes have uniform computing power. In contrast, the computing power of network nodes does not hold uniformity in PoW. Consequently, the attacker exploits this vulnerability in a shard-based consensus protocol. The successful BCP attack impacts the IDs generation process of nodes, preventing the insertion of transactions in a blockchain, and harms the business assets. For example, the BCP attack negates the integrity of ID generation, consensus process, and transaction availability.

The authors [124] perform numerical analysis to identify the success probability of a BCP attack (P_{BCP}) by using different settings of computing power, the number of shards, and network nodes (Table 21). For example, if the attacker gains 25% computing power, the system has at most 16 shards and at least 600 nodes, then P_{BCP} is $\leq 10^{-4}$. Proportionally, more computing power increases the P_{BCP}, in this manner, when computing power is between 33%-53% the P_{BCP} is ≥ 0.8 and P_{BCP} becomes 1 when computing-power in $\geq 56\%$.

Table 21. Numerical analysis of BCP attack [124]

Computing-power	# of shards	# of nodes	P_{BCP}
$\leq [25\%]$	at most 16	at least 600	$\leq 10^{-4}$
$[33\% - 53\%]$	at most 16	at least 600	≥ 0.8
$\geq [56\%]$	at most 16	at least 600	1

The simulation to prevent BCP attack depends on different settings of shards, nodes, and their computing power in the blockchain system [124]. For example, monitor the computing power of nodes, and if some node starts gaining computing power, then either restrict his computing power or proportionally increase the number of shards and nodes to prevent a BCP attack.

3.3.2. Generate Fake Transaction

Similar to the BCP attack, the Sybil-based generate fake transaction (GFT) attack is possible on shard-based blockchain systems [124]. This attack aims to use the Sybil nodes and create fake (or invalid) transactions in blocks to invalidate the state of the ledger or simply corrupt the transactions. Unlike the BCP attack, in the GFT attack, the attacker uses Sybil nodes to control and manipulate the shard-based consensus process to reach a final consensus on fake transactions to

Table 22. security risk analysis of Sybil attack

Blockchain characteristic		Risk-related concept		Asset-related concept		Risk treatment concept
Type	**Consensus**	**Threat**	**Vulnerability**	**System asset**	**Business asset**	**Countermeasure**
Shard-based permissionless	PoW, PBFT, Intra-committee consensus	Break consensus protocol [124]	Computing-power of nodes do not hold uniformity in PoW	Nodes, P2P Network, Intra-committee consensus protocol	Consensus process (I), ID generation (I), Transaction (A)	Monitor computing-power [124]
						Number of shards [124]
						Increase network nodes [124]
Shard-based permissionless	PoW, PBFT, Intra-committee consensus	Generate fake transaction [124]	Assumption of only verifiable nodes participate in consensus	Nodes, P2P Network, Consensus process	Transaction (I), Ledger (I)	Monitor computing-power [124]
						Number of shards [124]
						Increase network nodes [124]
Customized permissionless	Trustworthiness of nodes	Tampering nodes reputation [118]	Calculating nodes reputation with Sybil nodes	Nodes, P2P Network, Subjective work graph	Nodes reputation (I)	Accounting-based algorithm (NetFlow) [118]
Permissionless	PoW	Nodes isolation attack [125, 18, 122, 126, 84, 127]	Lack of computing power	Nodes, P2P Network, Computing power	Network reputation (I)	Increase computing power [18]
						Monitor computing-power [18]
			BGP does not validate routing origin	Nodes, P2P Network, Consensus	Transaction verification (I)	Monitor round-trip time [126]
			No proper authentication of nodes	Nodes, P2P Network, Network reputation, Transaction	Transaction verification (I)	Network joining fee [84]
						Validating node connection [84]
						Monitoring nodes behavior [84]
Customized permissionless	Respect Score	Routing table insertion [122, 128]	Sybil nodes can update the routing table information	Nodes, P2P network, Nodes Identifier, Routing table	Message route (I, A), Routing table (I)	Registration of nodes [122]
						Monitor activities of node [122]
						Reward of respect score [122]
						Route prediction model [128]
Permissionless	PoW, PoS, DPoS	Sybil-based linking [88]	Pairing with the same users multiple times during mixing	Mixer, Mixing process, Users, Transactions	User Identity (C), Transaction (C)	Non-refundable deposit to create node identity [88]
						Time locking on funds [88]
						Coin-age [88]
Permissionless	PoW, PoS, DPoS	Sybil-based DoS [129, 88, 130, 131, 132]	Sybil nodes refuse pairing with honest nodes	Mixer, Mixing protocol, Users,	Mixer functioning (A)	Non-refundable deposit to create node identity [88]
			Sybil nodes can participate in mining	Nodes, P2P network, Mining protocol	Mining process (A), Mining pool (A)	Use computational constraint-based techniques [88]
			Dusting transactions	Transactions, P2P network	Services (A)	Anti-dust model [132]

add them to the block. The assumption of only verifiable nodes participating in the consensus process allows the attacker to exploit the shard-based consensus process. The successful GFT attack harms the integrity of transaction and ledger.

Table 23. Numerical analysis of GFT attack [124]

Computing-power	# of shards	# of nodes	P_{GFT}
$\leq [25\%]$	at most 16	at least 600	0
$[33\% - 53\%]$	at most 16	at least 600	$\backsimeq 0.005$
$[56\%]$	at most 16	at least 600	$\backsimeq 0.001$
$\geq [66\%]$	at most 16	at least 600	$\geqslant 0.75$

To summarize, the numerical analysis in [124] identifies the success probability of the GFT attack (P_{GFT}). The results (Table 23) show that the attacker requires relatively higher computing power to initiate a GFT attack. Similar to the BCP attack, to prevent a GFT attack, monitor the computing power of nodes and use different settings of shards and nodes depending on the computing power.

3.3.3. Tampering Nodes Reputation

TrustChain [118] is a blockchain-based tamper-proof and scalable data structure designed to create reputation-based distributed trust. TrustChain includes one transaction per block and together form a directed acyclic graph (Fig. 15) where each transaction block has two incoming and two outgoing pointers. The system detects the violation of this rule and considers it fraud that lowers the reputation of the node. Meanwhile, other nodes involve and reach a consensus on the transaction. Furthermore, the system uses a subjective work graph to model network node interactions. The system stores and shares blocks with other network nodes only once it verifies incoming and outgoing pointers, sequence numbers, transaction data, and signatures. In TrustChain, the attacker uses the subjective work graph model to increase his reputation using Sybil nodes or lowers the reputation of honest nodes by initiating the interactions with them. The subjective work graph models the partial view of the interactions of the nodes in the network. The TrustChain assumes that all participant nodes contribute to other nodes in the network, and each successful contribution earns them some reputation.

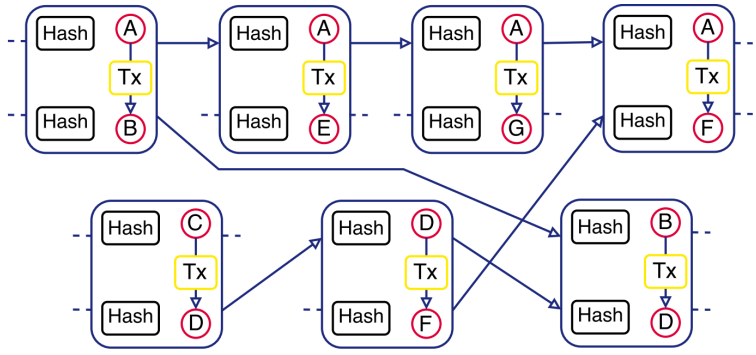

Figure 15. TrustChain data structure to record transactions, adapted from [118]

The attacker creates the Sybil nodes to boost his reputation and damages the reputation of honest nodes (Fig. 16). In Fig. 16, the honest nodes *(A, B, and C)* contributing $n-units$ work to each other over a P2P network. For instance, node A contributes 4 units work to node B ($A \xrightarrow{4} B$) and receives 9 units work contributions ($A \xleftarrow{9} B$) from node B, node B contributes 8 units work to node C ($B \xrightarrow{8} C$) and receives 3 units work contributions ($B \xleftarrow{3} C$), and node C contributes 5 units work to node A ($C \xrightarrow{5} A$) and receives 3 units work contributions ($C \xleftarrow{3} A$). These interactions are network-wide, and nodes do not know about all such interactions. Therefore, node C has also contributed a total of 2 units work to the attacker node D ($C \xrightarrow{2} D$). The attacker node creates the three Sybil nodes *(SD1, SD2, and SD3)* and performs work for them e.g., contributing 5 units work to each Sybil node ($D \xrightarrow{5} SD1, D \xrightarrow{5} SD2, D \xrightarrow{5} SD3$) to boost his contributions to get more reputation. The nodes that contribute the most are rewarded with a higher reputation score.

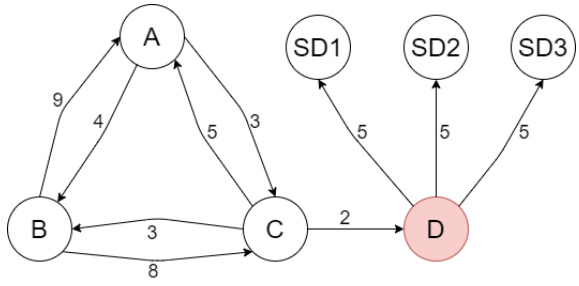

Figure 16. Attacker node (D) tampering nodes reputation

The system calculates the reputation score of the node combining the Sybil nodes, and once the attacker receives more work from honest nodes, the attack profitability increases. The authors [118] determine the profitability of tampering nodes reputation attack by calculating the supremum (Eq. 3.1). The calculation of supremum accumulates the attacker and his Sybil nodes contributions.

$$Sup = \left[\frac{\sum(X^n)}{\sum(Y^n)} \ : \ n \ \varepsilon \ \mathbb{N}, \ X \neq 0 \right] \tag{3.1}$$

In Eq. (3.1), X^n presents a sum of work the attacker node D or his Sybil nodes performed after some $n - number$ of activities, and Y^n is a sum of work that obtains the attacker node D and his Sybil nodes from the network. The results in [118] show that the Sybil attack is strongly beneficial by tampering with nodes' reputation if supremum is infinite $(sup = \infty)$. If the supremum is finite but larger than 1 $(sup \neq \infty \ \& \ sup > 1)$ then weakly beneficial otherwise unprofitable.

TrustChain proposed an accounting mechanism-based Sybil-resistant algorithm called NetFlow [118] to limit the tampering of nodes' reputations. NetFlow does not mitigate the Sybil attack, but it lowers benefits by determining and employing node trustworthiness during the consensus process. NetFlow assumes that nodes who consume resources also contribute back to the network during the transacting process. NetFlow uses a subjective work graph (Fig. 16) along with a choice set of those nodes interested in receiving some work, so the nodes get a reputation score. The Sybil nodes do not contribute to the network, and their reputation decreases over time. As a result, the system does not assign more work to node D.

3.3.4. Node Isolation Attack

In a node isolation attack, the attacker hijacks the honest nodes and splits the network into two or more disjointed groups [125, 127]. In Fig. 17, the attacker first identifies the victim nodes, then replaces the victim nodes' peers with Sybil nodes. Next, the attacker isolates the victim nodes and disconnects their initiated transactions. The attacker nodes are now involving victim nodes in their governed

blocks. In this case, the attacker gains proportional control over the system and performs various operations, such as halting transaction and block propagation, validating fake or double-spend transactions, and obtaining mining incentives.

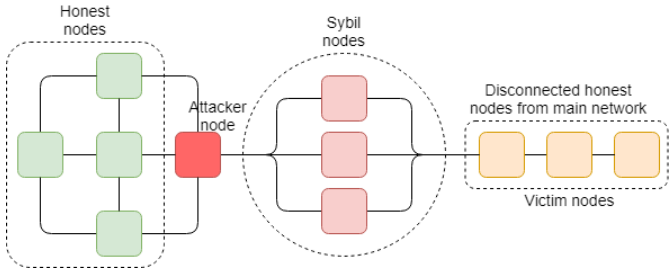

Figure 17. Node isolation attack on honest nodes

If the blockchain network has insufficient computing power, the attacker exploits this limitation [18] by using Sybil nodes [122]. For example, the blockchain network with a few participant nodes holding insufficient computing power, the attacker comes with higher computing power and Sybil nodes to connect with honest nodes. Using Sybil nodes, the attacker isolates the honest nodes and obstructs their communication with the main blockchain network. Once the attacker gains control of a fraction of the network, the attacker involves honest nodes in his mining process to gain more mining incentives or affect the transaction verification. The border gateway protocol (BGP) is a routing protocol, and it does not validate routing origin [126]. The attacker exploits this vulnerability by using the Sybil nodes to falsely announce some valid IP prefixes [126]. Once the honest nodes connect with Sybil nodes, the attacker can intercept the traffic and delay block propagation to affect the transaction verification. Moreover, the poor implementation of node authentication when joining the blockchain network [84]. For instance, no network joining fee, no validating IP address, or source of node connection. The attacker sees these limitations as an opportunity and builds as many Sybil nodes as possible.

There are currently no measures to mitigate the Sybil-based node isolation attack entirely, but there are some preventive measures to control this attack. For example, increase the computing power in the blockchain network according to the available nodes in the network and also monitor the nodes' computing power [18]. Monitor the round-trip time to detect irregular patterns [126] during nodes data exchange and accepting transmission. Once the node identifies the irregularities, it may disconnect itself and try to connect to other nodes on the network randomly. In addition, establish a node authentication process before joining the network [84]. For instance, use network joining fee, validate the source of node connection and monitor nodes' behavior over a certain period of time.

3.3.5. Routing Table Insertion

In blockchain systems, nodes keep their neighbor nodes' information in the routing table (RT) and periodically update the information in RT [122]. The entry in RT is a tuple containing nodeId, network connection IP, and port. Here, the routing table insertion attack [122, 128] refers to the insertion of Sybil nodes in the RT. The attacker uses Sybil nodes to isolate honest nodes from the network and force them to insert or update the compromised node(s) in the RT. On success, the Sybil nodes will become capable of breaking the routing system, modifying or diverting message routes, blocking access to the transaction, affecting the voting outcome, and interrupting network services.

To overcome the RTI attack, the authors [122] implement a decentralized registration process to create a new node. For example, the system identifies the new node Id request and places it under the miner. The miner continuously monitors the activities of the new node and records them on the blockchain. In this process, the contributing nodes get an incentive as a respect score that enables them to access different services in the system. Use the route prediction model [128] that is based on a freenet routing algorithm.

3.3.6. Sybil-based Linking (Deanonymization)

Even though blockchain systems are pseudo-anonymous, it is possible to link transactions and trace a user or company behind each transaction [88]. For example, in the dusting attack [133], the attacker sends a small amount of cryptocurrency transactions to a large number of addresses. The attacker conducts a combined analysis of dusted addresses to identify the respective user when the user transfers the dusted cryptocurrency. Blockchain platforms started using mixers as a service to enhance the privacy and anonymity of transactions by obfuscating the transaction flow [89]. For example, the user sends the transaction to the mixer that mixes with other addresses or breaks down the transaction into smaller transactions. The mixing process muddles the original transaction connection. This mechanism was working sufficiently until *Sybil-based linking*. Mixers operate over blockchain-based P2P systems; hence, they are susceptible to the Sybil attack. The attacker creates multiple funded addresses and joins a mixer [88]. In a mixer, the attacker creates a ring (e.g., CryptoNote mixer [134]) of his addresses and pairs them with the same users multiple times. The pairing with the same users helps the attacker build the profile that enables him to utilize paired user transactions to de-anonymize and originate the source, thus negating the confidentiality of the user identity and transaction. The attacker uses the transaction linking information to execute phishing or cyber-extortion threats.

A Sybil-based linking attack could not be mitigated completely, but there are some techniques to restrict it [88]. For example, a non-refundable deposit to create an identity makes it expensive for the attacker to create several identities in the blockchain. Time locking on the usage of funds will require a substantial amount

of funds to freeze for creating multiple identities from the attacker. The coin-age mechanism restricts the use of coins under a certain coinage. This technique takes a long time for the attacker to acquire enough aged coins to use in a mixer. If the attacker wants to participate in a mixer with lower coin-age, the system will alert broadcast to honest nodes to not join with him. The coin-age is similar to the staking mechanism in PoS consensus to show the possession of aged coins to participate in the mixing process.

3.3.7. Sybil-based DoS Attack

Despite being operating on a P2P network, blockchain is still vulnerable to DoS (or DDoS) attacks [130, 131]. Sybil-based DoS aims to disrupt the functioning of blockchain systems or worsen the user experience by delaying the transaction or block time. The Sybil-based DoS attack can target the mixing protocol [88], and blockchain network [129]. In the mixing protocol, the attacker uses Sybil nodes to participate in the mixers where participants exchange funds to mix them [88]. The Sybil nodes refuse to pair up with honest nodes in the pairing process. The honest nodes keep waiting to pair up with Sybil nodes, and the waiting time increases the transaction mixing time that disrupts the mixer functioning.

The attack on the blockchain network results in delaying the transaction or block time, exhausting the network resources, and disrupting the message transmission and consensus process [129]. First, the attacker and Sybil nodes create multiple wallets on the blockchain to trigger this attack. Second, the Sybil node attacker issues numerous dust transactions (e.g., 0.0000001 ETH in each transaction) between his Sybil nodes. The blockchains, by design, process a limited number of transactions per block in a given time. Also, the Sybil nodes participating in the consensus process do not share their verified transactions or blocks. Therefore, the large number of transactions with small value congest the blockchain network, deny services to legitimate users, and halt the mining process. Furthermore, the Sybil-based DoS affects the mining pool performance [135] by slowing down the mining task that would discourage future users from joining the victim pool, and current users might leave the pool [136].

The Sybil-based DoS attacks cannot be mitigated entirely but possible to restrict them. For example, use a non-refundable deposit to create an identity that makes it expensive for the attacker to create several identities [88]. Incorporate computational constraint-based Sybil resistance techniques like Bitcoin uses PoW [88]. However, the computational constraint-based techniques would be infeasible for low computing-power blockchain systems [137]. Moreover, utilize the anti-dust model [132] to identify and prevent dust attacks. The authors add the dust transaction pool with a certain pool capacity. The system analyzes the transaction structure based on various defined parameters (e.g., low transaction volume and fees) and adds it in the dust transaction pool; if there is a dust transaction, later the dust transaction may be discarded [132].

3.4. Double-spending

Double-spending is a data consistency attack, and it happens when spending the same digital money (or digital asset) twice. In general, Double-spending is a technique that can deceive someone about the state of a transaction. For instance, the attacker changes the transaction state and spends the same transaction twice [138] to gain monetary benefits [85]. The risk of Double-spending negates the integrity of the ledger. Several threats exist that can cause Double-spending, for example, Sybil-based Double-spending, 51% attack, etc. Algorithm 2 illustrates the procedure of Double-spending in the case of a 51% attack. The attack constitutes the honest nodes *(H)*, honest chain *(HC)*, attacker node *(A)*, attacker chain *(AC)*, and the attacker computing-power *(ACP)*. To initiate the attack, the attacker forks a private chain from an honest chain. Next, he gains 51% or more computing power in blockchain and creates two conflicting transactions $(TX1, TX2)$. The attacker sends one transaction in an honest chain $(TX1 \rightarrow HC)$ paying a merchant for some goods and another double-spend transaction to himself in his chain $(TX2 \rightarrow AC)$. The attacker starts mining blocks on his chain and generates blocks quicker than the honest nodes using his computing power. Once the attacker chain length is greater than the honest chain length $(AC > HC)$, the attacker chain becomes valid and honest nodes adapt it based on the longest chain rule. Thus, it makes the double-spend transactions valid, and the attacker receives his spent funds back himself. The 51% attack is mostly discussed as a cause of Double-spending. In contrast, several other approaches can trigger Double-spending (Table 24). Here, we discuss those in detail.

3.4.1. Sybil-based Double-spending

In Sybil-based Double-spending [29], the attacker uses Sybil nodes to influence communication/gossip protocol and exploits the threshold of waiting time. The attacker leverages the block propagation delay, and with 32% of computing power, he makes his fraudulent chain valid. First, the attacker node creates several Sybil nodes and initiates a transaction A_{T0} that disseminates to other nodes on the network. Honest nodes put A_{T0} in the memory pool after verifying the transaction. Second, the attacker establishes a private chain and initiates another double-spend transaction A_{T1}, before A_{T0} adds to the block. In this stage, the attacker keeps mining his private chain and uses the Sybil nodes to delay the block propagation time on the valid chain. The Sybil nodes freeze the block propagation process by not sending new block information to honest nodes. Once the waiting time threshold exceeds, the honest nodes stop waiting and start the next round of block mining. The attacker keeps aiming to catch up with the longest chain since the growth rate of the valid chain is delayed by Sybil nodes. If the attacker succeeds in making his private chain longer, the network nodes accept the attacker's longer chain according to the longest chain rule. Now, the attacker node in a valid chain, therefore, double-spends transaction A_{T1} becomes valid, and the attacker gets his

Algorithm 2: Double-spending by 51% attack

```
H ← Honest nodes;
HC ← Honest chain;
A ← Attacker node;
AC ← Attacker chain;
ACP ← Attacker computing-power;
while (true) do
    A → Forks private chain AC from HC;
    if (ACP ≥ 51%) then
        if (A → Creates conflicting transactions) then
            TX1 → HC;
            TX2 → AC;
            A → Starts mining AC ;
            if (AC.length > HC.length) then
                AC becomes Valid;
                H adopt AC;
                A gets spent funds back;
            end
        end
    end
end
```

spent coins back. This attack scenario is different from the 51% attack because the attacker uses Sybil nodes to delay the block propagation time. The delay supports the attacker to mine his private chain quickly by requiring only 32% of computing power. This attack can also cause blockchain forks and waste the computing power of honest nodes.

There exist several propositions to overcome this attack. For example, restrict miners not to mine consecutive blocks [139]. If the miner has mined a block already, he will not execute the mining process until he receives at least one block from other miners. This change could decline the total computing power in the blockchain network. Increase the number of confirmed blocks [29] then the attacker requires more computing resources to make the fraudulent chain longer. Furthermore, use a fee to create node identity [88] to restrict the attacker from creating multiple Sybil nodes.

3.4.2. 51% Attack

51% attack is a hash rate-based attack, and the most vicious [140, 141] where the attacker uses 51% or more computing power in the network to successfully execute Double-spending. For example, Bob is a malicious node and forks the ledger to create his private chain. Bob produces blocks in his private chain and does not broadcast them to the blockchain. Now, there are two versions of the

Table 24. Security risk analysis of Double-spending

Blockchain characteristic		Risk-related concept		Asset-related concept		Risk treatment concept
Type	**Consensus**	**Threat**	**Vulnerability**	**System asset**	**Business asset**	**Countermeasure**
Permissionless	PoW	Sybil-based Double-spending [29]	Increasing threshold of waiting time	Communication / gossip protocol, Nodes identities	Block propagation (A), Digital asset (A)	Restrict to mine consecutive blocks [139]
						Increase confirmed blocks [29]
						Fee to create node identity [88]
Permissionless	PoW	51% attack [86, 140, 34, 35, 141, 142, 37]	Nodes can hold a large portion of computing-power	Nodes, Ledger, P2P Network, Computing-power, Mining	Network reputation (I), Ledger (I)	Power monitoring tool [87]
						Increase confirmed blocks [140]
						Limit whale transactions [142]
						Transaction fee [86]
			Inappropriate consensus mechanism	Nodes, Ledger, P2P Network, Consensus	Consensus (I), Transaction (I)	Pluggable consensus [35, 37]
						Delayed proof of work [141]
Permissionless	PoS	Long-range attack [143]	Forging timestamps	Nodes, P2P Network, Ledger	Timestamp (I), Blocks (I)	Longest chain rule [143]
						Context-aware transactions [143]
						Economic finality [143]
			Obtains the keys of past validators	Nodes, Ledger, P2P Network, Private keys, Past validators	Blocks (I), Transaction (I), Private keys (I)	Key-evolving cryptography [143]
						Moving checkpoints [143]
						Trusted execution environment [143]
						Economic finality [143]
			Stake bleeding	Nodes, Ledger, P2P Network, Stakes, Transaction fees	Blocks (I), Transaction (I)	Longest chain rule [143]
						Plenitude rule [143]
						Economic finality [143]
Permissionless	PoW	Time advantage [144]	Secretly mining fraudulent blocks with time advantage	Miners, Mining process, Computing power, Mining time	Mining process (A), Ledger (I), Digital asset (A)	Limit time advantage [144]
						Monitor time differences [144]
						Increase confirmed blocks [144]
Permissionless	PoW	Eclipse-based Double-spending [145, 146]	Accepting unconfirmed transaction	Transaction verification, Block confirmations, Ledger	Fast payment (I, A), Digital asset (I)	Increase confirmed blocks [147]
			Node connects directly to incoming connections	Nodes, IP addresses, Node connection, Transaction	Communicating/ gossiping (A), Transaction verification (I)	Disable incoming connections [145]
						White-listed nodes [145]
						Random outgoing connections [148]
						Deterministic random eviction [145]
						Feeler connections [145]
						Anchor connections [145]
Permissionless	PoW	Border gateway protocol hijacking [126, 149]	Possible to hijack IP prefixes by corrupting routing table	Border gateway protocol, IP prefixes, Network traffic, ISP, Routing table	Block propagation (A), Routing table (I), Transaction (I)	Increase nodes connection diversity [126]
						Multi-homing of mining pool [126]
						Random outgoing connections [126]
						Monitor round-trip time [126]
						Anomaly detection mechanism [126]
						Hosting several gateway [126]
						Randomly request a block [126]
						Increase confirmed blocks [149]
						Selectively choosing peers [149]
						Encrypt bitcoin communication [126]
						Use randomized TCP port [126]
						Periodically send UDP messages [126]
Permissionless	PoW	0-confirmations race attack [125, 85, 138, 150, 151, 152, 153, 154]	Accepting unconfirmed transaction	Transaction, Transaction verification, Block confirmations, Ledger	Fast payment (I, A), Digital asset (I)	Increase confirmed blocks [138, 150]
						Closed-form formula probability [151]
						Enhance network policy [155, 152]
						Listening period [85]
						Insert observer [85, 153]
						Alerting honest nodes [85]
						Forward double-spend transaction [154]
						E-cash protocol [156]
						Enhanced observers [157]
Permissionless	PoW	Finney attack [155, 125, 145, 153, 158, 159]	Accepting 1-confirmation transaction	Nodes, Transaction, Transaction verification, Block confirmations, Ledger	Fast payment (I, A), Digital asset (I)	Increase confirmed blocks[125, 159]
						Logarithmic waiting time [158]
						Gambler's ruin problem [155, 160]
						Listening period [85]
						Insert observer [85, 153]
						Alerting honest nodes [85]
Permissionless	PoW	Vector76 attack [125, 145, 150, 161, 153]	Accepting 1-confirmation transaction	Nodes, Transaction, Transaction verification, Block confirmations, Ledger	Fast payment (I, A), Digital asset (I)	Increase confirmed blocks [125, 150]
						Listening period [85]
						Insert observer [85, 153]
						Alerting honest nodes [85]
			Possible to have multiple full nodes	Block confirmation, Exchanges, E-wallet, Static IP address	Digital asset (I)	Disable incoming connections [161]
						Well-connected inbound connection [161]
						Monitor outgoing node connection [161]

ledger (Fig. 18), one managed by Bob and another by the honest nodes. Bob sends 50 BTC to Alice (merchant) for some product. Bob spends his bitcoins in the honest nodes chain but did not add in his private chain.

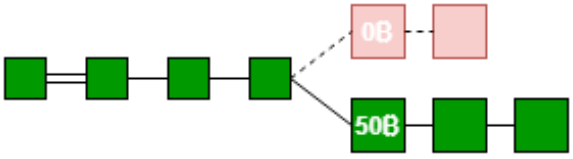

Figure 18. Honest nodes chain in green and fraudulent one in red

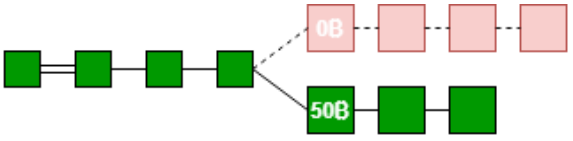

Figure 19. Attacker node succeeds to make the fraudulent chain longer

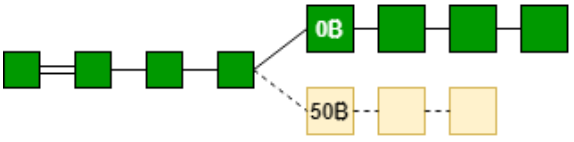

Figure 20. Attacker node chain is published and now it is valid one

The network nodes always adopt the longer chain. If Bob mines the blocks faster and builds a longer chain by gaining more computing power (e.g., 51% or more), then the honest nodes will adopt the Bob chain that contains a reverse transaction (Fig. 19). In this case, Bob will get his 50 BTC back, but Alice will not retain the 50 BTC sent by Bob (Fig. 20). Hence, the attacker can successfully trigger the 51% attack by accumulating a large portion of computing power. The attack weakens the P2P network and negates the integrity of the ledger. Also, the 51% attack can happen when the blockchains use an inappropriate consensus [34, 37]. For example, suppose the blockchain systems that contain a few participant nodes and possess insufficient computing power [18, 19] are using the PoW consensus. In that case, the attacker comes with higher computing power and sabotages the digital assets, the integrity of consensus, and transactions.

The practicality of a 51% attack is seemingly impossible on the large-sized blockchains having high computing power (e.g., Bitcoin and Ethereum) but possible on smaller blockchain systems holding an insufficient amount of computing power [18]. Blockchains that have suffered from 51% attack include Ethereum Classic (ETC), Feathercoin (FTC), Bitcoin Gold (BTG), Vertcoin (VTC), and Verge (XVG) [18, 19]. At the time of the attack, all of these held a limited number of nodes, insufficient computing power, and used PoW consensus. Double-spending is the one result of a 51% attack. The attacker can achieve selfish mining, prevent new transactions from gaining confirmations, and blockchain forks.

To protect against 51% attack, implement a power monitoring tool to monitor the computing power of nodes continuously and restrict when reaching a certain amount of computing power [87]. M. Rosenfeld [140] states that increasing the number of confirmed blocks (it means waiting for more confirmations before accepting the transaction e.g., 6 confirmations are required in Bitcoin) would decrease the success probability of hash rate-based attack. The PoW-based blockchains need a mechanism to limit the number of whale transactions, reduce the size of transactions fee, and prevent honest miners from colluding to mine a whale block for more profitable mining [142]. Incorporate transaction fee [86] as an incentive to keep nodes honest in a blockchain system. Use a pluggable consensus mechanism [35, 37] to facilitate consensus diversity based on the business models and requirements of the blockchain system. Komodo presents delayed proof of work (DPoW) [141] for unspent transaction output-based blockchain systems (e.g., Bitcoin) to add an impenetrable security layer for mitigating 51% attack. DPoW combines the notary node network and recycles the hash rate. The notary node network stores backups of individual blocks on multiple blockchain networks (called notarized blocks). Once notarization is complete, the history of every chain using dPoW becomes immutable.

3.4.3. PoS Long-range Attack

The PoS long-range attack [143] is similar to 51% attack. Currently, blockchain systems (e.g., Ethereum) attempt to switch over PoS to overcome PoW shortcomings [143]. In a PoS long-range attack, the attacker creates a private chain starting from the genesis block and overtaking the main chain. The attacker forges the blocks and adds double-spend transactions in his private chain. The attack succeeds once the attacker's private chain becomes longer than the main chain. There are three types of PoS long-range attacks: simple long-range, posterior corruption, and stake bleeding.

In a simple long-range attack, the attacker forges timestamps to produce blocks ahead of time to advance his private chain and overtake the main chain (Fig. 21). In the implementation of the PoS protocol, where the nodes do not verify the block timestamps, the attacker can exploit the block timestamps according to his will. As a result, both branches become valid. Once the attacker branch is ahead of the main branch, other nodes will accept the longer chain.

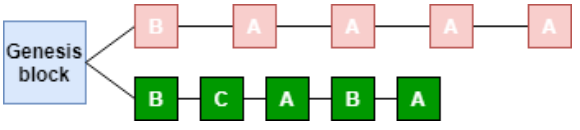

Figure 21. To compute blocks ahead of time and try to overtake the main chain, the attacker forges the timestamps.

In posterior corruption, the attacker uses private keys of such nodes (e.g., Node B) that leave the validating process and cash out the stakes. For example, *NodeB*

retires and removes his stakes after validating the first $n - blocks$ in the main chain. Now, *NodeB* is not a part of the block validation process and cannot sign new blocks. However, *NodeB* can still sign the first $n - blocks$ of any branch of that blockchain [143] using his private key. There are two possible ways the attacker can access the *NodeB* private key. Firstly, after leaving the validation process, the security of his private keys is not a priority, and the attacker somehow gets access to it. Secondly, the attacker can convince *NodeB* to launch the long-range attack. *NodeB* has already left the validation process, and nothing is at stake. Hence, there are likely high chances of performing this attack by teaming with the attacker. In this case, the attacker can sign valid blocks and increase his chances of generating blocks quicker than other honest nodes (Fig. 22).

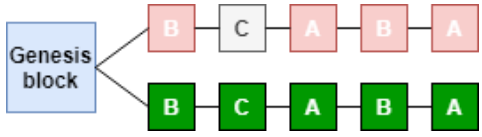

Figure 22. To make the private chain (upper chain) more competitive, Node B joins the attacker and attempts to conquer the main chain.

In stake bleeding (Fig. 23), the attacker starts stalling the main chain and gets more time to create blocks in his private chain [143]. For instance, when the attacker becomes a slot leader, he skips the chance to stretch the main chain, and no new block will be added to this slot. The attacker earns no incentives, and his stakes will decrease (or bleed) in the main chain. However, he keeps publishing blocks in his private chain and raises his stakes through transaction fees. The attacker's private chain grows faster than the main chain from this point on and gradually becomes longer. The attacker adds one transaction to redistribute the stakes to other validators before the forged private chain is released, which would not violate the "honest majority" presumption. Therefore the attacker's private chain will become valid.

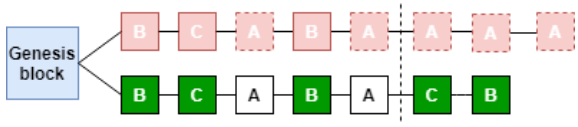

Figure 23. Attacker stakes in his private chain (upper chain) steadily increase and are more often chosen as a block validator while he tends to lose stake in the main chain and is thus less frequently chosen as a block validator in the main chain.

Various techniques are available to overcome the PoS long-range attack [143]. Individually, these techniques can not completely mitigate the attack but can restrict it to a certain level by combining these solutions [143]. In PoS, due to weak subjectivity, when the nodes come online, they can not determine which node is longer/valid because of having many different branches on the blockchain. Therefore, the nodes may be tricked into accepting the attacker branch and participating in his block validation process. Like PoW, the PoS longest chain rule can over-

come weak subjectivity issues, and the newly added nodes will always join the longer chain. In context-aware transactions, universal hash time is utilized to establish transaction references bound to specific points in time. This technique restricts the attacker from copying valid transactions from the main chain to his private chain. If the attacker copies the valid transactions from the main chain, honest nodes will reject, considering inconsistency in the attacker's private chain.

The economic finality approach ensures that the validators participating in the validation process have something to lose once they misbehave. This technique does not eradicate the attack but, by financial penalty, discourages the attacker. The key-evolving cryptography technique makes used keys immediately useless. The technique forcefully creates and assigns a new private key to the validator for participating in the next block validation round. For example, in posterior corruption, if Node B wants to sign old blocks, then he will be unable to use the already used private key. Moving the checkpoints technique enables a quota that allows only $n - number$ of blocks to be reorganized. The quota changes according to the protocol settings and configurations. For example, in Peercoin [37] the quota is limited to one month of blocks, and the NXT coin [38] quota is just for a few days or hours. Use trusted execution environments (TEE), for example, Intel software guard extensions (SGX). The SGX allows the user to perform signing within a trusted domain, and signing private keys are not available beyond the TEE. The plenitude rule aims to calculate a branch's block density from the moment the branch is created. Assuming that a minority is always the attacker, the main branch will always be denser than the opposing branches. It is reasonably easy to define the main chain and defeat weak subjectivity by following this law. However, if more than 34% of the validators are malicious, the system fails.

3.4.4. Time Advantage

This attack scenario is similar to a 51% attack; the only difference is that the attacker uses time advantage [144] over honest miners along with hash rate to produce fraudulent blocks and perform Double-spending. Time denotes the average time in seconds required for the whole network (e.g., including both honest and attacker nodes) to mine a block. The time advantage assumes that the attacker has been mining fraudulent blocks secretly with a time advantage of $n - seconds$ over honest miner nodes. The attacker interrupts the mining process, affects the block propagation, and steals the digital assets upon a successful attack.

The attack scenario of time advantage is an extension of the hash rate-based Double-spending attack model by S. Nakamoto [162] and M. Rosenfeld [140]. The extended attack version uses two different approaches: the time-based generalized model and the time-based ledger state model. The time-based generalized model adds a time parameter to the mining process, enabling the attacker to secretly mine blocks with enough time advantage to achieve a Double-spending. In contrast, the time-based ledger state model considers the times at which the

honest and attacker nodes last mined a block. This model represents the states in terms of the length of the chains where the attacker and the honest nodes are mining, and they can be different. The comparison shows [144] that the probability of carrying Double-spending with time advantage is coinciding. The authors [144] calculate the probability of a Double-spending using time advantage, number of confirmations, and computing power. Along with time advantage, if the number of confirmations decreases or the attacker's computing power increases, the Double-spending is doable. For example, if the attacker gains control of 40% of the network's computational power, the Double-spending is almost impossible to contain. This situation is very unlikely in high computing-power blockchains (e.g., Bitcoin and Ethereum) but possible in low computing-power blockchains.

To mitigate this attack, monitor time differences after the new block is produced [144]. Additionally, implement a mechanism to limit the time advantage when the honest miner and the attacker last mined a block [144]. Increase the number of block confirmations [144, 140] to make it expensive for the attacker to carry out this attack.

Blockchain reorganization: Sybil-based Double-spending, 51%, PoS long-range, and time advantage attacks are categorized under the blockchain reorganization (also known as alternative history) attack. In these attacks, the attacker creates a private chain that includes Double-spending transactions and relies on his computing power (and stakes in PoS) to find blocks quicker than honest nodes. If the attacker succeeds in making his private chain longer, the nodes adopt the longest chain of blocks according to the longest chain rule. Otherwise, the attacker has wasted his resources. Blockchain reorganization can also happen due to an accidental fork when two or more miners (honest nodes) find the next block at approximately the same time and compete for their own chain validity. The nodes with more computing power add new blocks faster and make a longer chain. The blocks of the losing chain are marked as orphaned blocks. In accidental forks, there is no Double-spending, but the losing chain nodes (e.g., miners and mining pool) lose their work and incentives [163].

3.4.5. Eclipse-based Double-spending

The Sybil attack is a network-wide attack; in contrast, an eclipse attack only targets a particular node(s) [29]. The attacker floods the victim node with his IP addresses to disengage the victim node from the blockchain network and directly attaches himself to the victim node. The attacker prevents the victim node from learning the rest of the network by not communicating/gossiping with other peer nodes. The eclipse attacks are performed on a specific node(s), such as prominent miners or merchants. The studies [145, 146] explain the possibility of Eclipse-based Double-spending on the blockchain systems, where the attacker exploits 0-confirmations or N-confirmations in fast payments to trigger double-spend.

To make fast payments desirable for both the merchant and customer, the 0-confirmations approach is followed [85]. The 0-confirmations transaction (also known as an unconfirmed transaction) is stored in the mempool of the honest nodes and is not yet included in the blockchain. The attacker exploits it and refers 0-confirmations transaction to a merchant that releases goods by assuming that confirmations will occur. The attacker eclipsed the merchant node (Fig. 24) that accepted a transaction A_{T0} with no confirmation. Next, the attacker creates a new transaction A_{T1} for double-spend and broadcasts it to the rest of the network. The merchant releases the goods to the attacker, and till this stage merchant does not know that his connections are eclipsed and cannot tell the blockchain network about A_{T0}, later blockchain network confirms A_{T1} as valid and discards A_{T0}. Hence, the attacker gets his goods without paying.

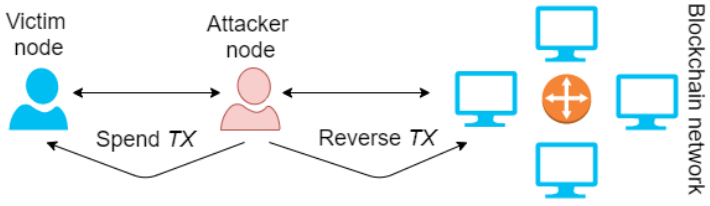

Figure 24. Eclipse-based 0-confirmations Double-spending

In N-confirmations Double-spending, the attacker eclipses the $n - fraction$ of miners along with the merchant (Fig. 25). The eclipsed merchant waits until block depth $N - 1$ confirmations before releasing the goods. The attacker shows the confirmations to a merchant from eclipsed miners. The merchant is convinced and sends the goods to the attacker. Upon completing a goods purchase, the attacker sends the actual blockchain view that makes eclipsed miners blockchain orphan. Thus the attacker procures goods without paying.

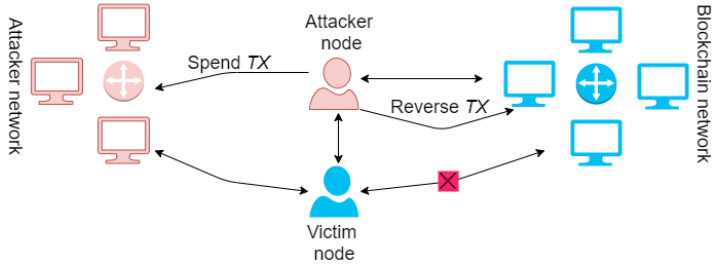

Figure 25. Eclipse-based N-confirmations Double-spending

To overcome Eclipse-based Double-spending, the study [147] asserts to wait for more block confirmations (e.g., 10 block confirmations to lessen the probability of this attack). However, waiting for more block confirmations will make the payment slow between merchant and customer, but the payment will be safe. The authors [145, 148] apply the combination of different techniques to control Eclipse-based Double-spending. For example, disable the direct incoming con-

nections. Use the white-listed nodes to choose outgoing connections, for instance, well-connected peers/miners. Random selection of addresses when making outgoing connections. The deterministic random eviction approach follows the *new* and *tried* addresses tables. The system deterministically hashes each address to a single bucket and assigns a single slot in a *new* table. If the connection succeeds, then the address moves from *new* to *tried* table. This approach reduces the attack addresses that attackers use for making outgoing connections. The feeler connections in which the nodes establish short-lived test connections to randomly selected addresses, and if the connection succeeds, the address is inserted into white-listed nodes. In anchor connections, integrate an anchor table to record the address of current outgoing connections. Once the node restarts and tries to make outgoing connections again, the system selects old addresses from an anchor table and two additional connections to persist rotation rate.

3.4.6. Border Gateway Protocol Hijacking

BGP is a routing protocol to exchange network routing information between independently operated networks such as internet service providers (ISPs) or autonomous systems (AS) [129, 126] (e.g., smart contract-based decentralized application). The BGP hijacking is a routing attack, and the attacker influences the ISPs/AS to make false announcements over the routing system to divert traffic. The attacker hijacks IP prefixes to partition the blockchain or obstruct the block propagation to perform Double-spending. Monthly, an average of 100 BGP exploitation occurs [18] over a Bitcoin blockchain. The attacker can isolate up to 50% hash rate on the Bitcoin network by hijacking fewer than 100 BGP IP prefixes [129]. The attacker uses BGP hijacking and slows down the propagation of new blocks to perform Double-spending. The attacker exploits block propagation delay to render $0-$ or $N-confirmations$ Double-spending. Moreover, the attacker can engineer block races, waste mining pool computing power, blockchain forks, selfish mining attacks, or decrease miners' revenue.

The study [126] discusses a few preventive measures against BGP hijacking attacks. For example, increasing the diversity of node connections means multihoming of the mining pool on the blockchain via multiple and distinct paths. A single-homed node can use a virtual private network (VPN) service to create multiple and distinct paths. Connect with random peers while making outgoing connections to avoid biased decisions. During BGP hijacking, round-trip time (RTT) increases; thus, by monitoring RTT, a node can quickly detect the attack and add extra random connections to protect against it. Deploy anomaly detection mechanisms and monitor sudden changes, such as the distribution of connections, the time elapsed between request and answer, and simultaneous disconnections of peers. Once an outgoing connection fails, the network refreshes its connections to embrace intentional or attacker-based connection churn. Hosting several gateways in different AS would make blockchain more robust to routing attacks. The

nodes should randomly request a block from multiple connections so the attacker does not keep waiting for the node to deliver a block.

In the Ethereum platform, to minimize the risks of BGP hijacking-based Double-spending, the authors [149] increase the number of block confirmations needed before transaction finality. Also, selectively choosing peers for querying transaction status, where a merchant verifies whether an issued transaction is committed or not before completing the purchase. Moreover, there are a few countermeasures [126] specifically for the Bitcoin blockchain. For example, encrypt Bitcoin communication or use a message authentication code to validate that the communication message content has not changed. Using distinct control and data channels means, using a randomized transmission control protocol (TCP) port rather than relying on the default port (8333) to communicate with other nodes on the network. Periodically send UDP messages that enable a node to realize that he is out-of-sync and establish new connections.

3.4.7. 0-Confirmations Race Attack

0-Confirmations race attack targets fast payment transactions when a merchant accepts payments immediately with 0-confirmations [125, 85]. The attacker sends two conflicting transactions in the blockchain with the same inputs but different transaction fees. One transaction (A_{T0}) to the merchant directly to pay for goods and the second transaction (A_{T1}) to himself with a higher transaction fee. Here, the attacker tries to exploit the intermediate time required for the initiation and confirmation of two transactions. Once the A_{T1} will be mined and accepted because of the higher transaction fee by the nodes, then A_{T0} becomes invalid. This attack is different from the Eclipse- and BGP hijacking-based 0-confirmations attack because, in the 0-confirmations race attack, the attacker indulges in a race to make his double-spend transaction valid by exploiting the intermediate time between two conflicting transactions and using a higher transaction fee.

The authors [138] suggested waiting for at least one block confirmation before sending out the goods, but this approach could lead to Finney or vector76 attacks [155, 125]. The merchant should wait for a few more confirmations like coinbase crypto exchange requires only three confirmations to mark bitcoin transaction final [150]. The waiting time for block confirmations diminishes the acceptability of fast payments. The study [151] presents a closed-form formula (combining transaction validation time) to calculate the probability of Double-spending in a race attack. In addition, enhance network policy [155, 152] to guide how to set a block confirmation number considering the value of the transaction.

The authors [85] present three techniques to detect 0-confirmations race attacks. The listening period (approx. 3.354 seconds) helps the merchant monitor transaction legitimacy before sending the goods. Inserting an observer is an extra layer on a listening period where the merchant inserts an observer that directly relays to him. The observer detects a double-spend transaction in a few seconds and

informs the merchant about this attempt. Alerting the vendor on the blockchain once other honest nodes on the network receive a double-spend transaction. Similarly, forward double-spend transactions to warn the peers in the network [154]. Furthermore, the authors [153] insert the observers in the transaction pool to detect if another transaction with the same inputs, E-cash protocol [156] incorporates a large group of trustworthy hosts as observers to detect Double-spending in real-time. The enhanced observers (ENHOBS) [157] combine the listening period and observers to monitor the Double-spending transaction.

3.4.8. Finney Attack

Finney attack is a 1-confirmation pre-mining attack [155, 145]. Similar to the 0-confirmations race attack, the attacker utilizes two conflicting transactions with the same inputs [125]. Firstly, the attacker per-mined a block with a transaction to himself (A_{T0}) and did not broadcast the block. Secondly, the attacker sends a second transaction (A_{T1}) with the same input to the merchant directly to pay for goods. Once a merchant accepts the transaction and sends the goods; then the attacker releases his privately pre-mind block into the blockchain. Now, the attacker A_{T0} will take precedence over merchant A_{T1} thus, the attacker would receive his coins back, and the merchant loses his payment and product. However, the Finney attack can fail, and the attacker loses his pre-mined block reward. For example, the probability is t/T that another block will be mined earlier than the attacker block [164]. Here, t is the time from finding the block until the attacker sends a payment and the merchant accepts, and T is the average time to find a block. The Finney attack is not a hash rate-based attack but slightly relies on hash rate; if the attacker has a low hash rate, he is less likely to carry out this attack.

The possible countermeasure is to increase block confirmations [125, 159] before completing the transaction. The authors [158] present the logarithmic waiting time for large transactions. For instance, the recipients of large transactions are advised to wait a time logarithmic in the chain's length. The gambler's ruin problem [155, 160] simulates the probability of the Finney-based Double-spending. The results show the probability increases with the attacker's mining power. The techniques (listening period, insert observer, and alerting nodes) [85] presented for 0-confirmations race attack can help to limit the Finney attack.

3.4.9. Vector76 Attack

Vector76 attack is also a 1-confirmation attack that combines 0-confirmations race and Finney attacks [125] to disrupt the fast payments. The attacker targets the exchanges and e-wallets that connect with a static IP address and accepts direct incoming connections. The attacker withdraws coins immediately when exchanges/e-wallets accept the transaction with 1-confirmation.

To launch a vector76 attack, the attacker maintains two full nodes (e.g., Node A and Node B), Node A is directly connected with exchange/e-wallet, and Node

B with the rest of the blockchain network. The attacker creates two conflicting transactions where one transaction (A_{T0}) on Node A (connected with exchange/e-wallet) and the second transaction (A_{T1}) on Node B (connected with other nodes on a blockchain). So far, both transactions have not been broadcast to the blockchain. The attacker starts mining the block with A_{T0} on Node A. Once the attacker gets a block solved, then instead of releasing the block publicly, the attacker convinces exchange/e-wallet with 1-confirmation and at the same time release A_{T1} on Node B. The A_{T0} is known only to the attacker and a connected exchange/e-wallet that will deposit the transaction into the attacker account after 1-confirmation (because of a block on Node A). The attacker would immediately withdraw the coins from his account and later a well-connected Node B transaction will become valid; thus Double-spending attack is successfully carried out. The attacker also pays the price (e.g., 1 Block that he mined on Node A) to execute this attack, but the reward is much higher than the cost.

To protect against vector76 attack, the node should avoid 1-confirmation transaction [150]. The countermeasures (listening period, insert observer, and alerting nodes) [85] presented for the 0-confirmations race, and Finney attack [125, 159] can limit the Vector76 attack. In addition, repudiate inbound connections or define inbound connections from well-connected nodes [161]. Monitor outgoing connections to detect false information from the attacker (e.g., double-spend transaction and blockchain connection).

3.5. Evaluation

In this section, we discuss the use of a newly developed framework. We selected two blockchain-based healthcare applications that are built upon the Ethereum platform. Healthcare data is sensitive, and it must be integral. Falsified or misplaced health records can cause major issues during the patient treatment process [165]. In previous years, several research studies have been conducted to preserve patients' medical health data using blockchain to ensure data integrity [76], patient ownership of his data [78], easy exchange of medical data [166], and medical insurance claims [167].

3.5.1. Use Cases

The **MedRec** [78] is a blockchain-based electronic health record (EHR) and medical research data management application (Fig. 26). The application enables patients to control their data and access their medical information across providers and treatment sites. It utilizes the blockchain to achieve fast access to medical data, system interoperability, and improved data quality. MedRec includes hospitals, patients, and researchers (e.g., doctors, institutes, and public health authorities) as network participant nodes. Blockchain node combines client node, smart contracts, mining, mining protocol, and immutable ledger. In MedRec, each participant node manages one client node, where an Ethereum client (e.g., PyEthApp)

implements the Ethereum platform specification (e.g., connection with P2P network, encoding and sending transactions). MedRec service monitors the real-time changes to signal the EHR manager that issues a patient notification and syncs the off-chain database using a database gatekeeper. Backend libraries communicate with an Ethereum client to facilitate the MedRec service. The hospitals are responsible for adding the patient's medical health record. The working procedure of MedRec (Fig. 26) is discussed in detail in Table 25.

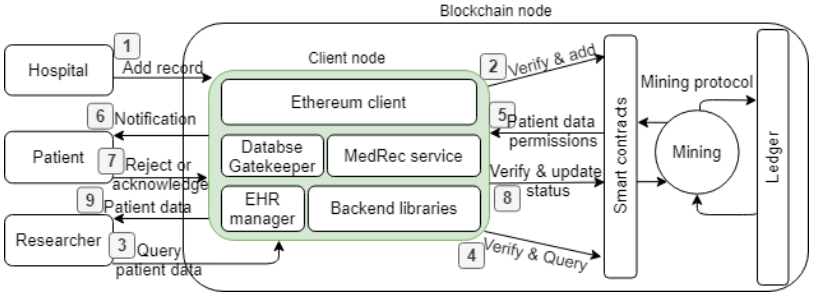

Figure 26. MedRec application orchestration

Another healthcare application is a **MIStore** [167]. MIStore is a blockchain-based medical insurance storage application (Fig. 27). MIStore utilizes blockchain ledger immutability property to provide high-credibility insurance services to users.

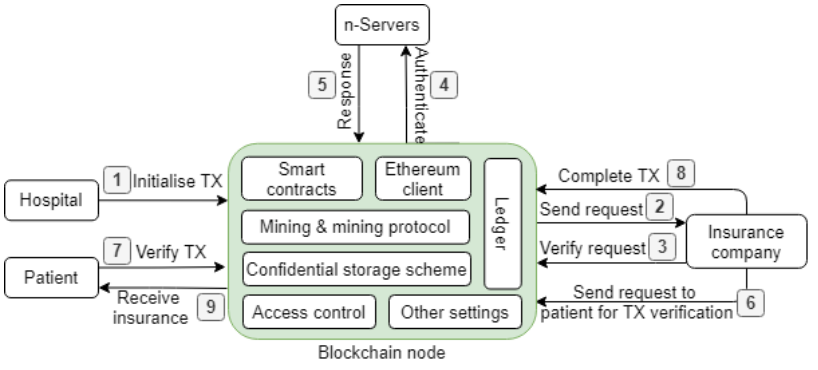

Figure 27. MIStore application orchestration

MIStore includes hospitals, patients, the insurance company, and n-servers as network participant nodes. N-servers are nodes that verify the authenticity of a transaction. Blockchain node combines Ethereum client to implement the Ethereum platform specification, smart contracts, mining, mining protocol, and immutable ledger. Confidential data storage scheme enables confidentiality for patients medical data, access control manages operations of the insurance company and other nodes accessing patients medical data, and other settings facilitate the operations of MIStore. The hospitals are responsible for adding the medical treatment costs of patients that initialize the insurance claim transaction. The working procedure of MIStore (Fig. 27) is discussed in detail in Table 25.

Table 25. Working procedure of MedRec and MIStore, and components

	MedRec	**MIStore**
Initialization	*1. Add record* - Hospital initiates add record transaction *3. Query patient data* - Researcher query patient data for medical-related research *7. Reject or acknowledge* - Patient can reject or acknowledge the data request	*1. Initialize TX* - Hospital initiates the insurance claim on behalf of patient *2. Send request* - System forwards request to insurance company *6. Request for TX verification* - Insurance company sends request to patient for verification of the insurance claim transaction
Verification	*2. Verify and add record* - System verifies the add record request from hospital *4. Verify and query* - System verifies the researcher request for patient data *8. Verify and update status* - System verifies the patient response on researcher data request	*3. Verify request* - Insurance company asks to verify TX request *4. Authentication and Verification* - n-Servers authenticate and verify the request *7. Verify TX* - Patient verifies the TX request
Processing	*5. Check data permissions* - System check the permission settings if already assigned to data for sharing *10. Incentives* - Researcher and hospital get incentives as patient medical data	*8. Complete TX* - System completes the insurance claim TX request initiated by the hospital on behalf of a patient *10. Incentives* - Hospital and n-Servers get incentives
Response	*6. Notification* - System notifies the patient about his decision to reject or acknowledge the data request *9. Patient data* - Researcher gets patient data	*5. Response* - n-Servers send response to system after verifying and authenticating the TX request *9. Receive insurance* - Patient receive insurance
Components	Transaction verification, P2P network, cryptography, data validators & miners, ledger, transactions, Ethereum virtual machine (EVM), Ethereum client, consensus, mining protocol, smart contracts, wallet, digital assets, data hash id, *EHR manager, DB Gatekeeper, backend libraries*	Transaction verification, P2P network, cryptography, data validators & miners, ledger, transactions, Ethereum virtual machine (EVM), Ethereum client, consensus, mining protocol, smart contracts, wallet, digital assets, data hash id, *n-Servers, access control, confidential storage scheme*

The working procedure of both applications is categorized into four stages (initialization, verification, processing, and response) to execute a transaction (Table 25). The table also shows the components of both applications. We utilize Table 25 and identify the commonalities in MedRec and MIStroe to build an architecture (Fig. 28) that characterizes the system assets at different layers. For example, the *application layer* provides an assertion to system users for accessing different resources. The *user layer* exposes the users who interact with the application via the *interface layer* of the application. The user interacts with the services, which are available in the *service layer*. The *Ethereum blockchain* and *Controls* layers show the different components and tools associated with the Ethereum platform. The *application specific components layer* presents the additional settings in both applications. The *network layer* is responsible for transmitting communication using different protocols and routing when interacting with external systems and databases. The *server layer* hosts off-chain services and resources. The *data storage layer* shows the database as off-chain storage.

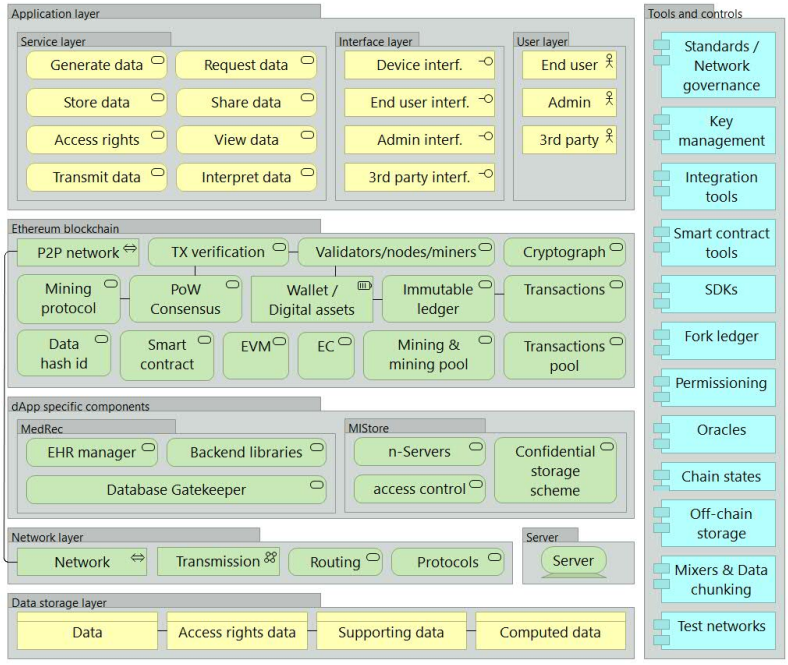

Figure 28. Components of Ethereum-based MedRec and MIStore

3.5.2. Threat Model

We evaluate the Sybil attack in MedRec and Double-spending in MIStore (Table 26) by following the framework (Table 22 & 24). We identify the security threats belonging to the Sybil attack in MedRec, and security threats that could trigger Double-spending in MIStore. Also, to alleviate the identified security threats, we accumulate countermeasures to integrate into both applications.

In MedRec, the nodes are limited, and the attacker can create Sybil nodes to trigger a node isolation attack. Using a node isolation attack, the attacker uses honest nodes to participate in the attacker governed blocks or consensus process. The results of this attack can lead the attacker to validate wrong data (e.g., to register an invalid patient), steal and sell the patient medical data, or verify illegal drug prescriptions. Furthermore, in MedRec, the attacker can perform a Sybil-based DoS attack to halt the services of MedRec.

In MIStore, Double-spending can be used for insurance frauds. In Eclipse-based Double-spending, the attacker will eclipse the node of a patient to perform Double-spending. The hospital initiates the insurance claim transaction (I_{T0}) and sends it to an insurance company (IC). IC verifies the legitimacy of I_{T0} and sends I_{T0} to a patient for verification. In this stage, the attacker receives I_{T0} transaction and sends a conflicting transaction (A_{T1}) to a patient for verification. The patient verifies the A_{T1} and sends a response message to the IC. Here, IC only knows I_{T0}, therefore completing the I_{T0} transaction. Later, the system invalidates the

Table 26. SRM of MedRec and MIStore

	Risk	Threat	ID	Vulnerability	Affected asset	Countermeasure
MedRec	Sybil attack	Node isolation attack	**V#1**	Lack of computation power	P2P network, PoW consensus	Increase computing power
			V#2	BGP does not validate routing origin	Transaction verification	Monitor round-trip time
			V#3	No proper authentication of nodes	Transaction verification, Validator/ nodes/ miners	Network joining fee
						Validating node connection
						Monitoring activities of node
		Sybil-based DoS	**V#4**	Sybil nodes can participate in mining process	Mining, Mining pool	Use computational constraint-based techniques
MIStore	Double-spending	Eclipse-based Double-spending	**V#5**	System connects node to incoming connections	Transaction verification, Validator/ nodes/ miners, Mining, Mining pool, Ledger, Digital assets	Disable incoming connections
						White-list nodes for outgoing connections
						Random selection of new connection
			V#6	Accepting unconfirmed transactions	Transactions, Transactions pool, Ledger, Digital assets	Increase confirmed blocks
		0-confirmations race attack	**V#6**	Accepting unconfirmed transactions	Transactions, Transactions pool, Ledger, Digital assets	Increase confirmed blocks
						Listening period
						Insert observer
						Alerting honest nodes
						E-cash protocol
						Enhanced observers

conflicting transaction (A_{T1}), and the attacker gets insurance. Also, IC validating transactions with 0-confirmations, hence, suspect to Double-spending due to 0-confirmations race attack.

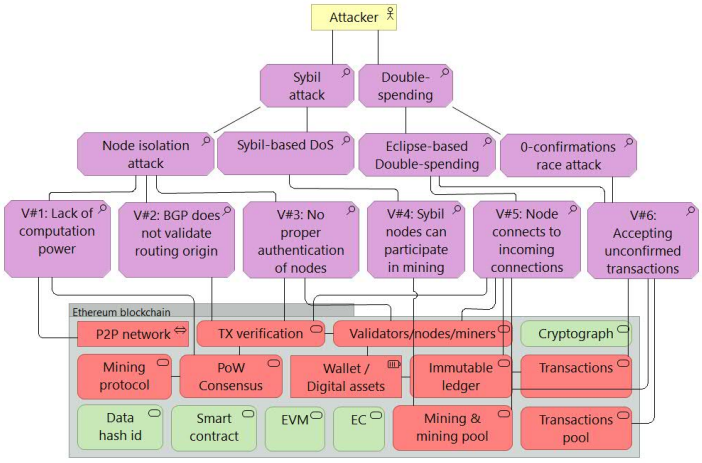

Figure 29. Threat architecture of MedRec and MIStore

In Fig. 29, an abstraction of Sybil and Double-spending risks are presented. The diagram illustrates both security threats, associated threats, vulnerabilities, and affected assets. For example, the attacker commands the Sybil or Double-spending risk using different threats by exploiting different vulnerabilities, which provoke the negative impact and negate the security criteria of the business assets. The vulnerabilities are connected to the system assets, and they depict their weaknesses that enable the attacker to harm valuable assets.

The diagram (Fig. 29) shows that the attacker can execute node isolation and Sybil-based DoS in MedRec. A node isolation attack has three different vulnerabilities. For example, insufficient computation power (V#1) in the network affects the P2P network and PoW consensus. The BGP does not validate routing origin (V#2), that affects the transaction verification. Also, the application has a limited number of nodes controlling insufficient computing power. Improper authentication of nodes (V#3) makes the transaction verification, validators/nodes/miners vulnerable. In a Sybil-based DoS, the Sybil nodes can participate in mining (V#4), thus halting the mining and mining pools operations. In MIStore, the attacker can trigger an Eclipse-based Double-spending and 0-confirmations race attack. The diagram (Fig. 29) shows the system connects the node to incoming connections (V#5) without checking the attacker's IP addresses flooding the victim node connections. V#5 affects transaction verification, validators/nodes/miners, mining, mining pools, ledger, and digital assets. Also, accepting unconfirmed transactions (V#6) affects the transactions, transaction pool, ledger, and digital assets.

3.5.3. Countermeasures

The countermeasures (Table 26) are set based on the framework (Table 22 & 24) to mitigate Sybil and Double-spending vulnerabilities within MedRec and MIStore. For instance, V#1 can be handled by adding more computing power. To overcome V#2, monitor the round-trip time to detect irregular patterns. Vulnerability V#3 is restrained by node authentication before joining the blockchain network, such as asking network joining fees, validating node connections, and monitoring node activities. V#4 is controlled by managing the computing power using a computational constraint-based Sybil resistance technique that requires spending computational resources proportional to the number of identities produced. Vulnerability V#5 can be controlled by disabling incoming connections, introducing white-listed nodes to make outgoing connections, and using only random selection to establish new connections. In Eclipse-based Double-spending to minimize V#6, increase confirmed blocks. In a 0-confirmations race attack, the V#6 can be mitigated by using more confirmed blocks, adding a listening period, inserting an observer, alerting honest nodes, E-cash protocol, and enhanced observers.

3.6. Related Work

The acceptance of blockchain technology continues to increase in various fields such as finance, healthcare, resource monitoring, digital rights management, supply chain [16]. For example, the authors [78] build the patient medical health record system using blockchain, and the study [167] provides a blockchain-based tamper-proof insurance claim system. The growth of blockchain-based applications maximizes the research of blockchain security. There exist a few studies that evaluated the security of various blockchain systems. The related works fall short in various directions that the current work addresses.

Nicolas et al. [155] perform a systematic literature study to explore defensive strategies of blockchain systems that protect against Double-spending and selfish mining. The study identifies the six defensive strategies: monitoring, alert forwarding, alert broadcasting, inform, detection, and conceptual research design. The authors focus on the countermeasures of Double-spending and selfish mining that are classified in these six strategies. In contrast, our study provides a framework based on the SRM domain model to explore Sybil and Double-spending in blockchain systems. Moreover, we focus on the threats of Sybil and Double-spending, the assets to secure in blockchain systems, the vulnerabilities, and what are the countermeasures for risk treatments. In [129], the authors identify different attacks in blockchain systems, mainly emphasizing attack surfaces. For example, the security attacks on blockchain cryptographic constructs, distributed architecture (P2P network), and blockchain-based applications. The study does not discuss any attack in detail or its associated vulnerabilities. Another work [125] that classifies security threats in blockchain technology. Similarly, this study does not explore the threats, vulnerabilities, or affected assets in detail. Conversely, we present a framework to explore Sybil and Double-spending risks detailing threats, vulnerabilities, affected assets, and countermeasures.

Pinzon et al. [144] present the Double-spending with a time advantage on the Bitcoin blockchain. The paper introduces the two different time advantage-based approaches and algorithmic comparison to show the probability of both approaches to carry Double-spending. The research work does not present attack scenarios, vulnerabilities, affected assets, or viable risk treatments. In [18], the authors assess blockchain consensus mechanisms against the 51% attack, which can cause Double-spending. The study managed a comparison of different consensus mechanisms along with their weaknesses. Zhang et al. [29] present the attack model that combines a Double-spending with a Sybil attack in the Bitcoin network. The study exemplifies the attack states, the success probability of the proposed combined attack, and suggestions to improve the Bitcoin blockchain.

The related works mentioned earlier rely primarily on identifying the security risks associated with the blockchain systems. None has explained the security risks by addressing their associated threats, vulnerabilities, affected assets, what assets to protect, and countermeasures. These related studies fail to report on the security risk management of blockchain systems because they do not follow any SRM domain model. Furthermore, only Nicolas et al. [155] built the framework for systematic evaluation of countermeasures against security risks. In this work, we utilize the BbRM which is based on the SRM domain model to develop a framework and conduct security risk management of Sybil and Double-spending, report the threats and vulnerabilities, the assets to secure, and classify affected assets within blockchain systems, and countermeasures for risks treatments.

3.7. Discussion

Security risk management is an iterative process because new security threats and vulnerabilities can emerge. For example, BitMex research reported the double-spent transaction in Bitcoin on the 21st of Jan 2021 [168]. The source and attack method of this double-spend transaction is not yet known. However, it indicates that the attacker continuously builds new techniques to sabotage the confidentiality, integrity, and availability of blockchain-based applications. To communicate security threats to blockchain developers, practitioners, and other associated stakeholders, we need a comprehensive blockchain security reference model.

To ensure the quality of empirical studies in software engineering, assessing threats to validity is important [169]. Our current research has several limitations, and we discussed following the Zhou et al. [169] threats to validity mapping. The relevant threats are restricted time span, publication bias, subjective interpretation, and lack of expert evaluation. The restricted time-span is that the researcher cannot predict other applicable studies beyond the time span. For example, blockchain is relatively new but continuously evolving, and not all the possible security threats are researched. It reflects the likelihood that a wide variety of security threats will emerge in the future. The threat concerning publication bias is that the related studies are more likely to report positive results than negative results. Also, various security threats and countermeasures are unclear, or there is no real implementation. The threat of subjective interpretation exists since we might have different interpretations and opinions related to identified threats, vulnerabilities, and countermeasures. Moreover, a lack of expert evaluation may also lead to a subjective interpretation and erroneous conclusion.

Blockchain technology looks promising because many blockchain-based applications have recently appeared, but their security threats need to be evaluated on a larger scale. In the next chapters, we develop an ontology-based security reference framework, and it may solve the threats to the validity of the current work. For example, ontology by design is dynamic, and researchers can update at any time, resolving the restricted time span and publication bias threats. Furthermore, the practitioners and researchers will be able to study and update the ontology online. As a result, the threats related to subjective interpretation and lack of expert evaluation will be minimized. Overall, resolving the above-mentioned limitations can bring richer insights and lead to a more in-depth contribution to performing security risk management of blockchain-based applications.

3.8. Summary

This chapter answers our RQ1: What is the current state of the art for performing the SRM of traditional applications utilizing blockchain and the SRM of blockchain-based applications? This chapter produces a blockchain-based reference model (BbRM) for security risk management (SRM) of traditional appli-

cations using blockchain. The reference model is further extended to support the SRM of blockchain-based applications. We use the SRM domain model constructs to develop the reference model. We first present the architecture, which includes the components of traditional applications, then using the conceptual model we map the security threats of traditional applications and blockchain as a countermeasures solution. Like traditional applications' SRM, we extended the BbRM with the security threats that may appear in blockchain-based applications and countermeasures to mitigate them. The BbRM defines the system assets to secure, business assets, their security criteria, and the countermeasures.

We evaluate the reference model by looking how the proposed BbRM may assist the SRM of traditional and blockchain-based applications? Following this, we use BbRM to explore further the data tampering threat of traditional applications and how blockchain might operate as a countermeasure solution. To realize the true potential of blockchain-based applications, we extended our analysis using BbRM to look at the security threats of blockchain-based applications. Notably, we explore the Sybil and Double-Spending risk. The result is a comprehensive framework for the SRM of both security threats. The framework illustrates the assets to secure, the classification of threats that the attacker can trigger using the Sybil attack, the identification of threats that cause the Double-spending, the vulnerabilities of identified threats, and their countermeasures. We evaluated this framework by performing the SRM of Sybil and Double-spending threats in Ethereum-based healthcare applications (e.g., MedRec and MIStore). This research's findings can support the software developers and decision-makers regarding Sybil and Double-spending while building blockchain-based applications.

In general, the BbRM enables a go-to strategy for identifying and explaining the security threats of traditional and blockchain-based applications, along with their countermeasures.

4. ONTOLOGY-BASED REFERENCE FRAMEWORK FOR MANAGING SECURITY RISKS

The blockchain-based reference model (BbRM) for the SRM of traditional and blockchain-based applications is presented in Chapter 3. However, the BbRM depicts the static knowledge base and inefficient instantiation of information security knowledge for the SRM of domain-specific applications. To fill these gaps, this chapter focuses on the second research question, RQ2: How can SRM domain model abstraction be used to build an ontology-based reference framework? As a result, we build an upper-level reference ontology (ULRO) as a foundation ontology using the SRM domain model to encode information security concepts following the fundamental constructs of SRM. The ULRO provides semantic interoperability, general concepts common to all domains, and enables a common foundation for ontology representations in the information security domain. The ULRO presents an ontology-based structural representation to support the dynamic knowledge encoding and instantiation with information security knowledge for the SRM of domain-specific applications.

The remainder of the chapter is structured as follows: Section 4.1 discusses the ULRO to explain the ontology artifacts, such as concepts, relationships, annotations, and description logic. In Section 4.2, we describe the use case to evaluate the ULRO. The case belongs to the smart healthcare application where the patient himself monitors his health condition using smart devices. For instance, blood pressure measurements are necessary for the healthcare and insurance providers for a successful patient's insurance claim. In Section 4.3, we present the instantiation of the ULRO using the data tampering threat and Sybil attack from Chapter 3. We map the ULRO instantiation on smart healthcare processes using the colored Petri nets for evaluation. Furthermore, we discuss the ULRO with the domain experts to check the completeness and correctness of the ULRO according to the constructs of the SRM. Section 4.4 discusses the related work. Section 4.5 is the discussion, and Section 4.6 concludes this chapter.

4.1. Upper-Level Reference Ontology

The process of creating an ontology representation is time-consuming, and error-prone [170]. Upper-level ontology representations in this scenario assist in guiding the process and provide a consistent framework based on established concepts and relationships [171]. Therefore, we create the ULRO to provide a foundational and consistent knowledge model for the SRM by incorporating the information security concepts. We use the SRM domain model to build ULRO, a shared conceptualization to support the SRM. As we discussed in chapter 3, BbRM is built upon the SRM domain model; the ULRO includes the same constructs (Fig. 30). Therefore, the BbRM, in this case, encourages the usage of SRM domain model

constructs for ULRO construction. According to [170, 171, 172], the upper-level ontology is domain-independent and provides a common foundation in the context to derive domain-specific ontologies. In accordance with this, the ULRO is a high-level, domain-independent ontology intended to be reused. It provides a common foundation from which domain-specific ontology representations for the SRM may be derived. Moreover, the ULRO provides semantic interoperability because it contains generalized concepts and relationships that stay the same in all domain-specific ontology representations. For instance, the ULRO defines the fundamental concepts of SRM, such as assets, security criteria, threats, vulnerabilities, countermeasures, and their relationships (Table 27).

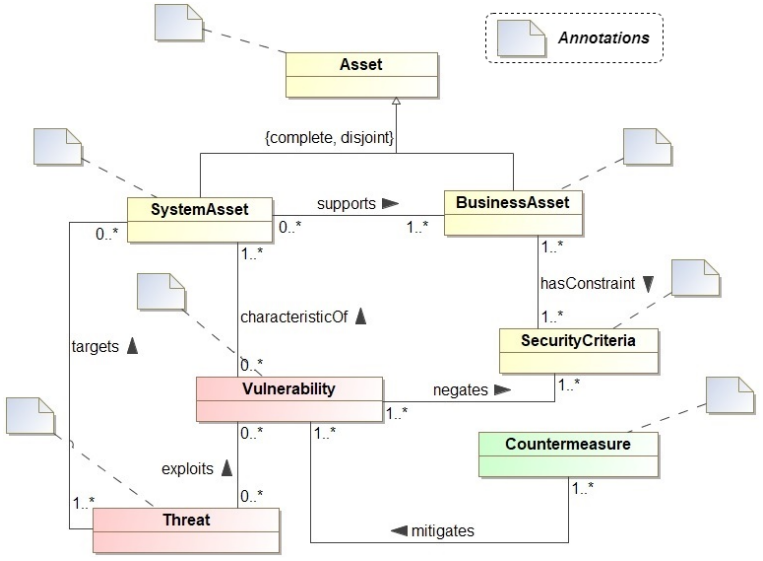

Figure 30. Upper-level reference ontology architecture

In URLO, the concepts are encoded as classes. We present ULRO "is-a" based taxonomic structure (Fig. 31) that illustrates the upper-level class hierarchies. The ULRO follows a semantic web language OWL, and OWL ontology begins from owl:Thing class that encapsulates every individual of the ontology. Therefore, each user-defined class is implicitly a subset ($\subseteq$) of $owl:Thing\ (T)$ [57]. The asset class ($Asset \subseteq T$) has two subclasses ($SystemAsset \subseteq Asset$ and $BusinessAsset \subseteq Asset$), where the system asset supports the business asset. The business asset has constraint security criteria ($SecurityCriteria \subseteq T$) that must be met for a system to be evaluated and deemed acceptable. The threat class ($Threat \subseteq T$) is another top-level concept that categorizes the threats that may exploit some vulnerabilities in the system. The vulnerability class ($Vulnerability \subseteq T$) categorizes the vulnerabilities that are characteristics of the system assets to depict their weakness that the attacker may exploit to trigger a particular threat. The countermeasure class ($Countermeasure \subseteq T$) combines the controls as countermeasures to mitigate the vulnerabilities to protect the system against security threats.

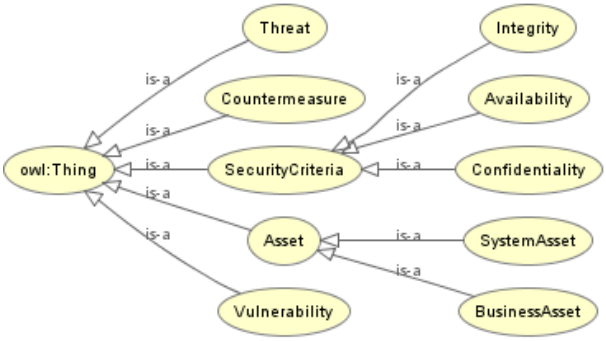

Figure 31. Class hierarchies of upper-level reference ontology

Relationships in URLO present object properties that define the link between two concepts (Table 28). For example, the object property *exploits* linking the threat and vulnerability concepts (*threat* ***exploits*** *vulnerability*) (Fig. 32a). The object properties in ULRO architecture are asymmetric, which means exploits property relates threat to vulnerability, but it cannot relate vulnerability to threat. To create such relationships, we defined corresponding inverse object properties in ULRO (Table 28). For example, Threat exploits Vulnerability, and its inverse is Vulnerability isExploitedBy Threat (Fig. 32a). Similarly, SecurityCriteria is a constraintOf BusinessAsset and its inverse is BusinessAsset hasConstraint SecurityCriteria (Fig. 32b). Because of the value of a business asset in an organization, the security criteria confine it and describe the security needs in terms of confidentiality, integrity, and availability [23]. We add the domain and range classes on object properties to describe how the particular object property connects a subject to an object (Table 28). The domain class specifies the subject class to which the given property belongs, and a range is an object class. For example, in the previous expression, the threat is a domain, and vulnerability is a range.

Table 27. ULRO classes and description logic

Class	**Description logic**
Asset	*isHarmedBy* some Vulnerability
SystemAsset	*supports* some BusinessAsset
BusinessAsset	*hasConstraint* some SecurityCriteria
SecurityCriteria	$constraintOf$ some BusinessAsset
Threat	*exploits* some Vulnerability
	targets some SystemAsset
Vulnerability	$characteristicOf$ some SystemAsset
	negates some SecurityCriteria
	harms some Asset
Countermeasure	*mitigates* some Vulnerability

Moreover, we use annotation properties for adding additional information (metadata) to explain the defined concepts. The annotations highlight the context, a systematic summary of the concept [173], so we used them in ULRO. In addition, the annotations cover important details and support active learning that improves comprehension and retention of information. We define the annotation proper-

Table 28. Domains and ranges of the ULRO object properties

Property	Domain → Range	Inverse	Domain → Range
supports	SystemAsset→BusinessAsset	isSupportedBy	BusinessAsset→SystemAsset
constraintOf	SecurityCriteria→BusinessAsset	hasConstraint	BusinessAsset→SecurityCriteria
exploits	Threat→Vulnerability	isExploitedBy	Vulnerability→Threat
targets	Threat→SystemAsset	isTargetedBy	SystemAsset→Threat
characteristicOf	Vulnerability→SystemAsset	hasCharacteristic	SystemAsset→Vulnerability
negates	Vulnerability→SecurityCriteria	isNegatedBy	SecurityCriteria→Vulnerability
harms	Vulnerability→Asset	isHarmedBy	Asset→Vulnerability
mitigates	Countermeasure→Vulnerability	isMitigatedBy	Vulnerability→Countermeasure

ties using datatype xsd:string (Table 29). Protégé provides the built-in resource description framework schema (rdfs) annotation properties such as rdfs:label for describing the concept and rdfs:comment for a longer definition of the concept. For instance, rdfs:comment annotation linking the class Asset to the data type xsd:string "Asset definition" (Fig. 32c). We create a few Dublin Core (dc) annotation properties (Table 29) [174], e.g., dc:isReferncedBy to add the information about the source from which the knowledge is retrieved for a particular concept. To add references, we introduce the data properties and individuals. Individuals represent the concrete object for dc:isReferencedBy annotation property to add a particular reference for a particular concept. The individual object includes data properties such as the title of the article, authors, DOI, publisher, year, number of pages, type (e.g., journal, conference, book, or web), and link to a web article.

Table 29. Annotation properties of the ULRO

Annotation	Type	Detail
rdfs:label	xsd:string	Built-in annotation property to describe the concept.
rdfs:comment	xsd:string	Built-in annotation property for a definition of the concept.
rdfs:seeAlso	xsd:string	Built-in annotation property for supplementary information.
dc:isReferencedBy	xsd:string	Dublin core annotation property to add source of knowledge.
dc:creator	xsd:string	Dublin core annotation property to define ontology creators.
dc:title	xsd:string	Dublin core annotation property to define ontology title.
dcterms:license	xsd:string	Dublin core annotation property to define license.
dcterms:isPartOf	xsd:string	Dublin core annotation property to determine if the threat is connected to traditional or blockchain-based applications.

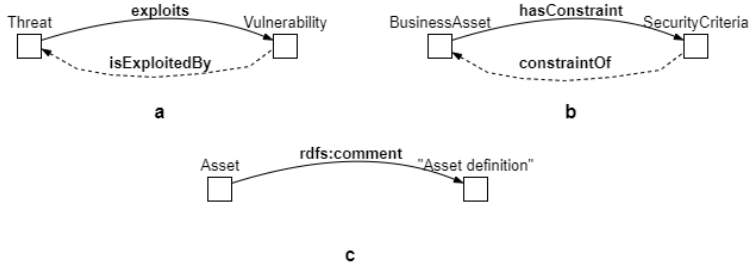

Figure 32. Example of object property, inverse object property, and annotation

The ULRO is accessible online (Table 30), and as we previously discussed, we use the Protégé to code the concepts and relationships (Fig. 33) using the OWL to build a readable semantic infrastructure that supports the triplet format

(e.g., subject-predicate-object) for describing the concepts. For example, in this triplet *(Threat **exploits** Vulnerability)*, *Threat* is a **subject**, *exploits* relationship is a **predicate** and *Vulnerability* is an **object**. We use the pellet reasoner to check the consistency of the ULRO and compute the concepts' classification hierarchy.

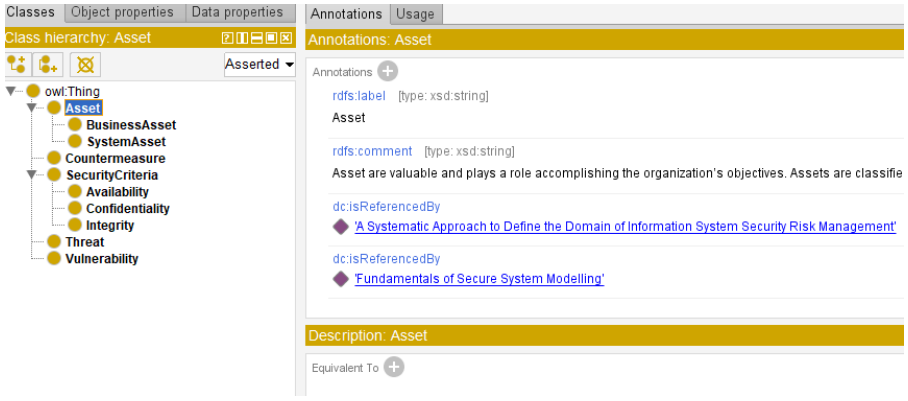

Figure 33. Visualization of the ULRO in the Protégé

Table 30. ULRO resources

Resource	URL
GitHub	`https://github.com/mubashar-iqbal/upper-level-reference-ontology`
MMISW repo.	`https://mmisw.org/ont/~mubashar/ULRO`
	`https://mmisw.org/ont/~mubashar/ULRO-instantiation`

4.2. Use Case

Figure 34 describes the smart healthcare application case where a patient can monitor himself at home with different autonomous devices such as a smartwatch, blood pressure monitoring device, and components of a smart home such as air- and water quality sensors, these devices collect data about a patient's activities and ecological environment. The devices communicate with each other sharing the data if needed. In this work, we take an example of blood pressure monitoring that is required for initiating the insurance claim. This data is continuously collected, enabling the feedback loop for a healthcare provider and professional who monitors the patient. In parallel, during the patient's visit to a healthcare professional, the latter creates electronic health records (EHR) that include laboratory-test results and anamnesis. The healthcare provider and professional have access to the EHR. After the treatment, the healthcare provider and professional send their medical reports to the insurance provider requesting the insurance claim.

During the blood pressure monitoring and sharing, the patient can face a data tampering threat due to a lack of security in either smart devices (e.g., blood pressure monitoring device), or other linked applications (e.g., communication channels). The healthcare providers collect and share the patient's healthcare records (PHR) to request compensation from the insurance provider. The tampering with

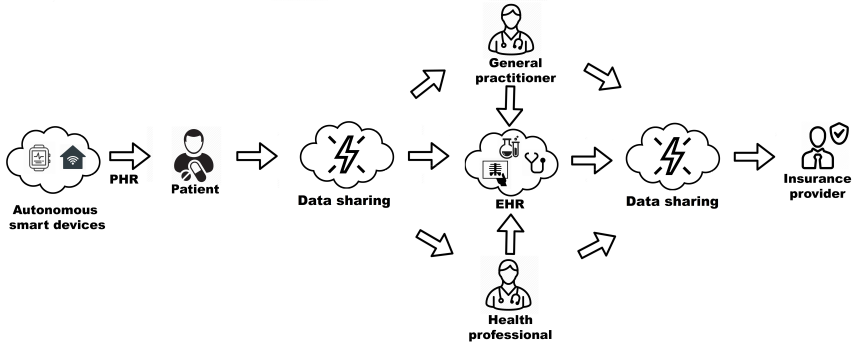

Figure 34. Data sharing and processing in smart healthcare system

PHR leads to conflict with the insurance provider, who does not process the insurance or process insurance to the wrong person (e.g., to the attacker). Such security threats hurdle the usage of smart healthcare services. Moreover, the PHR can be used in the internal processes of healthcare, personalized medicine, and research. As such, the data has been tampered with, and the system cannot ensure its integrity; thus, inappropriate results may occur. Also, privacy conflicts may occur when sharing the PHR with different stakeholders. However, our main focus is on the insurance claim scenario with the risk of data tampering in this work. We introduce the data tampering threat in a smart healthcare application that may occur when the patient collects the data during home monitoring.

4.3. Evaluation

We use the two evaluation approaches (Fig. 35) inspired by the ontology engineering method SABiO [175] for the verification and validation of the ULRO. First, we reached the domain experts from the information security and ontology domain to assess the suitability and correctness of ULRO for the SRM of traditional and blockchain-based applications. The expert evaluation helps us verify the ULRO to ensure whether we are building the ontology correctly and fulfilling our requirements. Second, we use the colored Petri nets (CPNs) models and streams of the smart healthcare application to map the encoded concepts of SRM from the URLO. Such as system assets, business assets, security criteria, threats, vulnerabilities, and countermeasures. The CPNs-based evaluation helps to validate the ULRO that whether we are building the right ontology based on the requirements for the SRM of traditional and blockchain-based applications.

For the CPNs-based evaluation, we instantiated the ULRO with the data tampering threat and Sybil attack from chapter 3. The ULRO instantiation also answers how the ULRO can be extended to a domain-specific ontology representation. The instantiation of the ULRO is accessible online (Table 30). The classes hierarchy (Fig. 36) of the instantiated ULRO includes the added concepts of smart healthcare application, such as the assets, data tampering threat that may appear

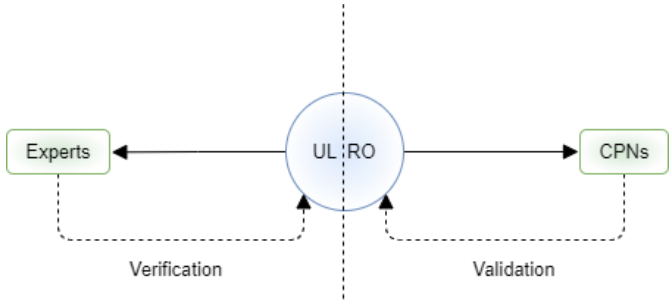

Figure 35. URLO evaluation methods

in a traditional smart healthcare application, blockchain as a countermeasure solution to mitigate it, Sybil attack that may appear in blockchain-based smart healthcare application, and its countermeasure.

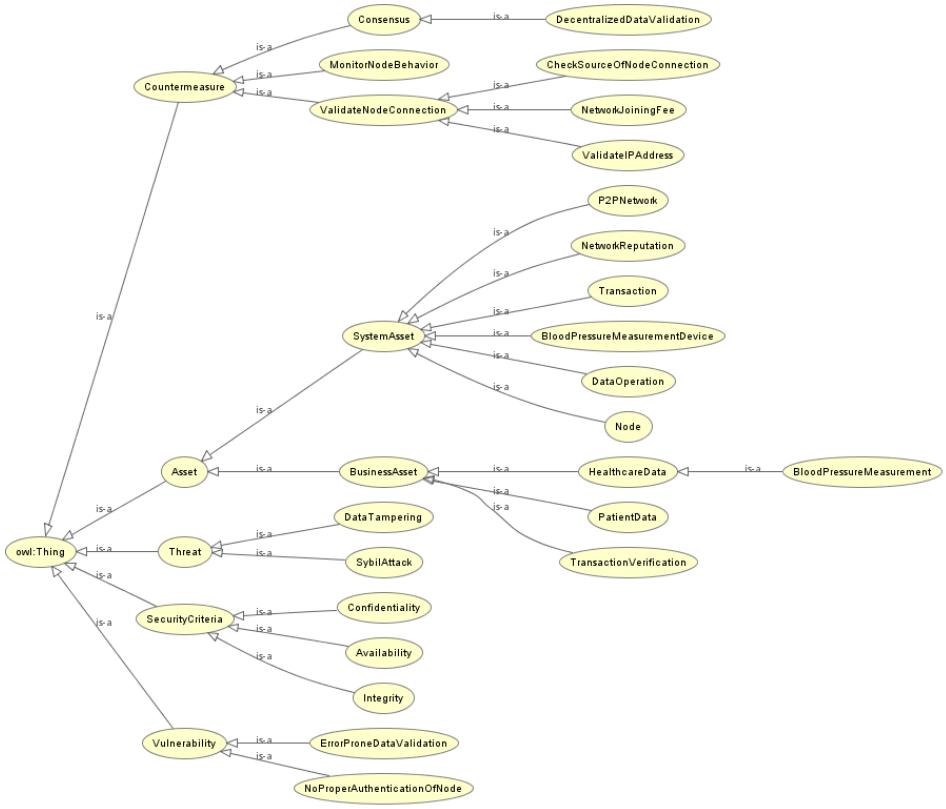

Figure 36. URLO instantiation using smart healthcare application case

4.3.1. Experts Discussion

We approached three domain experts to participate in the study to assess the ULRO's suitability and correctness in terms of encoded concepts and relation-

ships for dealing with the SRM. The *Expert 1* is working as a security engineer in a software company, and her professional experience consists of expertise related to information security, security risk analysis, and the internet of things. The *Expert 2* is a researcher and mainly has experience in ontology development, user privacy in socio-technical systems, and knowledge of information security. The *Expert 3* is a researcher mainly doing research in the domain of blockchain-enabled business processes, secure smart contracts, and ontology development. We prepared the seven interview questions (Table 31) for the discussion with the experts. We conducted face-to-face discussions with each domain expert individually, and each discussion session lasted between 20-30 minutes.

Table 31. Experts discussion questions. ($\odot$) agree, and ($\oplus$) agree with suggestions

Id	Question	Expert 1	Expert 2	Expert 3
Q1	Does the ULRO cover fundamental concepts of SRM?	$\odot$	$\odot$	$\odot$
Q2	Are the ULRO relationships well defined?	$\oplus$	$\odot$	$\oplus$
Q3	Is the ULRO complete for capturing the knowledge of blockchain as a countermeasure solution?	$\odot$	$\oplus$	$\oplus$
Q4	Is the ULRO complete for capturing the information security knowledge of blockchain-based applications?	$\odot$	$\odot$	$\oplus$
Q5	Are the annotation terms clear?	$\odot$	$\oplus$	$\odot$
Q6	Do the annotations explain the concepts clearly?	$\odot$	$\odot$	$\oplus$
Q7	Is the ULRO easy to extend?	$\odot$	$\odot$	$\odot$

Expert 1 agrees with the Q1, Q3, Q4, Q5, Q6, and Q7 and concludes that the ULRO is correct according to the required fundamental concepts of SRM. In Q2, Expert 1 suggests using a relationship "hasContraint" to define the security criteria constraint directly on the business asset in the ULRO. Expert 2 agrees with the Q1, Q2, Q4, Q6, and Q7 and provides a discussion on Q3 and Q5. In Q3, Expert 2 was asked to address the priority and likelihood of the security threats and vulnerabilities. Both concepts are out of scope from the current research. However, in the future, this information can be added by defining the annotation properties. In Q5, he suggested adding additional annotation properties (dc:isReferencedBy, dc:creator, dc:title) to define references from where the concepts are accumulated, the creator of the ontology representation, and ontology title. Expert 3 agrees with the Q1, Q5, and Q7 and supports the categorization of the system and business assets. In Q2, Expert 3 is asked to define the inverse relationships (Table 28) explicitly for each encoded concept to reveal the information about the relationship between two concepts when interpreting in the opposite direction. In Q3 and Q4, he asked to define an annotation property (dcterms:isPartOf) when defining the threats to distinguish which threat is mitigated using the blockchain and which threat appeared within a blockchain-based application. In Q4, he also asked to define annotation property (rdfs:seeAlso) on individuals to clarify whether a particular individual is related to the reference. Overall, the experts' discussion results demonstrate that the ULRO concepts and relationships properly cover the aspects they aim to represent for dealing with the SRM.

4.3.2. CPNs-based Evaluation

CPNs cover many phases of system development ranging from requirements to software design, validation, and implementation [176]. The SABiO method [175] and ontology of software defects, errors, and failures (OSDEF) [177] perform the validation by instantiating their built ontologies using a real-world case linked to the ontology's domain. The primary objective of such validation is to determine whether the correct ontology is being developed and whether it is serving its intended purposes. Following that, we use the case of smart healthcare application and instantiate it with the SRM concepts of data tampering and Sybil attack (Fig. 36 & 37). We go a step further and use the CPNs to provide insights into the actual properties and execution logic of the ULRO during the software design and implementation. The CPNs-based evaluation helps us ensure that the correct ontology is being built and can serve its intended purpose.

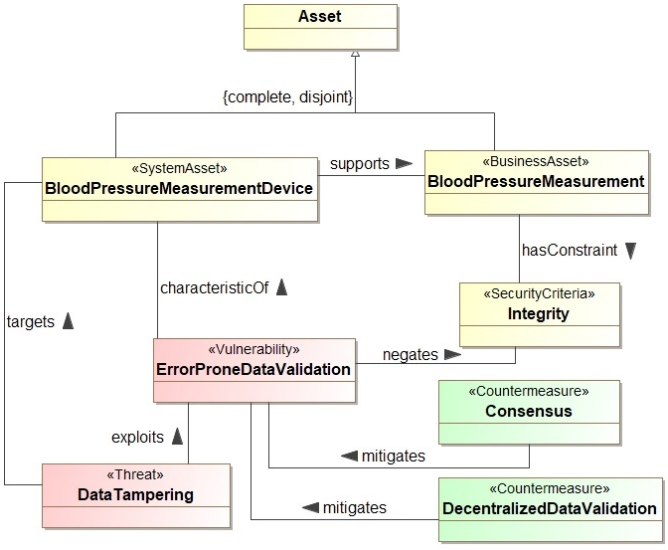

Figure 37. URLO instantiation using data tampering threat for CPNs-based evaluation

We use the *CPN tools* tool to build and execute CPNs models; the tool combines the Petri nets, programming language constructs, graphical notations, and the semantic foundation for communication in the system and system's simulation. The CPNs models we constructed are related to the smart healthcare application for assessing the ULRO's applicability by mapping the encoded concepts of the SRM (e.g., the SRM of data tampering threat (Fig. 37)). This may also solve system challenges related to the implementation and security in the design phase rather than in the later software development phases. We do a security risk analysis of smart healthcare applications and identify the assets to protect as part of the CPNs-based evaluation. We then identify the threats and vulnerabilities that the attacker can exploit. For example, we take data tampering in a traditional application, Sybil attack in a blockchain-based smart healthcare application, and

countermeasure solutions. We encode these concepts (e.g., assets, threats, vulnerabilities, countermeasures) by extending the ULRO. During the design phase of a smart healthcare application, we apply an extended version of the ULRO to determine where we need to protect the assets, where the vulnerability can arise or which system assets are susceptible, and where to introduce the countermeasure.

The CPNs models simulate the smart healthcare application by creating compact and parameterized patterns to provide insights on how to incorporate the aggregated knowledge for the SRM of data tampering threat. In general, the CPNs-based evaluation explains the execution logic of the ULRO for the SRM from the application point of view by characterizing its structural and behavioral properties. The CPNs models we created are related to the patient-internal process, healthcare provider internal process, healthcare professional internal process, and sub-models collect data, data tampering, consensus, and validate data. Here, we discuss only the patient-internal process, collect data, data tampering, consensus, nodes validation, and insurance claim models. The data flow across the CPNs models is depicted by token colors (Table 32). The tokens are specified in the first column and a brief explanation in the second column.

Table 32. Colors in the CPNs models

Color	Detail
cm	Collected blood pressure measurement value.
vm	Bloodpressure measurement for validation.
tm	Tampered value for blood pressure measurement.
pm	Original data for blood pressure measurement.
isDataTampered	Flag indicating that attack was successful and tampered data is used instead of original one.
cnt	Validaton node index.
nodeCount	Number of nodes performing the data validation.
valResult	Node data validation result.
totalValid	Number of nodes considering that data is valid.
totalInvalid	Number of nodes considering that data is invalid.
srcData	Original data taken from the data source to compare with incoming data during the validation.
claimData	Blood pressure data selected along with the insurance claim for the successful insurance claim decision.
isInsuranceDataValid	Flag indicating that data collected for the insurance claim is valid or not.

Here, we provide an abstract overview of the simulation flow (Fig. 38) of the CPNs models to illustrate the execution logic for the SRM of data tampering threat using the ULRO. The simulation begins by collecting the patient BPMs ((1,125), (2, 150)) and sharing them with the network nodes. The attacker can tamper with the BPMs (1, 85) before sending them to the network to be processed. The modified BPMs are identified and marked invalid by the blockchain-based consensus and decentralized validation process, on which the insurance claim decision is determined. Next, we discuss the CPNs models in detail.

Patient data collection model: This model (Fig. 39a) illustrates the patient's internal process to monitor and collect blood pressure measurement. The insurance

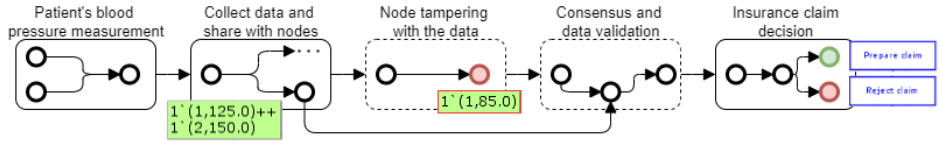

Figure 38. Simulation flow of CPNs models

provider asks the patient to provide a blood pressure measurement (BPM) to prepare the insurance claim. First, the patient checks if the requested BPM data exists in his PHR. If not, he measures his blood pressure using the BPM device. Once the required data is collected, it combines with PHR and sends it to the healthcare provider and healthcare professional for further processing. The patient data collection model has two sub-models: *(i)* collect data and *(ii)* data tampering. In the collect data model (Fig. 39b), we assume that the patient uses the BPM device to monitor and collect the blood pressure upon the data request. In this model, we assume the BP data source is a BPM device that produces two valid measurements (125, 150) based on the process ids (1, 2). The BP data source, a BPM device, is our system asset and BPMs are the business asset. If the BPM data is correct, then it sends the output for the next stage wherein the data tampering model (Fig. 40) the attacker gets the original BPM data and tampers it. On the stage of P-share PHR, there is a risk that the shared data has been tampered and the healthcare provider and healthcare professional received the tampered BPM data.

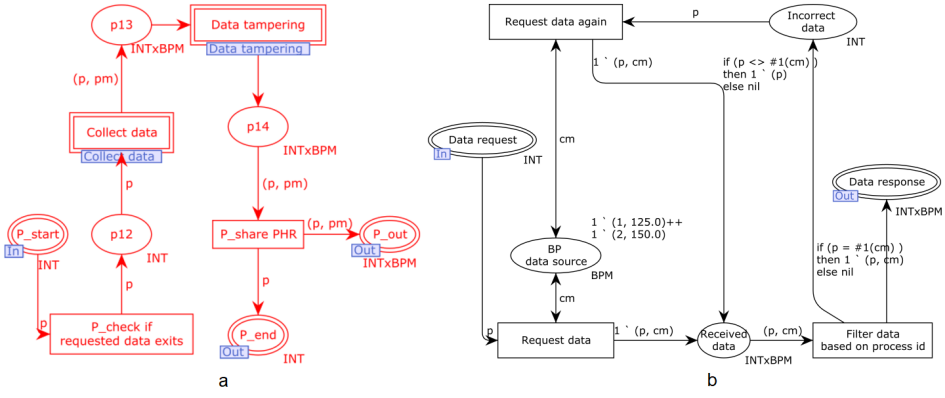

Figure 39. Patient's internal process for data collection

Data tampering model: The patient data collection model is based on a centralized infrastructure where the data tampering threat can appear when the patient monitors the blood pressure and sends it to the healthcare providers and healthcare professional for the insurance claim. Query#1 fetches the results from the ULRO instantiation related to the threat, its associated vulnerability, and affected system asset. For instance, in the data tampering model (Fig. 40), according to the query results, the application has the vulnerability of error-prone data validation. It means the traditional application does not impose the mechanism to check the

data correctness properly. The attacker gets the original BPM data and tampers it with the false BPM value (e.g., 85). Moreover, we can also assume that the centralized authority manages the data validation controls that tend to make or cause errors. Here, to highlight and send the tampered BPM values, we check if the data is tampered, then send tamper values rather than the original data.

```
Query#1: SPARQL query to fetch data tampering threat details
from the ULRO instantiation.
----------------------------
SELECT DISTINCT ?Threat ?Vulnerability ?SystemAsset WHERE {
    ?Threat rdfs:subClassOf ?Vulnerability .
    ?Threat rdfs:subClassOf ?SystemAsset .
    ?Vulnerability owl:onProperty ULRO_INST:exploits .
    ?SystemAsset owl:onProperty ULRO_INST:targets .
    ?Threat rdfs:seeAlso ?Domain .
    FILTER regex(?Domain, "^Mitigated")
}
```

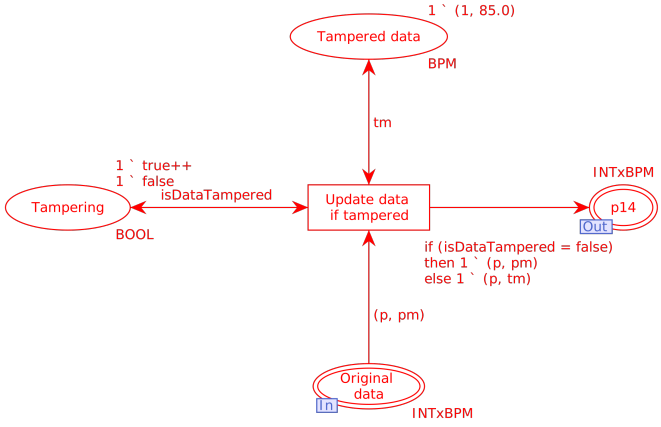

Figure 40. Data tampering model

Countermeasure model: We use the decentralized consensus as a countermeasure where decentralized distributed nodes perform data validation before processing it further or saving it in the ledger or database. For instance, Query#2 fetches the results from the ULRO instantiation related to the countermeasure and which vulnerability it mitigates (e.g., Consensus mitigates ErrorProneDataValidation). The countermeasure model (Fig. 41) illustrates the consensus process, where we use a variable *cnt* that defines the number of nodes required to perform the consensus. It is set to 4 nodes now. The consensus mechanism iterates through the defined number of nodes that perform the data validation while comparing the input value with the original data in the data source. The validation results are summarized based on the total valid and invalid responses. For example, the countermeasure model has Validate sub-model (Fig. 42) that illustrates the data validation performed by the nodes. The nodes compare the received data with the original data from the data source. If the data differs, it counts as an invalid result;

otherwise a valid result. The validation is successful when most nodes consider the input value valid. In the end, this process results in true or false that indicate the BPM data the system has received is integral or tampered with.

```
Query#2: SPARQL query to fetch countermeasure from the ULRO
instantiation for mitigating the data tampering.
----------------------------
SELECT DISTINCT ?Countermeasure ?Vulnerability WHERE {
    ?Countermeasure rdfs:subClassOf ULRO_INST:Countermeasure .
    ?Countermeasure rdfs:subClassOf ?Vulnerability .
    ?Vulnerability owl:onProperty ULRO_INST:mitigates .
    ?Countermeasure rdfs:label ?Label .
    FILTER regex(?Label, "^Consensus") .
}
```

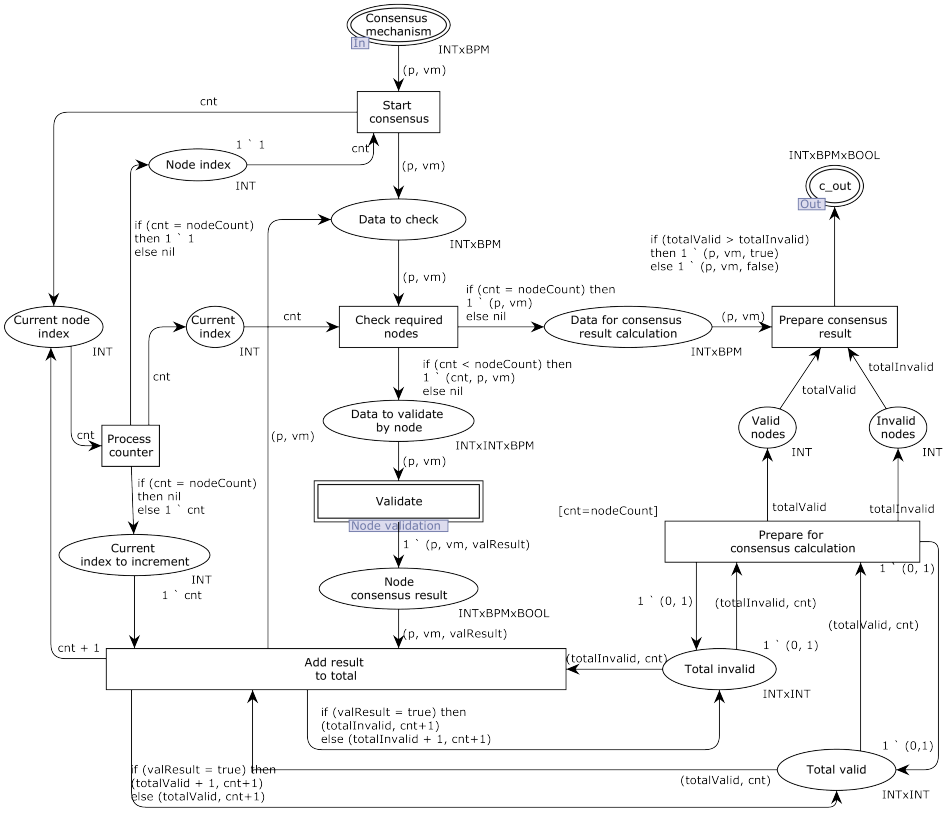

Figure 41. Consensus as a countermeasure model

According to [18, 84], the consensus mechanism in blockchain-based P2P networks may suffer from a Sybil attack. For example, if the system has fewer nodes, the computing power is insufficient, or the system allows to generate the fake nodes. However, this countermeasure model assumes that the current blockchain-based smart healthcare application has a required number of nodes and is easy to scale. As we discussed previously, many other approaches (e.g., fee for creating

the node identity, node validation before joining the network, etc.) can be used to restrict the Sybil attack.

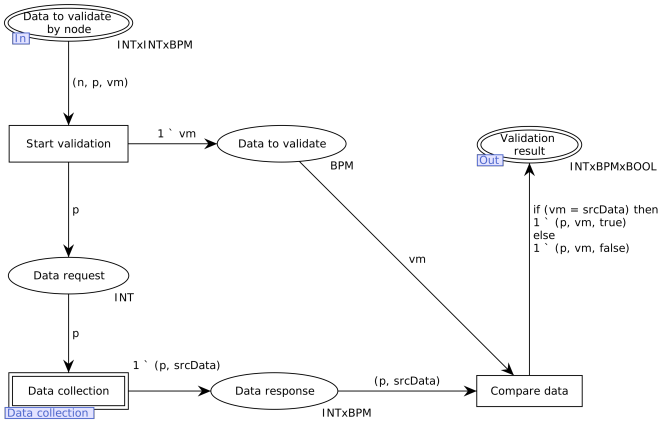

Figure 42. Validation model that performs data validation through a node

Insurance claim model: Insurance claim model (Fig. 43) is a part of the overall CPNs model of smart healthcare application that highlights the results of the insurance claims. For example, the sub-model *Validate claim data* receives the BPM data input from the patient, healthcare provider, and healthcare professional. If the data has been tampered with and the validation process also approved, the system will discard the tampered readings and not process the claim. If the values have not been tampered with, the BPM data is integral and approved by the validation process; then the system processes the insurance claim.

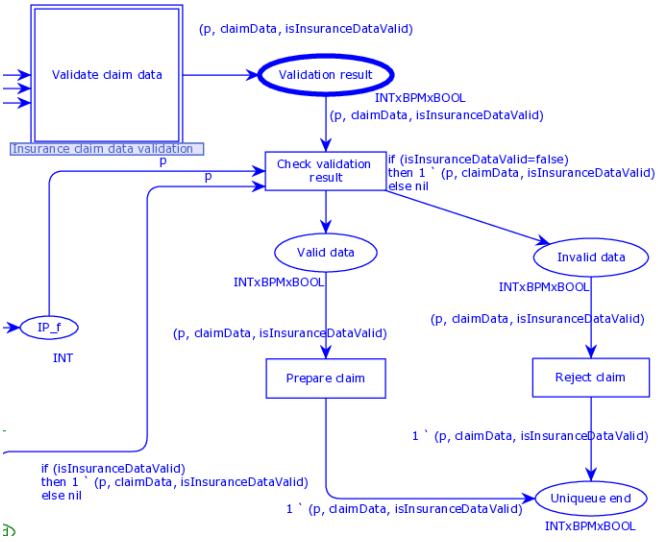

Figure 43. Insurance claim process

CPNs-based evaluation lets us know the ontology's feasibility to incorporate during the system design. The evaluation allows us to better understand the ULRO's

applicability in the context of its appropriateness for a particular domain (e.g., blockchain-based smart healthcare application), task (e.g., SRM of smart healthcare application using blockchain), and application (e.g., smart healthcare application). In addition, it helps to check the behavior and completeness of the ULRO because of formal verification of concepts and properties. For example, the concepts and properties are evaluated using the CPN tools by formalizing the business processes of a smart healthcare application. Based on the CPNs-based evaluation, we can conclude that the correct ontology is being constructed for the SRM of traditional and blockchain-based applications and that it fulfills its intended purpose. However, the CPNs are not a panacea because the use of CPNs does not address how to carry out the necessary initial domain analysis for building the ontology or the correctness of the constructs and concepts we used in the ULRO. Moreover, burdensome to encode the technical aspects that are required to incorporate within the system, for example, the technicality of a particular countermeasure for a particular vulnerability.

We perform the **state-space analysis** to investigate and fix the behavioral properties and correct the internal state of the CPNs models. The state-space analysis describes the system's behavior, such as the absence of deadlocks, the possibility of always being able to reach a given state, and the guaranteed delivery of a given service [178]. The basic idea underlying state spaces is to compute all reachable states and state changes of the CPNs models and represent these as a directed graph where nodes represent states and arcs are occurring events. While performing the state-space analysis, using the CPNs models for all the processes were used to calculate the reachable states and state transitions. As we have limited processes that prevented state explosion, it was feasible to perform a full computational verification of the defined CPNs models. After state calculation, a direct graph is presented, and it contains nodes corresponding to the set of all reachable markings, whereas arcs are linked to the occurring binding elements [179]. This graph helps to derive the properties of the CPNs model and system. To perform the state-space analysis, we use the pre-built functionalities of CPN tools. Based on the directed graph, we also calculated the strongly connected components graph, which showed that there are 30547 nodes and 94298 arcs.

Table 33. Findings of state-space analysis

Loop	Home marking	Dead marking	Dead transition	Live transition
No	30547	30547	7	No

As shown in Table 33, there were no loops that represent the absence of infinite occurrences of the execution path and ensure the completion of the module. Also, from the result of the state-space analysis, it can be seen there are home markings. A Home marking is a marking that can be reached from any other reachable marking, which means that it is not possible to have a sequence of occurrences that cannot be extended to reach the home marking. Dead markings are detected in the analysis results, and the cause is either customized input values or purposely

deactivating some parts of the CPNs models that prevent a state-space explosion. The presence of dead marking ensures the completion/termination of executable action at a specified point which prevents infinite runtime. As the results indicate the presence of dead marking, it is certain that there was no live transition. Live transition means we can always find an occurrence sequence that contains a transition from any reachable marking [95]. However, there are no dead transitions. A transition is dead if there is no reachable marking that allows the transition.

4.4. Related Work

Several ontology representations have been developed for dealing with information security and the SRM. For example, Herzoget al. [62] presents the publicly available OWL-based ontology SecOnt which models the assets, security threats, vulnerabilities, and countermeasures. These concepts are instantiated to provide the domain vocabulary of information security. Gao et al. [59] create an ontology related to the network and computer attacks for security assessment. The ontology follows the same constructs of SecOnt [62]. Unlike ULRO, both ontology representations have no categorization of system and business assets, and security criteria are not defined on the business assets. However, it uses the security goal and defense strategy, which are less refined concepts and do not specify the security requirements of business assets. In both ontology representations, countermeasures protect the security goal, but it is unclear which countermeasures mitigate which vulnerabilities or security threats. Moreover, they do not follow the upper-level ontology or any SRM domain model; therefore, the defined relationships are not reliable for performing the SRM.

Fenz et al. [55] performed the analysis of various SRM approaches and extracted the concepts and relations from them. The main objective of this work was to establish unified and formal knowledge models of the information security domain for supporting and enhancing existing risk management approaches. Similarly, this ontology does not provide assets categorization. The vulnerabilities classification is not provided, e.g., which threat exploits which vulnerability and the threats are confused with security and safety categories. This ontology follows the national institute of standards and technology (NIST) special publication "800-12: An Introduction to Information Security" and other analyzed SRM approaches that make it unnecessarily difficult to interpret.

Obrst et al. [180] present the Cyber ontology that mainly focuses on malware, their attack methods, and outcomes. This ontology presents the upper-level and domain ontology together to address the malware and its attack methods. However, a core of SRM is missing; for example, assets, vulnerabilities, and countermeasures are not modeled. Mozzaquatro et al. [53] present IoTSec, a reference ontology combining IoT and security concepts. The study followed a systematic literature review and MENTOR methodology to gather and harmonize knowledge of threats and vulnerabilities. The ontology does not specify the business assets

and their security criteria explicitly. The IoTSec uses the term security mechanism, which refers to control, countermeasures, and safeguard techniques and tools. But in ontology, the concept is unclear, and the relation with security threats and vulnerabilities is not modeled. Moreira et al. [181] present the methodology to create the security ontologies for information security management and governance. The study presents CoreOnt (includes countermeasures), OntoSec (includes assets, threats, and consequences), and OntoVul (includes vulnerabilities) with the concept of easily extending and reusing. However, the constructs of the ontology do not follow the principles of SRM, and the relationships of different concepts are not well defined (e.g., threats, vulnerabilities, and countermeasures).

Overall, the related works mentioned earlier rely primarily on identifying the security threats and vulnerabilities and do not fully cover the fundamental principles of SRM because they fail to report on the principles of SRM. The relationships between the classifications related to the assets, security criteria, threats, vulnerabilities, and countermeasures are not explicitly defined. For instance, we present in the instantiation of ULRO where each corresponding threat has a vulnerability, and each vulnerability has a corresponding countermeasure. In contrast, we utilize the SRM domain model constructs to create the ULRO. We report on SRM principles and classify the system assets to be secured, business assets, their security criteria, threats, vulnerabilities, and countermeasures. Moreover, the ontologies presented in related works are domain-centric and do not use the upper-level ontology. In comparison, the ULRO is a high-level, domain-independent ontology representation and provides a common foundation from which we can derive domain-specific ontology representations.

4.5. Discussion

An upper-level ontology is a top-level ontology that provides a foundational framework including a modular collection of extendable classes and relationships [171, 170]. Such ontology presentation is domain-independent and shares a common knowledge foundation from which more domain-specific ontologies can be created [180]. There are many examples where upper-level ontology representations are provided, for example, unified foundational ontology (UFO), basic formal ontology (BFO), and descriptive ontology for linguistic and cognitive engineering (DOLCE) [170]. The goal of such ontology representations is to provide a uniform semantic infrastructure for information collection, retrieval, analysis, and integration while taking into account common language concerns and instances [170, 182]. Moreover, such ontology representations are considered to facilitate the semantic integration of domain ontologies. Following a similar context, we created the ULRO to provide a semantic and foundational framework, including the fundamental concepts of SRM. The ULRO presents a taxonomical representation that can support the dynamic knowledge encoding and instantiation with information security knowledge for the SRM of domain-specific applications.

Here, we want to emphasize the usage of UFO that can provide theoretical foundations to the conceptual constructs and relationships of our ontology [177, 183, 184]. For example, the UFO can align the concepts of the SRM domain model and provide conceptual foundations to elucidate theoretical details, allowing for a better understanding of the semantic meaning and interoperability of the SRM concepts. Despite the importance of UFO, we could not employ it in this work since the main focus of this study is to abstract the fundamental concepts of SRM from the SRM domain model and to develop an ontology-based reference framework for managing security risks of traditional and blockchain-based applications. Undoubtedly, integrating UFO in the ULRO will add conceptual clarity and semantic explicitation to the SRM concepts and relationships. For instance, Adach et al. [184] built a combined security ontology, and Duarte et al. [177] built the ontology for the software system anomalies and their associated risks using the UFO. However, the concepts used do not rely concretely on UFOs and their stated theoretical aspects, but they can serve as a starting point for expanding this work in the direction of incorporating UFOs into our ontology.

In this work, we constructed the ULRO using the SRM domain model. The SRM domain model fulfills the ISO/IEC 27001 information security standard [48]. As a result, the ULRO offers a solid foundation for the SRM, and it can be instantiated using information security knowledge for any information system domain. Furthermore, the instantiation of a smart healthcare application demonstrates that the ULRO is feasible to encode the knowledge for blockchain-based applications' SRM. We examined the ULRO for the SRM of traditional and blockchain-based apps using two distinct approaches (e.g., expert discussion and CPN tools) to assess the correctness and practicality of the ULRO. Our work has several limitations. For instance, the ULRO has been evaluated only with the three-domain experts. Also, the risk of subjective interpretation when analyzing the experts' discussion data. While a limited number of experts gave results quickly, we cannot draw firm conclusions regarding the dependability, precision, and generalizability of our findings. Furthermore, the ULRO is missing the concepts related to a threat's likelihood and impact analysis. For example, these are essential considerations in the risk assessment process. Another limitation, as previously stated, is that the ULRO is not founded on fundamental top-level ontology (e.g., UFO). We discuss these limitations in our future work.

4.6. Summary

This chapter answers our RQ2: How can SRM domain model abstraction be used to build an ontology-based reference framework? Correspondingly, we present an upper-level reference ontology (ULRO) as a foundation ontology by abstracting the concepts of the SRM from the SRM domain model. For instance, the ULRO defines the fundamental concepts of the SRM, such as assets, security criteria, threats, vulnerabilities, and countermeasures. The ULRO offers a common and

consistent structure for creating an ontology representation in the information security domain. The ULRO is a high-level and domain-independent ontology from which more domain-specific ontology representations may be derived. We use the Protégé editor to code the concepts and relationships of the ULRO using the OWL to build a readable semantic infrastructure that supports the triplet format (e.g., subject-predicate-object) for describing the SRM concepts.

We evaluate the ULRO using the experts to illustrate that the ULRO concepts and relationships adequately cover the aspects they aim to represent for dealing with the SRM. In addition, we use CPNs-based evaluation to provide insights into the system's actual properties and workflow, considering the blockchain as a countermeasure solution. To perform the CPNs-based evaluation, we used the smart healthcare application case for the SRM using blockchain and instantiated the ULRO with the data tampering threat and Sybil attack. Overall, CPNs-based evaluation lets us know the feasibility and applicability of the ULRO to incorporate during the system design and explains the execution logic of the ULRO for the SRM from the application point of view by characterizing its structural and behavioral properties. Also, it helps to check the completeness of the ULRO because of the formal verification of concepts and properties.

5. PERMISSIONED BLOCKCHAIN-BASED SECURITY ONTOLOGY

The advent of blockchain technology introduces new concepts to revolutionize the financial industry because the constant security threats challenge the centralized infrastructure [185, 4]. For example, the attackers stole $81 million by exploiting the SWIFT credentials [5], and increasing data breaches in the financial industry [6, 7]. To address such security challenges of the financial industry, permissioned blockchains are transpiring that can provide privacy-oriented secure infrastructure to deal with financial transactions securely. The objective of this chapter is to build a permissioned blockchain-based security ontology to answer our third research question, RQ3: How might permissioned blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissioned blockchain-based applications? To address this research question, we perform the security risk analysis of post-trade matching and confirmation and present a permissioned blockchain-based Corda platform as a countermeasure solution. We applied this security risk analysis to the ULRO, resulting in a Corda-based security ontology (CordaSecOnt). CordaSecOnt provides extensible knowledge for the SRM of post-trade matching and confirmation, including the notions of blockchain and Corda. The CordaSecOnt also enables unified and formal knowledge models to support the comprehension and communication of the security threats within the Corda-based applications (CorDapps).

The remainder of the chapter is structured as follows: Section 5.1 discusses the capital market's post-trade matching and confirmation in the context of centralized and decentralized infrastructure. The section describes the business process models of centralized and decentralized infrastructure to explain the interaction of different parties and architectures to elaborate the components of the systems. Section 5.2 includes the security risk analysis of post-trade matching and confirmation and collects the security threats that can be mitigated using CorDapp and the security threats that may appear in the CorDapp. Following the results of Section 5.2, in Section 5.3 we present the Corda-based security ontology, CordaSecOnt. This section presents and illustrates the classifications related to the systems assets, business assets, security criteria, threats, vulnerabilities, and countermeasures for elaborating the encoded knowledge. In Section 5.4, we evaluate the CordaSecOnt using three different approaches. Section 5.5 discusses the related work. Section 5.7 is the discussion, and Section 5.8 concludes this chapter.

5.1. Capital Market Post-Trade Matching and Confirmation

Post-trade is a component of the capital market's value chain that occurs after a trade is completed [186]. At this stage, the investors examine trade facts, ratify the transaction, change ownership records, and arrange the transfer of financial instru-

ments [17]. The participants of the capital market's post-trade are investors (e.g., private individuals, investment banks, insurance companies, etc.), financial organizations (e.g., banks) as custodians, clearinghouses, regulators, and exchanges (e.g., public marketplace). In this section, to learn how the Corda transforms the existing centralized infrastructure and business processes in the capital market, we present the context of the capital market's post-trade matching and confirmation in centralized and decentralized infrastructure.

Centralized infrastructure: A capital market is a part of the financial industry and processes financial instruments (e.g., securities, futures, and other assets). Mainly, the trading system has three modules: front-office, middle-office, and back-office [187]. The front-office is in charge of trade execution and accommodates trading operations and direct interactions with buyers, sellers, and exchange services. Following that, several post-trade activities are carried out and processed through the middle-office and back-office, involving third parties (e.g., regulators and clearinghouses) to ensure that trades are valid and necessary trade information is provided in accordance with regulatory requirements [188]. For example, the important action in the middle-office is the trade matching and confirmation (Fig. 44) process between counter-parties (e.g., Bank X is a buyer and Bank Y is a seller) [189]. These operations ensure that trade details and payment information are correct between involved counter-parties. For example, the financial organization (e.g., Bank X) performs trade matching with Bank Y after receiving the trade details from the front-office.

Once trade matching is finished, the details are passed to the investor, who possesses information regarding trade specifics, agreements, and funds transfer. The trade confirmation happens when each counter-party accepts and agrees to the trade details. The trade confirmation carries out various activities such as clearing instructions and payment operations. In a centralized infrastructure, trade matching is performed manually, so there are high chances of security risks. Back-office is also responsible for trade reporting to respective authorities (e.g., regulators), processing the settlement procedures, and payment executions. The authorities can levy severe penalties and sanctions if financial organizations fail to report trades correctly. Thus, the third parties ensure that the trade is genuine and relevant information is given to avoid such mistakes.

Post-trade matching and confirmation architecture in centralized infrastructure (Fig. 45) presents an abstraction of the system components that are organized mainly in three layers. The Presentation layer exposes the interaction interface where users interact with the application. The user interacts with the services and performs different operations, which are present in an Application layer. There are three modules (e.g., front-office, middle-office, and back-office) responsible for interactions and support the trade life cycle. The engines are in charge of interacting with the front-, middle-, and back-office modules. The SWIFT interface is used to engage with the SWIFT network, which enables the exchange of informa-

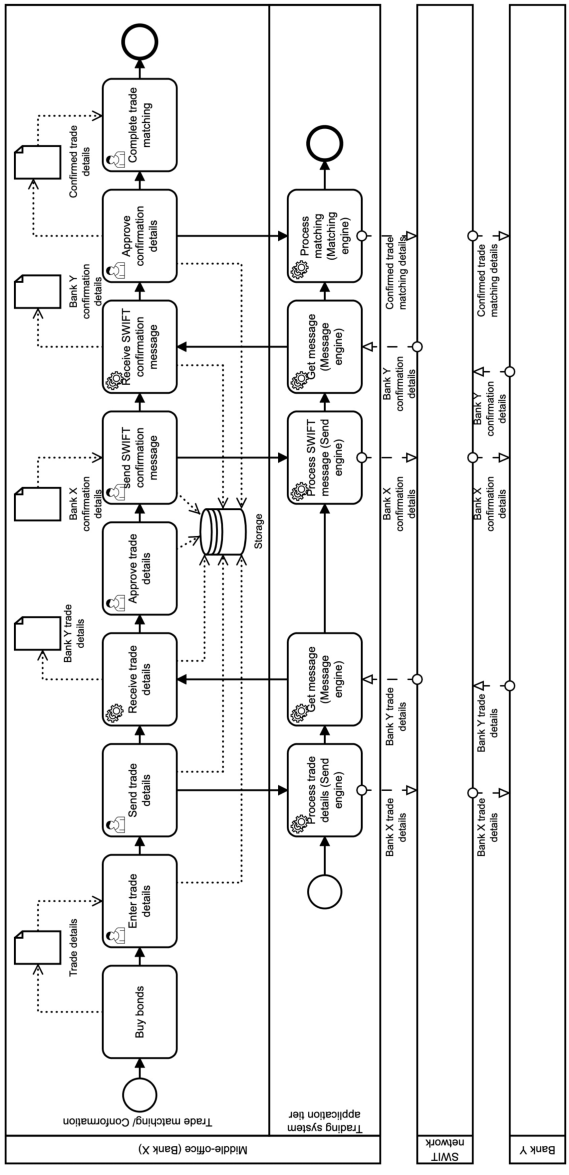

Figure 44. Post-trade matching and confirmation in centralized infrastructure

tion between counter-parties. The message bus is responsible for handling events, cache, data services, and transferring relevant information to particular system modules. The data server is responsible for executing and processing information from the database, whereas the engine server is in charge of custom and standard engine interactions. The Data layer combines the database as persistent storage. Also, it includes access rights details, trade details (e.g., reports, documents), and logs (e.g., access, error, activity logs).

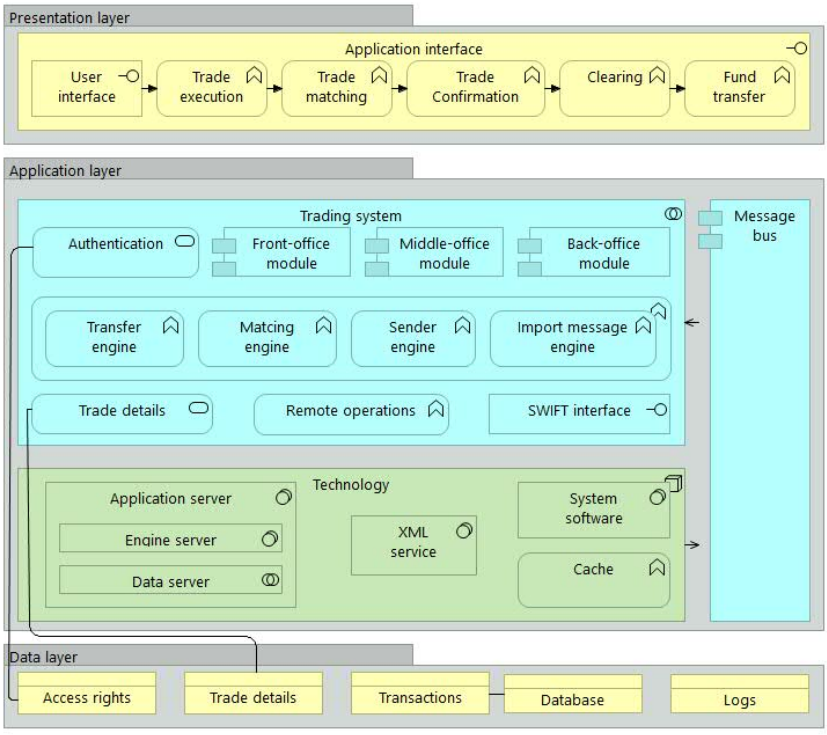

Figure 45. Centralized infrastructure of post-trade matching and confirmation

Decentralized infrastructure: The business process model of CorDapp (Fig. 46) enables blockchain-based states, contracts, flows, and a vault to interact with counter-parties to perform the assets exchange [189]. The counter-parties (e.g., Bank X & Y) perform the operations without relying on the manual operations and trusted third parties. The CorDapp performs the validation of a transaction by a notaries-based consensus [42]. The counter-parties receive validated data from a distributed Corda ledger over a P2P network. The involved counter-parties provide necessary details of the transaction by Corda node to CorDapp that validates and completes the post-trade matching and confirmation. The process increases efficiency and reduces the risks of manual operations.

As compared to the centralized post-trade matching and confirmation architecture, the CorDapp will change the existing system components according to the Corda node architecture, how nodes interact within the Corda network, and process information exchange. The architecture (Fig. 45) is extended to (Fig. 47) by integrating the Corda platform. Similarly, the architecture (Fig. 47) has three different layers along with the auxiliary components (e.g., client, node network). CorDapp's counter-parties establish an agreement on ledger changes by specifying flows that Corda node owners can activate through the remote procedure call (RPC) interface. In a post-trade matching and confirmation CorDapp, the Corda node contains the service hub, notaries, vault, messaging, flow state machine man-

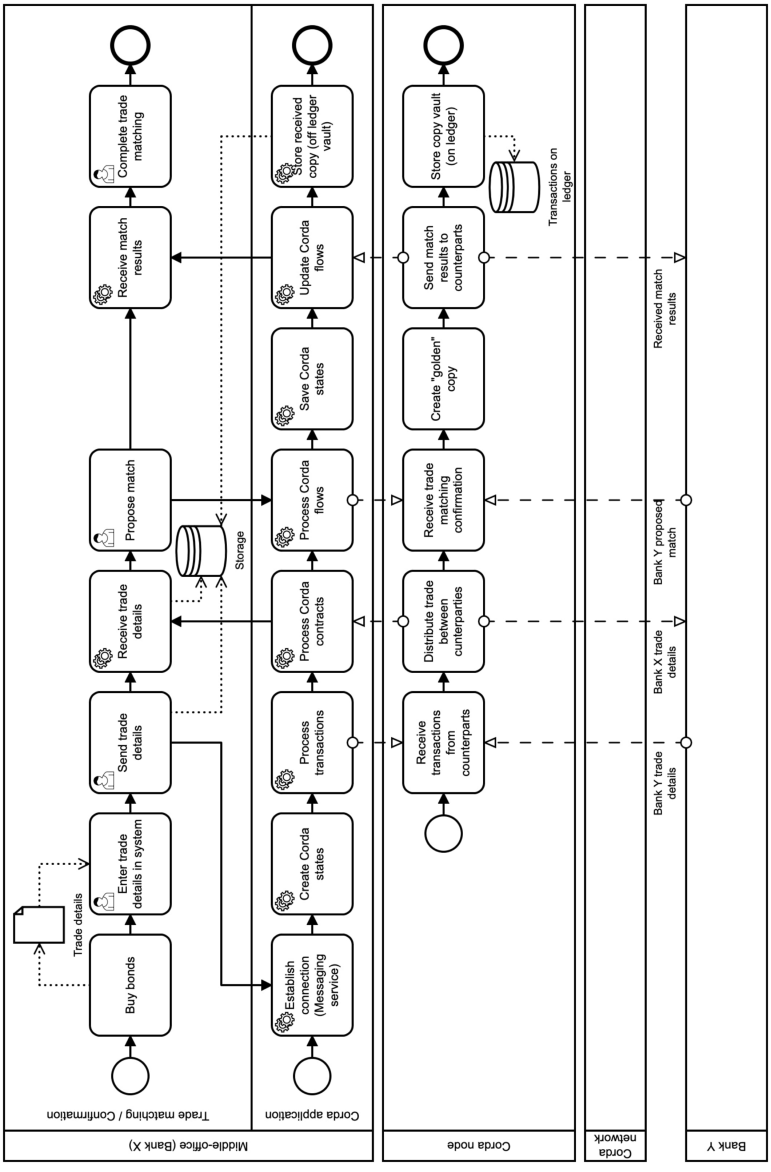

Figure 46. Post-trade matching and confirmation in CorDapp

ager (Flow SMM), identity management, permission settings, and storage service to store data in a permanent database. Furthermore, Corda supports Java programming, SQL databases, and the integration of Corda in a present as-is post-trade architecture. Also, Corda and SWIFT are collaborating to provide smooth payment processing between financial organizations and other stakeholders [187, 42].

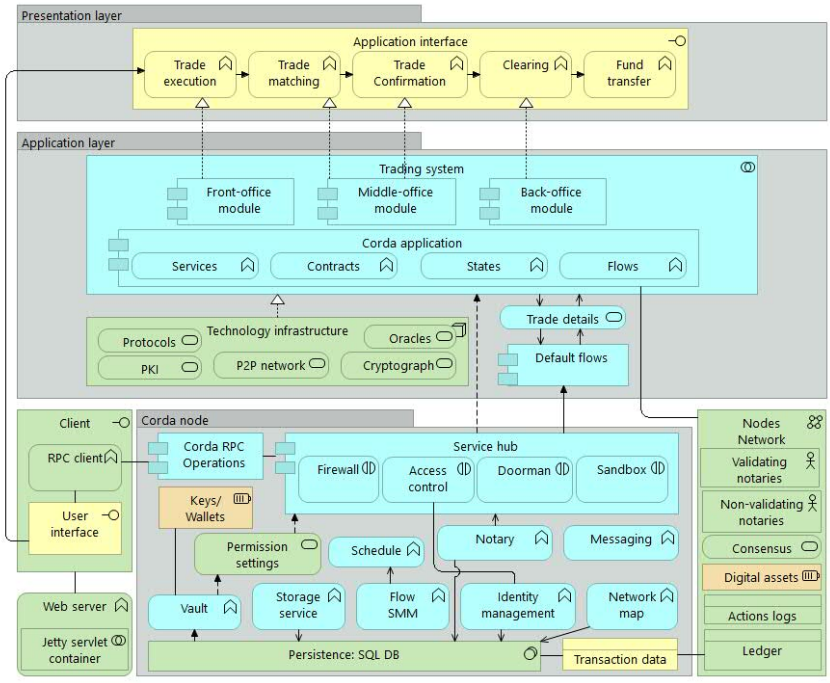

Figure 47. CorDapp-based infrastructure of post-trade matching and confirmation

5.2. Security Risk Analysis of Post-Trade Matching and Confirmation

In this section, we undertake a security risk analysis of post-trade matching and confirmation to determine which security threats can be reduced using the CorDapp and which security threats may surface in CorDapp.

5.2.1. CorDapp to Mitigate Security Threats

We use the STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privileges) threat model [21] that supports a systematic analysis to identify and explain the potential security threats of capital market post-trade matching and confirmation. STRIDE also supports the security risk analysis activity of the SRM domain model, and based on it; we identify the business assets to protect, their security criteria, system assets that support the business assets, and what security threats target system assets by exploiting various vulnerabilities in a centralized infrastructure of post-trade matching and confirmation (Table 34). Furthermore, we illustrate the CorDapp-based countermeasures to mitigate the security threats (Table 35).

The V#1 belongs to missing, or no proper implementation of communication protocols that the attacker can exploit to trigger a spoofing threat [190, 191]. The insecure transmission of data (V#2) can be exploited by spoofing or tampering

Table 34. Vulnerabilities that can be exploited in post-trade matching and confirmation

	Vulnerability	System asset	Business asset	Threat
V#1	Missing communication protocol	Import message engine, Sender engine	Import message (I)	*S*
V#2	Insecure transmission of data	Import message engine, Sender engine	Import message (I), Trade data (I)	*S, T*
V#3	Error-prone validation of data	Trade matching, Trade confirmation, Database	Trade data (I), Trade detail (I)	*T*
V#4	Missing access control	Import message engine, Sender engine, Trade detail	Import message (I), Trade data (I, C)	*S, T, R*
V#5	Ineffective logging	Trade detail, Logs	Trade data (I, C)	*R, Id*
V#6	Missing requests filtering	Import message engine, Sender engine, Server, Database	Trade matching (A), Trade confirmation (A)	*D*
V#7	Unauthorized remote operation	Access right, Remote operation, Trade execution	Trading (I), Fund transfer (I)	*E*

threats [5, 192]. In current post-trade matching and confirmation infrastructure, the error-prone validation of data (V#3) [4, 193] due to centralized controls and data validation rules. Thus, the attacker can enable the tampering threat. The poorly implemented or having no access control (V#4) could enable spoofing, tampering, and repudiation threats [192, 193]. The ineffective logging (V#5) [192, 193] can happen because of insignificant log messages, logging sensitive information, unprotected logs, centralized control on logs, or having no backup of logs. The attacker exploits this to trigger repudiation and information disclosure threats. The attacker targets the system with a large number of requests to deny the services of post-trade matching and confirmation to the legit users. If a system has no proper mechanism to filter a large number of requests (V#6) [194, 192] the attacker can successfully trigger this threat. Due to centralized access control and various dependency components, the attacker can get elevated privileges in the system to execute unauthorized remote operations (V#7) [192, 193]

We present CorDapp as a countermeasure solution (Fig. 47 & Table 35) to mitigate the security threats and protect the assets. The architecture (Fig. 47) illustrates the assets of post-trade matching and confirmation and presents the CorDapp countermeasures (CC). For example, to mitigate V#1, CorDapp considers authorized nodes over a P2P network (CC#1) where nodes behave both as client and server [116]. Also, the CorDapp incorporates a mutually authenticated TLS connection (CC#2) [195] to protect the communication between nodes. The insecure transmission of data can lead to spoofing and tampering threats (V#2) [5, 192]. The malicious user may intercept the plain-text data transmission and gain unauthorized access to the system. It would negate the confidentiality and integrity of data. The CorDapp mitigates by incorporating PKI-based cryptography combined with Intel SGX (CC#2) [195] that brings the CPU-based P2P encryption and allows one to encrypt the entire ledger [42]. Also, a hardware security module (HSM) (CC#2) can be applied to manage and protect digital keys [195].

To mitigate V#3, the transaction validation mechanism is utilized. The Corda platform uses a notaries-based decentralized consensus (CC#3) to validate a transaction and ensure authenticity and integrity [195, 42]. In a centralized approach,

Table 35. Corda-based countermeasures to mitigate security threats

	Countermeasure	**Mitigate**
CC#1	Distributed P2P network and authorized nodes	*V#1, V#4, V#7*
CC#2	Mutually-authenticated TLS connection, PKI, and HSM	*V#2, V#6*
CC#3	Consensus mechanism	*V#3*
CC#4	Decentralized access control	*V#4*
CC#5	Distributed immutable ledger	*V#5*
CC#6	Firewall	*V#6*
CC#7	JVM sandbox	*V#7*

the attacker can trigger a man-in-the-middle (MitM) attack and modify the transaction [196]. Similarly, in CorDapp, the attacker can perform MitM to modify the transaction, but the notaries-based consensus protects and guarantees the integrity of the transaction [195]. The V#4 is mitigated by decentralized access control (CC#4) in CorDapp. Also, only authorized nodes can join the network (CC#1), limiting this vulnerability. Multi-user applications are subject to repudiation because the system allows a user to perform/deny the malformed actions. The system should ensure that the user actions are recorded in order to protect against insider security risks. To mitigate V#5, CorDapp manages the records in a decentralized immutable ledger (CC#5). It provides tamper-proof transparent traceability and auditing. Also, the CorDapp logs each action of a participant node that replicates over a P2P network [195]. The V#6 is mitigated by introducing the requests rate-limiting firewall (CC#6). In CorDapp, the P2P communication is authenticated as a part of the TLS protocol (CC#2), which means the attacker can not join the Corda network to launch a DoS attack [195]. To protect against unauthorized remote operations (V#7), the CorDapp utilizes the secure communication protocols (CC#1) along with the concept of a custom-built JVM sandbox (CC#7) [195] and prevent unauthorized remote operations and execution of code.

5.2.2. CorDapp Security Threats

Similarly to Section 5.2.1, we identify and explore the security threats that may appear within CorDapp, as well as how to mitigate them. First, we identify the business assets, their security criteria, system assets, security threats, and vulnerabilities (Table 36). For example, i) endpoint vulnerability (EV) (such as keys lost, weak passwords, physical access to digital wallets, and devices), ii) quantum computing threat (QCT), iii) privacy violation (PV), iv) deanonymization (DA), v) smart contract attack (SCA), and vi) denial-of-state attack (DSA). Second, we investigated the countermeasures against the security threats of CorDapp.

The security threats belonging to CorDapp post-trade matching and confirmation are present in Table 36 to visualize which vulnerabilities are exploited, and targeted assets. For example, the lack of awareness and knowledge (CV#1) about security could trigger the endpoint vulnerability [197]. For example, if the attacker learns about the private key, he can use it to acquire access and ownership of data. Quantum computing research is advancing and in blockchain, currently employed cryptography techniques are at high risk in a post-quantum era [197]

Table 36. Vulnerabilities that can appear in post-trade matching and confirmation CorDapp

	Vulnerability	System asset	Business asset	Threat
CV#1	Lack of awareness	Digital wallet, Keys, Devices, User	CorDapp service (I, A), Digital asset (I)	*EV*
CV#2	No QC resistant cryptography	Cryptography, Ledger, P2P network	Transaction (I)	*QCT*
CV#3	Sharing content with validating notaries	Notaries, Consensus	Transaction content (C), Customer data (C)	*PV*
CV#4	Linking user account with the transaction ID	Transaction, Validating notary	Counter-parties (C)	*DA*
CV#5	Faulty and error-prone smart contracts	Smart contract, Ledger, Non-validating notary	Transaction (I), Transaction validation (I), Digital asset (I)	*SCA*
CV#6	Non-validating notaries consume state	Smart contract, Ledger, Node, State reference	Transaction state (A), Digital asset (I), Transaction (I),	*DSA*

because current techniques are not quantum-resistant (CV#2). In CorDapp, validating notary observe the full content of the transaction to validate it, and sharing the full content of the transaction with validating notaries (CV#3) could lead to privacy violation [41]. In CorDapp, it is possible to link individuals' data that could trigger deanonymization threat [198, 41]. Error-prone smart contracts [199] (CV#5), for example, a smart contract may have a logical bug, missing error handling, missing input validation, or misuse of programming language constructs. In CorDapp, the malicious actor is able to create a transaction, and a non-validating notary consumes a state [41] by considering it a valid transaction (CV#6).

We collect various countermeasures (C) to overcome the security threats of CorDapp. For example, the lack of knowledge and awareness (CV#1) led the attacker to steal information [200] by social engineering, phishing [194], or the company individual accidentally exposing the secure information [192]. The organizations should educate the system users (C#1) about the possible security threats if exposing their protected information [197]. Moreover, the organizations should arrange staff security training and establish a disciplinary process, promote an incident reporting culture within an organization, and imply user security policies. Also, incorporate hardware security modules (HSM) (e.g., AWS cloud HSM, Azure key vault, Futurex, GemaltoLuna, N-cipher N-shield, Securosys Primus X or Utimaco) to generate, protect, and store keys (C#2) [195, 197].

Table 37. Countermeasures to mitigate security threats of CorDapp

	Countermeasure	Mitigate
C#1	Security awareness and knowledge	*CV#1*
C#2	Hardware security module	*CV#1*
C#3	Quantum-resistant cryptography	*CV#2*
C#4	Transaction tear-off	*CV#3, CV#4*
C#5	Code analyzer	*CV#5*
C#6	Trusted execution environment (TEE) and zero-knowledge proof (ZKP)	*CV#6*

The quantum computing threat is real [201] and CorDapp does not provide any mechanism (CV#2) to tackle this threat in a post-quantum era. The possible way is to implement quantum-resistant cryptography schemes (C#3) (e.g., lattice-

based, multivariate, hash-based, code-based, symmetric key quantum resistance) to secure against quantum computing threats [201, 41]. The study [41] suggested to use transaction tear-off (C#4) to protect against CV#3 and CV#4. For example, this concept within CorDapp would increase privacy because it would tear-off the information and show a minimum amount of information that should be kept confidential from the transaction [199]. Smart contracts deployed on the blockchain are immutable and can not be modified (or fixed). Thus, the lack of exception handling and error-prone smart contracts (CV#5) can lead the attacker to disrupt the services and harm valuable assets. For example, Ethereum smart contract's reentrancy attack when an adversary stole $60 million Ethers [26]. To minimize the risks of the smart contracts, the smart contract's code should be analyzed by using the smart contract's code analyzer (C#5) [26] to detect errors, identify race conditions and sanitize the smart contract code before deployment. Koens et al. [41] suggested using trusted execution environments (e.g., IntelSGX) and zero-knowledge proofs (C#6) to protect against CV#6.

5.3. Corda-based Security Ontology

We present a publicly available, OWL-DL, and Corda-based domain-oriented security ontology CordaSecOnt (Table 38). The CordaSecOnt models the system and business assets, security criteria, threats, vulnerabilities, countermeasures, and their relationships by extending the ULRO. The CordaSecOnt gives reasoning in natural language about the encoded concepts of capital market's post-trade matching and confirmation's information security, Corda platform, and answers the competence questions such as: What assets to secure? Which system asset supports business assets? Which particular threat exploits which vulnerability? Which Corda-based countermeasure mitigates which vulnerability? What threats and vulnerabilities may appear in CorDapp? and What are the countermeasures to mitigate the vulnerabilities that appear within CorDapp?

Table 38. CordaSecOnt resources

Resource	URL
GitHub	`https://github.com/mubashar-iqbal/corda-security-ontology`
MMISW repo.	`https://mmisw.org/ont/~mubashar/CordaSecOnt`

5.3.1. Assets Classification

We identify the assets to secure from security threats; for example, assets have value and need protection against security threats. The assets are classified as business and system assets (Fig. 48). Security criteria constrain business assets, and system assets support business assets. For example, a business asset "ImportMessage" hasConstraint Integrity. System assets "SenderEngine and ImportMessageEngine" support the business asset "ImportMessage". The following statements are an example of DL for relations "hasConstraint" and "supports" belong to the business and system assets respectively.

```
- supports some BusinessAsset
- hasConstraint some SecurityCriteria
```

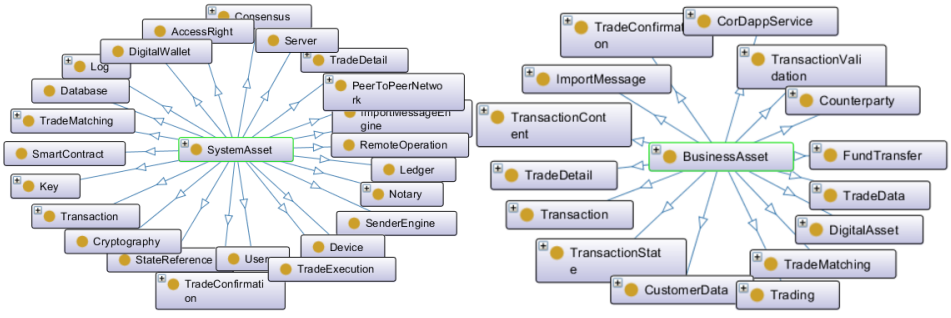

Figure 48. System and business assets classification

System assets, at the same time, can be business assets; it depends on what asset value to protect. For example, system assets "TradeDetail, Transaction, and LogFile" support a business asset "ProcessedTrade". Here, "TradeDetail" is an example of a business asset that is supported by the system assets "Database, SenderEngine, MatchingEngine". The class definition of Asset is:

```
Class (Asset SubClass (
    BusinessAsset SystemAsset
) class BusinessAsset (
    restriction (hasConstraint
        someValuesFrom ( SecurityCriteria )
) class SystemAsset (
    restriction (supports someValuesFrom ( BusinessAsset )
)
```

5.3.2. Security Threats Classification

Security threats classification is built upon the threats that are mitigated and appear within CorDapp (Fig. 55). For example, in the financial industry, security threats related to spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege are mitigated by using a blockchain-based solution (e.g., CorDapp). Some threats may appear in CorDapp (e.g., endpoint, quantum computing, privacy violation, deanonymization, error-prone smart contract, and denial-of-state). Security threats exploit vulnerabilities and target some business asset(s). The DL for relation "exploits" and "targets" is:

```
- exploits some Vulnerability
- targets some BusinessAsset
```

Threat class has subclasses (e.g., Tampering, Repudiation) and a restriction "exploits" on someValuesFrom the Vulnerability. Another restriction "targets" on someValuesFrom the SystemAsset. The someValuesFrom restriction illustrates that a particular threat exploits particular vulnerabilities fully or partially

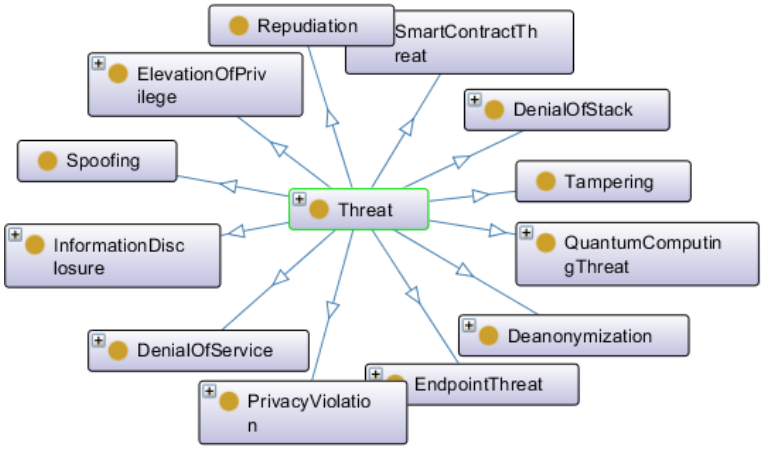

Figure 49. Security threats classification

and targets SystemAsset. For example, in CorDapp, the attacker can create a transaction, and a non-validating notary consumes a state by considering it a valid transaction. The “DenialOfStack” threat exploits a vulnerability “NonValidating-NotariesConsumeState” within CorDapp when non-validating notaries consume the state. The “DenialOfStack” threat targets both “NonValidatingNotary" and “StateReference” SystemAsset.

```
Class (
    Threat SubClass (
        ActiveThreat PassiveThreat
    ) restriction (
        exploits someValuesFrom ( Vulnerability )
    ) restriction (
        targets someValuesFrom ( SystemAsset )
    )
)
```

5.3.3. Vulnerabilities Classification

Vulnerabilities classification (Fig. 56) is built by identifying the weakness within the system that enables a particular security threat. A vulnerability is a characteristic of one or more system assets and negates the security criteria of one or more business assets. The DL for relation “characteristicOf” and “negates” is:

```
- characteristicOf some SystemAsset
- negates some SecurityCriteria
```

The vulnerability class definition explains that it contains various vulnerabilities characteristic of system assets and negates the security criteria of business assets. For example, the missing or improper implementation of access control presents a weakness within a system. The attacker can exploit this vulnerability and get unauthorized access. A vulnerability “MissingAccessControl” is

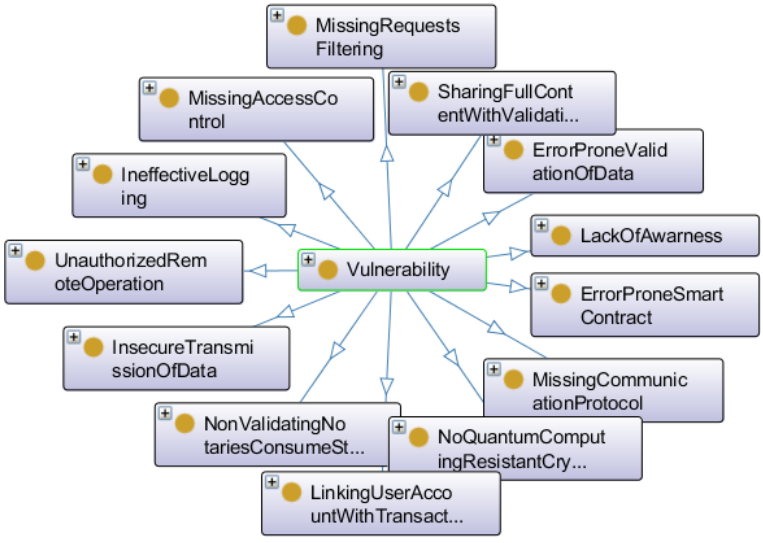

Figure 50. Vulnerabilities classification

a characteristicOf "SystemAsset" (ConfirmationProcess, Database, ImportMessageEngine, LogFile, MatchingEngine, SenderEngine, TradeDetail, TradeMatching, Transaction) and negates some (Availability or Confidentiality or Integrity).

```
Class (
    Vulnerability SubClass (
        MissingAccessControl
        ErrorProneSmartContract
        .....
    ) restriction (
        negates someValuesFrom ( SecurityCriteria )
    ) restriction (
        characteristicOf someValuesFrom ( SystemAsset )
    )
)
```

5.3.4. Countermeasures Classification

Countermeasures classification (Fig. 57) presents the Corda-based counteracts for mitigating the security threats of centralized post-trade, along with the countermeasures for Cordapp's security threats. The DL for relation "mitigates" is:

```
- mitigates some Vulnerability
```

Countermeasure class definition explains that it contains various countermeasures that mitigate the vulnerabilities. For example, countermeasure "AccessControl" mitigates "RestrictRemoteOperation" and "RestrictUnauthorizedAccess" vulnerabilities.

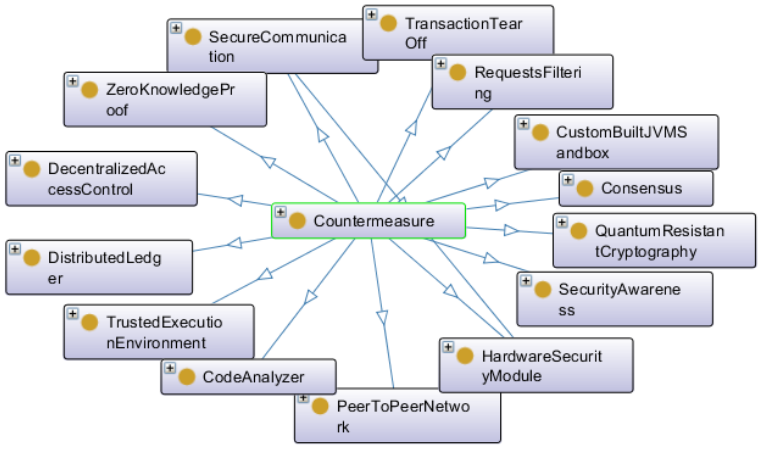

Figure 51. Countermeasures classification

```
Class (
    Countermeasure  SubClass (
        AccessControl
        SecureCommunication
        .....
    ) restriction (
        mitigates someValuesFrom ( Vulnerability )
    )
)
```

5.4. Evaluation

Ontology evaluation is an essential part of ensuring the correctness of ontology, the meaning of ontological reasoning, and the effective use of an ontology representing a knowledge domain [202]. To evaluate CordaSecOnt, we use three different approaches. First, we use the automated tools, e.g., the pellet reasoner we used throughout the development of CordaSecOnt, to check the consistency. Also, OOPS! tool to scan common pitfalls that may appear during the concepts encoding. Second, we evaluate the CordaSecOnt based on the qualitative assessment criteria. Third, we utilize the tasks-based evaluation where we define the SPARQL queries corresponding to each defined task to request data from CordaSecOnt to verify the retrieved results are correct and based on the defined relationships, thus fulfilling the task. Such evaluations may also bring the conception to determine the correctness of encoded concepts for organizational fitness.

5.4.1. Tools-based Evaluation

Throughout the CordaSecOnt development process, we employ the pellet reasoner available in Protégé to ensure that stated concepts, their relationships, data, and

object properties are consistent. It also serves as a proof-of-concept solution for meeting the W3C's OWL implementation criteria [203]. OOPS! is an OntOlogy Pitfall Scanner! available online[1] that evaluates the ontology to detect the common pitfalls appearing when developing it [204]. OOPS! provides a catalog of 41 pitfalls that can be classified into three different categories [205]. For instance, consistency pitfalls to check inconsistencies in defined concepts, their relationships, and restrictions, completeness pitfalls to check if any missing elements in the ontology, and conciseness pitfalls that check for irrelevant or redundant elements. We upload the CordaSecOnt to the OOPS! in RDF format and evaluate it based on the catalog of 41 pitfalls. The evaluation report (Fig. 52) presents only the six pitfalls that existed in the CordaSecOnt that are labeled as critical, important, and minor. We utilize the ULRO that was tested with domain experts and the pellet reasoner to verify CordaSecOnt regularly during the development process. Therefore, several pitfalls have already been identified and fixed. The critical pitfalls should be fixed; otherwise, issues may arise in ontology consistency, reasoning, and applicability [204]. Important ones do not affect the ontology function but can create confusion in understanding. Therefore, it is important to fix all such pitfalls. Minor pitfalls do not create any real problem but correcting them makes ontology nicer. Following the OOPS! evaluation report, we fixed all the detected pitfalls following the OOPS! catalog guidelines.

Results for P07: Merging different concepts in the same class.	1 case | Minor
Results for P08: Missing annotations.	11 cases | Minor
Results for P13: Inverse relationships not explicitly declared.	8 cases | Minor
Results for P19: Defining multiple domains or ranges in properties.	1 case | Critical
Results for P24: Using recursive definitions.	1 case | Important
Results for P41: No license declared.	ontology* | Important

Figure 52. Ontology pitfall scanner evaluation report

5.4.2. Qualitative Assessment

We use the qualitative assessment criteria of [206, 207] (Table 39) to evaluate the CordaSecOnt. This approach helps in the early phase of the CordaSecOnt development process to check whether the coded concepts model the real-world domain knowledge for which the ontology was created.

5.4.3. Tasks-based Evaluation

Furthermore, we follow the tasks-based evaluation approach for CordaSecOnt to determine whether the encoded concepts and relationships have been encoded as intended and resulted in certain outputs [208]. This approach may also help us evaluate CordaSecOnt periodically once it updates the new concepts. To perform this evaluation, we created a list of tasks in Table 40, where each task covers

[1] https://oops.linkeddata.es/catalogue.jsp

Table 39. CordaSecOnt evaluation based on the qualitative assessment criteria

Criteria	Detail
Accuracy	We utilize the scientific literature to define classes, properties, and individuals. The relationships are derived from the SRM domain model, and also the CordaSecOnt is based on the ULRO that is well tested.
Adaptability	CordaSecOnt provides a conceptual foundation for a range of anticipated tasks (e.g., assets, threats, vulnerabilities, and countermeasures)
Clarity	Definitions related to CordaSecOnt concepts and relationships are documented. For instance, we use the annotation properties to explain all the concepts.
Completeness	BbRM and the URLO enable the richness and granularity of CordaSecOnt. We also use the OOPS! to ensure that there are no missing elements in the CordaSecOnt.
Computational efficiency	We use the Pellet reasoner to process the CordaSecOnt, and SPARQL for querying results. Pellet reasoner and SPARQL are fast and computationally efficient tools.
Conciseness	CordaSecOnt includes only the essential terms, and we use the OOPS! to check for irrelevant or redundant elements that may exist in the CordaSecOnt.
Consistency	We use the Pellet reasoner to check CordaSecOnt consistency and avoid contradictions in ontology concepts, relationships, and restrictions. Moreover, the OOPS! helps to maintain and report any inconsistencies in the CordaSecOnt.
Organizational fitness	We follow the well-defined ontology construction method. CordaSecOnt is available online, and it can be extended, reused, or integrated with other security ontologies.

different aspects of CordaSecOnt to get answers from it. We use the SPARQL queries to retrieve information from the CordaSecOnt to fulfill the tasks. The following header code will remain the same for all the SPARQL queries. The PREFIX rdf defines the RDF schema for the RDF vocabulary terms, PREFIX owl contains built-in classes and properties that together form the basis of the RDF/XML syntax of OWL, PREFIX rdfs is the RDF schema vocabulary, and PREFIX xsd describes the XML Schema namespace.

Table 40. List of tasks to perform on CordaSecOnt using SPARQL queries

Id	Task
Task 1	Select a list of systems assets that support business assets.
Task 2	Select a list of business assets that have security criteria constraints.
Task 3	Select a list of security threats and vulnerabilities that may appear in traditional post-trade matching and confirmation.
Task 4	Select a list of security threats and vulnerabilities that may appear within the CorDapp.
Task 5	Select a list of countermeasures that mitigate the vulnerabilities.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX CordaSecOnt:
        <https://mmisw.org/ont/~mubashar/CordaSecOnt#>
```

System assets: The following SPARQL query retrieves the system assets that support the business assets and successfully fulfills Task 1. The property related to the rdfs:subClassOf (CordaSecOnt:SystemAsset) has a relationship owl:onProperty (CordaSecOnt:supports) to state and fetch all the system assets that support the business assets. For example, one of the results of this query is *ImportMessageEngine supports ImportMessage*.

```
SELECT DISTINCT ?SystemAsset ?BusinessAsset
WHERE {
    ?SystemAsset rdfs:subClassOf CordaSecOnt:SystemAsset .
    ?SystemAsset rdfs:subClassOf ?BusinessAsset .
    ?BusinessAsset owl:onProperty CordaSecOnt:supports .
}
```

Business assets: The following SPARQL query gets the business assets that have security criteria constraints (confidentiality, integrity, or availability). We use the hasConstraint relationship to aggregate the results of this query to fulfill Task 2. One of the results of this query is *ImportMessage hasConstraint Integrity*.

```
SELECT DISTINCT ?BusinessAsset ?Constraint
WHERE {
    ?BusinessAsset rdfs:subClassOf CordaSecOnt:BusinessAsset .
    ?BusinessAsset rdfs:subClassOf ?Constraint .
    ?Constraint owl:onProperty CordaSecOnt:hasConstraint .
    { ?Constraint owl:someValuesFrom
                CordaSecOnt:Confidentiality . } UNION
    { ?Constraint owl:someValuesFrom
                CordaSecOnt:Integrity . } UNION
    { ?Constraint owl:someValuesFrom
                CordaSecOnt:Availability . }
}
```

Threats of traditional post-trade matching and confirmation: The following SPARQL query brings the threats, and vulnerabilities of traditional post-trade matching and confirmation that are mitigated by using the CorDapp. The query also brings what system assets are targeted, thus fulfilling Task 3. The exploits relationship fetches the vulnerabilities, and the target's relationship fetches the system assets that are targeted by the threats. We use the rdfs:seeAlso annotation property to filter the threats that are mitigated using the CorDapp. One of the results of this query is *Tampering exploits MissingAccessControl targets Database*.

```
SELECT DISTINCT ?Threat ?Vulnerability ?SystemAsset
WHERE {
    ?Threat rdfs:subClassOf ?Vulnerability .
    ?Threat rdfs:subClassOf ?SystemAsset .
    ?Vulnerability owl:onProperty CordaSecOnt:exploits .
    ?SystemAsset owl:onProperty CordaSecOnt:targets .
    ?Threat rdfs:seeAlso ?Domain .
    FILTER regex(?Domain, "^Mitigated")
}
```

Threats appear in CorDapp post-trade matching and confirmation: The following SPARQL query brings the threats that appear within the CorDapp and fulfills Task 4. The query results show the threats that appear in the CorDapp, vulnerabilities, and system assets that are targeted by the threats. Similarly, we use the rdfs:seeAlso annotation property to filter the threats that can appear in the

CorDapp. One of the results of this query is *DenialOfStack exploits NonValidatingNotariesConsumeState targets NonValidatingNotary*.

```
SELECT DISTINCT ?Threat ?Vulnerability ?SystemAsset
WHERE {
    ?Threat rdfs:subClassOf ?Vulnerability .
    ?Threat rdfs:subClassOf ?SystemAsset .
    ?Vulnerability owl:onProperty CordaSecOnt:exploits .
    ?SystemAsset owl:onProperty CordaSecOnt:targets .
    ?Threat rdfs:seeAlso ?Domain .
    FILTER regex(?Domain, "^Appeared")
}
```

CorDapp countermeasures: The following SPARQL query brings a list of countermeasures to mitigate the vulnerabilities of traditional and Corda-based post-trade matching and confirmation. The query fulfills Task 5, and one of the results of this query is *DistributedLedger mitigates IneffectiveLogging*.

```
SELECT DISTINCT ?Countermeasure ?Mitigates
WHERE {
    ?Countermeasure rdfs:subClassOf
                            CordaSecOnt:Countermeasure .
    ?Countermeasure rdfs:subClassOf ?Mitigates .
    ?Mitigates owl:onProperty CordaSecOnt:mitigates .
}
```

5.5. Related Work

In the beginning, we explore various literature studies that build ontology representations to formalize and structure the domain of information security (Table 41). For example, the authors [62, 55] present OWL-based information security ontology that includes the ontology hierarchy and reasoning capabilities by classifying assets, threats, vulnerabilities, and countermeasures in taxonomic structures. The ontology is created using the Protégé OWL tool and SPARQL to retrieve data from the ontology. Security requirements are difficult to elicit, analyze, and manage [209]. Souag et al. [209] use the ontological approach to facilitate the security requirements elicitation process. The ontology uses the concepts of organization dimensions (e.g., assets), risk dimensions (e.g., threats, vulnerabilities), and treatment dimensions (e.g., countermeasures). The ontology uses OWL, and Semantic Query-Enhanced Web Rule Language (SQWRL) to query data from the ontology. The authors performed the controlled experiment to demonstrate that the ontology helps in the security requirements eliciting process.

Substantial research has been conducted to formalize and structure the knowledge of information security. The above-related works represent examples of generic information security ontologies, there also exist domain-specific ontologies to structure knowledge of information security for a specific domain. For

Table 41. Summary of related work that perform the SRM by using security ontologies

Paper reference	Domain	Findings and addressed topics					
		Threats/Risks	Counter-measures	Blockchain as counter-measure	Business/ System Assets	Vulnerabilities	Use of SRM model
Herzog et al. (2007)	Information security ontology	◉	◉	○	○ / ○	◎	○
Fenz et al. (2009)	Information security ontology	◉	◎	○	○ / ◉	◎	○
Gao et al. (2013)	Computer attacks, Securty assessment	◉	◎	○	○ / ◉	◎	○
Souag et al. (2015)	Security requirements elicitation	◉	◉	○	◎ / ◉	◉	◎
Silva and Rafael (2017	Network security, Network monitoring	◉	◉	○	○ / ◉	◎	○
Mozzaquatro et al. (2018)	Security ontology, IoT	◉	◉	◎	◉ / ◉	◉	○
Zamfira et al. (2018)	Cyber defense system	◎	◎	○	○ / ○	○	○
Vega-Barbas et al. (2019)	Dynamic risk management	◎	◎	○	◎ / ◉	◎	○
CordaSecOnt	Financial application security	◉	◉	◉	◉ / ◉	◉	◉

◉ - detailed discussion; ◎ - limited discussion; ○ - not addressed

example, Vega-Barbas et al. [210] present an ontology for dynamic risk management in administrative domains. The work collected and modeled the assets, threats, and vulnerabilities in the administrative domain. Mozzaquatro et al. [211] develop an ontology-based cybersecurity framework for the IoT to improve security by focusing on the enterprise monitoring, analyses, and classification of security vulnerabilities. The IoT cybersecurity ontology framework deals with the security of IoT at design time, run time, and an integration layer. The ontology uses OWL and Protégé to classify assets, vulnerabilities, threats, security properties, and security mechanisms in the IoT security domain. Zamfira et al. [63] build the ontology of cyber-operations in networks of computers. The ontology improves the detection capabilities of attacks at various levels of application. Silva and Rafael [212] present the ontology for network security, and Gao et al. [213] created an ontology-based security assessment framework for network and computer attacks. The ontology explains the taxonomy of attacks in the context of attack impact, attack vector, attack target, vulnerability, and defense strategies.

The related works so far develop either generic or domain-specific ontology of information security focusing on the centralized applications where traditional SRM concepts are encoded. The definition of assets is not concrete to categorize system and business assets. Also, missing the security criteria of assets to define the security goals. The relations between the threats and countermeasures are not explicitly described to explain what countermeasures mitigate which vulnerability. The information security concepts are defined mainly based on assumptions and neglected the use of fundamental concepts of SRM. In contrast, our ontology focuses on the organization of information security knowledge for permissioned blockchain-based decentralized applications. We utilize the Corda platform and build the ontology for the financial industry case, where we present CorDapp as a countermeasure solution. Moreover, our ontology uses the foundational ontology such as the ULRO which is based on the fundamental concepts of SRM. Thus, the ontology explicitly explains the concepts related to the system and business assets, security criteria, threats, vulnerabilities, and countermeasures.

5.6. Challenges and Implications

Permissioned blockchains are appearing in the financial sector because of the suitable technological framework for developing decentralized financial applications. However, in addition to the benefits, various obstacles impede the uptake of permissioned blockchains. For example, there is a lack of transparency and decentralization because permissioned blockchain-based applications have a limited number of nodes, and activities are controlled by a private group or authorized authority [214, 215]. This problem can also potentially lead to collusion and the overriding of consensus norms. Another challenge is shifting the legacy system to blockchain [216]. For instance, the current financial applications stakeholders are comfortable using the current centralized systems, and various organizations don't want to share the data and their operations because they are managing in their convenient way. Despite the fact that permissioned blockchains execute transactions quicker and have higher transaction throughput than permissionless blockchains, they face performance challenges as the number of nodes increases. For example, Khan et al. [217] conducted a performance analysis of HLF and found that transaction throughput decreased as the number of nodes increased. Blockchain interoperability generally tackles the ability to share states and transact across different blockchains [218]. Currently, there is no sophisticated method to allow one blockchain to transmit information to another blockchain [219, 218].

In the current work, the first research challenge is a subjective interpretation [169] that exists since we might have different interpretations and opinions related to identified threats, vulnerabilities, and countermeasures. Another challenge is the security risk analysis of emerging security threats in blockchain systems. For example, blockchain is a new technology that is continually changing, and permissioned blockchain-based financial applications are still in their infancy. As a result, this study does not examine all the security threats that can be addressed or exist in permissioned blockchain-based applications. Furthermore, oracles are used by blockchain-based financial applications as a mode of communication to interact with the off-chain applications to send and receive relevant information or to execute dependent operations. However, this work does not examine the usage and security of oracles. In the scientific community, research in this field is also sparse, and this gap presents an attraction for academics to do more empirical analysis on the aforementioned platform.

Permissioned blockchains allow only pre-verified nodes in the network, and the access control layer governs the actions of participants. Such blockchains are energy-efficient because they do not require energy-waste consensus. They provide a high rate of transactions per second (TPS), decentralized operations, and data storage in a robust governance structure. Also, ensure user privacy, confidentiality, and cost-effective transactions by removing mining fees. Therefore, permissioned blockchain-based applications are becoming apparent in industry-

level enterprises and businesses where security and privacy are imperative (e.g., finance). The security and implementation challenges of permissioned blockchains compared to permissionless blockchain systems are relatively low or controlled to a certain level. The attacks are less beneficial and burdensome. For example, permissioned blockchain systems do not directly include monetary assets, pre-verification before joining the network, and control the participant's operations by permissioned settings and access control layer. These concepts lead to permissioned blockchain adoption among industry-level enterprises (e.g., logistic partners, supply vendors, financial institutions, etc). Because security, user privacy, and transaction confidentiality are important aspects along with decentralization. The ontology representation we created can help with interoperability between different security solutions, offering a unique approach to express SRM in financial applications. An example related to the capital market's post-trade matching and confirmation shows how the ontology may be incorporated into high-level decision-making for the SRM by utilizing permissioned blockchain as a countermeasure solution. Moreover, our work opens up other avenues for future research in terms of theory development and concept validation. Therefore, further research is required to improve and expand on our findings. The discussion section expands on such potential future research implications.

5.7. Discussion

Blockchain technology can revolutionize the financial industry by protecting the centralized infrastructure from the constant security threats [185, 4] that harm the business processes and valuable assets leading to reputation loss and financial sanctions. We chose the financial industry case and Corda blockchain because the SLR (Section 2.4) results that show the financial domain is most studied using permissioned blockchains. The post-trade services have grown across Europe, but several inefficiencies are challenging its centralized infrastructure (e.g., regulatory harmonization) and security risks, including other risks (e.g., credit, liquidity, operational, legal, and systemic risks) [186, 220]. European Central Bank [185] investigated the use of permissioned blockchains to overcome the security challenges of the post-trade services and to keep control over the sensitive information regarding the holdings and human-readable account identifiers. Such evidence motivates us to use the Corda, which provides an appropriate technological framework for developing a blockchain-based post-trade matching and confirmation application in order to address security challenges while respecting privacy. It is also scalable and provides ease of integration with existing systems [195, 199].

This chapter focuses on increasing financial industry security through an ontological analysis using Corda platform. We employ OWL to construct a semantic knowledge base that provides a valuable tool for assessing and communicating post-trade matching and confirmation security elements, resulting in timely deci-

sions to correct them. In information security, an ontological representation also removes conceptual ambiguity and semantic gaps. Our current research has several limitations. For instance, the current approach lacks a comprehensive analysis of security threats. For example, the likelihood and impact of a threat are essential considerations in the risk assessment process. Also, the threat agent profile and attack method are missing in the current ontology representations. Corda looks promising but is still in its infancy and moderately adopted in the industry, so its security needs to be evaluated on a larger scale. Many security issues, for example, have yet to be investigated. Also, we gathered security threats through literature studies and assessed the case of traditional post-trade matching and confirmation. The security threats we obtained focus on the asset level, neglecting technical specifics and defects that constitute a source of security threats. We discuss these constraints in our future work that will extend to the current work.

5.8. Summary

This chapter answers our RQ3: How might permissioned blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissioned blockchain-based applications? Accordingly, we extended the ULRO using the case from a financial industry related to the capital market's post-trade matching and confirmation and permissioned blockchain-based Corda platform. The result is a Corda-based security ontology representation, CordaSecOnt. CordaSecOnt provides an extensible semantic knowledge base for the SRM of post-trade matching and confirmation, including the notions of blockchain and Corda. To build the CordaSecOnt, we define the scope of our ontology and develop the classifications related to system assets, business assets, security criteria, threats, vulnerabilities, and countermeasures. We explore the security threats from two different perspectives. For example, the security threats that are mitigated (such as spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege) by using the CorDapp, and the security threats that appear (such as endpoint threat, quantum computing threat, privacy violation, deanonymization, smart contract threat, and denial-of-state) within the CorDapp.

To evaluate the CordaSecOnt, we use automated tools (e.g., pellet reasoner, and OOPS!) to examine the consistency of the encoded concepts and relationships. We also employ the qualitative assessment criteria to check whether the coded concepts model the real-world domain for which the ontology representation was created. Then, we utilize the tasks-based evaluation where we build the SPARQL queries corresponding to the created tasks to request data from the CordaSecOnt to verify the retrieved results are correct and based on the defined relationships. Overall, the CordaSecOnt can support the SRM while developing the financial industry applications. Also, the CordaSecOnt encodes the knowledge of Corda information security to dynamic ontology-based knowledge that can be extended, reused, or integrated with other security ontologies.

6. PERMISSIONLESS BLOCKCHAIN-BASED SECURITY ONTOLOGY

This chapter answers our fourth research question RQ4: How might permissionless blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissionless blockchain-based applications? The result is a comprehensive ontology representation in the healthcare domain combining the permissionless blockchain as a countermeasure solution. The healthcare domain is subjected to a variety of security threats, and blockchain is gaining traction to improve their security by turning healthcare operations into decentralized, transparent, and immutable manners. However, blockchain does not become a panacea for securing healthcare applications because various security threats are observed in blockchain-based applications. Also, when it comes to the SRM of traditional healthcare applications (THAs) using blockchain or the SRM of blockchain-based healthcare applications (BbHAs), there are conceptual ambiguities and semantic gaps that hinder treating security threats effectively. Therefore, to address these issues, we present a permissionless blockchain-based healthcare security ontology, HealthOnt. HealthOnt provides the classifications of system assets, business assets, threats, vulnerabilities, and countermeasures and offers coherent and formal information models to treat security threats of THAs and BbHAs.

The remainder of the chapter is structured as follows: Section 6.1 discusses the use case of back-pain patients' healthcare application that we use to evaluate the HealthOnt. In Section 6.2, we discuss the security threats of THAs and present blockchain as a countermeasure solution. Section 6.3 presents the security threats that may appear in blockchain-based healthcare applications. Both sections present the framework combining the knowledge and fundamental concepts of SRM that we used to extend the URLO with permissionless blockchains. Section 6.4 constitutes the HealthOnt that illustrates the different classifications to elaborate the encoded knowledge of SRM of healthcare applications. Section 6.5 is the evaluation of HealthOnt where we perform the SRM of the back-pain patients' healthcare application case using the HealthOnt. Moreover, we conduct competence questions-based interviews to evaluate the adequacy and applicability of the HealthOnt. Section 6.6 outlines the related work. Section 6.8 is the discussion, and Section 6.9 concludes this chapter.

6.1. Back-pain Patients' Healthcare Application Case

In this section, we discuss a case of the back-pain patients' healthcare application (BPPHA) that we used to evaluate the ontology. The BPPHA is operating at *Farhat Hached University Hospital in Sousse, Tunisia* to illustrate our proposal. The case scenario is shown in Fig. 53, where the main stakeholders are the medical advisor, patient, and expert doctor. The scenario starts when the patient

contacts the medical advisor for consultation. After the appointment, the medical advisor prepares the CNAM letter[1] (including questions to the expert doctor) and attaches the necessary medical reports. The patient is then in charge of delivering the CNAM letter and the medical reports to the expert doctor. The expert doctor registers the patient's data and collects additional information during the interview in order to define illness type (e.g., work accident or long-duration illness). For example, during the interview, the expert doctor collects whether the patient suffers from low back-pain, the type of sciatica, whether the patient is diabetic, and also asks for personal information (e.g., marital status, number of children, and the last job type). Thereafter, the expert doctor identifies the illness and studies the necessary documents related to the work accident or the long-term illness.

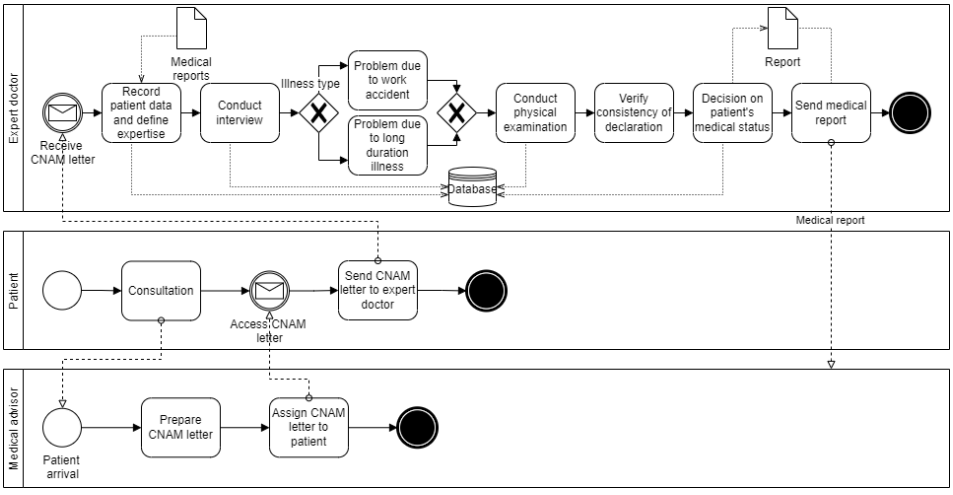

Figure 53. Case of back-pain patients' healthcare application

Next, the expert doctor performs the patient's physical examination and records results (e.g., weight, height, build, limp, and gait). During the physical examination, the expert doctor can check and verify the consistency of the claim. Then, the expert doctor writes a conclusion based on the gathered data (e.g., on the patient's details, interview, and physical examination outcomes). The expert doctor writes a medical report and sends it to the medical advisor. This report includes the conclusion about the patient's medical status and guides the medical advisor regarding the decision (e.g., whether the medical leave is needed, what is the duration of the medical leave, and when the patient could return to work). We will consider this back-pain patients' healthcare application case to illustrate the security threats and how they can be mitigated using blockchain technology.

[1]Caisse Nationale d'Assurance Maladie (*French*) – National Health Insurance Fund

6.2. Security Threats Mitigated

We examine the literature studies that describe how blockchain can alleviate the security threats of THAs. Following the constructs of BbRM, we develop a framework (Table 42) and discuss the security threats of THAs in detail. The framework describes THAs' assets to protect, threats, vulnerabilities, and countermeasures.

6.2.1. Data Tampering

The traditional approaches lack control over data security, which is a major concern for healthcare organizations because it can put patients' lives at risk. In THAs, the access control is managed by a designated authority that can be error-prone. The weak centralized access control [2, 11, 94, 221] describes a case when the healthcare application fails to restrict unauthorized access to the resources. The attacker compromises the security and performs unauthorized actions that negate medical records' integrity and patient data confidentiality. The attacker uses unauthorized access to gain elevated privileges, execute commands, or bypass the security mechanisms to tamper with medical data. Moreover, THAs often rely on manual techniques or third-party providers to perform data verification and validation. These techniques lack the proper mechanisms to verify and validate the authenticity of data [78, 222, 223]. Consequently, the attacker can submit malicious content that the system can process and negate the integrity of medical records and confidentiality of patient data.

Blockchain allows smart contracts-based distributed access control [224] that regulates the users access to stored medical data. The system authenticates and identifies associated users according to their access rights deployed in a decentralized and distributed environment. Also, strong cryptography (e.g., attribute-based encryption) [225] helps to build fine-grained access control. The records are difficult to modify or delete because of the ledger redundancy, and append-only structure [8]. The healthcare applications on permissionless blockchain use PoW consensus to verify the executed transaction and data validation without requiring a third party before saving on the ledger [222]. Using the SHA-256 hashing algorithm, blockchain computes a unique hash id of original data that can be used to verify the authenticity of data [221]. Chen et al. [78] use the trusted authorized nodes to verify and validate the authenticity of data. Blockchain is tamper-evident [94, 221] and detects unauthorized modifications. Blockchain builds strong audit trails in an immutable ledger by keeping a log of each performed action [94] over time that could be used to verify and validate the authenticity of data.

6.2.2. Data Theft

In healthcare, data theft has been on the rise over the past ten years, in 2020 reported 642 data theft incidents [1] only in the United States. Databases are one of the most compromised assets [54] and centralized databases have improper security controls to protect against insider or outsider threats [225, 8].

Table 42. Framework for security risk analysis of traditional healthcare applications

Risk-related concept		Asset-related concept		Risk treatment concept
Threat	**Vulnerability**	**System asset**	**Business asset**	**Countermeasure**
Data tampering	Weak centralized access control mechanism	Healthcare database, Access control	Medical records (I), Patient data (C)	Distributed access control mechanism
				Access control with cryptographic primitives (e.g., attribute-based encryption)
	No mechanism to verify and validate the authenticity of data	Healthcare database, Medical transactions	Medical records (I), Patient data (C), Data validation (I, A)	Distributed (shared) and append-only ledger
				Proof of work-based consensus mechanism
				Data validation without requiring third party
				Unique hash id of original data
				HLF-based trusted authorized nodes
				Decentralized and tamper-resistant
				Immutable logging and data provenance
Data theft	Improper security controls for centralized database	Healthcare system, Data access right	Healthcare database (I), Medical records (C)	Blockchain-based P2P network
				Voting process to determine data access
				Permissioned settings to restrict data access
				Access control with cryptographic primitives
	Weak centralized access control mechanism	Access control	Medical records (C)	Distributed access control mechanism to control data leak
	No proper cryptographic controls	Healthcare system	Medical records (C)	Encrypts data and store on/off chain
				Store the encrypted and obfuscated data
Medical records mishandling	Patients have weak control over their medical records	Data access right	Medical records (I, C)	Blockchain enables patients to control the access to their data
	Relying on a third-party	Healthcare database	Medical records (C)	Data validation without requiring third party
	No guarantee of electronic medical records authenticity			Decentralized and tamper-resistant
				Consensus mechanism
Counterfeit drugs	Weak traceability controls in pharmaceutical supply chain	Drugs details, Supply chain	Drug traceability (I)	Immutable and traceable drug trails
Man in the middle attack	Weak controls to secure communication	Network, Data exchange	Communication (I)	Distributed IPFS for storage
				P2P-based encrypted communcation
	Lack of anonymization of patient medical records	Healthcare system	Medical records (I, C)	Blockchain anonymize the data
Single point failure	Relying on centralized server	Healthcare database and system	Server (A), Services (A)	Decentralized distributed P2P network
	Weak implementation to handle large number of requests			
Repudiation	Weak controls to prove illegal data changes by authorized users	Healthcare system	Medical records (I)	Blockchain-based versioning scheme to track each performed operation
	Lack of immutable logs	Action logs	Medical records (I)	Immutable log of all performed activities
Insurance fraud	No proper authenticity to verify the insurance claim	Medical bills, Insurance data	Insurance claim (I)	Decentralized verification of insurers
				Verified records are distributed among nodes
Clinical trial fraud	Inadequate clinical trials data	Clinical trial data, Data access right	Data processing (I, C)	Distributed nature and use of cryptography
	Improper patient recruitment and lack of data access			Blockchain provides data ownership
				Data saved on blockchain cannot be altered
Tampering device settings	Weak controls on settings of medical devices	IoT devices	Device settings (I, A)	Storing devices settings in distributed immutable ledger
Social engineering	Possible to manipulate employess to get data access	Employees, Stakeholders	Medical records (I)	Only relevant employees have access to particular information or part of information

The threats imposed by this vulnerability include abuse of elevated privileges, unauthorized access, backup storage exposure, database injection, default database accounts and configurations, and the human factor [54]. Overall it negates the integrity of the healthcare system and the confidentiality of medical data. Similar to data tampering, the attacker can steal medical data due to weak centralized access control [225, 226] that leads the attacker to gain unauthorized access, elevated privileges, or bypass security mechanisms. As a result, it negates the confidentiality of medical data. THAs use cryptography to save data securely and achieve information security objectives. However, it lacks cryptographic control [226] since the centralized authority (or individual) is responsible for the administration of the database (e.g., keeping elevated privileges, encryption/decryption keys). If the security of the system compromises, the attacker can steal the medical data.

Blockchain works on a P2P-based distributed network where nodes behave as a server and client to send and receive data directly. This mechanism helps to protect the data leakage to unauthorized network users [78]. The solution proposed in [8] uses the voting process (e.g., QuorumChain algorithm) to determine which nodes are allowed to access certain types of data. The permissioned blockchains define permission settings to restrict data access only to authorized nodes [221, 226]. Similar to data tampering countermeasures, the strong cryptographic primitives (e.g., attribute-based encryption) [225] and smart contracts-based distributed access control mechanism [222] allows only authorized users to access medical data. The Ancile framework [8] uses the proxy re-encryption to encrypt the data and store hashes data on- and off-chain. Esposito et al., [225] suggest data obfuscation to protect data on- and off-chain.

6.2.3. Medical Records Mishandling

Healthcare staff must ensure that medical records are kept private and safe. But medical records mishandling is one of the common HIPAA violations [3]. The medical institutions control and manage the patient's medical data where non-relevant individuals can access it. Thus, patients have weak control over their medical records [227]. Also, the patient is unaware of how his data is processed or with whom it is shared. In some cases, the individual from a medical institution involves in illegal medical data trade [228] that negates the integrity and confidentiality of medical records. Hospitals and healthcare applications rely on third-party [223, 225] vendors (e.g., IT vendors, pharmacies, insurance companies, etc.) daily to perform their routine functions. These third-party vendors have access to the patient's medical data. They could intentionally sell medical data to data brokers or become a source of data breach and negate the confidentiality of medical data. Also, the medical data is managed by a designated authority/individual, and the system administration governs the system with elevated privileges. If any such point is compromised, the attacker can manipulate and negate the integrity of medical data without leaving traces. Therefore, THAs cannot guarantee the authenticity [223] of digital medical records.

Blockchain-based permission settings and distributed access control enable patients to handle their medical data [11, 229]. The blockchain performs data validation before saving on the ledger during the consensus process. For example, blockchain provides a transparent platform to define data validation rules which are agreed upon by decentralized and distributed network nodes [110]. Then, all the nodes follow those rules to validate the data. The blockchain-based applications detect and discard all the unauthorized changes [229] if the majority of the network is honest (e.g., the adversary does not control 51% computing power). This process establishes a tamper-resistant environment [230].

6.2.4. Counterfeit Drugs

For years, the pharmaceutical supply chain has been struggling to monitor its products and avoid fake medicine. According [11, 227], 10-30% (worth $200 billion) of drugs sold worldwide each year are counterfeit. Counterfeit drugs are on the rise, posing significant health risks. In pharmaceuticals, after manufacturing, drugs are moved from production stocks to wholesale distributors, which then move to retail firms. Customers purchase drugs from retailers. Due to weak traceability controls (e.g., ineffective data sharing, no traceable records) [10, 11, 227] in the pharmaceutical supply chain, there is a risk of fake medicines being introduced during this process.

Blockchain offers a solution to enable pharmaceutical traceability, real-time access to data, and supply chain validation by creating a log to track each step [10, 11, 227]. For example, IBM Research uses blockchain to reduce or eliminate the drug counterfeiting problems in Kenya [227] by using immutable and traceable logs at each stage of the pharmaceutical supply chain.

6.2.5. Man in the Middle Attack

Man in the middle (MitM) attacks are rising in healthcare applications to gain sensitive information [231]. The attacker can exploit the weak controls of secure communication [2] in THAs and negate the integrity of communication assets. For example, not properly implementing (or having) cryptographic functionality or lack of fine-grained access control mechanism. Moreover, due to lack of anonymization of patient medical records [232] the medical data is associated directly with patient identity. The attacker can get the data to trigger a ransomware attack, publish it online or deny access to it.

Blockchain-based distributed interplanetary file system for storage and data encryption to reduce the communication and computation overhead that establishes a secure communication channel [2]. Blockchain works on a P2P-based distributed network where nodes behave as a server and client to exchange encrypted data directly. This feature resists the attacker from intercepting communication or data analysis/sniffing [11, 78]. Blockchain maintains pseudo-anonymity; the patients and their medical data are linked with a public address. Also, the data processing on a blockchain is anonymous [11], and blockchain anonymizes the medical data to hide the actual identity [232].

6.2.6. Single Point Failure

Like any other system, the attacker can find faults in the system's design, implementation, or centralized dependency components to disrupt the healthcare services. Currently, the healthcare system uses a centralized server model [2] that can pose a threat of single-point failure and performance bottleneck. The weak mechanism to handle large numbers of requests [229] allows the attacker to target the server and services of the system to halt them for legit users.

Blockchain is resilient to this threat with the advantage of a decentralized distributed P2P network [10]. Moreover, blockchains do not rely on a single or central point server [2, 229].

6.2.7. Repudiation

The patient's medical data is sensitive and life-critical. The healthcare system should trace all actions performed (intentionally or unintentionally) on a patient's medical data and easily identify how it was performed. In THAs, there are weak controls to prove illegal data changes by authorized users [80]. For example, almost every stakeholder within a medical institution has access to the patient's medical data that can be viewed, modified, or deleted. Moreover, unintentional data changes that later are not traceable during data processing can happen. THAs manage centralized and mutable logs [79] that are handled (or have access) by a system administrator or other IT staff. Also, if the system is compromised, the attacker can easily remove the actions he performed from the logs. Therefore, the authenticity of logs can not be proved on centralized systems.

Blockchain keeps immutable logs [79] to track who and when the particular operation was performed. Also, the study [80] uses the blockchain-based versioning scheme to track each performed operation over time.

6.2.8. Insurance Fraud

Healthcare insurance frauds are increasing, involving filing dishonest healthcare claims. For example, the value of challenged healthcare claims surged from $11 billion to $54 billion annually [10]. In THAs, there is a lack of proper authenticity [227] to verify the insurance claim because of complex information systems, administrative burdens, expensive & manual validation and verification of provider directories, and record-keeping mistakes that attracts the attackers.

The blockchain enables the decentralized verification of insurers based on the predefined set of rules [227] before registering on the ledger. Once the insurer is verified and registered, the records are distributed among other nodes to keep track of valid and invalid insurers in the system.

6.2.9. Clinical Trial Fraud

Healthcare institutions and research groups suffering from clinical trial fraud [233] and medical decisions made by researchers on the premise of fraudulent data could leave patients at risk. The data frauds in clinical trials include deliberate fabrication, falsification, or plagiarism in proposing, performing, or reviewing research and research results [233]. The inadequate clinical trials data [227] emerge due to lack of data integrity and provenance. Also, the current infrastructure has inefficiencies in patient recruitment and access to medical data [10].

Blockchain has a distributed nature, and using cryptography ensures data is authentic [227]. Also, blockchain provides data ownership to patients [8] to control

the access of their data; once data is saved on the blockchain, it cannot be altered. Thus, eliminating the threat of clinical trial fraud.

6.2.10. Tampering Device Settings

Medical devices connected to the internet and the internet of things (IoT) enable healthcare professionals to be more watchful and connected with the patients. Progressively, IoTs are becoming the heart of digital healthcare, but new security challenges are appearing. In healthcare, the medical devices are subject to heedless settings [234] (e.g., lack of network segmentation, insufficient access control, and reliance on legacy systems). The intentional changes in device settings (e.g., from the attacker) or unintentional changes (e.g., from the authorized user) can lead to false readings that put the patient's life in danger.

Blockchain follows the append-only structure to save data. Thus, device settings stored in blockchain are distributed and immutable [234].

6.2.11. Social Engineering

According to [235], only 1% of cyber-attacks in the year 2019 were exploited due to hardware or software vulnerabilities, and 99% of cyber-attacks utilized some form of human intervention (e.g., phishing, fake identity, honey trap, etc). In healthcare, the healthcare staff is one of the weakest points, and the attackers use social engineering techniques [232] to target them to get patients' medical data. Healthcare staff is vulnerable to social engineers because they naturally trust others, do not want to be rude, have a desire to be helpful, and find it difficult to remember everyone in a large healthcare environment [236].

Blockchain implements smart contract-based distributed access control that ensures only relevant users have access to particular information or part of the information [224, 8]. Thus, protecting medical data against unauthorized user access. However, the threat of social engineering can not be eliminated through new technology or a more secure password, but it can be restricted to an acceptable level by proper training of employees [236].

The security risk analysis of THAs shows that blockchain can help the healthcare sector to overcome the security threats of traditional technology infrastructure for preserving the medical data, data integrity, and patient ownership of his data. We use the constructs of the SRM domain model that fulfills the ISO/IEC 27001 standard [48] for defining the scope of our work and to assist in building a framework for structuring the security risk analysis of traditional and blockchain-based healthcare applications. This framework (Table 42) presents blockchain as a countermeasure solution for mitigating the security threats of THAs. Blockchain provides technology infrastructure with unique characteristics for building healthcare applications. For example, blockchain operates over a P2P network, uses consensus mechanism and cryptography, is immutable, decentralized, tamper-evident,

and provides permission settings, provenance, and pseudo-anonymity. However, we cannot deny the security aspects of BbHAs because, in recent years, various security threats have appeared in blockchain-based solutions. Hence, we discuss such security threats in the next section.

6.3. Security Threats Appeared

We analyze the literature studies that describe the security threats of BbHAs. We identify those security threats and categorize them using the constructs of BbRM and develop a framework (Table 43). The framework illustrates the BbHAs' assets to protect, threats, vulnerabilities, and countermeasures.

6.3.1. Sybil Attack

Sybil attack is a P2P network attack [83] where the attacker creates numerous fake identities and connects with victim nodes to isolate them from other honest nodes. Blockchain systems run over the P2P network, and therefore they are susceptible to Sybil attack. The attacker can control several nodes on BbHAs by creating fake identities [237, 238] to gain disproportionately large influence. Once Sybil nodes gain recognition, the attacker forces victim nodes to process blocks under his control, out-votes (or block) the honest nodes, interrupts the flow of information, distort the block generation process, and refuses to receive or transmit information [29]. Also, if the blockchain system has insufficient computing-power [18], the attacker with higher computing power and Sybil nodes can disrupt the healthcare operations. Moreover, the poor implementation of node authentication [84] (e.g., no network joining fee, not validating IP address, or source of node connection) negates the integrity of the transaction verification process.

To overcome Sybil attacks in BbHAs, incorporate network joining fee [84] and stake requirements in PoS consensus [239] to make identity creation more expensive. Monitor node behavior [84] to spot any unusual activity of nodes and disconnect them from the network. If the system uses PoW consensus, the network should have enough computational power [18] based on the network's available nodes. Regularly monitor the computing power to ensure no one is misusing it. In addition, before joining the blockchain network, perform node authentication. For instance, requesting a network joining fee, validating node connections, and monitoring node activities [84].

6.3.2. Double-spending

The double-spending is categorized under data consistency attack [155] to spend the same transaction twice [85]. Similarly, in BbHAs, the attacker can change the transaction state and spend the same transaction twice. The attacker uses 51% or more computing-power to control the network [240] to weaken the P2P network to perform double-spending (e.g., insurance frauds) [237].

Table 43. Framework that presents security risks analysis of BbHAs

Risk-related concept		Asset-related concept		Risk treatment concept
Threat	**Vulnerability**	**System asset**	**Business asset**	**Countermeasure**
Sybil attack	Possible to create fake identities in the network	Nodes (miners), Nodes identity, P2P Network, Transactions	New nodes (A), Information flow (A), Ledger (I, A), Block generation (A)	Network joining fee
				Monitor nodes behavior
				Stake requirements in PoS consensus
	Lack of computing power	Nodes, P2P Network, Computing power	Network reputation (I), Healthcare operations (A)	Increase computing power
				Monitor computing power
	No proper authentication of nodes	Nodes, P2P Network, Network reputation, Transactions	Transaction validation (I)	Network joining fee
				Validating node connection
				Monitor nodes behavior
Double-spending	51% vulnerability	Computing power, Nodes (miners), P2P network	Transaction (I), Ledger (I), Network resources (A)	Insert observers
				Use power monitoring tool
				Transaction fee
				Pluggable consensus
	Accepting unconfirmed transactions	Transactions, Block confirmations	Fast transaction (I, A), Digital assets (I), Ledger (I)	Increase confirmed blocks
				Closed-form formula probability
				Enhance network policy
				Listening period
				Insert observers
				Alerting honest nodes
Eclipse attack	Poisoning nodes' routing table	Nodes, IP addresses, Node connection, Transactions, Routing table	Communicating/ gossiping (A), Transaction validation (I), Medical data (C, I)	Disable direct incoming connections
				White-listed nodes
				Random outgoing connections
				Deterministic random eviction
				Incorporate feeler and anchor connections
Smart contracts attacks	Faulty and error-prone smart contracts	Smart contracts, Transaction validation, Ledger	Digital assets (I), Transaction (I), Medical data (C, I, A)	Smart contracts code analyzers (e.g., SmartCheck)
				Penetration testing tool
Block withholding delay	Possible to delay the submission of valid blocks	Transaction validation, Blocks, Mining incentives	Healthcare operations (A), Information processing (A), Block confirmations (A)	Enforce immediate block submission scheme
				Increase risk of earning less incentives
Sybil-based DoS	Sybil nodes can participate in the consensus mechanism	Nodes, P2P network, Mining protocol	Healthcare operations (A), Mining process (A)	Use computational constraint-based techniques
	Dusting transactions	Transactions, P2P network, Ledger	Healthcare operations (A), Network resources (A)	Anti-dust model
Deanonymization attack	Network analysis and listening	Transactions	Medical data (C)	Use mixing techniques
				Use anonymity overlay networks (e.g., Tor)
				Ring signatures and zero-knowledge Proofs
Quantum computing threats	Not using quantum-resistant cryptography schemes	Cryptography	Transactions (I), Ledger (I), Medical data (C, I)	Quantum computing resistant cryptography
Endpoint security threats	Lack of awareness and knowledge	Wallets, Keys, Computers/devices, User	Healthcare operations (A), Digital assets (I), Medical data (C, I, A)	Multi-level authentication (MLA) method
				Security awareness
				Hardware security module (HSM)

This vulnerability can also affect the availability of network resources; the attacker can trigger selfish-mining, prevent new transactions from gaining confirmations, and blockchain forks [241]. However, this vulnerability is practically impossible on high computing power blockchains (e.g., Bitcoin and Ethereum) [237]. Moreover, accepting unconfirmed transactions [85] enables the attacker to indulge in a race to make his double-spend transaction valid by exploiting the intermediate time between two conflicting transactions and using a higher transaction fee. If it is successful, it negates the integrity and availability of fast transaction mechanisms and the loss of digital assets (such as insurance claims) and the ledger's integrity.

To mitigate double-spending, one should implement a power monitoring tool to monitor the computing power of nodes continuously and restrict when reaching a certain amount of computing-power [87]. Also, incorporate transaction fee [86] as an incentive to keep nodes honest in a blockchain system. Use a pluggable consensus mechanism [35] to facilitate consensus diversity based on the blockchain

system's requirements. Furthermore, the study [140] states that increasing the number of confirmed blocks would decrease the double-spending threat. Use a closed-form formula to calculate the likelihood of double-spending in a race attack [151]. In addition, enhance network policy [155] to guide how to set a block confirmation number considering the value of the transaction.

6.3.3. Eclipse Attack

In an eclipse attack, the attacker takes control of all the neighboring peers of the victim node and hides the correct ledger from the victim node [238]. Eclipse attack targets particular node [29] by flooding with his IP addresses. The attacker poisons victim node's routing table [238] by filling it with his IP addresses. Once the node restarts, it loses its current outgoing and incoming connections and makes the new connections with the attacker's IP addresses. If the attack is successful, the attacker inhibits the victim node from learning and communicating with other peer nodes, disrupting the transaction verification process and gaining access to the medical data. Moreover, this attack allows the attacker to alter transactions to perform double-spending, and selfish mining [238].

The first countermeasure is to stop direct incoming connections [242, 145] and make incoming and outgoing connections via white-listed nodes [145], such as well-connected peers/miners, to prevent the eclipse attack. Include a random outgoing connections method [242] to prohibit all connections with the attacker's IP addresses. Use deterministic random eviction [145] to keep track of new and tried connections. It minimizes the number of attack addresses used by the attacker when making connections. Moreover, the feeler connections [145] to make short-lived test connections with randomly-selected addresses. If the connection is successful, the address includes in white-listed nodes. The anchor table method [145] allows to keep track of current outgoing connections, and when the node restarts, it makes a connection with the old addresses from the anchor table.

6.3.4. Smart Contracts Attack

The security of smart contracts has become a major concern in recent years [243] as a result of different security issues originating in blockchain-based applications from the execution of smart contracts. The security issues in smart contracts are associated with the bugs in the source code (e.g., transaction-ordering dependency, timestamp dependency, mishandled exceptions, reentrancy, unpredictable state, transaction overflow, underflow, etc.) [243, 137]. According to [66], in Ethereum around 45% smart contracts are vulnerable, and the attacker can exploit faulty and error-prone smart contracts [137] to harm the valuable assets in BbHAs. For example, Ethereum smart contract reentrancy attack on the decentralized autonomous organization (DAO) when the attacker stole $60 million Ethers [243]. Many blockchain platforms are introducing smart contracts to construct decentralized applications, but their security has yet to be fully studied [137]. In BbHAs,

the attacker can exploit these vulnerabilities and target the digital assets, steal or modify the medical data, and interrupt the medical operations [244].

The developers should employ smart contract code analyzers to discover flaws, race situations, and sanitize the smart contract code before deploying it on a blockchain. For example, the SmartCheck [244] to detect vulnerabilities in the smart contract at different severity levels, Oyente tool [244] to detect callstack depth and re-entrancy attacks. On top of these, use penetration testing tool [245] to test blockchain-based applications before deployment.

6.3.5. Block Withholding Delay

In PoW-based blockchains, a block withholding delay is common. The attacker miner joins a victim mining pool and refuses to submit blocks on time [246]. The attacker miners deliberately delay the submission of valid blocks [246] that results in discarding of the blocks that can distort the operations of BbHAs. Also, the strategy leads the attacker to gain higher rewards than honest mining nodes [230]. In BbHAs, this attack can hinder medical operations and delay block confirmations needed for the transaction finality.

To mitigate this attack, the system should enforce an immediate block submission [247] to submit the block as soon as it is found. Moreover, implement an incentive payoff scheme [246] to increase the risk of earning fewer incentives to demotivate those who deliberately delay block submissions.

6.3.6. Sybil-based DoS

Blockchain-based applications operate over a P2P network. Despite being operating on a P2P network, they are still vulnerable to DoS attacks [247]. By design, permissionless blockchains let anybody participate in the consensus process. The attacker takes advantage of this situation by participating in the consensus with his Sybil nodes [88] to postpone medical operations and interrupt the mining process. Also, the attacker creates numerous dust transactions [132] between his Sybil nodes, and blockchains process a limited number of transactions per block in a given time. The Sybil nodes participating in the consensus do not share their verified transactions or blocks. Thus, the large number of transactions with small values congest the blockchain network [237], delay the medical operations, exhaust the network resources, and halt the mining process.

The Sybil-based DoS attacks cannot be mitigated entirely but possible to restrict them. For example, incorporate computational constraint-based Sybil resistance techniques like Bitcoin uses PoW [88]. Moreover, utilize the anti-dust model [132] that checks different parameters in the transaction (e.g., transaction volume and fees) to identify and prevent dust attacks [132].

6.3.7. Deanonymization Attack

Anonymization is a characteristic of blockchains that refers to hiding an identity, but still possible to link a user or company behind each transaction [88]. Patients' privacy is the utmost requirement in healthcare. However, in BbHAs, it is possible to identify the patient by performing network analysis and listening [248, 249]. For example, analyzing the transaction contents, transaction relationship with other transactions, and the way the transaction is broadcasted. Moreover, the attacker can perform graph analysis [248] on publicly available transactions to deanonymize the identities of patients.

As a countermeasure, use the mixers as a service to enhance the privacy and anonymity of transactions by obfuscating the transaction flow [89]. The study [249] suggests using anonymity overlay networks such as Tor. Moreover, incorporate ring signatures, and zero-knowledge proofs [248] to achieve the required level of privacy on medical data, and users get only relevant information.

6.3.8. Quantum Computing Threat

Quantum computing research is advancing, and many cryptographic protocols in use currently are vulnerable to quantum computing [250]. Blockchains rely on cryptographic protocols that are also vulnerable to quantum computing because they are not using quantum-resistant cryptography [11, 197] to tackle it. Thus, the BbHAs are vulnerable to quantum computing [213] in a post-quantum era. For example, blockchain platforms use an elliptic curve digital algorithm (ECDSA), and it could be solved by quantum computers [213].

The blockchains should implement quantum computing resistant cryptography schemes (e.g., lattice-based, multivariate, hash-based, code-based cryptography) [201]. For example, Yin et al., [201] implemented the anti-quantum transaction authentication scheme using lattice-based cryptography. This study [213] presents the post-quantum blockchain using a lattice-based delegation algorithm.

6.3.9. Endpoint Vulnerability

The easy way of attacking technology solutions is through endpoint vulnerabilities, which occur where humans and technology interact [197]. Hence, the protection of endpoints is paramount in BbHAs [197]. The attacker coerces the victim through social engineering, phishing, or other strategies that are under his control [13]. For example, the flawed key generations and signatures tool exposes users' private keys. Moreover, the lack of awareness and knowledge about security could trigger the endpoint vulnerability [197]. For example, if the attacker learns about the private key, he can utilize it to acquire access and ownership to data. Anyway, endpoint vulnerabilities remain susceptible through social engineering, real-world theft, or physical access to user wallets, phones, or computers [197].

To minimize endpoint vulnerabilities, implement multi-level authentication method [13] when accessing wallets or generating wallet keys, use multi-signature

wallets, cold wallets, and do not share private keys of wallets with anyone. Users should be aware of social engineering, always use authentic and legitimate sources to protect against phishing, and utilize hardware security modules [13, 197].

We build this framework (Table 43) aiming to provide the details about the security threats that may appear in BbHAs, and the controls to mitigate them. Both frameworks (Table 42 & 43) complement one another in the context of the SRM constructs we used. However, the aforementioned frameworks represent the knowledge base in a static manner and are difficult to update when new security threats, vulnerabilities, or countermeasures appear. To overcome these issues, we build a permissionless blockchain-based healthcare security ontology, HealthOnt, where these frameworks serve as a foundation.

6.4. Healthcare Security Ontology

HealthOnt is publicly available, OWL-DL based healthcare security ontology that encapsulates the security threats of THAs and BbHAs (Table 44). We utilize the ontology construction method [64] that has been applied in the previous chapter to build CordaSecOnt. We start the ontology-building process by identifying its purpose and scope. Then, we collect the domain information (e.g., concepts and relations) and categorize it in the frameworks (Table 42 & 43). This process refines the concepts and improves the technical domain language related to system assets, business assets, security criteria, threats, vulnerabilities, and countermeasures. Thereby, the frameworks provide a coherent structure and required level of understanding for a successful implementation of HealthOnt. HealthOnt can support the iterative process of SRM of THAs and BbHAs, and it can be updated continuously when new security threats, vulnerabilities, or countermeasures emerge. We use the Protégé to code the concepts and categorize them in taxonomic structures for refining the concepts and improving the technical domain language related to the system assets, business assets, security criteria, threats, vulnerabilities, and countermeasures. HealthOnt, like CordaSecOnt, is built using the ULRO; hence, the class definitions and DLs of classifications are identical to the CordaSecOnt we discussed in Chapter 5. Therefore, in this section, we only present the classifications of the coded concepts linked to the healthcare domain and skipped the class definitions and DLs.

Table 44. HealthOnt resources

Resource	URL
GitHub	`https://github.com/mubashar-iqbal/HealthOnt`
MMISW repo.	`https://mmisw.org/ont/~mubashar/HealthOnt`

6.4.1. Assets Classification

The asset class has two sub-classes, and restrictions hasConstraint and supports. For example, we identify and classify the assets as business and system assets

(Fig. 54). Security criteria is a constraint of business assets, and system assets support the business assets. For example, business asset "MedicalRecord" hasConstraint "Integrity", and System assets "AccessControl" supports "MedicalRecord".

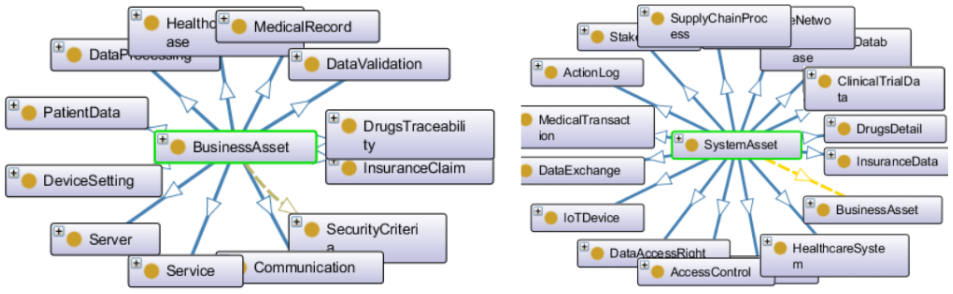

Figure 54. Business and system assets classification

6.4.2. Security Threats Classification

Security threats classification (Fig. 55) is built upon the threats that exploit vulnerabilities and target system assets. For example, the attacker can trigger “DataTampering” threat by exploiting a vulnerability “ErrorProneAuthenticityOfData” or “WeakAccessControl”. The “DataTampering” threat targets the system assets (e.g., AccessControl, HealthcareDatabase, or MedicalTransaction).

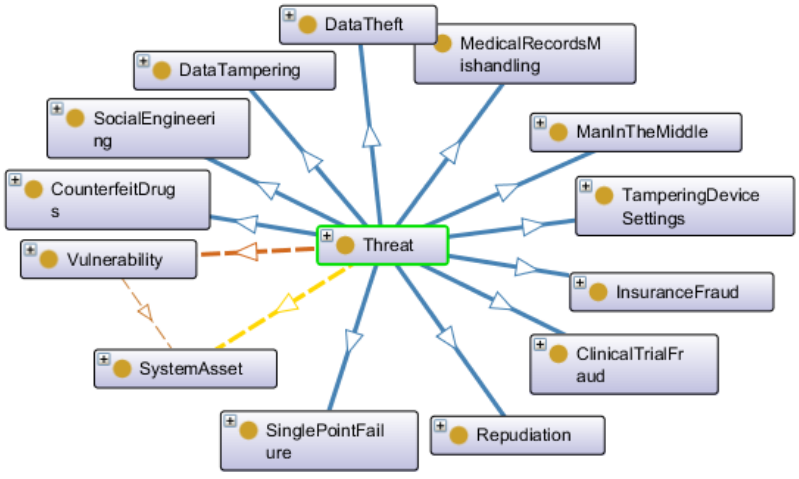

Figure 55. Security threats classification

6.4.3. Vulnerabilities Classification

Vulnerabilities classification (Fig. 56) is built upon the weaknesses in healthcare applications that enable the security threats. The vulnerabilities are the characteristic of system assets and negates the security criteria of business assets. For

example, a vulnerability “WeakAccessControl” is a characteristicOf "AccessControl" and "HealthcareDatabase", and negates their integrity or confidentiality.

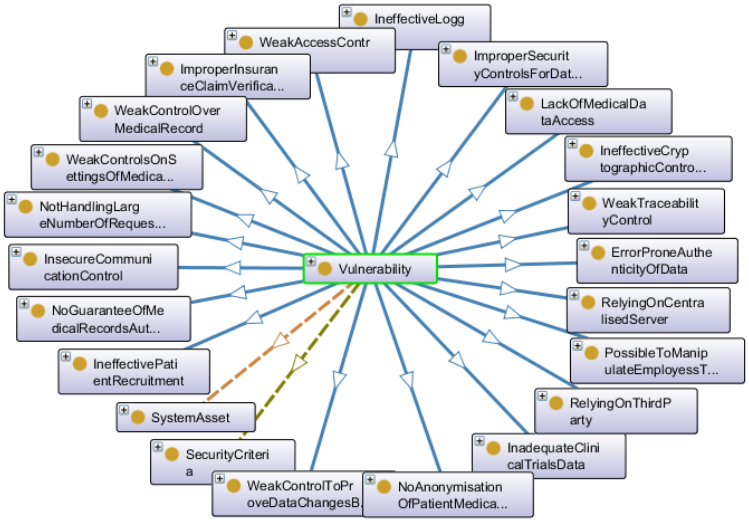

Figure 56. Vulnerabilities classification

6.4.4. Countermeasures Classification

Countermeasures classification (Fig. 57) presents the counteract to mitigate the vulnerabilities for improving the system’s security. For example, the countermeasure “DistributedAccessControl” mitigates “WeakAccessControl” vulnerability.

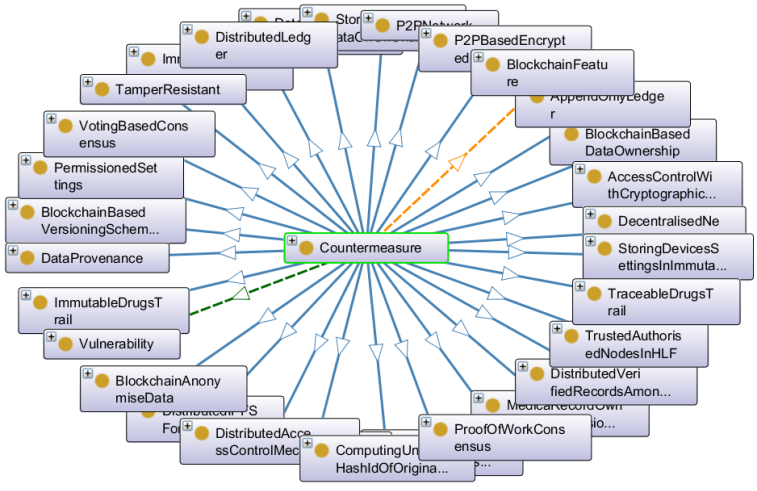

Figure 57. Countermeasures classification

6.5. Evaluation

Ontology evaluation is important to ensure the correctness of ontology, the meaning of ontological reasoning, and the effective use of an ontology [202]. For the HealthOnt, similar to the CordaSecOnt, we use the automated tools (e.g., pellet reasoner and ontology pitfalls scanner), the qualitative assessment criteria [206], and tasks-based evaluation using the SPARQL queries. Such approaches help to check whether the coded concepts model the real-world domain for which the ontology is built, and contribute to the quality of ontology. However, these approaches do not address how good and applicable the developed ontology is? Therefore, we use a back-pain patient's healthcare application to answer this question to map the HealthOnt.

6.5.1. Analysis of Back-pain Patients' Healthcare Application Using HealthOnt

We use HealthOnt to map the healthcare applications' security knowledge on a BPPHA, described in Section 6.1. HealthOnt helps to identify the BPPHA security threats that are highlighted as threat points in Fig. 58.

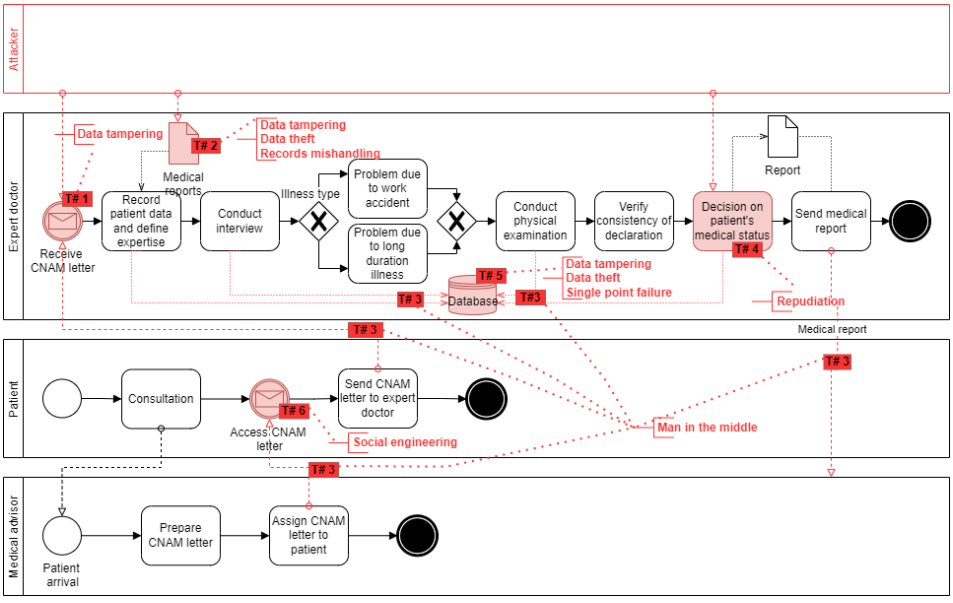

Figure 58. Mapping of threats that can appear in traditional BPPHA

Threat point 1: In BPPHA, due to the weak access control to share CNAM letter, the unauthorized user (e.g., attacker) can get access to the CNAM letter and tamper it. Also, there exists a possibility that a patient can intentionally or unintentionally tamper with the CNAM letter. In both cases, the system does not have a proper mechanism to verify and validate the legitimacy of the CNAM letter.

Threat point 2: In BPPHA, the data tampering threat can happen on medical reports because the system has a weak centralized access control mechanism. Thus, an unauthorized user can access the patient's medical reports and tamper with them. In the current system, the medical reports are stored in human-readable formats (e.g., PDF, docs, Xrays, etc.), and no proper cryptographic controls (e.g., encryption) are implemented. The attacker can access them to pursue various activities (e.g., insurance frauds, wrong drug prescriptions, and ransomware attacks). The data theft undermines the confidentiality of medical reports and patient privacy, eventually jeopardizing the integrity and trust in the BPPHA. Furthermore, the patients have weak control over their medical reports in BPPHA. For example, medical institutions control and manage patients' medical data where non-relevant individuals can access and manipulate it. Also, the BPPHA does not guarantee the authenticity of electronic medical reports.

Threat point 3: The attacker can exploit the weak controls of secure communication to get medical records, medical reports, and CNAM letter. Moreover, due to the lack of anonymization of data, the medical records are associated directly with the patient's identity. With the MitM, the attacker can sniff the data to pursue various activities (e.g., publish the data online or deny a patient access to it, or ransomware attack). A successful MitM attack can affect the data exchange, medical records, medical reports, CNAM letter, and communication assets.

Threat point 4: The patient's medical data is life-critical, and the healthcare system should trace all actions performed (either intentionally or unintentionally). The BPPHA does not use immutable logs to maintain track of all actions taken on a patient's medical data over time. As a result, the existing system lacks a means for proving the modifications on a decision of a patient's medical status.

Threat point 5: Weak centralized access control in BPPHA refers to a situation in which the system fails to prevent unauthorized access to the database. The attacker breaches security and performs unauthorized actions that negate the integrity of medical records and healthcare database. Currently, the BPPHA does not have any security or cryptography controls to protect the database from data theft attacks. Overall, this threat negates the confidentiality of medical records, healthcare database, and patient privacy. Also, the BPPHA has a centralized database server and network services. The attacker can locate the flaw in the design or implementation of the systems and cause database overhead or disables the medical services, essentially shutting down the whole system.

Threat point 6: BPPHA is also vulnerable to social engineering, where patients and hospital personnel are the weakest links. The attacker can target them using social engineering tactics to get the CNAM letter.

6.5.2. Blockchain as a Countermeasure Solution

In this section, we present blockchain as a countermeasure solution. Figure 59 illustrates the blockchain-based BPPHA that implements various security controls

by design and mitigates security threats of traditional BPPHA. For example, the blockchain-based BPPHA enables role-based access control (RBAC), which is a decentralized distributed access control to restrict who has access to the CNAM letter. Moreover, it provides a consensus mechanism to verify and validate the CNAM letter transaction without requiring a third party, a unique hash id of the original CNAM letter stored in the blockchain to verify its authenticity, and an immutable ledger to keep track of each performed action. Similarly, medical reports can be protected against data tampering using RBAC and blockchain-based controls to verify and validate the authenticity of medical reports. RBAC and the use of cryptography (e.g., to store only encrypted medical reports on-chain and off-chain) overcome the threat of data theft. Also, the permission settings and RBAC enable patients to control their medical reports, and the tamper-resistant environment of blockchain guarantees the authenticity of medical reports.

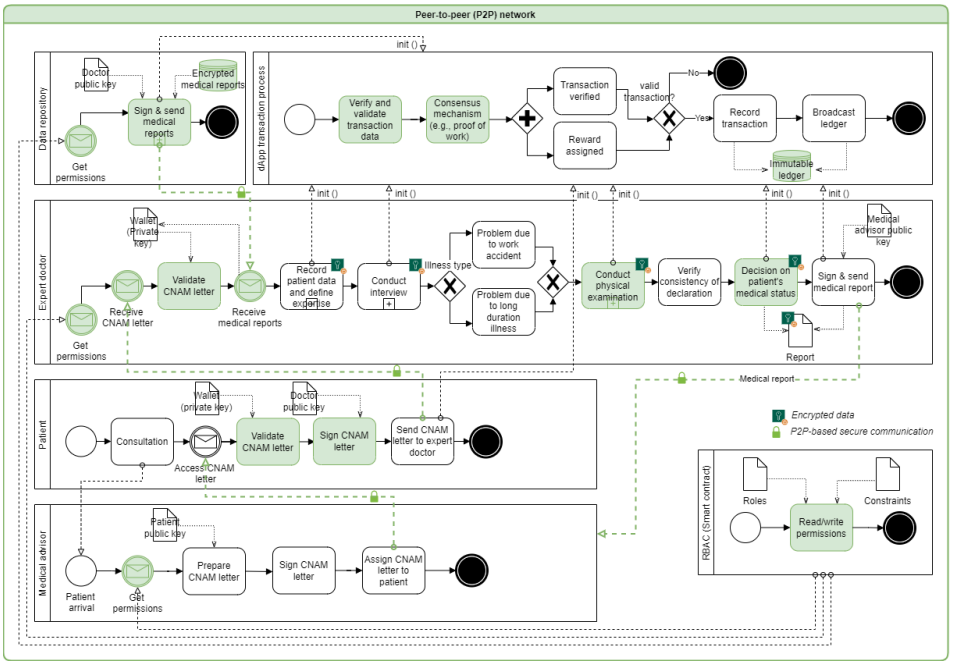

Figure 59. Blockchain as a countermeasure for the threats of traditional BPPHA

Blockchain-based BPPHA works on a P2P-based distributed network to exchange data (e.g., CNAM letter, medical reports, medical records). It makes it hard for an attacker to intercept the communication, data analysis, or sniffing. Blockchain enables pseudo-anonymity because in BPPHA the patients and their medical data are linked with a public address. Blockchain-based BPPHA has an immutable ledger that keeps immutable logs to track who and when the particular operation (intentional or unintentional) was performed. Thus, overcoming the repudiation threat. Medical records and healthcare database can be protected against data tampering by using RBAC and blockchain-based controls to verify

and validate the authenticity of medical records and healthcare databases. Decentralized and distributed access control (e.g., RBAC) and cryptographic (e.g., to store only encrypted medical reports on-chain and off-chain) overcome the threat of data theft. BPPHA is decentralized, operates over a P2P network, and does not rely on a single or central point server and service. Thus, it is resilient to a single-point failure. Blockchain-based BPPHA employs RBAC to guarantee that only relevant people have access to specific information, and unauthorized users cannot access medical data in BPPHA.

6.6. Related Work

While healthcare applications are getting ubiquitous, researchers are working to improve the security and privacy of these applications to an acceptable level. However, a number of surveys and literature studies have only focused on the technical perspective of security threats in healthcare applications (Table 45). The studies neglected the business context and the impact of security threats on business assets, also not following the SRM domain model to describe the relationships of security threats with the system. Moreover, the THAs are not fully leveraging the benefits of emerging technology (e.g., blockchain).

For instance, the authors [251, 252] review the common security threats and corresponding countermeasures considering only the technical side of the healthcare systems components. Similarly, Wani et al. [253] investigated a few notable vulnerabilities in the hospitals connected to bring-your-own-device usage, the study reviews the countermeasures to mitigate them. Still, they do not explicitly pinpoint the assets in the healthcare system targeted by the security threats and what business assets to protect. Sardi et al. [254] explore the variety of security threats in healthcare facilities solely and briefly mention key assets. They lack risk assessment based on the specific needs of healthcare facilities and processes. Some studies focus on controls to secure complex mobile, ubiquitous, and connected IoT healthcare systems [255, 256]. However, they do not consider the context of such measures and do not describe how they can contribute to EHRs protection. At the same time, Yeng et al. [257] present a relatively complex security and privacy analysis of healthcare systems by investigating what assets to protect in healthcare, their vulnerabilities, and countermeasures. Most of the literature reviews similar to our work present a rather limited scope of analysis. Noteworthy also that only a few studies mention blockchain technology as a countermeasure to THAs' security threats. However, various organizations started working on BbHAs; for example, IBM-Blockchain [258] is integrating blockchain in healthcare for better data sharing between healthcare providers without compromising data security, to overcome the drug counterfeiting [227], and so on.

In recent years, blockchain technology has gained interest in the healthcare domain, and researchers presented blockchain as a countermeasure solution to mitigate security threats of THAs. For example, Saha et al. [14] present a com-

Table 45. Summary of related work w.r.t the SRM of traditional healthcare applications

Paper reference	Domain	Findings and addressed topics					
		Threats/Risks in healthcare	Counter-measures	Blockchain as counter-measure	Business/ System Assets	Vulnerabilities in healthcare	Use of SRM model
Fatima and Colomo-Palacios (2018)	Healthcare; Security	◉	◉	○	○ / ○	○	○
Ahmadi et al. (2019)	Healthcare; Security	○	◎	○	○ / ◎	○	○
Iwaya et al. (2020)	mHealth and uHealth; Security & Privacy	○	◎	◎	○ / ○	○	○
Sardi et al. (2020)	Healthcare; Security	◉	○	○	◎ / ○	○	○
Wani et al. (2020)	Healthcare; Security	◉	◉	○	○ / ○	◎	○
Semantha et al. (2020)	Healthcare; Privacy	◉	◉	◎	◉ / ○	○	○
Aljedaani and Babar (2021)	mHealth; Security	◎	◎	○	○ / ○	○	○
Yeng et al. (2021)	Healthcare; Security	◎	◎	○	◎ / ○	◎	○
HealthOnt	Healthcare; Security	◉	◉	◉	◉ / ◉	◉	◉

◉ - detailed discussion; ◎ - limited discussion; ○ - not addressed

parative analysis of healthcare applications that use blockchain-based healthcare solutions to protect against data tampering and data leakage. The survey of [54] addresses the security and privacy concerns in healthcare. The authors explore the timeline of security attacks on medical data and various traditional security algorithms to defend against them. The traditional security algorithms are shown to be ineffective, and blockchain is used as an advanced architecture for the safe and secure execution of medical transactions and to maintain the security and privacy of digital medical records. Koo et al. [25] describe the fundamental principles of blockchain to address the security and privacy issues of THAs. The study also discusses the technical advantages of blockchain in healthcare (e.g., faster and easier interoperability). This study [259] presents the different use cases to address the security and interoperability challenges of THAs. Chukwu et al. [260] perform the SLR to explore the trust, security, and privacy constraints of traditional EHRs and how blockchain plays a role in overcoming them. The SLR of [261] investigates the security challenges, including how blockchain can protect medical data from potential data loss, corruption, or intentional security attacks. Jin et al. [262] present blockchain in healthcare for secure and privacy-preserving medical data sharing. The study argues that blockchain's tamper-evidence and decentralization features could help build a secure medical data-sharing network.

The related works explore various security aspects without addressing vulnerabilities, what assets to protect, blockchain characteristics, and not adhering to any SRM domain model. Furthermore, the related works do not address the security threats and vulnerabilities that may arise in BbHAs. In contrast, we use the SRM domain model to analyze and compile the security threats of THAs and BbHAs. We also investigated the countermeasures to minimize them. To ease the SRM of healthcare applications, we provide an SRM domain model-based ontological framework that offers a dynamic knowledge base of security threats of THAs and BbHAs, vulnerabilities, what assets to protect, and countermeasures to mitigate the security threats of both THAs and BbHAs.

6.7. Challenges and Implications

In this section, we outline emerging security and implementation challenges (e.g., scalability, privacy, regulatory) in blockchain systems, research challenges (e.g., threats to validity), and research implications. For instance, there exist numerous possible security threats (Table 46) that have yet to be studied in BbHAs but may appear. Therefore, blockchain developers and practitioners should be aware of these security threats. Furthermore, blockchain technology is advancing in the healthcare domain, and along with the security issues, it is also facing scalability, privacy, and regulatory challenges.

Table 46. Security threats not yet investigated in BbHAs but may appear

Threat	Detail
BGP hijacking	The attacker can intercept the blockchain network by manipulating the border gateway protocol (BGP), after which data can be routed, and the traffic can be modified to the attacker's favor [243].
Liveness attack	This attack can delay the transaction confirmation time and proceeds in three stages: preparation (build private chain), transaction denial (delay the genuine block), and blockchain delay (decrease the rate at which the chain transaction grows) [243].
Timejacking	Timejacking exploit the handling of blockchains' timestamps. The attacker can forge or broadcast a false timestamp of a transaction when connecting to a network node allowing him to change the node's network time and trick it into accepting an alternative blockchain. This attack can cause a double-spending [247].
Blockchain poisoning	The attacker add stolen data (e.g., addresses, credit card numbers), illegal files (e.g., malware), and malicious content and force blockchain nodes to download such content [239]. Blockchain poisoning can lead to DoS or DDoS attacks or disrupt the operations of a blockchain network.
Transaction malleability	The attacker alters the transaction signature responsible for generating unique identifiers of the transaction. The attacker changes the transaction identifier before the transaction confirmation on the network to pretend the transaction did not happen. This technique causes the victim to pay twice [239, 247].
Selfish mining	This attack happens on mining pools to earn extra mining rewards. The attacker holds a mined block in his private chain. Once his chain is longer, he broadcasts the blocks in the network at once and makes other miners lose their blocks. The purpose of selfish mining is to waste the efforts and rewards of honest miners [239, 246].
Balance attack	Balance attack combines mining power with communication delay to affect fork-able blockchains (e.g., Ethereum). The attacker isolates a blockchain branch from one subgroup and convinces another competing subgroup to influence the branch selection process. This successful attack can lead to a double-spending [243].
Race attacks	In race attacks, the attacker sends two or more conflicting transactions in the network and exploits the fast transaction mechanism where the merchant (a victim) accepts a transaction with 0 confirmations [238].

Scalability is a big challenge in blockchains' widespread adoption [239]. For example, blockchains have prefixed block size, block creation time, and process a fixed number of transactions per block. These settings help to achieve immutability, tamper-evident feature, ledger redundancy, and decentralized verification and validation of transactions but make the transaction processing (throughput) slow. For example, the Ethereum platform processes only 15 transactions per second [263]. Also, blockchains maintain a ledger starting from the first (genesis) block that grows over time (e.g., Ethereum full node sync size is now 1+ Terabytes and increasing[2]). The blockchain network shares the ledger with all the participant

[2]`https://etherscan.io/chartsync/chaindefault`

nodes. Therefore, each node requires tremendous network resources and storage to store the ledger. Various solutions are explored to overcome scalability issues (e.g., permissioned blockchains, lighting protocol, sharding, delegated proof of stake, directed acyclic graph) [243]. These techniques can help to increase the volume of transactions, although more work is needed in this direction.

Privacy is the main concern in permissionless blockchains that have privacy issues by design [11]. The ledger is disseminated across network nodes, and transactions are publicly accessible. The attacker can utilize the ledger and apply different approaches (graph analysis, social engineering, phishing, transaction linkage) to track user activity and get private information. These privacy concerns are growing, and it is restraining the use of blockchain in healthcare applications since it distributes personal information in a publicly accessible database [11]. To overcome privacy challenges, different privacy-preserving proposals (e.g., secure multi-party computation, zero-knowledge proof, homomorphic encryption, ring signatures, transaction mixers) [264] and blockchain platforms (such as Enigma, Zcash, Monero) [265] are advancing to preserve the privacy of confidential information. These solutions primarily address overall transaction privacy in cryptocurrency transactions. Therefore, more research is required in the area of privacy-preserving blockchains for healthcare applications.

Regulatory challenges emerge in blockchain due to disintermediation where nobody takes responsibility for providing services, controls, and associated data sets. Privacy laws (e.g., EU general data protection regulation (GDPR), health insurance portability and accountability act (HIPPA)) can overwhelm the standardization and regulations for BbHAs. For example, under GDPR, the users are controllers of their data, but the immutable ledger cannot let the user delete (or update) their data [266, 11]. In governance, a key question for regulators is who should be held accountable for breaches of laws and regulations. Many organizations are collaborating on regulatory guidance (such as a legal framework for data storage and sharing over blockchains) [11]. However, more research is needed to standardize blockchains for healthcare applications.

To ensure the quality of empirical research, we expanded our discussion by outlining the challenges related to the **threats to validity** [169]. The relevant threats are restricted time span, publication bias, subjective interpretation, and lack of expert evaluation. The researcher cannot forecast further relevant studies beyond the defined time period because of the restricted time span. For example, blockchain is a relatively new technology that is constantly evolving. As a result, a wide range of countermeasures will arise in the future. The publication bias is the tendency of linked research to disclose good outcomes rather than negative results. The threat of subjective interpretation exists since we might have different interpretations and opinions related to identified threats, vulnerabilities, and countermeasures. Moreover, a lack of expert evaluation may also lead to a subjective interpretation and erroneous conclusion.

Security risk management is an iterative process because new security threats and vulnerabilities can emerge. For example, BitMex research reported a double-spent transaction in Bitcoin on the 21st of Jan 2021 [168]. The source and attack method of this double-spend transaction is not yet known. However, it indicates that the attacker continuously builds new techniques to sabotage the integrity, confidentiality and availability of the blockchain systems. To communicate security threats to blockchain developers, practitioners, and other associated stakeholders, we build an ontology-based comprehensive blockchain-based healthcare security ontology using the concepts of the SRM domain model. The ontology representation presents blockchain as a countermeasure solution and supports the decision to build blockchain-based healthcare applications to mitigate the security threats of traditional healthcare applications. This work also addresses the issues related to the conceptual ambiguities and semantic gaps when performing the SRM of traditional and blockchain-based applications. Compared to the previous works, the ontology representation we presented encodes the information into a dynamic ontology-based knowledge that can be extended, reused, or integrated with other security ontology representations. Also, it supports the iterative process of SRM and can be updated continuously when new security threats, vulnerabilities, or countermeasures emerge. Our findings hint at a distinct set of SRM-related traits, information sources, decision techniques, and attitudes toward knowledge management that may distinguish the work from the existing SRM approaches. Although the practical implementation of the ontology representation using real-world healthcare application is outside the scope of the current work, future research implications have already been formed to see how this might be accomplished in the near future. Moreover, our work opens up other avenues for future research, both in terms of theory development and concept validation. Therefore, further research is required to improve and expand on our findings. The discussion section expands on such potential future research implications.

6.8. Discussion

For this contribution, rather than focusing on a single use case or blockchain, we picked general healthcare applications and permissionless blockchains. According to the SLR (Section 2.4), healthcare is the most studied domain using permissionless blockchains. Permissionless blockchains are more transparent and decentralized as compared to the permissioned blockchain and provide robust data integrity. Therefore, permissionless blockchain-based applications are becoming apparent in data-centric applications (e.g., healthcare). We present a permissionless blockchain-based healthcare security ontology that offers coherent and formal information models to treat security threats of THAs and BbHAs [267].

Our current research has several limitations. For instance, similar to the previous contribution, the current approach lacks a comprehensive analysis of security threats. The security threats we obtained focus on the asset level, neglecting

technical specifics and defects that constitute a source of security threats. The HealthOnt was used to perform the SRM of a real-world running healthcare application. That healthcare application, however, has not yet been built. Therefore, the current ontology representations are not yet formally validated with a real-world application. It is necessary to explore how domain experts (e.g., healthcare specialists, blockchain engineers, and security analysts) perceive the significance of HealthOnt's contribution to deriving the missing security components, to determine the comprehensiveness and technical correctness of the healthcare system. As noted in Sections 6.2.11 and 6.3.9, the human aspect is crucial in healthcare. Hence, it is pertinent to investigate how blockchain might overcome challenges linked to a human factor and the relevance of various vulnerabilities associated with human interaction. We addressed these constraints in our future work that will extend to the current work.

6.9. Summary

This chapter answers our RQ4: How might permissionless blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissionless blockchain-based applications? As a result, we present a permissionless blockchain-based healthcare security ontology, HealthOnt, by extending the ULRO. HealthOnt includes security risk analysis of traditional healthcare applications (THAs) and blockchain as a countermeasure. HealthOnt supports the decision to build blockchain-based healthcare applications (BbHAs) to mitigate the security threats of THAs. However, blockchain-based applications are also not secure because there exist several ways to negate their security. Thus, the HealthOnt includes the security threats that can appear in the BbHAs. In HealthOnt, we define the classifications of system assets, business assets, security criteria, threats, vulnerabilities, and countermeasures. HealthOnt is publicly available and encodes the information into a dynamic ontology-based knowledge that can be extended, reused, or integrated with other security ontology representations. HealthOnt can support the iterative process of SRM and can be updated continuously when new security threats, vulnerabilities, or countermeasures emerge.

To evaluate the HeathOnt, we use the automated tools (e.g., pellet reasoner and ontology pitfalls scanner), the qualitative assessment criteria, and tasks-based evaluation using the SPARQL queries. In addition, we use the back-pain patients' healthcare application to map the coded knowledge of healthcare security using HealthOnt. The results show that HealthOnt can support the SRM of THAs using blockchain as a countermeasure. Also, it can assist in modeling and analysis of real-world situations and helps to address the important (from a stakeholder point of view) security concerns.

7. WEB-BASED ONTOLOGY PARSING TOOL

Previous chapters present blockchain-based security ontology representations in finance (CordaSecOnt) and healthcare (e.g., HealthOnt). Both representations help in the SRM of traditional and blockchain-based applications, reuse, and maintainability of the information security knowledge by combining the blockchain as a countermeasure solution. However, practitioners must first master a few techniques before they can begin using the ontology. For example, one should learn about the concepts of ontology and how to interpret encoded information, how to traverse around the concepts using Protégé or another tool, and how to obtain structured information using SPARQL queries. Such dependencies may hurdle the use and applicability of both ontology representations. Therefore, to address these problems, we build a web-based ontology parsing tool (OwlParser) [268] to leverage the developed ontology representations without acquiring knowledge of ontology, an editing tool, or querying the concepts through SPARQL queries. OwlParser provides a web-based explorer feature that allows one to dive into the SRM concepts and assist the practitioners in finding and navigating through the concepts that fit their needs. To assess the usability of the OwlParser, we conduct interview-based surveys with the experts. The survey yielded positive findings since the participants were able to use the tool in the context of understanding the SRM of traditional and blockchain-based applications and discovered that the tool allows easy examination and navigation through the encoded concepts.

The remainder of the chapter is structured as follows: Section 7.1 addresses the implementation of the OwlParser, in which we show the architecture as well as explain the web interface and guidelines to use it. Section 7.2 is the evaluation of the tool where we approached the experts to test its usability. Section 7.3 concludes this chapter.

7.1. Implementation

The OwlParser is built using ReactJs[1]. The tool is freely available and accessible online[2]. The code is available on GitHub[3], and it can be extended with the new features in the future. This section goes through the architecture and resources that we used to create the OwlParser. Furthermore, we explain the web interface and guidelines about using the OwlParser.

7.1.1. Architecture

The architecture of OwlParser (Fig. 60) consists of five main components (interface, parser, pre-processor, mapping, navigation tree builder) that are intercon-

[1] https://reactjs.org

[2] https://owlparser.cs.ut.ee

[3] https://github.com/mubashar-iqbal/OwlParser

nected to interact with each other for parsing the content of ontology and displaying it in a structured format on the web interface. For example, the loading interface allows uploading the ontology file from the computer or loading it from the MMISW ontology repository according to the provided internationalized resource identifier (IRI). We use the npmjs package $rdfxml-streaming-parser$ to parse the ontology file. This package takes the RDF/XML data and outputs RDFJS-compliant quads. The pre-processor takes the parsed content and extracts information based on the defined triplets relationships (e.g., subject-predicate-object). The extracted information maps on the SRM domain model constructs, for example, the classes and relationships mapping. This activity allows the user to verify the correctness of the mapping, and if the mapping is wrong, the tool allows the user to change and correct it based on the SRM domain model. The information about concepts and relationships is found and dynamically creates a navigation tree on the web page. Once the navigation tree is ready, the ontology content displays on the browsing interface. The current version of the tool is intended to work with such ontology representations that follow the SRM domain model. The tool also presents some extra features, for example, threats and countermeasures filtering depending on the type of application (e.g., traditional or blockchain-based application).

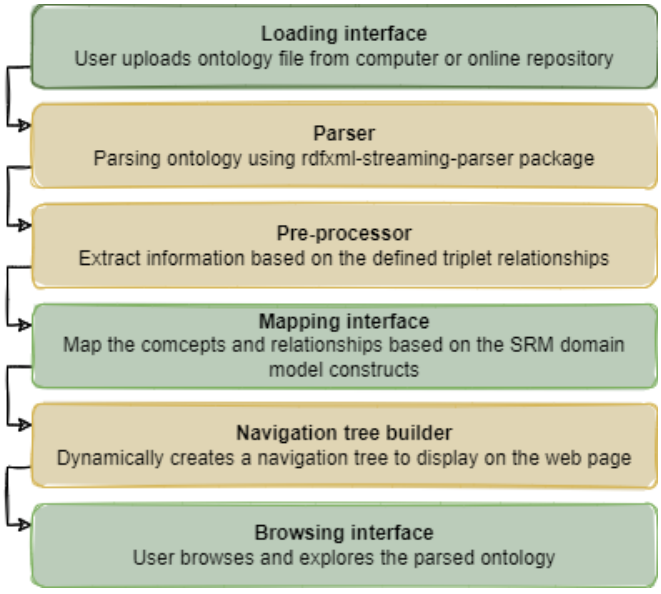

Figure 60. Web-based ontology parsing tool architecture

7.1.2. Web Interface and Guidelines

A landing page is the loading interface (Fig. 61) that has the option to upload the ontology file from the computer or the web using an IRI. Moreover, there is an option to load an example ontology. When the ontology is loaded, the application

provides an interface to map the classes and relationships (Fig. 62) in the loaded ontology according to the SRM domain model, which is pre-filled with guessed mappings. The mappings can be changed and must be confirmed to continue. After clicking “Continue”, the application displays a browsing interface with the parsed content of the ontology.

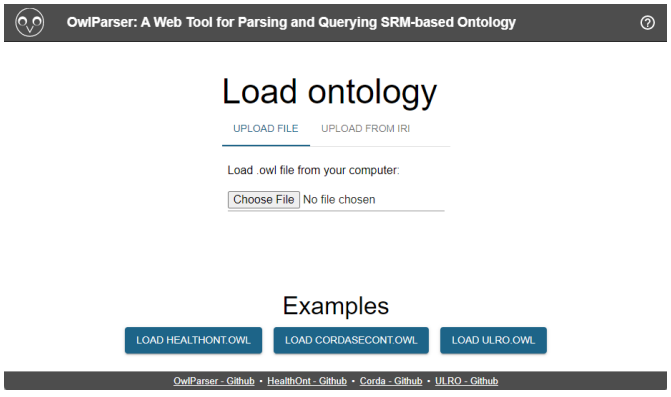

Figure 61. OwlParser’s loading interface

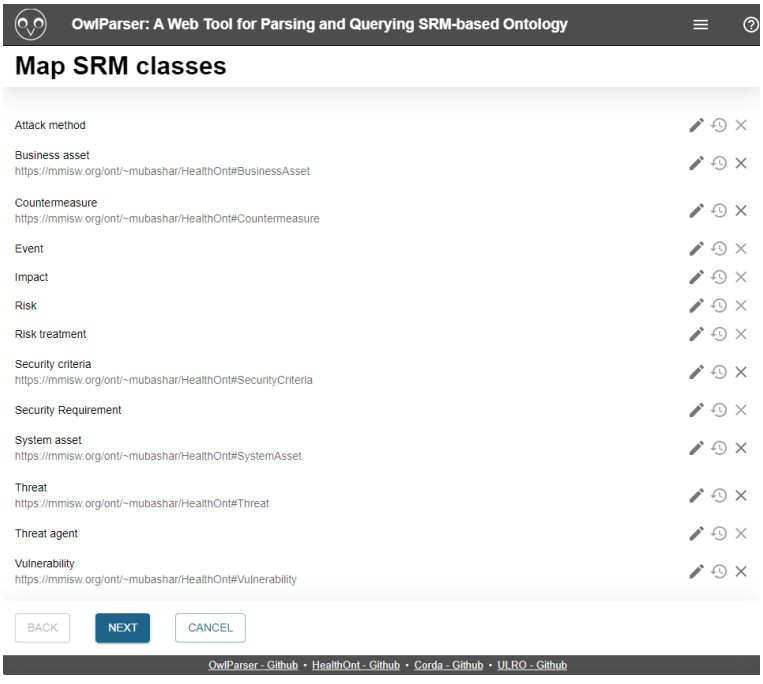

Figure 62. OwlParser’s mapping interface

The browsing interface (Fig. 63) consists of two panels. The left-side panel contains the SRM-based navigation tree and provides an option to hide or show the categories of SRM classes not present in the loaded ontology. An extra details section under the SRM-based navigation tree contains ontology classes that do

not belong to the SRM model. The left-side panel also has a "Threats" tab which displays the threat classes based on the type of application they target. Clicking on a class link in the left panel (e.g., DataTampering) displays information about that class in the right-side panel. Also, the concepts displayed on the right have links, and clicking them opens the details of the concept.

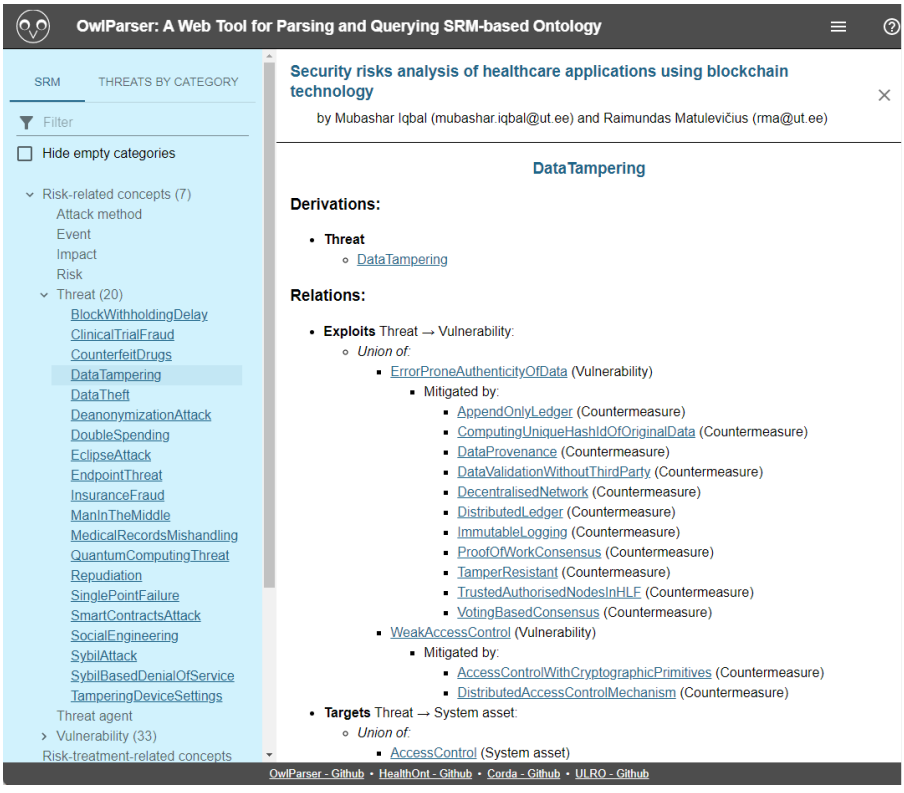

Figure 63. OwlParser's browsing interface

7.2. Evaluation

We undertake usability testing to see whether the implemented tool is capable of leveraging the ontology representations. We engage target users and conduct usability testing to determine how easy the application is to learn and use, what flaws exist in the design, and chances to improve the application, as well as to learn about the target users' behavior and preferences [269]. In this section, we describe the study design first, followed by the results of usability testing.

7.2.1. Study Design

We approach five target users (Table 47) from three user groups (e.g., blockchain and ontology specialist, security specialist, and software developer) to participate in the study. We conduct the interviews with each of them separately based on

the predefined interview protocol. The interview protocol consists of a short introduction of the tool, asking the interviewee to perform a few tasks using the OwlParser. Later, we ask about the predisposition of the user and the interview questions (Appendix A) covering the area of OwlParser's applicability, ease of use, ease of learning, task efficiency, subjective satisfaction, and understandability. The interview questions eventually answered about OwlParser's usability. For the evaluation purposes, we assume that the target users include software developers, security specialists, and blockchain and ontology specialists who are willing to learn about blockchain security, specifically about the concepts concerning the SRM of traditional applications using blockchain as a countermeasure solution. We create a list of tasks (Table 48) to perform by using the HealthOnt ontology.

Table 47. List of participants

	Expertise	**Mode**	**Time**
Expert 1	Blockchain, Ontology	Face-to-face	55 m
Expert 2	Security specialist	Face-to-face	45 m
Expert 3	Software development	Face-to-face	55 m
Expert 4	Software development, Security specialist	Face-to-face	40 m
Expert 5	Software development, Blockchain, Ontology	Zoom	50 m

Table 48. List of tasks to perform by using the HealthOnt

Id	**Task**
Task 1	Load the ontology file and map the concepts according to the SRM domain model.
Task 2	Find out which business assets does AccessControl system asset support?
Task 3	What are the vulnerabilities that are exploited by the data theft threat?
Task 4	What are the countermeasures to mitigate the vulnerabilities related to the data theft threat?
Task 5	Find out the threats that may appear within blockchain-based applications.
Task 6	Find SRM domain model categorize including the empty categorize.

The facilitator (e.g., who conducts the interview), tasks, and participants are the three main components of usability testing (Fig. 64). The facilitator assigns tasks to participants and takes them through realistic activities they perform in real life [269]. The tasks we develop are based on practical activities to be completed utilizing the HealthOnt. The tasks are open-ended, and the facilitator gave them both vocally and in writing to the participants. The participant gives behavioral and verbal feedback on the tasks, operations, and interface he uses while executing those tasks. For example, participants are frequently requested to think aloud, and the facilitator may urge them to narrate their actions and thoughts. On the basis of the OwlParser tool we constructed, we create three user groups and approach participants based on our personal understanding of their abilities as our target users and inquire about their willingness to participate in the study.

We conducted four face-to-face interviews, and one was conducted online using Zoom. First, we explained the tool for five minutes to participants and then assigned ten minutes to perform the defined tasks. After the tasks were completed, we asked the two task-related questions. For example, how complicated were the tasks? and did you remember how to start the task? Each participant spent 45-

Figure 64. Usability testing flow, adapted from [269]

60 minutes on average during this procedure. Overall, the tasks were completed efficiently by the participants. However, there were some inconsistencies; for example, one participant confused the system and business assets and required a hint to complete task 2. Another participant had difficulty locating the empty category. Next, we accumulate and discuss the results of expert interviews.

7.2.2. Results

The qualitative metrics are the main emphasis of the usability testing, such as gathering data on how users interact with the OwlParser. The findings from the usability testing are listed below.

Applicability: According to experts, it is difficult for non-IT practitioners to comprehend the ontology's relationships and overall difficult to interpret the knowledge in it. For instance, non-IT practitioners may utilize this tool to examine relationships easily and browse through different terms. Two experts supported using the ontology to illustrate blockchain security since many people believe that blockchain is a security, but the issue is that blockchain is also vulnerable. They believed that employing such tools would make it simpler to explain and educate the individuals involved in the blockchain field. One expert had constructed the ontology in healthcare for conflict resolution using Protégé and validated it with an ontology reasoner called Hermit. He had never used such web-based ontology parsing tools before. In comparison to the Protégé, he believes that this tool allows him to investigate ontological knowledge more rapidly. One expert encountered some difficulties in locating the ontology file to upload. He also urged to give a download option if someone wanted to investigate it in the ontology editor. Another expert was unfamiliar with ontology. According to him, utilizing this tool appears to be model-driven engineering, where we construct a model and define the fundamental elements of the model and their interactions.

Ease of use: The loading page is simple and straightforward, and it loads quickly. On the web interface, mapping is instinctive, and knowledge is structured based on this mapping. The tool is simple in that and accomplishes what is expected of it. One expert proposed adding a previous button to return from this section to the prior one, rather than just canceling. For him, it was also difficult to understand the starting point while using the browsing interface. One expert asked to explain the idea behind the mappings and mapping interface. He also advised using the hide button to conceal categories rather than the display button. Another expert was perplexed in locating the vulnerability. He proposed adding a search box to search for a particular threat and then displaying all connected information about that threat. One expert advised moving the load example ontology button since it gives the idea that one must click on it after uploading the file. One expert suggested testing the program by importing the large ontology to see how long it would take. Another expert advised that the concepts be organized alphabetically. Currently, the concepts are organized based on the SRM domain model.

Ease of learning: The brief description at the start helped the experts comprehend the tool better. Overall, the tool is simple and offers three primary interfaces: loading, mapping, and browsing. The information is properly structured in two panels, and there is little noise on these pages; just what is important is displayed. One expert suggested adding a tooltip to explain different terms. Another expert asked for a page that should explain the SRM domain model and its details. For example, he understood that system and business assets are the same in the beginning. According to him, adding details and definitions of terms will be easier for people to understand better the terms used.

Task efficiency: The tasks were adequately executed, and they were not difficult to complete. Furthermore, the interface did not disrupt or confound the tasks. The search bar would be an excellent addition to improving the efficiency of the tasks. For example, when the ontology is large enough, there may be a search option that can help in finding the knowledge. According to one expert, mapping is a bit unclear. Thus, he requested additional markup to identify whether mappings are incorrect or correct. Then it will be easier and faster to understand where I need to update the mapping.

Subjective satisfaction: According to one expert, obtaining information from ontology takes time. If I had to explain ontology to someone, it could take a long time to do so. This application makes the knowledge extremely well organized and easy to navigate. Protégé is a technical tool; however, with this web-based tool, one may go directly to any class, and knowledge will appear on the right panel. Another expert stated that it supports the SRM domain model and allows me to go from one category to another. One expert proposed increasing the gap between the rows since it appears overly dense.

Understandability: According to experts, they grasp the purpose and functionality of the tool since it was simple to scroll, loading ontology, and the overall design was intelligible. One expert said the description before utilizing the tool aided in understanding how to use it. He asked if it would be preferable to include such information on a page of the web tool.

According to the usability factors, the findings illustrate how easy it is for the end-users to use the OwlParser. Overall, usability testing enables us to determine if users can successfully execute given tasks and how satisfied they are with the OwlParser. It also assists in identifying the improvements that are necessary to increase the tasks' efficiency and the potential of using this tool.

7.3. Summary

This chapter provides a web-based ontology parsing tool to leverage the CordaSecOnt and HealthOnt without acquiring knowledge of ontology, an editing tool, or SPARQL queries. The OwlParser is built using ReactJs, and it is freely available online. The OwlParser consists of five components (interface, parser, pre-processor, mapping, navigation tree builder) interconnected to interact with each other for parsing the content of SRM domain model-based ontology and displaying it in a structured format on the web interface.

To assess the usability of the OwlParser, we conduct interview-based surveys with the five experts. The surveys yielded positive findings since the participants were able to use the tool in the context of understanding the SRM of traditional and blockchain-based applications and discovered that the tool allows easy examination and navigation through the encoded concepts.

8. CONCLUSION AND FUTURE WORK

This chapter outlines contributions concerning the research questions and opens up several prospects for future work, which we discuss in Section 8.2.

8.1. Conclusion

This thesis addresses the problem of traditional applications' security risk management (SRM) using blockchain as a countermeasure solution and the SRM of blockchain-based applications. The thesis produced five contributions to resolve these problems that collectively resulted in an ontology-based security reference framework for managing the security risks using blockchain. The framework also supports the SRM of blockchain-based applications. The framework establishes common ground and systematic understanding that can assist developers, researchers, practitioners, and other associated stakeholders by explaining how blockchain can mitigate security threats, what security threats may appear within blockchain-based applications, and what security countermeasures should be implemented. According to the research problem, this thesis addressed: How can a reference framework be developed for managing the security risks of traditional and blockchain-based applications? We proposed the four research questions, and our findings are summarized below based on these research questions.

RQ1: What is the current state of the art for performing the SRM of traditional applications utilizing blockchain and the SRM of blockchain-based applications?
We conducted a systematic literature review to explore blockchain security from two different perspectives. First, we identify the security threats of traditional applications (e.g., data tampering, data theft) and how blockchain can act as a countermeasure solution. Second, we scrutinize the security threats that may appear in blockchain-based applications (e.g., Sybil attack, Double-spending) and what countermeasures exist to overcome them. The answer to RQ1 resulted in a blockchain-based reference model (BbRM) comprising the fundamental concepts of the SRM domain model. The BbRM includes the blockchain components in addition to the countermeasures that address the security threats of blockchain-based applications. We evaluate the BbRM utilizing it to investigate the data tampering, Sybil attack, and Double-spending in detail. The results show that the BbRM can support the SRM of both traditional and blockchain-based applications.

RQ2: How can SRM domain model abstraction be used to build an ontology-based reference framework?
We use the SRM domain model to build the upper-level reference ontology (ULRO) that answers the RQ2. The ULRO provides semantic knowledge and fundamental concepts of the SRM domain model common to performing the SRM. The ULRO

enables a common foundation and provides an ontology-based structural representation that can support the dynamic knowledge encoding and instantiation with information security knowledge for the SRM of domain-specific applications. We use the smart healthcare application case for instantiating the ULRO using the data tampering and Sybil attack. To assess the correctness of the ULRO, we conduct an experts-based evaluation. The results demonstrate that the ULRO concepts and relationships adequately cover the aspects they aim to represent for dealing with the SRM. Also, we perform the colored Petri nets-based (CPNs) evaluation, considering the blockchain as a countermeasure solution. The evaluation allows us to know the ontology's feasibility to incorporate it during the system design and to understand the applicability in the context of its appropriateness for a particular domain, task, and application.

RQ3: How might permissioned blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissioned blockchain-based applications?

The answer to this question brings the permissioned blockchain-based security ontology. We perform the security risk analysis of the capital market's post-trade matching and confirmation process. We use Corda-based permissioned blockchain to mitigate the security threats in the traditional capital market's post-trade infrastructure. Further, we investigate the security threats of Corda-based permissioned blockchain and provide countermeasures to mitigate them. Using Protégé, we encoded the components of the Corda-based permissioned blockchain and presented the ontology, CordaSecOnt. CordaSecOnt provides the classifications of assets, security criteria, threats, vulnerabilities, and countermeasures. CordaSecOnt is evaluated using automated tools, qualitative assessment criteria, and tasks-based evaluation. The evaluation of CordaSecOnt determines the correctness of encoded concepts, its organizational fitness and has the potential to improve the SRM of the financial industry.

RQ4: How might permissionless blockchain be used for the SRM of traditional applications, and how to perform the SRM of permissionless blockchain-based applications?

To answer this research question, we perform a systematic literature review to define the healthcare applications' context and assets. Later, we perform a security risk analysis of traditional healthcare applications using permissionless blockchains and the security risk analysis of permissionless blockchain-based healthcare applications. The result of RQ4 is a permissionless blockchain-based security ontology. Using Protégé, we encoded the components of permissionless blockchains and presented the healthcare security ontology (HealthOnt). The HealthOnt compliments the CordaSecOnt because it follows the same foundations as we presented and used in the CordaSecOnt. For instance, the HealthOnt is an ontological representation of healthcare applications' security. Similar to Cor-

daSecOnt, the HealthOnt combines blockchain and provides the classifications of assets, security criteria, threats, vulnerabilities, and countermeasures. HealthOnt can support the SRM of both traditional and blockchain-based healthcare applications. Along with the automated tools, qualitative assessment criteria, and tasks-based evaluation, we take a back-pain patients' healthcare application to evaluate the healthcare security ontology. The results show that HealthOnt offers coherent and formal information models that delineate blockchain as a countermeasure solution for the SRM of traditional healthcare applications.

In addition to the contributions based on the research questions, we have another contribution pertaining to a web-based ontology parsing tool, OwlParser. We built OwlParser to leverage the CordaSecOnt and HealthOnt without acquiring a knowledge of ontology, an editing tool or querying the concepts using SPARQL queries. The OwlParser was created primarily for easy navigation through the encoded knowledge that should be accessible to all types of users. The OwlParser is built using ReactJs, and it is freely available online. The OwlParser consists of different components (e.g., interface, parser, pre-processor, mapping, navigation tree builder) that are interconnected to interact with each other. The tool parses the SRM domain model-based ontology content and displays it in a structured format on the web interface. We assessed OwlParser's usability using the interview-based surveys. The surveys yielded positive findings since the participants were able to use the tool in the context of understanding the SRM of traditional and blockchain-based applications and discovered that the tool allows easy examination and navigation through the encoded knowledge.

Our contributions support the justifications discussed in Table 4. For instance, by creating two ontology representations and assessing them using expert and qualitative evaluation, we validate that the ontology shares a common understanding and reuse of the information security knowledge model for the SRM of traditional and blockchain-based applications. By nature, ontology is dynamic and enables a seamless process for adding new information, as we explicitly describe in the ULRO, CordaSecOnt, and HealthOnt. As a result, ontology provides users with explicit specifications and explanations about the properties of blockchain and the ability to easily adapt as new concepts emerge. The expert evaluation, tasks-based evaluation, and OwlParser tool ensure that the ontology describes information security knowledge from its ontology components by the required specification, and formalizing information security into an ontological knowledge domain allows security experts to analyze the security of traditional and blockchain-based applications. However, the practical implementation is the uttermost necessity to validate these ontology-building reasons using real-world cases to apply the created ontology representations. However, this is beyond the scope of the current work and has already been considered in future work.

Considering the contributions, the result of this thesis is a reference framework for managing security risks using blockchain. The framework can enable a

systematic evaluation and establish a common ground and understanding for the developers, practitioners, and researchers about the security of traditional applications, blockchain as a countermeasure solution, and security of blockchain-based applications. Moreover, it can communicate the security needs and assist in the SRM of blockchain-based applications. Ultimately, the reference framework can potentially lessen the security threats of traditional and blockchain-based applications. The contributions addressed in this thesis adhere to the SRM domain model's fundamental concepts. We use the Protégé tool to create ontology representations, which are then published to the MMISM ontology repository and the GitHub public repository. OwlParser's source code is also accessible on GitHub. Appendix B contains information on the resources.

8.2. Future Work

The research and contributions presented in this thesis open up several directions for future work, which we discuss in this section.

- The current approach lacks a comprehensive analysis of security threats. For example, a threat's likelihood and impact analysis are essential considerations in the risk assessment process. Also, the threat agent profile and attack method are missing in the current ontology representations. To address these gaps, the ontology representations should be extended to add support for threat likelihood, impact analysis, threat agent classification, and attack method. The ontology representations will be more detailed as a result of this future work.
- UFO can provide foundations to the conceptual modeling constructs; as we previously mentioned, UFO can align the concepts of the SRM domain model and provide conceptual foundations to elucidate theoretical details, allowing for a better understanding of the semantic meaning and interoperability of the SRM concepts. In the future, we will integrate the UFO into the ULRO, which will bring conceptual clarity and semantic explicitation to the SRM concepts and relationships, allowing us to better comprehend the fundamental constructs of the SRM from an ontological standpoint.
- Blockchain looks promising but is still in its infancy. Many blockchain platforms have recently appeared, but their security needs to be evaluated on a larger scale. Many security issues, for example, have yet to be investigated on the blockchain. In this thesis, we gathered security threats through literature studies and evaluated traditional and blockchain-based applications. The security threats we obtained are focused on the asset level, neglecting technical specifics and defects that constitute a source of security threats. In the future, the technical aspects of traditional and blockchain-based application security threats should be investigated.
- CordaSecOnt and HealthOnt are built on a specific platform and applica-

tion domain. For example, the CordaSecOnt is built on a Corda-based permissioned blockchain and includes post-trade matching and confirmation knowledge. HealthOnt, on the other hand, contains permissionless blockchains aimed toward the SRM of healthcare applications. One future research direction is to build platform-agnostic and application-independent ontology representations for permissionless and permissioned blockchains. For instance, generalizing the permissioned blockchain concepts irrespective of platform and application can be used for the SRM of any application that builds on permissioned blockchains. Similarly, to generalize the concepts of permissionless blockchains for the SRM of any application that builds on permissionless blockchains.

- Integrate ontology representations with model-driven engineering tools that generate code and configurations for blockchain applications to introduce the security concepts. For example, these studies [270, 271] used model-driven engineering to automatically generate the code for blockchain applications, whereas HealthOnt can be used while modeling a healthcare application using a permissionless blockchain. In this context, this approach can ensure protection against known threats and vulnerabilities and early validation of required configurations in blockchain platforms.
- We used the HealthOnt to perform the SRM using a real-world healthcare application case. That healthcare application, however, has not yet been built. Therefore, the current ontology representations are not yet formally validated with a real-world application. In the future, the ontology representations shall be integrated with real-world projects. In addition, we should thoroughly validate the encoded concepts with domain experts. For example, evaluating the HealthOnt using different healthcare scenarios, including various stakeholder perspectives, to explore how domain experts (e.g., healthcare specialists, blockchain, and security analysts) perceive the significance of HealthOnt's contribution to deriving the missing security components, to determine the comprehensiveness and technical correctness.
- As noted in Sections 6.2.11 and 6.3.9, the human aspect is crucial in healthcare. Blockchain may offer a suitable infrastructure to solve security concerns related to the human factor. So, another potential future work is to investigate how blockchain might overcome the challenges and the relevance of various vulnerabilities associated with human interaction.
- Extending OwlParser to make it a decision support tool. The tool will help to decide on the security countermeasures based on the security threats and vulnerabilities. The process of extending the OwlParser will follow the guidelines of the SRM domain model. The tool will also enable a dynamic and seamless approach for adding new knowledge and interpreting the security threats of blockchain-based applications. For example, to describe the assets to secure, security threats, vulnerabilities, and countermeasures.

BIBLIOGRAPHY

[1] HIPAA. *2020 Healthcare Data Breach Report: 25% Increase in Breaches in 2020*. 2021. URL: https://www.hipaajournal.com/2020-healthcare-data-breach-report-us/ (visited on 06/01/2021).

[2] Jie Xu et al. "Healthchain: A Blockchain-Based Privacy Preserving Scheme for Large-Scale Health Data". In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8770–8781. DOI: 10.1109/JIOT.2019.2923525.

[3] Laurie Zabel. *10 common HIPAA violations and preventative measures to keep your practice in compliance*. 2016. URL: https://bit.ly/34E8Hrx (visited on 06/02/2021).

[4] Malik Al-essa. "The Impact of Blockchain Technology on Financial Technology (FinTech)". In: *Università degli Studi di Salerno Dipartimento di Scienze Aziendali Management and Innovation Systems* (2019). (Visited on 08/21/2019).

[5] Ariana Polyviou, Pantelis Velanas Soldatos, and John. "Blockchain Technology : Financial Sector". In: (2019), pp. 1–5. DOI: 10.3390/proceedings2019028007. (Visited on 07/17/2020).

[6] Rob Sobers. *98 Must-Know Data Breach Statistics*. 2022. URL: https://www.varonis.com/blog/data-breach-statistics (visited on 01/02/2022).

[7] IBM. *How much does a data breach cost?* 2021. URL: https://www.ibm.com/security/data-breach (visited on 01/02/2022).

[8] Gaby G. Dagher et al. "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology". In: *Sustainable Cities and Society* 39.December 2017 (2018), pp. 283–297. DOI: 10.1016/j.scs.2018.02.014.

[9] Steve Mansfield-Devine. "Your life in your hands: The security issues with healthcare apps". In: *Network Security* 2016.4 (2016), pp. 14–18. DOI: 10.1016/S1353-4858(16)30038-1.

[10] Naga Ramya et al. *Blockchain Applications in Healthcare – A Review and Future Perspective*. Vol. 1. Springer International Publishing, 2020, pp. 198–218. ISBN: 9783030596385. DOI: 10.1007/978-3-030-59638-5.

[11] Ibrar Yaqoob. "Blockchain for healthcare data management : opportunities , challenges , and future recommendations". In: *Neural Computing and Applications* (2021).

[12] Lanxiang Chen et al. "Blockchain based searchable encryption for electronic health record sharing". In: *Future Generation Computer Systems* 95 (2019), pp. 420–429. DOI: https://doi.org/10.1016/j.future.2019.01.018.

[13] B. L. Radhakrishnan, A. Sam Joseph, and S. Sudhakar. “Securing Blockchain based Electronic Health Record using Multilevel Authentication”. In: *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019* March 2019 (2019), pp. 699–703.

[14] Arijit Saha et al. “Review on “Blockchain technology based medical healthcare system with privacy issues””. In: *Security and privacy* 2.5 (2019), pp. 1–14. DOI: 10.1002/spy2.83.

[15] Wang Xiaoding et al. “Enabling Secure Authentication in Industrial IoT with Transfer Learning empowered Blockchain”. In: *IEEE Transactions on Industrial Informatics* 3203.c (2021), pp. 1–9. DOI: 10.1109/TII.2021.3049405.

[16] Mubashar Iqbal and Raimundas Matulevičius. “Blockchain-Based Application Security Risks: A Systematic Literature Review”. In: *Springer Nature Switzerland AG* (2019), pp. 1–26.

[17] Investopedia. *Post-trade processing*. 2022. URL: https://www.investopedia.com/terms/p/post-trade-processing.asp (visited on 03/02/2022).

[18] Sarwar Sayeed and Hector Marco-Gisbert. “Assessing blockchain consensus and security mechanisms against the 51% attack”. In: *Applied Sciences (Switzerland)* 9 (2019).

[19] Elikem Attah. *Five most prolific 51% attacks in crypto: Verge, Ethereum Classic, Bitcoin Gold, Feathercoin, Vertcoin*. 2019. URL: https://cryptoslate.com/prolific-51-attacks-crypto-verge-ethereum-classic-bitcoin-gold-feathercoin-vertcoin (visited on 03/15/2020).

[20] Minhaj Ahmad Khan and Khaled Salah. “IoT security: Review, blockchain solutions, and open challenges”. In: *Future Generation Computer Systems* 82 (2018). DOI: 10.1016/j.future.2017.11.022.

[21] Fabian Ruffy, Wolfgang Hommel, and Felix Von Eye. “A STRIDE-based Security Architecture for Software-Defined Networking”. In: *ICN 2016 : The Fifteenth International Conference on Networks* c (2016), pp. 95–101.

[22] Éric Dubois et al. “A Systematic Approach to Define the Domain of Information System Security Risk Management”. In: *Intentional Perspectives on Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 289–306. DOI: 10.1007/978-3-642-12544-7_16.

[23] Raimundas Matulevičius. *Fundamentals of Secure System Modelling*. 1st ed. Springer International Publishing, 2017, p. 225. ISBN: 978-3-319-61717-6. DOI: 10.1007/978-3-319-61717-6.

[24] Nguyen Binh Truong et al. “GDPR-Compliant Personal Data Management: A Blockchain-Based Solution”. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 1746–1761. DOI: 10.1109/TIFS.2019.2948287. eprint: 1904.03038.

[25] Laure A. Linn and Martha B. Koo. “Blockchain For Health Data and Its Potential Use in Health IT and Health Care Related Research”. In: *ONC/NIST Use of Blockchain for Healthcare and Research Workshop*. Gaithersburg, MD, USA: NIST, 2016, pp. 1–10.

[26] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. “A Survey of Attacks on Ethereum Smart Contracts SoK”. In: (2017), pp. 164–186. DOI: 10.1007/978-3-662-54455-6_8. URL: https://doi.org/10.1007/978-3-662-54455-6_8.

[27] C. Liu et al. “ReGuard: Finding reentrancy bugs in smart contracts”. In: *International Conference on Software Engineering* (2018), pp. 65–68. DOI: 10.1145/3183440.3183495.

[28] Ivan Homoliak et al. “A security reference architecture for blockchains”. In: *2019 2nd IEEE International Conference on Blockchain, Blockchain 2019* (2019), pp. 390–397. DOI: 10.1109/Blockchain.2019.00060. eprint: 1904.06898.

[29] Shijie Zhang and Jong Hyouk Lee. “Double-Spending with a Sybil Attack in the Bitcoin Decentralized Network”. In: *IEEE Transactions on Industrial Informatics* 15.10 (2019), pp. 5715–5722. DOI: 10.1109/TII.2019.2921566.

[30] Karl Wüst and Arthur Gervais. “Do you Need a Blockchain?” In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2018, pp. 45–54. DOI: 10.1109/CVCBT.2018.00011.

[31] Saqib Ali et al. “A Blockchain-Based Decentralized Data Storage and Access Framework for PingER”. In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2018, pp. 1303–1308. DOI: 10.1109/TrustCom/BigDataSE.2018.00179.

[32] Lin William Cong and Zhiguo He. “Blockchain Disruption and Smart Contracts”. In: *The Review of Financial Studies* 32.5 (Apr. 2019), pp. 1754–1797. DOI: 10.1093/rfs/hhz007. eprint: https://academic.oup.com/rfs/article-pdf/32/5/1754/28275179/hhz007.pdf.

[33] Vitalik Buterin. *A Next-Generation Smart Contract and Decentralized Application Platform*. 2014. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[34] Elli Androulaki et al. “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”. In: *Thirteenth EuroSys Conference*.

EuroSys '18. Porto, Portugal: Association for Computing Machinery, 2018. ISBN: 9781450355841. DOI: 10.1145/3190508.3190538.

[35] Tien Tuan Anh Dinh et al. "Blockbench: A Framework for Analyzing Private Blockchains". In: SIGMOD '17 (2017), pp. 1085–1100. DOI: 10.1145/3035918.3064033.

[36] Hyperledger. *Hyperledger fabric introduction*. 2019. URL: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html (visited on 01/02/2019).

[37] Rui Yuan et al. "ShadowEth: Private Smart Contract on Public Blockchain". In: *Journal of Computer Science and Technology* 33 (2018), pp. 542–556.

[38] Daniel Macrinici, Cristian Cartofeanu, and Shang Gao. "Smart contract applications within blockchain technology: A systematic mapping study". In: *Telematics and Informatics* 35.8 (2018), pp. 2337–2354. DOI: https://doi.org/10.1016/j.tele.2018.10.004.

[39] Ethereum$_{O}$*rg*. *Nodes and clients*. 2022. URL: https://ethereum.org/en/developers/docs/nodes-and-clients (visited on 01/02/2022).

[40] Muhammad Zubair. "A Blockchain Solution for Auditing of Timber-to-Charcoal Process". In: *University of Tartu, Master thesis* (2021). URL: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=73720&year=2021.

[41] Tommy Koens et al. "Solutions for the Corda Security and Privacy Trade-off : Having Your Cake and Eating It". In: *ING Bank* (). URL: https://mondovisione.com/_assets/files/Corda_DoSt_v1.6.pdf (visited on 02/09/2020).

[42] Mike Hearn. "Corda: A distributed ledger (Whitepaper)". In: *Whitepaper* (2016), pp. 1–56. URL: https://docs.corda.net/_static/corda-technical-whitepaper.pdf (visited on 01/02/2020).

[43] R3 Corda. *Writing the flow*. URL: https://docs.r3.com/en/platform/corda/4.8/open-source/key-concepts-flows.html (visited on 02/09/2020).

[44] R3 Corda. *What is a Corda node*. URL: https://docs.r3.com/en/platform/corda/4.8/enterprise/node/component-topology.html (visited on 02/09/2020).

[45] R3 Corda. *Corda nodes*. URL: https://docs.r3.com/en/platform/corda/4.6/open-source/key-concepts-node.html (visited on 02/09/2020).

[46] Donald Firesmith. "Specifying Reusable Security Requirements". In: *J. Object Technol.* 3 (2004), pp. 61–75.

[47] Donald G Firesmith. "Cite this column as follows: Donald Firesmith: Engineering Security Requirements". In: *Journal of Object Technology, Pub-*

lished by ETH Zurich, Chair of Software Engineering ©JOT, 2003 2.1 (2003), pp. 53–68.

[48] Daniel Ganji et al. "Approaches to Develop and Implement ISO/IEC 27001 Standard - Information Security Management Systems: A Systematic Literature Review". In: *International Journal on Advances in Software* (2019), pp. 228–238.

[49] Nicola Guarino and Daniel Oberle. "What Is an Ontology?" In: *Handbook on Ontologies* (2009), pp. 1–17. DOI: 10.1007/978-3-540-92673-3.

[50] Willem Nico Borst and W.N. Borst. "Construction of Engineering Ontologies for Knowledge Sharing and Reuse". Undefined. PhD thesis. Netherlands: University of Twente, Sept. 1997. ISBN: 90-365-0988-2.

[51] Oxfod University Press. *Ontology*. URL: https://www.oxfordlearnersdictionaries.com/definition/english/ontology (visited on 09/01/2019).

[52] Natalya F. Noy and Deborah L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". In: *Stanford Knowledge Systems Laboratory* (2001), pp. 1–25.

[53] Bruno A. Mozzaquatro, Ricardo Jardim-Goncalves, and Carlos Agostinho. "Towards a reference ontology for security in the Internet of Things". In: *2015 IEEE International Workshop on Measurements and Networking, M and N 2015 - Proceedings* (2015), pp. 117–122. DOI: 10.1109/IWMN.2015.7322984.

[54] Jigna J. Hathaliya and Sudeep Tanwar. "An exhaustive survey on security and privacy issues in Healthcare 4.0". In: *Computer Communications* (2020), pp. 311–335.

[55] Stefan Fenz and Andreas Ekelhart. "Formalizing information security knowledge". In: *4th International Symposium on ACM Symposium on Information, Computer and Communications Security, ASIACCS'09* (2009), pp. 183–194. DOI: 10.1145/1533057.1533084.

[56] Jing Zhou et al. "What is Theoretical Contribution? A Narrative Review". In: *Sarhad Journal of Management Sciences* 3.2 (2017), pp. 261–271. DOI: 10.31529/sjms.2017.3.2.6.

[57] OWLWorkingGroup. *OWL Web Ontology LanguageGuide*. Version OWL 2. 2009. URL: https://www.w3.org/TR/owl-guide (visited on 01/01/2020).

[58] Alan Rector et al. "OWL pizzas: Practical experience of teaching OWL-DL: Common errors and common patterns". In: *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)* 3257 (2004), pp. 63–81. DOI: 10.1007/978-3-540-30202-5_5.

[59] Jian Bo Gao et al. "Ontology-based model of network and computer attacks for security assessment". In: *Journal of Shanghai Jiaotong Univer-*

sity (Science) 18.5 (2013), pp. 554–562. DOI: 10.1007/s12204-013-1439-5.

[60] Ugarte-Rojas Hector and Chullo-Llave Boris. "BLONDiE: Blockchain Ontology with Dynamic Extensibility". In: *ArXiv* (2020). eprint: 2008.09518.

[61] Eric Miller. "An Introduction to the Resource Description Framework". In: *Bulletin of the American Society for Information Science and Technology* 25.1 (1998), pp. 15–19. DOI: https://doi.org/10.1002/bult.105.

[62] Almut Herzog, Nahid Shahmehri, and Claudiu Duma. "An Ontology of Information Security". In: *IJISP* (2007), pp. 1–23.

[63] Andrei C. Zamfira and Horia Ciocarlie. "Developing an ontology of cyber-operations in networks of computers". In: *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing, ICCP 2018* (2018), pp. 395–400. DOI: 10.1109/ICCP.2018.8516644.

[64] Michael Uschold and Michael Gruninger. "Ontologies : Principles , methods and applications". In: *Knowledge Engineering Review* (1996).

[65] Douglas R. Skuce. "How We Might Reach Agreement on Shared Ontologies: A Fundamental Approach". In: *9th Knowledge Acquisition for Knowledge Based Systems Workshop* (1995).

[66] Xiaoqi Li et al. "A survey on the security of blockchain systems". In: *Future Generation Computer Systems* (2017). eprint: 1802.06993.

[67] Barbara Kitchenham and Stuart Charters. "Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3". In: *Engineering* 45.4ve (2007), p. 1051. DOI: 10.1145/1134285.1134500. eprint: 1304.1186.

[68] Chitu Okoli. "A Guide to Conducting a Standalone Systematic Literature Review". In: *Communications of the Association for Information Systems* 37 (2015), pp. 879–910.

[69] Arlene Fink. "Conducting Research Literature Reviews: From the Internet to Paper". In: *SAGE Publications* (2019), p. 304.

[70] Mouna Jouini, Latifa Ben Arfa Rabai, and Anis Ben Aissa. "Classification of security threats in information systems". In: *Procedia Computer Science* 32.October 2017 (2014), pp. 489–496.

[71] Bojana Koteska and Anastas Mishev. "Blockchain Implementation Quality Challenges : A Literature Review". In: *CEUR Workshop Proceedings* September (2017), pp. 11–13.

[72] Yusuf Muhammad Tukur, Dhavalkumar Thakker, and Irfan-Ullah Awan. "Edge-based blockchain enabled anomaly detection for insider attack prevention in Internet of Things". In: *Transactions on Emerging Telecommu-*

nications Technologies 32.6 (2021). e4158 ETT-20-0337.R1, e4158. DOI: https://doi.org/10.1002/ett.4158.

[73] Stewart Robinson et al. "Conceptual modeling: Definition, purpose and benefits". In: *2015 Winter Simulation Conference (WSC)*. 2015, pp. 2812–2826. DOI: 10.1109/WSC.2015.7408386.

[74] Roel Wieringa. "Design Science Methodology for Information Systems and Software Engineering". In: *Springer Berlin Heidelberg*. 2014, p. 332. DOI: https://doi.org/10.1007/978-3-662-43839-8.

[75] Bin Yu et al. *Platform-Independent Secure Blockchain-Based Voting System*. Vol. 2433. Springer International Publishing, 2018, pp. 369–386. ISBN: 978-3-540-44270-7. DOI: 10.1007/3-540-45811-5.

[76] Peng Zhang et al. "FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data". In: *Computational and Structural Biotechnology Journal* 16 (2018), pp. 267–278. DOI: 10.1016/j.csbj.2018.07.004. eprint: 1807.03227.

[77] 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.February (2018), pp. 42–52. DOI: 10.1016/j.jnca.2018.05.005.

[78] Jieying Chen et al. "A Blockchain Application for Medical Information Sharing". In: *TEMS-ISIE 2018 - 1st Annual International Symposium on Innovation and Entrepreneurship of the IEEE Technology and Engineering Management Society* (2018), pp. 1–7. DOI: 10.1109/TEMS-ISIE.2018.8478645.

[79] Kristen N. Griggs et al. "Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring". In: *Journal of Medical Systems* (2018), pp. 1–7.

[80] Athina Styliani Kleinaki et al. "A Blockchain-Based Notarization Service for Biomedical Knowledge Retrieval". In: *Computational and Structural Biotechnology Journal* (2018), pp. 288–297.

[81] Muhammad Muzammal, Qiang Qu, and Bulat Nasrulin. "Renovating blockchain with distributed databases: An open source system". In: *Future Generation Computer Systems* 90 (2018), pp. 105–117. DOI: 10.1016/j.future.2018.07.042.

[82] Sana Moin et al. "Securing IoTs in distributed blockchain: Analysis, requirements and open issues". In: *Future Generation Computer Systems* 100 (2019), pp. 325–343. DOI: 10.1016/j.future.2019.05.023.

[83] John R. Douceur. "The sybil attack". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2429 (2002), pp. 251–260.

[84] P. Swathi, Chirag Modi, and Dhiren Patel. "Preventing Sybil Attack in Blockchain using Distributed Behavior Monitoring of Miners". In: *2019*

10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019 (2019), pp. 6–11.

[85] Cristina Pérez-Solà et al. "Double-spending prevention for Bitcoin zero-confirmation transactions". In: *International Journal of Information Security* (2019), pp. 451–463.

[86] Kevin Jonathan and Anny Kartika Sari. "Security Issues and Vulnerabilities on A Blockchain System: A Review". In: *2019 2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019* (2019), pp. 228–232.

[87] Ramon Alcarria et al. "A blockchain-based authorization system for trustworthy resource monitoring and trading in smart communities". In: *Sensors (Switzerland)* 18.10 (2018). DOI: 10.3390/s18103561.

[88] Mikerah Quintyne-Collins. "Short Paper: Towards Characterizing Sybil Attacks in Cryptocurrency Mixers". In: *IACR Cryptology ePrint Archive* 2019/1111 (2019).

[89] Arvind Narayanan et al. "Bitcoin and Cryptocurrency Technologies". In: 2016, p. 336. ISBN: 978-1-4302-6383-8.

[90] Binance Academy. *What Is a Replay Attack?* 2019. URL: https://academy.binance.com/en/articles/what-is-a-replay-attack (visited on 01/01/2021).

[91] Pierluigi Gallo and Uy Quoc Nguyen. "BlockSee: Blockchain for IoT video surveillance in smart cities Suporn Pongnumkul NECTEC Thailand". In: *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)* (2018), pp. 1–6.

[92] Michael Fimin. *Five early signs of data tampering*. 2017. URL: https://www.itproportal.com/features/five-early-signs-of-data-tampering/.

[93] Hongyu Li et al. "Blockchain-Based Data Preservation System for Medical Data". In: *Journal of Medical Systems* (2018), pp. 1–13.

[94] Md Zakirul Alam Bhuiyan et al. "Blockchain and Big Data to Transform the Healthcare". In: ICDPA 2018 (2018), pp. 62–68. DOI: 10.1145/3224207.3224220.

[95] Shi Cho Cha et al. "A Blockchain Connected Gateway for BLE-Based Devices in the Internet of Things". In: *IEEE Access* 6 (2018), pp. 24639–24649. DOI: 10.1109/ACCESS.2018.2799942.

[96] Friorik P. Hjalmarsson et al. "Blockchain-Based E-Voting System". In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (2018), pp. 983–986. DOI: 10.1109/CLOUD.2018.00151.

[97] Claudia Pop et al. "Decentralizing the Stock Exchange using Blockchain An Ethereum-based implementation of the Bucharest Stock Exchange". In: (2018), pp. 459–466. DOI: 10.1109/ICCP.2018.8516610.

[98] Patrick Sylim et al. “Blockchain technology for detecting falsified and substandard drugs in distribution: Pharmaceutical supply chain intervention”. In: *Journal of Medical Internet Research* 20.9 (2018). DOI: 10.2196/10163.

[99] Daisuke Ichikawa, Makiko Kashiyama, and Taro Ueno. “Tamper-resistant mobile health using blockchain technology”. In: *JMIR mHealth and uHealth* 5.7 (2017), pp. 1–11. DOI: 10.2196/mhealth.7938.

[100] Martin Nuss B, Alexander Puchta B, and Michael Kunz B. *Towards Blockchain-Based Identity and Access Management for Internet of Things in Enterprises*. 2018. DOI: 10.1007/978-3-319-98385-1.

[101] Mohammad Jabed Morshed Chowdhury et al. “Blockchain as a Notarization Service for Data Sharing with Personal Data Store”. In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (2018), pp. 1330–1335. DOI: 10.1109/TrustCom/BigDataSE.2018.00183.

[102] Steven Trout. *The Impact of Weak Protocols and Ciphers on Healthcare Servers*. 2019. URL: https://getreferralmd.com/2019/11/impact-of-weak-protocols-ciphers-on-healthcare-servers (visited on 11/15/2018).

[103] CypressDataDefense. *How To Prevent Data Tampering In Your Business*. 2020. URL: https://www.cypressdatadefense.com/blog/data-tampering-prevention (visited on 11/15/2018).

[104] Victoria Drake Larry Conklin. *Threat modeling process*. 2021. URL: https://owasp.org/www-community/Threat_Modeling_Process (visited on 01/01/2021).

[105] Jiaxing Li, Jigang Wu, and Long Chen. “Block-secure: Blockchain based scheme for secure P2P cloud storage”. In: *Information Sciences* 465 (2018), pp. 219–231. DOI: 10.1016/j.ins.2018.06.071.

[106] David A. McGrew and Scott R. Fluhrer. “Multiple forgery attacks against Message Authentication Codes”. In: *IACR Cryptol. ePrint Arch.* 2005 (2005), p. 161.

[107] Kazuo Ohta and Mitsuru Matsui. “Differential Attack on Message Authentication Codes”. In: *13th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO ’93. Berlin, Heidelberg: Springer-Verlag, 1993, pp. 200–211. ISBN: 3540577661.

[108] crypto-it.net. *Message Authentication Code (MAC)*. 2020. URL: https://www.crypto-it.net/eng/theory/mac.html (visited on 02/09/2021).

[109] Hao Dai et al. “TrialChain: A Blockchain-Based Platform to Validate Data Integrity in Large, Biomedical Research Studies”. In: *arXiv* (2018), pp. 1–7. eprint: 1807.03662.

[110] Shawn Dexter. *How Are Blockchain Transactions Validated? Consensus VS Validation*. 2018. URL: https://www.mangoresearch.co/blockchain-consensus-vs-validation/.

[111] Shaan Ray. *Blockchains versus Traditional Databases*. 2017. URL: https://hackernoon.com/blockchains-versus-traditional-databases-c1a728159f79.

[112] Owasp. *A10-Insufficient Logging and Monitoring*. 2017. URL: https://bit.ly/3dijcbu.

[113] CC0 Public Domain. *From Yahoo to Uber, major hacks of data*. URL: https://phys.org/news/2018-05-yahoo-uber-major-hacks.html.

[114] Daniel Mellado et al. "A systematic review of security requirements engineering". In: *Computer Standards and Interfaces* 32.4 (2010), pp. 153–165. DOI: 10.1016/j.csi.2010.01.006.

[115] Umesh Hodeghatta Rao and Umesha Nayak. "Understanding Networks and Network Security". In: *The InfoSec Handbook: An Introduction to Information Security*. Berkeley, CA: Apress, 2014, pp. 187–204. ISBN: 978-1-4302-6383-8. DOI: 10.1007/978-1-4302-6383-8_9.

[116] Gil Dagan. *The Actual Networking behind the Ethereum Network: How It Works*. 2018. URL: https://medium.com/orbs-network/the-actual-networking-behind-the-ethereum-network-how-it-works-6e147ca36b45.

[117] Angello Pozo. *Ethereum: Signing and Validating*. 2017. URL: https://medium.com/@angellopozo/ethereum-signing-and-validating-13a2d7cb0ee3.

[118] Pim Otte, Martijn de Vos, and Johan Pouwelse. "TrustChain: A Sybil-resistant scalable blockchain". In: *Future Generation Computer Systems* (2017). DOI: 10.1016/j.future.2017.08.048.

[119] Jung Ki So and Douglas S. Reeves. "Defending against sybil nodes in BitTorrent". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6641 LNCS (2011), pp. 25–39. DOI: 10.1007/978-3-642-20798-3_3.

[120] Muhammad Numan et al. "A Systematic Review on Clone Node Detection in Static Wireless Sensor Networks". In: *IEEE Access* 8 (2020), pp. 65450–65461. DOI: 10.1109/ACCESS.2020.2983091.

[121] Aziz Mohaisen. "The Sybil Attacks and Defenses: A Survey". In: *The Smart Computing Review* 3.6 (2013). DOI: 10.6029/smartcr.2013.06.009. eprint: 1312.6349.

[122] Srikanta Pradhan, Somanath Tripathy, and Sukumar Nandi. "Blockchain based Security Framework for P2P Filesharing system". In: *International Symposium on Advanced Networks and Telecommunication Sys-*

tems, ANTS 2018-Decem (2019), pp. 1–6. DOI: 10.1109/ANTS.2018.8710078.

[123] Loi Luu et al. "A Secure Sharding Protocol For Open Blockchains". In: CCS '16 (2016), pp. 17–30. DOI: 10.1145/2976749.2978389.

[124] Tayebeh Rajabi et al. "On the feasibility of sybil attacks in shard-based permissionless blockchains". In: *arXiv* (2020). eprint: 2002.06531.

[125] Jamal Hayat Mosakheil. "Security Threats Classification in Blockchains". In: *Culminating Projects in Information Assurance* (2018), p. 141.

[126] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies". In: *IEEE Symposium on Security and Privacy* (2017), pp. 375–392. DOI: 10.1109/SP.2017.29. eprint: 1605.07524.

[127] Muoi Tran et al. "A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network". In: *2020 IEEE Symposium on Security and Privacy (SP)* (2020), pp. 894–909. DOI: 10.1109/SP40000.2020.00027.

[128] Todd Baumeister et al. "A routing table insertion (RTI) attack on freenet". In: *2012 ASE International Conference on Cyber Security, CyberSecurity 2012* SocialInformatics (2012), pp. 8–15. DOI: 10.1109/CyberSecurity.2012.8.

[129] Muhammad Saad et al. "Exploring the Attack Surface of Blockchain: A Systematic Overview". In: *IEEE Communications Surveys Tutorials* (2019), pp. 1–30. eprint: 1904.03487.

[130] Lester Coleman. *Ethereum Responds to Recent DDoS Attack*. 2016. URL: https://www.ccn.com/ethereum-responds-to-recent-ddos-attack (visited on 01/01/2020).

[131] Marie Vasek, Micah Thornton, and Tyler Moore. "Empirical Analysis of Denial-of-Service Attacks in the Bitcoin Ecosystem". In: *Financial Cryptography and Data Security* (2014). Ed. by Rainer Böhme et al., pp. 57–71.

[132] Yunpeng Wang et al. "Anti-Dust: A Method for Identifying and Preventing Blockchain's Dust Attacks". In: *2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*. 2018, pp. 274–280.

[133] Binance Academy. *What Is a Dusting Attack?* 2020. URL: https://academy.binance.com/en/articles/what-is-a-dusting-attack (visited on 02/09/2020).

[134] Nicolas van Saberhagen. *CryptoNote v 2.0*. 2013. URL: https://bytecoin.org/old/whitepaper.pdf (visited on 01/02/2018).

[135] Michael Mirkin et al. "BDoS: Blockchain Denial-of-Service". In: *2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA, 2020, pp. 601–619. ISBN: 9781450370899.

[136] Mohiuddin Ahmed et al. “A Poisoning Attack Against Cryptocurrency Mining Pools”. In: *DPM/CBT@ESORICS*. 2018.

[137] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. “Smart Contract: Attacks and Protections”. In: *IEEE Access* 8 (2020), pp. 24416–24427.

[138] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. “Double-Spending Fast Payments in Bitcoin”. In: *2012 ACM Conference on Computer and Communications Security*. CCS ’12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 906–917. ISBN: 9781450316514. DOI: 10.1145/2382196.2382292.

[139] Puwen Wei, Quan Yuan, and Yuliang Zheng. “Security of the Blockchain Against Long Delay Attack”. In: *Advances in Cryptology – ASIACRYPT 2018*. Vol. 11274. Lecture Notes in Computer Science. Springer, 2018, pp. 250–275. DOI: 10.1007/978-3-030-03332-3_10.

[140] Meni Rosenfeld. “Analysis of Hashrate-Based Double Spending”. In: *arXiv* (2014), pp. 1–13. eprint: 1402.2009.

[141] Delton Rhodes. *51% Attack Security: Delayed Proof of Work (dPoW)*. 2018. URL: https://blog.komodoplatform.com/en/delayed-proof-of-work (visited on 01/02/2019).

[142] Kevin Liao and Jonathan Katz. “Incentivizing Double-Spend Collusion in Bitcoin”. In: *Financial Cryptography and Data Security* (2017). URL: https://fc17.ifca.ai/bitcoin/schedule.html.

[143] 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. DOI: 10.1109/ACCESS.2019.2901858.

[144] Carlos Pinzón and Camilo Rocha. “Double-spend Attack Models with Time Advantange for Bitcoin”. In: *Electronic Notes in Theoretical Computer Science* 329 (2016), pp. 79–103. DOI: 10.1016/j.entcs.2016.12.006.

[145] Ethan Heilman et al. “Eclipse attacks on Bitcoin’s peer-to-peer network”. In: *24th USENIX Security Symposium* August (2015), pp. 129–144.

[146] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. “Low-Resource Eclipse Attacks on Ethereum’s Peer-to-Peer Network.” In: *IACR Cryptology ePrint Archive* 2018.January (2018), p. 236. URL: https://eprint.iacr.org/2018/236.pdf.

[147] George Dean Bissias et al. “An Analysis of Attacks on Blockchain Consensus”. In: *ArXiv* abs/1610.07985 (2016).

[148] Sebastian Henningsen et al. “Eclipsing Ethereum Peers with False Friends”. In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 2019, pp. 300–309. DOI: 10.1109/EuroSPW.2019.00040.

[149] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. "Double-Spending Risk Quantification in Private, Consortium and Public Ethereum Blockchains". In: *ArXiv* abs/1805.05004 (2018).

[150] Coinbase. *Bitcoin glossary*. URL: https://help.coinbase.com/en/coinbase/getting-started/crypto-education/bitcoin-glossary (visited on 01/02/2022).

[151] Cyril Grunspan and Ricardo Pérez-Marco. "Double spend races". In: *arXiv* (2017), pp. 1–36. eprint: 1702.02867.

[152] Jehyuk Jang and Heung-No Lee. "Profitable Double-Spending Attacks". In: *Applied Sciences* 10.23 (2020). URL: https://www.mdpi.com/2076-3417/10/23/8477.

[153] Ghassan O. Karame et al. "Misbehavior in Bitcoin: A Study of Double-Spending and Accountability". In: *ACM Trans. Inf. Syst. Secur.* 18.1 (May 2015). DOI: 10.1145/2732196.

[154] Matthias Grundmann, Till Neudecker, and Hannes Hartenstein. "Exploiting Transaction Accumulation and Double Spends for Topology Inference in Bitcoin". In: *Financial Cryptography and Data Security*. Ed. by Aviv Zohar et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 113–126. ISBN: 978-3-662-58820-8.

[155] Kervins Nicolas et al. "Blockchain System Defensive Overview for Double-Spend and Selfish Mining Attacks: A Systematic Approach". In: *IEEE Access* 9 (2021), pp. 3838–3857. DOI: 10.1109/ACCESS.2020.3047365.

[156] Ivan Osipkov et al. "Combating double-spending using cooperative P2P systems". In: *International Conference on Distributed Computing Systems* (2007), pp. 41–50. DOI: 10.1109/ICDCS.2007.91.

[157] John P. Podolanko and Jiang Ming. "Countering Double-Spend Attacks on Bitcoin Fast-Pay Transactions". In: 2017.

[158] Yonatan Sompolinsky and Aviv Zohar. "Bitcoin's Security Model Revisited". In: *arXiv* (2016). eprint: 1605.09193. URL: http://arxiv.org/abs/1605.09193.

[159] Gholamreza Ramezan, Cyril Leung, and Z. Jane Wang. "A Strong Adaptive, Strategic Double-Spending Attack on Blockchains". In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2018, pp. 1219–1227. DOI: 10.1109/Cybermatics_2018.2018.00216.

[160] A. Pinar Ozisik and Brian Neil Levine. "An Explanation of Nakamoto's Analysis of Double-spend Attacks". In: *arXiv* (2017), pp. 1–15. eprint: 1701.03977.

[161] Tobias Bamert et al. "Have a snack, pay with Bitcoins". In: *IEEE P2P 2013 Proceedings*. 2013, pp. 1–5. DOI: 10.1109/P2P.2013.6688717.

[162] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. In: *Www.Bitcoin.Org* (2008), p. 9. DOI: 10.1007/s10838-008-9062-0. eprint: 43543534534v343453. URL: https://bitcoin.org/bitcoin.pdf.

[163] Anna Katrenko and Mihail S. *Blockchain Attack Vectors: Vulnerabilities of the Most Secure Technology*. 2020. URL: https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors (visited on 01/02/2019).

[164] Rosenfeld Meni. *What is a Finney attack?* 2012. URL: https://bitcoin.stackexchange.com/questions/4942/what-is-a-finney-attack (visited on 01/02/2019).

[165] Yiwen Du et al. “A Medical Information Service Platform Based on Distributed Cloud and Blockchain”. In: *2018 IEEE International Conference on Smart Cloud (SmartCloud)* (2018), pp. 34–39. DOI: 10.1109/SmartCloud.2018.00014.

[166] Qi Xia et al. “MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain”. In: *IEEE Access* 5 (2017), pp. 14757–14767. DOI: 10.1109/ACCESS.2017.2730843.

[167] Lijing Zhou, Licheng Wang, and Yiru Sun. “MIStore: a Blockchain-Based Medical Insurance Storage System”. In: *Journal of Medical Systems* 42.8 (2018). DOI: 10.1007/s10916-018-0996-4.

[168] BitMEX research. *Stale block candidates at height 666833*. 2021. URL: https://twitter.com/bitmexresearch/status/1351855414103715842 (visited on 01/20/2021).

[169] Xin Zhou et al. “A map of threats to validity of systematic literature reviews in software engineering”. In: *Asia-Pacific Software Engineering Conference, APSEC* 0 (2016), pp. 153–160. DOI: 10.1109/APSEC.2016.031.

[170] Linda Elmhadhbi, Mohamed Hedi Karray, and Bernard Archimède. “Toward the use of upper-level ontologies for semantically interoperable systems: An emergency management use case”. In: *I-ESA Conferences* 9 (2019), pp. 131–140. DOI: 10.1007/978-3-030-13693-2_11.

[171] Stefan Schulz and Martin Boeker. “BioTopLite : An Upper Level Ontology for the Life Sciences . Evolution , Design and Application 2 BioTop and BioTopLite : Evolution and Design”. In: *iospress* (2017), pp. 1889–1899.

[172] Giancarlo Guizzardi and Gerd Wagner. “Towards Ontological Foundations for Agent Modelling Concepts Using the Unified Fundational Ontology (UFO)”. In: *Agent-Oriented Information Systems II*. Ed. by Paolo Bresciani et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 110–124. ISBN: 978-3-540-31946-7.

[173] Matthew Horridge et al. “A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins”. In: *The University Of Manchester* (2011).

[174] DublinCore. *DCMI Metadata Terms*. 2020. URL: https://www.dublincore.org/specifications/dublin-core/dcmi-terms (visited on 01/01/2022).

[175] Ricardo de Almeida Falbo. “SABiO: Systematic Approach for Building Ontologies”. In: *Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering co-located with 8th International Conference on Formal Ontology in Information Systems, ONTO.COM/ODISE@FOIS 2014, Rio de Janeiro, Brazil, September 21, 2014*. Ed. by Giancarlo Guizzardi et al. Vol. 1301. CEUR Workshop Proceedings. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1301/ontocomodise2014_2.pdf.

[176] Lars Michael Kristensen, Jens Bæk Jørgensen, and Kurt Jensen. *Application of coloured Petri nets in system development*. Vol. 3098. February. 2004, pp. 626–685. ISBN: 9783540277552. DOI: 10.1007/978-3-540-27755-2_18.

[177] Bruno Borlini Duarte et al. “An ontological analysis of software system anomalies and their associated risks”. In: *Data Knowledge Engineering* 134 (2021), p. 101892. ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2021.101892.

[178] Mathias Weske. *Business process management architectures*. Springer, 2007.

[179] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. “Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems”. In: *International Journal on Software Tools for Technology Transfer* 9.3 (2007), pp. 213–254.

[180] Leo Obrst, Penny Chase, and Richard Markeloff. “Developing an ontology of the cyber security domain”. In: *CEUR Workshop Proceedings* 966 (2014), pp. 49–56.

[181] Edson dos Santos Moreira et al. “Ontologies for information security management and governance”. In: *Information Management and Computer Security* 16.2 (2008), pp. 150–165. DOI: 10.1108/09685220810879627.

[182] Stefano Borgo and Claudio Masolo. “Ontological Foundations of dolce”. In: *Theory and Applications of Ontology: Computer Applications*. Ed. by Roberto Poli, Michael Healy, and Achilles Kameas. Dordrecht: Springer Netherlands, 2010, pp. 279–295. ISBN: 978-90-481-8847-5. DOI: 10.1007/978-90-481-8847-5_13.

[183] M.E.M. van Wingerde and H. Weigand. “An ontological analysis of artifact-centric business processes managed by smart contracts”. In: *2020 IEEE 22nd Conference on Business Informatics (CBI)*. Vol. 1. 2020, pp. 231–240. DOI: 10.1109/CBI49978.2020.00032.

[184] Malina Adach, Kaj Hänninen, and Kristina Lundqvist. “A Combined Security Ontology based on the Unified Foundational Ontology”. In: *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*. 2022, pp. 187–194. DOI: 10.1109/ICSC52841.2022.00039.

[185] ECB. *The potential impact of DLTs on securities post trading harmonisation and on the wider EU financial market integration*. 2017. URL: https://www.ecb.europa.eu/paym/intro/governance/shared/pdf/201709_dlt_impact_on_harmonisation_and_integration.pdf (visited on 02/09/2020).

[186] Frank; Striegel and Julia; Florian Klein Petry. *The future of post-trade: A glimpse into the future*. 2018. URL: https://www2.deloitte.com/de/de/pages/strategy/articles/future-of-post-trade.html (visited on 01/15/2020).

[187] Justs Placans. “Security risk management in corda-based application for capital markets”. In: *Riga technical university, Institute of Applied Computer Systems* (2019).

[188] Robert P. Baker. “The Trade Lifecycle Behind the Scenes of the Trading Process”. In: Wiley, 201o, p. 320.

[189] Mubashar Iqbal and Raimundas Matulevičius. “Managing Security Risks in Post-Trade Matching and Confirmation Using CorDapp”. In: *Databases and Information Systems*. Ed. by Tarmo Robal et al. Cham: Springer International Publishing, 2020, pp. 325–339. ISBN: 978-3-030-57672-1.

[190] Chad Brubaker et al. “Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations”. In: *IEEE Symposium on Security and Privacy* (2014), pp. 114–129. DOI: 10.1109/SP.2014.15.

[191] POA Network. *Proof of Authority: consensus model with Identity at Stake*. 2017. URL: https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256 (visited on 01/02/2019).

[192] Tim Maurer, Ariel Levite and George Perkovich. “Toward a global norm against manipulating the integrity of financial data”. In: *Carnegie endowment* (2017), pp. 0–41.

[193] AccentureSecurity. *Future Cyber Threats : Extreme But Plausible Threat Scenarios In Financial Services*. 2019. URL: https://www.accenture.com/_acnmedia/pdf-100/accenture_fs_threat-report_approved.pdf.

[194] Sakshi Agarwal. *Cybersecurity essentials for capital markets firms in the digital age*. URL: https://www.wipro.com/securities-and-capital-markets/cybersecurity-essentials-for-capital-markets-firms-in-the-digital-age (visited on 01/01/2020).

[195] Corda Docs. *Corda Threat Model*. URL: https://docs.corda.net/design/threat-model/corda-threat-model.html (visited on 02/09/2020).

[196] Ryogo Kubo. "Detection and Mitigation of False Data Injection Attacks for Secure Interactive Networked Control Systems". In: *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)* (2018), pp. 7–12.

[197] John Velissarios, Justin Herzig, and Unal Didem. *Blockchain's potential starts with security*. 2019. URL: https://www.accenture.com/us-en/insights/blockchain/potential-starts-security.

[198] Jana Moser. *The Application and Impact of the European General Data Protection Regulation on Blockchains*. 2017. URL: https://www.r3.com/wp-content/uploads/2018/04/GDPR_Blockchains_R3.pdf.

[199] R3 Corda. *Transaction Tear-Offs*. 2020. URL: https://docs.corda.net/docs/corda-os/4.6/key-concepts-tearoffs.html (visited on 02/09/2020).

[200] Xavier Bellekens et al. "Pervasive eHealth services a security and privacy risk awareness survey". In: *2016 International Conference on Cyber Situational Awareness, Data Analytics and Assessment, CyberSA 2016* (2016), pp. 1–4. DOI: 10.1109/CyberSA.2016.7503293.

[201] W E I Yin et al. "An Anti-Quantum Transaction Authentication Approach in Blockchain". In: 6 (2018).

[202] Christina M. Steiner and Dietrich Albert. "Validating domain ontologies: A methodology exemplified for concept maps". In: *Cogent Education* 4.1 (2017). DOI: 10.1080/2331186X.2016.1263006.

[203] Evren Sirin et al. "Pellet: A practical OWL-DL reasoner". In: *Journal of Web Semantics* 5.2 (2007). Software Engineering and the Semantic Web, pp. 51–53. DOI: https://doi.org/10.1016/j.websem.2007.03.004.

[204] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. "OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), pp. 7–34.

[205] Mohamad Gharib, Paolo Giorgini, and John Mylopoulos. "COPri v.2 — A core ontology for privacy requirements". In: *Data and Knowledge Engineering* 133.April (2021). DOI: 10.1016/j.datak.2021.101888.

[206] Joe Raad and Christophe Cruz. "A Survey on Ontology Evaluation Methods". In: *HAL archives-ouvertes* (2018).

[207] Denny Vrandecic. "Ontology Evaluation". In: *Handbook on Ontologies* (2009), pp. 293–313.

[208] He Tan et al. "Evaluation of an application ontology". In: *CEUR Workshop Proceedings* 2050 (2017).

[209] Amina Souag et al. “A security ontology for security requirements elicitation”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8978 (2015), pp. 157–177. DOI: 10.1007/978-3-319-15618-7_13.

[210] Mario Vega-Barbas et al. “Ontology-based system for dynamic risk management in administrative domains”. In: *Applied Sciences (Switzerland)* 9.21 (2019). DOI: 10.3390/app9214547.

[211] Bruno Augusti Mozzaquatro et al. “An ontology-based cybersecurity framework for the internet of things”. In: *Sensors (Switzerland)* 18.9 (2018), pp. 1–20.

[212] Danny Velasco Silva and Glen Rodríguez Rafael. “Ontologies for network security and future challenges”. In: *12th International Conference on Cyber Warfare and Security, ICCWS 2017* (2017), pp. 541–547. eprint: 1704.02441.

[213] Yu Long Gao et al. “A Secure Cryptocurrency Scheme Based on Post-Quantum Blockchain”. In: *IEEE Access* 6.Part Ii (2018), pp. 27205–27213.

[214] Eugenia Politou et al. “Blockchain Mutability: Challenges and Proposed Solutions”. In: *IEEE Transactions on Emerging Topics in Computing* 9.4 (2021), pp. 1972–1986. DOI: 10.1109/TETC.2019.2949510.

[215] Manlu Liu, Kean Wu, and Jennifer Jie Xu. “How Will Blockchain Technology Impact Auditing and Accounting: Permissionless versus Permissioned Blockchain”. In: *Current Issues in Auditing* 13.2 (Aug. 2019), A19–A29. ISSN: 1936-1270. DOI: 10.2308/ciia-52540.

[216] Amrutanshu Panigrahi et al. “Application of Blockchain as a Solution to the Real-World Issues in Health Care System”. In: *Blockchain Technology: Applications and Challenges*. Ed. by Sandeep Kumar Panda et al. Cham: Springer International Publishing, 2021, pp. 135–149. ISBN: 978-3-030-69395-4. DOI: 10.1007/978-3-030-69395-4_8.

[217] Dodo Khan et al. “Empirical Performance Analysis of Hyperledger LTS for Small and Medium Enterprises”. In: *Sensors* 22.3 (2022), pp. 1–17. ISSN: 14248220. DOI: 10.3390/s22030915.

[218] Alexei Zamyatin et al. “XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 193–210. DOI: 10.1109/SP.2019.00085.

[219] Thomas Hardjono, Alexander Lipton, and Alex Pentland. “Toward an Interoperability Architecture for Blockchain Autonomous Systems”. In: *IEEE Transactions on Engineering Management* 67.4 (2020), pp. 1298–1309. DOI: 10.1109/TEM.2019.2920154.

[220] Stefan Mai. “The European Post-Trade Market - An Introduction”. In: *SSRN Electronic Journal* (2011). DOI: 10.2139/ssrn.663921.

[221] Huirui Han, Mengxing Huang, and Yu Zhang. “An Architecture of Secure Health Information Storage System Based on Blockchain Technology”. In: *ICCCS* (2018), pp. 578–588.

[222] Ahmed F. Hussein et al. “A medical records managing and securing blockchain based system supported by a Genetic Algorithm and Discrete Wavelet Transform”. In: *Cognitive Systems Research* (2018), pp. 1–11.

[223] Mingxiao Du et al. “An Optimized Consortium Blockchain for Medical Information Sharing”. In: *IEEE Transactions on Engineering Management* (2020), pp. 1–13.

[224] Di Francesco, Laura Ricci, and Paolo Mori. “Distributed Access Control Through Blockchain Technology Blockchain”. In: *ERCIM News: Blockchain Engineering* (2017), pp. 31–32.

[225] Christian Esposito et al. “Blockchain : A Panacea for Healthcare Cloud-Based Data Security and Privacy?” In: *IEEE Cloud Computing* (2018), pp. 31–37.

[226] Guojun Wang. “MediBchain: A Blockchain Based Privacy Preserving Platform for Healthcare Data”. In: *SpaCCS* (2017), pp. 534–543.

[227] F Dario De Martino et al. “Transforming the U . S . Healthcare Industry with Blockchain Technology”. In: (2019), pp. 1–7.

[228] Sam Thielman. *Your private medical data is for sale – and it’s driving a business worth billions*. `https://bit.ly/3ceaacp`. 2017. (Visited on 02/06/2021).

[229] Shuyun Shi et al. “Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey”. In: *Computers & Security* (2020).

[230] Deepak K. Tosh et al. “Security Implications of Blockchain Cloud with Analysis of Block Withholding Attack”. In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017* (2017), pp. 458–467.

[231] SpecOpsSoft. *The countries experiencing the most ‘significant’ cyber-attacks*. `https://bit.ly/3idba4m`. 2020. (Visited on 02/06/2021).

[232] Muhammad Salek Ali et al. “A decentralized peer-to-peer remote health monitoring system”. In: *Sensors (Switzerland)* (2020), pp. 1–18.

[233] Stephen L George and Marc Buyse. “Data fraud in clinical trials”. In: *Clinical Investigation* (2015), pp. 161–173.

[234] Thomas Mcghin et al. “Blockchain in healthcare applications: Research challenges and opportunities”. In: *Journal of Network and Computer Applications* (2019), pp. 62–75.

[235] HelpNetSecurity. *More than 99% of cyberattacks rely on human interaction*. `https://bit.ly/3uL3CYW`. 2019. (Visited on 02/06/2021).

[236] SecurityMetrics. *Healthcare: Recognize Social Engineering Techniques*. `https://www.securitymetrics.com/blog/healthcare-`

recognize - social - engineering - techniques. 2015. (Visited on 03/15/2022).

[237] Mubashar Iqbal and Raimundas Matulevičius. "Exploring Sybil and Double-Spending Risks in Blockchain Systems". In: *IEEE Access* 9 (2021), pp. 76153–76177.

[238] Sandi Rahmadika and Kyung Hyune Rhee. "Blockchain technology for providing an architecture model of decentralized personal health information". In: *International Journal of Engineering Business Management* 10.May (2018).

[239] Paritosh Banchhor et al. "A Systematic Review on Blockchain Security Attacks , Challenges , and Issues". In: *International Journal of Engineering Research and Technology* 10.04 (2021), pp. 386–391.

[240] Pranav Ratta et al. "Application of Blockchain and Internet of Things in Healthcare and Medical Sector: Applications, Challenges, and Future Perspectives". In: *Journal of Food Quality* 2021 (2021).

[241] Rania El-Gazzar and Karen Stendal. "Blockchain in health care: Hope or hype?" In: *Journal of Medical Internet Research* 22.7 (2020).

[242] Sebastian Henningsen et al. "Eclipsing ethereum peers with false friends". In: *4th IEEE European Symposium on Security and Privacy Workshops, EUROS and PW 2019* (2019), pp. 300–309. eprint: 1908.10141.

[243] Saurabh Singh, A. S.M. Sanwar Hosen, and Byungun Yoon. "Blockchain Security Attacks, Challenges, and Solutions for the Future Distributed IoT Network". In: *IEEE Access* 9 (2021), pp. 13938–13959.

[244] Ahmad Musamih et al. "A blockchain-based approach for drug traceability in healthcare supply chain". In: *IEEE Access* 9 (2021), pp. 9728–9743.

[245] Akashdeep Bhardwaj et al. "Penetration testing framework for smart contract Blockchain". In: *Peer-to-Peer Networking and Applications* (2020).

[246] Liang Liu et al. "A type of block withholding delay attack and the countermeasure based on type-2 fuzzy inference". In: *Mathematical Biosciences and Engineering* 17.1 (2020), pp. 309–327.

[247] Divya Guru, Supraja Perumal, and Vijayakumar Varadarajan. "Approaches towards blockchain innovation: A survey and future directions". In: *Electronics (Switzerland)* 10.10 (2021), pp. 1–15.

[248] Aisha Zahid Junejo, Manzoor Ahmed Hashmani, and Abdullah Abdulrehman Alabdulatif. "A survey on privacy vulnerabilities in permissionless blockchains". In: *International Journal of Advanced Computer Science and Applications* 11.9 (2020), pp. 130–139.

[249] Alex Biryukov and Sergei Tikhomirov. "Deanonymization and linkability of cryptocurrency transactions based on network analysis". In: *4th IEEE European Symposium on Security and Privacy, EURO S and P 2019* (2019), pp. 172–184.

[250] Stephen Shankland. *Cryptocurrency faces a quantum computing problem*. 2021. URL: https://www.cnet.com/personal-finance/crypto/cryptocurrency-faces-a-quantum-computing-problem (visited on 01/21/2022).

[251] Aqsa Fatima and Ricardo Colomo-Palacios. "Security aspects in healthcare information systems: A systematic mapping". In: *Procedia Computer Science* 138 (2018), pp. 12–19. DOI: https://doi.org/10.1016/j.procs.2018.10.003.

[252] Bakheet Aljedaani and M Ali Babar. "Challenges With Developing Secure Mobile Health Applications: Systematic Review". In: *JMIR Mhealth Uhealth* 9.6 (June 2021), e15654. DOI: 10.2196/15654.

[253] Tafheem Ahmad Wani, Antonette Mendoza, and Kathleen Gray. "Hospital Bring-Your-Own-Device Security Challenges and Solutions: Systematic Review of Gray Literature". In: *JMIR Mhealth Uhealth* 8.6 (June 2020), e18175. DOI: 10.2196/18175.

[254] Alberto Sardi et al. "Cyber risk in health facilities: A systematic literature review". In: *Sustainability* 12.17 (2020). DOI: 10.3390/su12177002.

[255] Hossein Ahmadi et al. "The application of internet of things in healthcare: a systematic literature review and classification". In: *Universal Access in the Information Society* 18.4 (2019), pp. 837–869. DOI: 10.1007/s10209-018-0618-4.

[256] Leonardo Horn Iwaya, Aakash Ahmad, and M. Ali Babar. "Security and Privacy for mHealth and uHealth Systems: A Systematic Mapping Study". In: *IEEE Access* 8 (2020), pp. 150081–150112. DOI: 10.1109/ACCESS.2020.3015962.

[257] Prosper Kandabongee Yeng et al. "Mapping the Psychosocialcultural Aspects of Healthcare Professionals' Information Security Practices: Systematic Mapping Study". In: *JMIR Human Factors* 8.2 (June 2021), e17604. DOI: 10.2196/17604.

[258] IBM-Blockchain. *Blockchain in healthcare*. https://www.ibm.com/blogs/blockchain/category/blockchain-healthcare. 2022. (Visited on 03/15/2022).

[259] David Randall, Pradeep Goel, Ramzi Abujamra, et al. "Blockchain applications and use cases in health information technology". In: *Journal of Health & Medical Informatics* 8.3 (2017), pp. 1–17.

[260] Emeka Chukwu and Lalit Garg. "A Systematic Review of Blockchain in Healthcare: Frameworks, Prototypes, and Implementations". In: *IEEE Access* 8 (2020), pp. 21196–21214. DOI: 10.1109/ACCESS.2020.2969881.

[261] Cornelius C. Agbo, Qusay H. Mahmoud, and J. Mikael Eklund. "Blockchain Technology in Healthcare: A Systematic Review". In: *Healthcare* 7.2 (2019). DOI: 10.3390/healthcare7020056.

[262] Hao Jin et al. “A Review of Secure and Privacy-Preserving Medical Data Sharing”. In: *IEEE Access* 7 (2019), pp. 61656–61669. DOI: 10.1109/ACCESS.2019.2916503.

[263] Ricardo Neisse, Gary Steri, and Igor Nai-Fovino. “A blockchain-based approach for data accountability provenance tracking”. In: *ACM International Conference Proceeding Series* Part F1305 (2017).

[264] Jorge Bernal Bernabe et al. “Privacy-Preserving Solutions for Blockchain: Review and Challenges”. In: *IEEE Access* 7 (2019), pp. 164908–164940.

[265] Nida Khan and Mohamed Nassar. “A Look into Privacy-Preserving Blockchains”. In: *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. 2019, pp. 1–6.

[266] Gamal Elkoumy et al. “Privacy and Confidentiality in Process Mining: Threats and Research Challenges”. In: *ACM Trans. Manage. Inf. Syst.* 13.1 (Oct. 2021). ISSN: 2158-656X. DOI: 10.1145/3468877.

[267] Raimundas Matulevičius et al. “Ontological Representation of Healthcare Applications Security Using Blockchain Technology”. In: *Informatica* 33.2 (2022).

[268] Mai Ristioja. “Parsing OWL-based Blockchain Security Ontology”. In: *University of Tartu, Bachelor thesis* (2022).

[269] Kate Moran. *Usability Testing 101*. 2019. URL: https://www.nngroup.com/articles/usability-testing-101 (visited on 05/15/2022).

[270] Yassine Ait Hsain, Naziha Laaz, and Samir Mbarki. “Ethereum’s Smart Contracts Construction and Development using Model Driven Engineering Technologies: a Review”. In: *Procedia Computer Science* 184 (2021). The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops, pp. 785–790. DOI: https://doi.org/10.1016/j.procs.2021.03.097.

[271] Victor Amaral de Sousa and Corentin Burnay. “MDE4BBIS: A Framework to Incorporate Model-Driven Engineering in the Development of Blockchain-Based Information Systems”. In: *2021 Third International Conference on Blockchain Computing and Applications (BCCA)*. 2021, pp. 195–200. DOI: 10.1109/BCCA53669.2021.9657015.

Appendix A. INTERVIEW QUESTIONS

We divided interview questions into six categories (e.g., applicability, ease of use, ease of learning, task efficiency, subjective satisfaction, and understandability) that cover the different aspects of the OwlParser to assess its usability.

Table 49. Questions related to the applicability

Id	Question
Q1	Are you familiar with ontologies? If yes, please specify your goals with ontologies.
Q2	What purposes have you had with ontologies?
Q3	Have you ever used ontologies to get an overview of some concepts?
Q4	Did you use ontologies before to construct or gain information?
Q5	Have you ever used ontologies to avoid misunderstandings?
Q6	Have you ever used ontologies to clarify some definitions?
Q7	What do you know about blockchain security?
Q8	How will you use this tool?
Q9	How applicable is this tool in general to perform the SRM?

Table 50. Questions related to the ease of use

Id	Question
Q10	How easy was it to use the tool in general?
Q11	How easy was it to use the loading page?
Q12	How easy was it to use the mapping interface?
Q13	How easy was it to use the left-side panel?
Q14	How easy was it to find information from the right-side panel?
Q15	On first use, did you expect some part(s) of the tool to work differently?
Q16	Which functionalities do you think are missing?
Q17	Which aspects of the application would you like to see improved and how?

Table 51. Questions related to the ease of learning

Id	Question
Q18	Did the short introduction to the tool make sense?
Q19	What were the aspects of the tool that were difficult to understand on first use?

Table 52. Questions related to the task efficiency

Id	Question
Q20	Which activities did you find cumbersome to perform?

Table 53. Questions related to the subjective satisfaction

Id	Question
Q21	How satisfied are you with the tool?
Q22	How likely would you use the tool in the future?
Q23	What do you think of the design of the user interface?
Q24	How satisfied are you with the performance and responsiveness of the application?

Table 54. Questions related to the understandability

Id	Question
Q25	How easy or difficult was it to understand the tool?
Q26	Did you understand the purpose and functionality of all elements of the application?
Q27	Do you have anything else to add or suggest?

Appendix B. RESOURCES

The resources section lists the tools we utilized, as well as the repositories of ontology representations and the OwlParser.

Repositories:

- Upper-level reference ontology repository (https://github.com/mubashar-iqbal/upper-level-reference-ontology) includes the RDF/XML based *ULRO.owl* file, ULRO architecture, ULRO instantiation, and SPARQL queries. The ULRO (https://mmisw.org/ont/~mubashar/ULRO) and ULRO instantiation (https://mmisw.org/ont/~mubashar/ULRO-instantiation) are hosted on MMISW registry (https://mmisw.org/ont).
- Corda-based security ontology repository (https://github.com/mubashar-iqbal/corda-security-ontology) includes the RDF/XML based *CordaSecOnt.owl* file, as-is classes hierarchy architecture, SPARQL queries, and how to use it. CordaSecOnt also hosted on MMISW registry (https://mmisw.org/ont/~mubashar/CordaSecOnt).
- Healthcare security ontology repository (https://github.com/mubashar-iqbal/HealthOnt) includes the RDF/XML based *HealthOnt.owl* file, as-is classes hierarchy architecture, SPARQL queries, and how to use it. HealthOnt also hosted on MMISW registry (https://mmisw.org/ont/~mubashar/HealthOnt).
- OwlParser code is available on GitHub (https://github.com/mubashar-iqbal/OwlParser) that is forked from (https://github.com/mairistioja/OwlParser-frontend). The tool is hosted on (https://owlparser.cs.ut.ee).

Tools:

- Protégé (https://protege.stanford.edu) is a free, open-source ontology editor to build knowledge-based solutions.
- OntoGraf (https://protegewiki.stanford.edu/wiki/OntoGraf) is a Protégé plugin for interactively navigating through the relationships of OWL-based ontology representations.
- Pellet reasoner (https://www.w3.org/2001/sw/wiki/Pellet) is a Protégé plugin to check the consistency of ontology, compute the classification hierarchy, explain inferences, and answer SPARQL queries.
- OOPS! (Ontology Pitfall Scanner!) (http://oops.linkeddata.es/response-advanced.jsp) is a web-based tool to detect the common pitfalls that may appear when creating an ontology.
- OWLViz (https://protegewiki.stanford.edu/wiki/OWLViz) is a Protégé plugin to build is-a class hierarchies for OWL-based ontology.

ACKNOWLEDGEMENTS

My thesis would not have been possible without the support of my supervisor, family, and friends. I would like to express my gratitude to my supervisor, Raimundas Matulevičius. His vast knowledge and wealth of experience have continuously inspired me in my daily life and academic research. He has been the ideal teacher, mentor, and thesis supervisor, providing guidance and inspiration with the perfect blend of wisdom and humor. I appreciate and am grateful for my time working with him during my Ph.D.

I am appreciative to my thesis reviewers, Agnes Koschmider and Hans Weigand, for their valuable comments and insightful criticism, which considerably enhanced my thesis.

I am grateful to my father, Zafar Iqbal, and mother, Musarrat Bibi, for their unwavering love and support, which keeps me motivated and confident. My accomplishments and success are the results of their faith in me. My fervent thankfulness goes to my sister and brothers, who always support my endeavors and remind me of what matters in life. Finally, I want to express my heartfelt appreciation to my wife, Maria Iqbal. I will be eternally grateful for the unwavering love and support I received during my Ph.D. My daughters, Muntaha and Minha, are my greatest blessings since both kept inspiring me to pursue this lifetime achievement.

I wish to extend my appreciation to my colleagues at the University of Tartu and friends in Pakistan and Estonia for their unending encouragement while I was working on getting my Ph.D. I would like to thank the technical and support team at the University of Tartu Delta center for their assistance. I also want to express my sincere gratitude to all of the frontline COVID-19 heroes around the globe during the pandemic who risked their lives to keep us safe.

SISUKOKKUVÕTE

Viiteraamistik turvariskide haldamiseks plokiahela abil

Turvariskide juhtimine on suunatud väärtuslike varade tahtliku volitamata kahjustamise riski vähendamisele tasemeni, mis on süsteemi sidusrühmadele vastuvõetav, ennetades pahatahtlikku kahju, väärkasutust, ohte ja riske ning reageerides neile. Turvariskide haldamine hõlmab turvaohtude tuvastamist, analüüsimist ja käsitlemist, et säilitada süsteemi konfidentsiaalsus, terviklikkus ja kättesaadavus. Traditsiooniline tehnoloogia infrastruktuur on haavatav erinevate turvaohtude suhtes (nt andmete rikkumine, andmete vargus, ühe punkti tõrge), mis võivad muuta süsteemi konfidentsiaalsuse, terviklikkuse ja kättesaadavuse olematuks. Turbeohtude suhtlemiseks ja vähendamiseks turvalise tarkvara loomiseks on olemas mitmesugused programmid (nt OWASP), ohumudelid (nt STRIDE), turvariskide juhtimise mudelid (nt ISturvariskide juhtimine) ja eeskirjad (nt GDPR). Turvaohud arenevad aga pidevalt, sest traditsiooniline tehnoloogiline infrastruktuur ei rakenda turvameetmeid kavandatult. Vaatamata suurenevale turvaintsidentide arvule ei pea paljud organisatsioonid tarkvara turvalisust esikohal. Plokiahela infrastruktuur on sobiv lahendus traditsiooniliste rakenduste turvaohtude leevendamiseks. Siiski puuduvad ühtsed ja ametlikud teadmiste mudelid, mis toetaksid turvariskide juhtimist ja selgitust selle kohta, kuidas plokiahelapõhised rakendused saavad traditsiooniliste rakenduste turvaohte leevendada.

Lisaks areneb plokiahela domeen pidevalt, pakkudes uusi tehnikaid ja sageli vahetatavaid disainikontseptsioone, mille tulemuseks on kontseptuaalne ebaselgus ja segadus turvaohtude tõhusal käsitlemisel. Kuigi plokiahelapõhiseid rakendusi peetakse vähem haavatavateks, ei saanud need erinevate turvaohtude (nt Sybili rünnak, topeltkulutamine) eest kaitsmise hõbekuuliks. Samamoodi on plokiahelapõhiste rakenduste turvariskide juhtimiseks vaja ühtseid ja ametlikke teadmiste mudeleid. Kokkuvõttes käsitleb käesolev lõputöö traditsiooniliste rakenduste turvariskide haldamise probleemi, kasutades plokiahelat vastumeetmena, ja plokiahelapõhiste rakenduste turvariskide juhtimist.

Tuvastatud uurimisprobleemi lahendamiseks püstitame peamise uurimisküsimuse: Kuidas saab välja töötada võrdlusraamistiku traditsiooniliste ja plokiahelapõhiste rakenduste turvariskide juhtimiseks? Uurimisküsimusele vastamiseks alustame uuringuga, kuidas plokiahelapõhised rakendused leevendavad traditsiooniliste rakenduste turvaohte, kus plokiahela turvaohte vaadeldakse kahest erinevast vaatenurgast. Näiteks traditsiooniliste rakenduste turvaohud, mida saab leevendada plokiahela abil, ja plokiahelapõhiste rakenduste turvaohud. Tuletame traditsiooniliste rakenduste komponendid ja koondame tuvastatud turvaohud, sealhulgas plokiahela komponendid. Tulemuseks on plokiahelal põhinev võrdlusmudel, mis järgib turvariskide haldamise domeenimudelit. Lisaks oleme laiendanud plokiahelapõhist võrdlusmudelit, et hõlmata plokiahelapõhiste rakenduste turvaohte ja nende vastumeetmeid. Hindame plokiahelapõhist võrdlusmudelit, uurides traditsiooniliste rakenduste andmete manipuleerimise ohtu ning plokiahelapõhis-

te rakendustega seotud Sybili ja topeltkulutamise ohte. Plokiahelal põhinev võrdlusmudel kujutab aga turvariskide haldamise staatilisi teadmisi ja ebaefektiivset instantsi. Esitame kõrgema taseme võrdlusontoloogiat kui alusontoloogiat nende lünkade ületamiseks plokiahelapõhise võrdlusmudeli kontseptualiseerimisega. Kõrgema taseme viiteontoloogia pakub semantilist koostalitlusvõimet, turvariskide haldamise üldkontseptsioone, mis on ühised kõikidele valdkondadele ja võimaldavad ühise aluse infoturbevaldkonna ontoloogiatele. Kõrgema taseme võrdlusontoloogia esitab struktuurse esituse, mis võib toetada dünaamilist teadmiste kodeerimist ja käivitamist infoturbealaste teadmistega domeenispetsiifiliste rakenduste turberiskide haldamiseks. Kõrgema taseme referentsontoloogia sobivuse ja õigsuse hindamiseks viisime läbi ekspertide arutelu ja värvilise Petri võrkude põhise hindamise, võttes vastumeetme lahenduseks plokiahela.

Kontseptsiooni tõestuseks pakume ülemise taseme viiteontoloogia kahte eksemplari. Kõrgema taseme võrdlusontoloogia esimene eksemplar sisaldab lubatud plokiahela komponente, mis kasutavad Cordat lubatud plokiahelana, ja finantsjuhtumit, mis on seotud kapitalituru tehingujärgse sobitamise ja kinnitamisega. Corda turvaontoloogiat hinnatakse automatiseeritud tööriistade, kvalitatiivsete hindamiskriteeriumide ja ülesannetepõhise hindamise abil. Ülemise taseme võrdlusontoloogia teine eksemplar sisaldab lubadeta plokiahelate komponente ja tervishoiujuhtumit. Kasutades automatiseeritud tööriistu, kvalitatiivseid hindamiskriteeriume ja ülesannetepõhist hindamist, kasutame tervishoiu turvalisuse ontoloogia hindamiseks ka seljavaluga patsientide tervishoiurakendust. Mõlemad ontoloogiaesitlused aitavad traditsiooniliste rakenduste turvariskide juhtimisel kombineerida plokiahelat kui vastumeetme lahendust ja plokiahelapõhiste rakenduste turvariskide juhtimist. Lisaks koostasime veebipõhise ontoloogia parsimise tööriista OwlParser. OwlParser kasutab väljatöötatud ontoloogiaesitusi ja pakub veebipõhist uurija funktsiooni, mis võimaldab sukelduda turberiskide juhtimise kontseptsioonidesse ja aidata kasutajatel leida nende vajadustele vastavaid kontseptsioone ja nende vahel liikuda. OwlParseri kasutatavuse hindamiseks viime läbi küsitluse tarkvaraarendajate, infoturbe spetsialistide, plokiahela ja ontoloogiaekspertidega.

Selle lõputöö panused järgivad turvariskide juhtimise domeenimudeli põhikontseptsioone. Me kasutame Protégét OWL-põhiste ontoloogiaesitluste loomiseks ja seejärel avaldame need MMISM-is ja GitHubi avalikus hoidlas. OwlParseri lähtekoodile pääseb juurde ka GitHubis. Hindasime selle lõputöö panust erinevate ülalkirjeldatud meetodite abil. Kaastööde tulemusel loodi ontoloogiapõhine turberaamistik turvariskide haldamiseks plokiahela abil. Raamistik loob ühisosa ja süstemaatilise arusaama, mis aitab teadlastel, arendajatel, praktikutel ja teistel sidusrühmadel selgitada traditsiooniliste rakenduste turvariskide juhtimist. Raamistik on dünaamiline, toetab turberiskide haldamise iteratiivset protsessi ja seda saab pidevalt värskendada, kui ilmnevad uued turvaohud, haavatavused või vastumeetmed. See saab edastada turvavajadusi ja aidata plokiahelapõhiste rakenduste turvariskide haldamisel. Kokkuvõttes, võib võrdlusraamistik potentsiaalselt vähendada traditsiooniliste ja plokiahelapõhiste rakenduste turvaohte.

CURRICULUM VITAE

Personal data

Name:	Mubashar Iqbal
Date of Birth:	05.06.1989
Citizenship:	Pakistani
Language Skills:	Urdu, English, Estonian
E-mail:	mubashar.iqbal@ut.ee

Education

2018–2022	University of Tartu, Tartu, Estonia, Doctor of Philosophy in Computer Science
2015–2018	Tallinn University, Tallinn, Estonia, Master of Science in Human Computer Interaction
2007–2011	National College of Business Administration and Economics, Lahore, Pakistan, Bachelor of Science in Computer Science

Employment

2019–	Junior Research Fellow, University of Tartu, Tartu, Estonia
2015–2018	Software Developer, Saule IT Services, Tallinn, Estonia
2013–2015	Software Developer, Confiz Solutions, Lahore, Pakistan
2011–2013	Software Developer, Amco IT Systems, Lahore, Pakistan

Scientific work

Main fields of interest:

- blockchain
- information systems
- information security

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Mubashar Iqbal
Sünniaeg:	05.06.1989
Kondakondsus:	Pakistani
Keeled:	urdu, inglise, eesti
Meiliaadress:	mubashar.iqbal@ut.ee

Haridus

2018–2022	Tartu Ülikool, Tartu, Eesti, Arvutiteaduse filosoofiadoktor
2015–2018	Tallinna Ülikool, Tallinn, Eesti, Inimese-arvuti interaktsiooni magistrikraad
2007–2011	Ärijuhtimise ja majanduse rahvuskolledž, Lahore, Pakistan, Arvutiteaduse bakalaureuse kraad

Teenistuskäik

2019–	Nooremteadur, Tartu Ülikool, Tartu, Eesti
2015–2018	Tarkvara arendaja, Saule IT Services, Tallinn, Eesti
2013–2015	Tarkvara arendaja, Confiz Solutions, Lahore, Pakistan
2011–2013	Tarkvara arendaja, Amco IT Systems, Lahore, Pakistan

Teadustegevus

Peamised uurimisvaldkonnad:

- plokiahel
- infosüsteemid
- infoturve

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

I. R. Matulevičius, M. Iqbal, E. Ammar, S. Ayachi, M. Bakhtina, S. Ghannouchi, Ontological Representation of Healthcare Applications Security Using Blockchain Technology. In *Informatica*, 2022, vol. 33, pp. 1–30.

II. M. Iqbal and R. Matulevičius, "Exploring Sybil and Double-Spending Risks in Blockchain Systems," in *IEEE access*, 2021, vol. 9, pp. 76153-76177.

III. M. Iqbal, and R. Matulevičius, "Corda Security Ontology: Example of Post-Trade Matching and Confirmation," in *Baltic journal of modern computing*, 2020, vol. 8, pp. 638–674.

IV. M. Iqbal, and R. Matulevičius, "Comparison of Blockchain-Based Solutions to Mitigate Data Tampering Security Risk," In *Business Process Management: Blockchain and Central and Eastern Europe Forum*, 2019, pp. 13–28.

V. M. Iqbal, and R. Matulevičius, "Blockchain-Based Application Security Risks: A Systematic Literature Review," In *Advanced Information Systems Engineering Workshops*, 2019, pp. 176–188.

Other published work

I. O. Levasseur, M. Iqbal, R. Matulevičius, Survey of Model-Driven Engineering Techniques for Blockchain-Based Applications. In *PoEM'21 Forum: 14th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling*, 2021, vol. 3045, pp. 11–20.

II. M. Iqbal, R. Matulevičius, Blockchain as a Countermeasure Solution for Security Threats of Healthcare Applications. In *BPM: Blockchain and Robotic Process Automation Forum*, 2021, vol 428, pp. 67–84.

III. B. Düdder, V. Fomin, T. Gürpinar, M. Henke, M. Iqbal, V. Janavičienė, R. Matulevičius, N. Straub and H. Wu, Interdisciplinary Blockchain Education: Utilizing Blockchain Technology From Various Perspectives. In *Frontiers of Blockchain*, 2021, vol. 3, article no. 578022, pp. 1–8.

IV. M. Iqbal, A Reference Model for Security Risk Management of the Blockchain-based Applications. In *Advanced Information Systems Engineering Doctoral Consortium*, 2020, vol. 2613 pp. 1–8.

V. K. Mammadzada, M. Iqbal, F. Milani, L. García-Bañuelos, R. Matulevičius, Blockchain Oracles: A Framework for Blockchain-Based Applications. In *Business Process Management: Blockchain and Robotic Process Automation Forum*, 2020, vol. 393, pp. 190034.

VI. M. Iqbal, R. Matulevičius, Managing Security Risks in Post-Trade Matching and Confirmation Using CorDapp. In *Databases and Information Systems (DB&IS)*, 2020, vol 1243, pp. 325–339.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.
26. **Tauno Palts.** A Model for Assessing Computational Thinking Skills. Tartu 2021, 134 p.
27. **Liis Kolberg.** Developing and applying bioinformatics tools for gene expression data interpretation. Tartu 2021, 195 p.
28. **Dmytro Fishman.** Developing a data analysis pipeline for automated protein profiling in immunology. Tartu 2021, 155 p.
29. **Ivo Kubjas.** Algebraic Approaches to Problems Arising in Decentralized Systems. Tartu 2021, 120 p.
30. **Hina Anwar.** Towards Greener Software Engineering Using Software Analytics. Tartu 2021, 186 p.
31. **Veronika Plotnikova.** FIN-DM: A Data Mining Process for the Financial Services. Tartu 2021, 197 p.
32. **Manuel Camargo.** Automated Discovery of Business Process Simulation Models From Event Logs: A Hybrid Process Mining and Deep Learning Approach. Tartu 2021, 130 p.
33. **Volodymyr Leno.** Robotic Process Mining: Accelerating the Adoption of Robotic Process Automation. Tartu 2021, 119 p.
34. **Kristjan Krips.** Privacy and Coercion-Resistance in Voting. Tartu 2022, 173 p.
35. **Elizaveta Yankovskaya.** Quality Estimation through Attention. Tartu 2022, 115 p.