



D4.4 – TBOM Security and Trust Mechanism (first version)

PROJECT	
Project Number	101120962
Project Acronym	RESCALE
Project Title	Revolutionised Enhanced Supply Chain Automation with Limited Threats Exposure
Start Date	01.10.2023
Programme	HORIZON-CL3-2022-CS-01-02
DELIVERABLE	
Deliverable Type	R - Document Report
Workpackage	WP4
Deliverable Lead	ICERT
Editors	Alessandro Visintin
Contributors	ISI, AEGIS, INT, EISI, CBRL, DIE, UNSPMF, ICERT, BNR
Dissemination Level	Public

Abstract

This document presents the security and trust mechanisms for TBOM within the RESCALE project. It outlines a robust framework that leverages blockchain technology, cryptographic methods, and smart contracts to ensure the integrity, authenticity, and confidentiality of TBOM data. The document details the selection criteria for the blockchain platform, with a focus on Hyperledger BESU, and explores the integration of IPFS for distributed storage. It also addresses compliance with international standards and regulatory requirements, including GDPR and cybersecurity regulations. The security framework encompasses identity-based access control, permissions management, and integration with external PKI services.

Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author’s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.



This project has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101120962

Document Revision & Quality Assurance

Internal Reviewers

1. Danijela Boberic Krsticev - (UNSPMF)
2. Konstantinos Latanis - (S5)

Revisions

Version	Date	Partner	Overview
0.1	22/07/2024	ICERT	ToC
0.2	02/09/2024	ICERT	Sections structured
0.5	21/11/2024	ICERT	Finished draft
0.6	29/11/2024	ICERT, UNSPMF, S5	Internal review and text updated
0.7	04/12/2024	ICERT	Text updated
0.9	07/12/2024	ISI, AEGIS	Final comments
1.0	10/12/2024	BNR	Final version

Table of Contents

1	Introduction	7
1.1	Scope & Contribution	7
1.2	Relation to Work Packages, Deliverables, and Activities	7
1.3	Contribution to Work Package 4 and Project Objectives	8
1.4	Structure of the Document	8
2	Security and Trust Framework for TBOM Management	9
2.1	Supply Chain Security, Trust Models, and Compliance	9
2.1.1	Security Challenges and Trust Models in Supply Chains	9
2.1.2	Compliance with Standards and Regulatory Requirements	11
2.2	Technical Mechanisms for TBOM Security	13
2.2.1	Core Security Features, Cryptography, and Additional Attributes	14
2.2.2	Identity-Based Access Control and Permissions Management	15
3	Ledger Infrastructure for the Trusted Bill of Materials	17
3.1	Selection Criteria for the Ledger Platform	17
3.1.1	Justification for Using Blockchain as a Ledger	17
3.1.2	Key Criteria for Selecting the Blockchain Platform	18
3.2	Overview of blockchain platforms	20
3.2.1	BESU as the Preferred Ledger for RESCALE	21
3.2.2	Consensus Mechanism, Security, and Governance in BESU	23
3.3	The Role of the InterPlanetary File System in the Trust Storage Architecture . .	26
3.3.1	Introduction to IPFS	26
3.3.2	Integration of IPFS in the Trust Storage System	28
4	Smart Contract Implementation for TBOM Management	30
4.1	General Properties of Smart Contracts	30
4.1.1	Introduction to Smart Contracts	30
4.1.2	Benefits of Smart Contracts for TBOM Management	32
4.2	Smart Contract Implementation	33
4.2.1	Basic Implementation: HashStorage.sol	33
4.2.2	Advanced Implementation: HashManager.sol	35
4.2.3	Comparative Analysis of Implementations	37
5	Rescale Trusted Storage Architecture and Implementation	40
5.1	Architecture Overview	40
5.1.1	Confidentiality, Integrity and Availability	40
5.1.2	The networking system	41
5.1.3	Certification Authority Integration	42
5.1.4	Key Management System	43
5.2	Implementation strategy	44
5.2.1	Trust Storage Node	44
5.2.2	Trust Storage Client	50
6	Potential Challenges and Risks	54
6.1	Scalability and Performance Concerns	54
6.2	Security Risks and Vulnerabilities	54

6.3	Legal and Regulatory Challenges	55
6.4	Integration Complexities	55
6.5	User Adoption and Training	55
7	Roadmap and Future Work	57
8	Conclusions	59

List of Figures

1	Certificate Issuance Flow	42
2	Certificate Verification Flow	43
3	Trust Storage module structure.	44
4	Sequence diagram of add_document() operation.	52
5	Sequence diagram of get_document() operation.	53
6	Sequence diagram of deprecate_document() operation.	53

List of Abbreviations

ABI	Application Binary Interface. 44
BFT	Byzantine Fault Tolerance. 18
CA	Certification Authority. 42, 43
CI/CD	Continuous Integration and Continuous Deployment. 10
CID	Content IDentifier. 27–29, 51, 52
CSR	Certificate Signing Request. 42
DPoS	Delegated Proof of Stake. 21
EEA	Enterprise Ethereum Alliance. 21, 23
EVM	Ethereum Virtual Machine. 21, 55
GDPR	General Data Protection Regulation. 1, 12, 13, 22, 55
IBAC	Identity-Based Access Control. 15, 16
IBFT	Istanbul Byzantine Fault Tolerance. 21–25
ICT	Information and Communication Technology. 12
IPFS	Inter-Planetary File System. 1, 8, 26–29, 40, 41, 44, 48, 50–52, 54, 59
JWT	JSON Web Token. 14
KMS	Key Management System. 43
PKI	Public Key Infrastructure. 1, 14–16, 57
PoA	Proof of Authority. 22, 24, 25
PoS	Proof of Stake. 18, 20, 21, 24, 25
PoW	Proof of Work. 18, 20, 22, 24
QBFT	Quorum Byzantine Fault Tolerance. 22–25, 44–46
RBAC	Role-Based Access Control. 13, 57
SBOM	Software Bill of Materials. 12
TBOM	Trusted Bill of Materials. 1, 7, 8, 11–18, 28–33, 38–41, 43, 54–57, 59
TPS	Transactions Per Second. 21, 22
TTP	Trusted Third-Party. 11
ZKP	Zero-Knowledge Proof. 14, 41, 57

1 Introduction

The RESCALE project works to improve supply chain security by design, focusing on automating the assessment of software and hardware components to minimize vulnerabilities. The purpose of this project is to develop detailed cybersecurity audit processes and build reliable systems with strong security assurances. Through systematic analysis and improvement of each layer in computing systems, RESCALE creates and applies new methods and tools throughout the supply chain. This document outlines the project's progress by detailing the security and trust mechanisms for Trusted Bill of Materials (TBOM), along with relevant background and considerations.

1.1 Scope & Contribution

This document outlines the security and trust approaches for the TBOM, including a review of current technologies and plans to implement them in practice. It gives the RESCALE project, especially Work Package 4 (Task 4.3), key details needed to build a secure and reliable TBOM system. The security and trust methods are described in enough detail to start development, but since this document comes out before full implementation begins, we expect that some unexpected issues may arise. We will summarize any needed changes to these methods in a later document.

1.2 Relation to Work Packages, Deliverables, and Activities

This document originates from Task 4.3 in Work Package 4, but it's also connected to Work Packages 3 and 5.

Connection to Work Package 3. Task 4.3 produces a secure storage module for TBOMs. This secure storage module provides a robust and reliable repository for storing the TBOMs generated in Work Package 3. The integration ensures that the security assessment results and TBOMs produced in Work Package 3 are stored securely and with integrity, enhancing the overall security posture of the RESCALE framework.

Connection to Work Package 5. The security and trust mechanisms developed in Task 4.3 will undergo testing and validation as part of the overall system demonstration in Work Package 5. This validation process in Work Package 5 will provide practical information on the effectiveness of the TBOM security framework and the secure storage module in real-world scenarios. Feedback from Work Package 5 demonstrations may inform potential refinements to the security and trust mechanisms.

Deliverable D4.4 is an important part of building a secure supply chain, focusing on making TBOMs safe and trustworthy. These security and trust features are key to the goals of the entire project.

1.3 Contribution to Work Package 4 and Project Objectives

Deliverable 4.4 is a result of Task 4.3 in Work Package 4. It contributes to achieving two key objectives of the RESCALE project:

1. **Objective 1:** Design and develop a complete toolbox to audit and increase the security of supply chain based on emerging technologies for both hardware and software modules.
2. **Objective 3:** Provide a Trusted BOM approach that will infuse trust in software and hardware supply chain and promote trusted updates.

For Objective 1, the deliverable proposes the design of the Trust Storage module that provides a secure way to store TBOMs in a decentralized fashion. This module is a core component of the toolbox for enhancing supply chain security, leveraging emerging technologies for both hardware and software modules.

As for Objective 3, the deliverable adds to the TBOM security construction by providing a platform where TBOMs can be stored, retrieved, and deprecated while ensuring core cryptographic properties. This contribution directly supports the Trusted BOM approach, infusing trust in the software and hardware supply chain and promoting trusted updates.

1.4 Structure of the Document

The rest of this document is structured as follows.

- Section 2 presents the Security and Trust Framework for TBOM Management. It covers supply chain security challenges, trust models, compliance requirements, and technical mechanisms for TBOM security.
- Section 3 describes the Ledger Infrastructure for TBOMs, including selection criteria, justification for using blockchain, and an overview of Hyperledger BESU. It also discusses the role of Inter-Planetary File System (IPFS) in the trust storage architecture.
- Section 4 details the Smart Contract Implementation for TBOM Management, covering general properties of smart contracts and specific implementations.
- Section 5 outlines the RESCALE Trusted Storage Architecture and Implementation, providing an architecture overview and implementation strategy.
- Section 6 discusses potential Challenges and Risks, including scalability concerns, security risks, and legal/regulatory challenges.
- Section 7 presents the roadmap and future work, covering planned enhancements, further smart contract development, and integration with other RESCALE components.
- Section 8 concludes the document.

Each section provides detailed information on key aspects of the TBOM security and trust mechanisms, from high-level frameworks to specific technical implementations and future plans.

2 Security and Trust Framework for TBOM Management

2.1 Supply Chain Security, Trust Models, and Compliance

This section explores the critical security challenges and evolving trust models within modern supply chains, focusing on the complexities and risks associated with global software and hardware inter-dependencies. Section 2.1.1 outlines the primary security risks and trust mechanisms emerging in response to the heightened threat landscape, covering areas such as code integrity, counterfeit hardware, and dependency management. Section 2.1.2 then addresses the importance of adhering to international standards and regulatory frameworks to ensure supply chain resilience and regulatory compliance.

2.1.1 Security Challenges and Trust Models in Supply Chains

In the contemporary digital landscape, supply chains for software and hardware components are increasingly distributed, complex, and interdependent. This inter-connectivity, while essential for accelerating innovation and reducing development costs, introduces considerable security vulnerabilities. The dependence on third-party components and global distribution networks has turned supply chains into lucrative targets for adversaries aiming to infiltrate systems at scale. Notably, recent high-profile incidents, such as the SolarWinds [2] and Log4j [12] breaches, have underscored the need for robust security measures within supply chains, prompting a shift towards new trust models and heightened security standards.

Supply chains encounter numerous security challenges [25] stemming from their reliance on third-party components and extensive integration processes. These challenges are multifaceted, affecting both software and hardware domains, and necessitate a holistic view of supply chain security:

Code Integrity and Software Tampering. In an era of modular software development, software applications rely heavily on third-party libraries, frameworks, and packages, often sourced from open-source or commercial vendors. This reliance introduces risks of code tampering or insertion of malicious code. Attackers can modify source code to insert malware or back-doors that may remain undetected, as third-party software undergoes limited scrutiny during integration. Ensuring code integrity through cryptographic signatures and strict version control mechanisms is critical to preventing such risks.

Counterfeit and Malicious Hardware Components. Hardware components sourced globally are vulnerable to counterfeiting and tampering. Counterfeit hardware can degrade system performance or introduce latent vulnerabilities, while malicious hardware may include embedded threats that can compromise downstream components. Hardware Trojans [30], for instance, are unauthorized modifications introduced during manufacturing that can be activated later to disrupt functionality or extract sensitive information. This threat underscores the need for robust validation processes and a verified chain of custody across hardware supply chains.

Dependency Management and Cascading Vulnerabilities. Modern software applications are

rarely developed from scratch; they are built on a web of dependencies, with each third-party component introducing its own set of dependencies. This dependency tree model is inherently vulnerable, as a single compromised component can propagate vulnerabilities through multiple layers. The complexity of tracking and securing each dependency in this tree structure is a significant challenge. Additionally, vulnerabilities in widely used components can lead to cascading risks, as seen with the Log4j vulnerability [12], which affected countless systems globally.

Side-Channel and Processor-Level Attacks. Side-channel attacks exploit the physical properties of hardware, such as power consumption, timing, and electromagnetic emissions, to gather information about a system's internal state without directly accessing its data. Processor-level attacks, like transient execution vulnerabilities (e.g., Spectre [18] and Meltdown [21]), target the architecture of processors to exploit speculative execution paths. These attacks highlight the potential risks at the hardware level that can be leveraged to circumvent traditional software protections.

Dependency on Open Source Components and Potential Exploits. Open source software components are integral to modern applications due to their accessibility and collaborative development model. However, the same accessibility that promotes innovation also opens the door for adversaries to study and exploit vulnerabilities. Security gaps in widely adopted open-source components can expose entire ecosystems to threats, with attackers able to embed malicious code in less-regulated repositories. Maintaining security across these components requires rigorous tracking, vulnerability assessment, and patching.

Supply Chain Attacks via CI/CD Pipelines and Update Mechanisms. Continuous Integration and Continuous Deployment (CI/CD) pipelines [31] facilitate rapid deployment of software updates but are increasingly targeted by attackers. By compromising CI/CD processes, attackers can inject malicious code into software before distribution. Similarly, update mechanisms for software and firmware are critical vectors, as seen in the SolarWinds [2] attack, where compromised updates were used to distribute malware. Securing these pipelines and ensuring authenticated, verified updates are essential to safeguarding downstream systems.

Human and Process-Related Security Risks. Beyond technical vulnerabilities, human error and insufficient process controls also contribute to supply chain risks. Misconfigurations, lack of adherence to security policies, and inadequate training in secure development practices can create weaknesses that attackers exploit. As supply chains involve multiple stakeholders across regions, establishing consistent, enforced security practices is vital to reducing risks introduced by human factors.

In response to the growing risks within supply chains, several trust models have emerged that emphasize transparency, decentralized verification, and continuous security assurance. These approaches fundamentally challenge traditional notions of implicit trust by establishing more rigorous verification and validation processes.

The **Zero-Trust Model** [5] forms the cornerstone of many modern supply chain security strategies, operating on the principle that no component, user, or system within a network should be implicitly trusted. Instead, zero-trust mandates continuous authentication and authorization of each interaction within the supply chain, with each request evaluated independently before access is granted. This model significantly reduces the risk of insider threats and mitigates unauthorized access, assuming that all internal and external sources are potentially compro-

mised. Zero-trust policies, therefore, require a combination of identity verification, access controls, and granular permissions, supporting a resilient and secure framework for supply chain security.

In addition to zero-trust principles, **blockchain technology** [8] offers a decentralized mechanism for enhancing traceability and transparency across supply chains. By implementing a blockchain-based ledger, every transaction and modification to supply chain components is immutably recorded, providing an independent verification of each component's origin, version, and security status. Blockchain's decentralized nature prevents unauthorized modifications and ensures that records remain tamper-proof, forming a comprehensive audit trail that addresses key challenges of forgery and unverified updates. Through blockchain-based models, stakeholders across the supply chain can verify a component's authenticity and gain confidence in its provenance.

Trusted Third-Party (TTP) [1] verification further supports this trust framework by incorporating independent entities responsible for auditing and certifying supply chain components. These third-party auditors assess each component for adherence to security standards and quality assurance metrics, providing additional verification for stakeholders. TTPs offer certifications based on stringent evaluations, addressing the limitations of self-attested security measures and enhancing confidence in the supply chain. However, it is essential that TTPs operate with transparency and adhere to standardized, impartial assessment protocols to maintain credibility and reduce potential biases.

Smart contracts [17], which are self-executing agreements coded onto blockchain platforms, enable automated enforcement of access controls and compliance rules within supply chains. By embedding these rules directly into smart contracts, organizations can ensure that permissions, access levels, and compliance policies are consistently enforced without manual oversight. This approach establishes a transparent and self-governing security framework, where actions are executed based on pre-defined rules and are independently verifiable. As a result, smart contracts enhance the security and reliability of access controls, reducing the need for intermediary verification steps and facilitating seamless compliance within complex, multi-tiered supply chains.

Finally, advanced trust models in supply chain security emphasize **dependency mapping** and **continuous monitoring** [13]. Dependency mapping involves creating a detailed profile of each component's interdependencies, allowing organizations to track and validate every external link within their products. Continuous monitoring tools then provide real-time tracking of the security posture of each component, enabling organizations to identify emerging threats and vulnerabilities as they arise. This proactive monitoring approach is essential for maintaining robust security across dynamic environments, ensuring that evolving threats do not compromise the supply chain's integrity.

2.1.2 Compliance with Standards and Regulatory Requirements

Ensuring compliance with international standards and regulatory frameworks is critical in building secure, transparent, and reliable supply chains. For the TBOM within the RESCALE project, aligning with recognized standards and regulatory requirements underpins the project's objectives of fostering security and resilience across software and hardware ecosystems. Compliance efforts within RESCALE focus on adhering to widely accepted standards for supply

chain security, managing cybersecurity risks, and ensuring data privacy in line with established legal frameworks.

A number of standards play a pivotal role in defining requirements and best practices for supply chain security, and RESCALE leverages these standards to structure its TBOM. Key standards include the ISO/IEC 20243 [14] (Open Trusted Technology Provider Standard), which establishes protocols for secure technology development, supply chain risk management, and delivery across Information and Communication Technology (ICT) providers. ISO/IEC 20243 provides a framework that aims to mitigate the risk of counterfeit components and unauthorized modifications in ICT products, which is crucial for maintaining the integrity of hardware and software supply chains. By setting forth guidelines for traceability, risk assessment, and supply chain transparency, ISO/IEC 20243 supports a comprehensive security approach across the lifecycle of technology products.

Another critical framework is NIST SP 800-161 [22] (Supply Chain Risk Management Practices for Federal Information Systems and Organizations) issued by the U.S. National Institute of Standards and Technology (NIST). This standard specifically addresses supply chain risk management for information systems, outlining key practices to identify, assess, and mitigate risks in complex supply chains. NIST SP 800-161 provides guidance on implementing layered security protections, ensuring trusted partnerships, and integrating risk management at each stage of the supply chain. This standard is particularly relevant in contexts where supply chains involve both commercial off-the-shelf and custom-built components that must meet stringent security requirements.

Standards specifically designed for Software Bill of Materials (SBOM), such as CycloneDX [7] and SPDX [27], also inform the structure and functionality of TBOMs within RESCALE. These SBOM standards enable comprehensive cataloging of software components, dependencies, and their origins. CycloneDX and SPDX specify standardized formats for documenting software components, allowing stakeholders to identify, track, and verify each piece of software used in a product. These standards provide crucial transparency, facilitating compliance with licensing and regulatory requirements and promoting greater visibility into the software's provenance, security status, and associated vulnerabilities. Integrating these SBOM standards into a TBOM ensures compatibility, consistency, and adherence to recognized practices across the software supply chain.

The RESCALE project adopts a compliance-first approach to TBOM management, ensuring that its operations align with both cybersecurity and data protection regulations. One of the central regulatory frameworks governing data privacy and protection in RESCALE is the General Data Protection Regulation (GDPR) [11], a cornerstone regulation for data privacy within the European Union. GDPR mandates stringent requirements for data handling, processing, and storage, including explicit user consent, data minimization, and robust data security practices. In compliance with GDPR, RESCALE's TBOM framework includes provisions for safeguarding personal and sensitive information, particularly as it pertains to data generated, stored, or shared within the supply chain ecosystem. GDPR compliance within RESCALE extends to data transfer protocols, access controls, and mechanisms to support data subject rights, ensuring that all TBOM operations uphold privacy standards.

Beyond GDPR, RESCALE adheres to additional cybersecurity requirements that focus on securing critical infrastructure and protecting supply chain data from cyber threats. These cybersecurity measures encompass guidelines for risk assessment, incident response, and security

auditing to ensure the ongoing resilience of the TBOM. The project's alignment with cybersecurity standards is structured to safeguard supply chain data integrity, protect sensitive information from unauthorized access, and reduce exposure to cybersecurity risks. By incorporating recognized cybersecurity practices, RESCALE's TBOM framework aligns with international standards, fortifying its overall approach to supply chain security.

Blockchain Compliance and Legal Considerations

As RESCALE's TBOM integrates blockchain technology for transparency, traceability, and immutability, several legal and compliance considerations arise regarding data storage, access control, and identity verification. Blockchain's inherent immutability—where records cannot be altered once entered—presents unique challenges and advantages in compliance contexts. For instance, data immutability aligns well with requirements for maintaining secure, tamper-proof records, as stipulated in various cybersecurity and audit standards. However, immutability also introduces potential issues with data governance, particularly when handling data that may be subject to deletion requests or amendments under privacy regulations like GDPR.

One important aspect of blockchain compliance is access control and permission management. Blockchain technology offers an inherently decentralized access model, where records are accessible across a distributed network. To align with regulatory requirements, it is essential to implement layered permissions and identity verification mechanisms. Role-Based Access Control (RBAC) [24] and identity-based authentication frameworks can restrict sensitive information to authorized personnel, meeting compliance mandates for data protection while ensuring that transparency and traceability are preserved within the TBOM.

Identity verification on blockchain platforms is another regulatory consideration, as it involves securely associating each entry and transaction with a verified user or entity. Identity verification on decentralized platforms is achieved through cryptographic keys, providing a secure method for validating the origin of each entry within the TBOM. Cryptographic identity frameworks ensure that only verified entities can modify or access sensitive information, supporting both compliance and security objectives. Legal frameworks related to identity verification in distributed environments continue to evolve, necessitating a proactive approach to align blockchain operations within the TBOM with emerging standards.

Additionally, the decentralized and cross-border nature of blockchain technology raises questions about data sovereignty and jurisdictional compliance. Data stored on distributed ledgers may reside on nodes across multiple jurisdictions, each with its own regulatory requirements for data protection and governance. Ensuring compliance in a cross-border context involves careful consideration of where blockchain nodes are hosted and understanding how local data protection laws impact the data stored on those nodes. Compliance frameworks are evolving to address the unique legal requirements of blockchain, providing guidance on privacy, security, and governance to support compliant blockchain deployments.

2.2 Technical Mechanisms for TBOM Security

This section details the cryptographic and access control mechanisms securing TBOMs. Section 2.2.1 covers core cryptographic features - hashing, digital signatures, PKI, and blockchain encryption - establishing data integrity and traceability. Section 2.2.2 focuses on identity-based

access control, using cryptographic signatures and smart contracts to ensure that only authorized users can interact with TBOM data, further strengthened by integration with external PKI services.

2.2.1 Core Security Features, Cryptography, and Additional Attributes

To ensure the integrity, authenticity, and confidentiality of TBOM data, cryptographic methods are integral to the TBOM framework. Cryptography not only safeguards the data within the TBOM but also enhances trust by ensuring that only authorized entities can access, modify, or validate the information contained within it.

Cryptographic Methods

At the core of TBOM security are cryptographic techniques that serve as the foundation for data protection. These methods establish a verifiable, secure, and reliable structure within the TBOM, ensuring that only trusted entities can access, modify, or validate information.

Cryptographic hashes [26] generate a unique, fixed-size hash value from the data in the TBOM, creating a digital fingerprint for each component and document. Commonly used hashing algorithms, such as SHA-256, ensure that any alteration to the TBOM's data is immediately detectable by observing changes in the hash value. This property is essential for verifying the integrity of each entry within the TBOM and helps establish a baseline of trust within the supply chain.

Public Key Infrastructure (PKI) [4] underpins secure communications within the TBOM framework, enabling encrypted data exchanges and secure authentication of entities. Using a pair of keys—one public and one private—PKI enables stakeholders to encrypt sensitive TBOM data, ensuring that only entities with the appropriate decryption key can access it. This framework provides a robust and secure communication channel within the TBOM, preserving both the confidentiality and integrity of shared data across the supply chain.

JSON Web Token (JWT)s [16] provide a secure way to transmit information as a JSON object and are especially useful for conveying identity and permissions within the TBOM framework. JWTs are cryptographically signed and can be verified to ensure the authenticity of the sender and the validity of the data. In the context of TBOM, JWTs are used to enforce access control policies by validating user identity, permissions, and access rights, supporting secure interactions with TBOM data.

Zero-Knowledge Proof (ZKP)s [28] are cryptographic protocols that allow one party to prove to another that a statement is true without revealing any information beyond the statement's validity. In TBOM management, ZKPs can be employed to verify a component's compliance with security policies or standards without exposing sensitive information about the component itself. This approach strengthens confidentiality within the TBOM, allowing stakeholders to verify security claims securely and privately.

Blockchain Encryption and Smart Contracts

In addition to traditional cryptographic techniques, blockchain-based encryption and smart contracts provide a powerful mechanism for enforcing data security within the TBOM.

Blockchain technology secures TBOM data through decentralized, immutable storage. Encryption within blockchain frameworks ensures that each entry in the TBOM is protected, with access restricted to authorized entities. By distributing the TBOM on a blockchain ledger, every modification to the TBOM is recorded and time-stamped, creating an immutable, auditable trail. This mechanism makes it extremely difficult for unauthorized entities to tamper with or alter TBOM data, as each change is permanently logged and can be independently verified.

Smart contracts, automated agreements stored and executed on blockchain platforms, are essential for maintaining the integrity and compliance of TBOM access permissions. These contracts can enforce access restrictions by executing predefined rules that allow only authenticated entities to access, view, or modify TBOM data. For instance, a smart contract could automatically validate a stakeholder's permissions and deny access if specific conditions are not met. By embedding these access policies directly into the smart contract, TBOM management benefits from a transparent and tamper-resistant enforcement mechanism.

Security Attributes in Trusted Bill of Materials

The TBOM framework incorporates several security attributes designed to protect supply chain data at multiple levels. These attributes are vital for establishing a resilient, verifiable structure that stakeholders can trust:

Metadata in the TBOM includes critical contextual information such as component version, creation date, supplier, and dependencies. Cryptographic measures protect this metadata to ensure that it remains unaltered, as any changes could impact the traceability and accountability within the supply chain.

A secure TBOM includes references to known vulnerabilities associated with each component, enabling stakeholders to assess risks accurately. Cryptographically securing these references is essential, as any unauthorized modification could misrepresent the component's risk profile. This transparency is crucial for security audits and proactive risk management.

The TBOM also encompasses proof of security testing, including results from vulnerability assessments and compliance checks. Testing proofs are stored in a verifiable, immutable format, ensuring that stakeholders can independently validate the security claims associated with each component. By maintaining cryptographically signed records of security testing, the TBOM reinforces trust in the integrity and robustness of each component within the supply chain.

2.2.2 Identity-Based Access Control and Permissions Management

Managing access and permissions is essential for maintaining a secure and trustworthy supply chain environment. By leveraging blockchain-based identity mechanisms aligned with Identity-Based Access Control (IBAC) principles, the Trust Storage module ensures that only verified and authorized entities can interact with or modify TBOM data. The blockchain infrastructure plays a crucial role in achieving these goals by integrating cryptographic signatures, smart contracts, and external PKI services to enhance verification and trust.

Blockchain technology inherently supports identity verification and access control by associating each transaction with a unique cryptographic identity. Within the TBOM framework, two main blockchain mechanisms - signature verification at transaction time and smart contract-based access control - work in tandem to enforce identity-based restrictions and ensure that

only authorized users can access the TBOM:

Signature Verification at Transaction Time. Each transaction on the blockchain requires a cryptographic signature from the sender, generated using the sender's private key, which corresponds to a unique public blockchain address. This signature serves as a digital identifier, establishing the sender's identity at the transaction level. When a user initiates a transaction, the blockchain network verifies the signature against the sender's public address, authenticating the transaction and confirming the request's origin. This signature verification process ensures that only entities with cryptographically verified identities can interact with TBOM data, enforcing IBAC principles by tying access to verified identities.

Additionally, the private/public key pairs used in this process can be associated with external PKI services, enhancing the level of trust in the system. By linking blockchain addresses to PKI-verified certificates, the framework gains higher assurance of the identity's authenticity, reducing risks associated with key compromise or impersonation. This integration with external PKI strengthens the TBOM's security by validating not only the ownership of cryptographic keys but also the legitimacy of each user's identity according to recognized PKI standards.

Smart Contract Access Verification through Sender Address. Beyond signature verification, smart contracts introduce an additional programmable layer of access control. Smart contracts within the TBOM framework contain predefined rules that enforce permissions based on the sender's blockchain address. When a transaction is submitted, the smart contract checks the sender's address against an authorized list or defined access levels embedded in the contract. If the sender's address is not recognized or lacks the required access level, the smart contract automatically rejects the transaction, preventing unauthorized access to TBOM data.

This smart contract-based verification mechanism allows for dynamic, role-based access control. TBOM administrators can define permissions according to roles such as supplier, auditor, or verifier—directly within the smart contract, linking access rights to specific identities. By embedding permissions into smart contracts, the TBOM framework aligns with IBAC principles, ensuring that only designated roles with verified identities can access, modify, or validate TBOM entries.

3 Ledger Infrastructure for the Trusted Bill of Materials

3.1 Selection Criteria for the Ledger Platform

This section examines the foundational technologies and architectural components underpinning the RESCALE project's TBOM framework. Section 3.1.1 provides a comprehensive justification for employing blockchain technology as the core ledger system, highlighting its unique attributes that align with RESCALE's requirements for security, transparency, and decentralization. Section 3.1.2 delves into the selection criteria for choosing an optimal blockchain platform, presenting a detailed comparison of prominent solutions.

3.1.1 Justification for Using Blockchain as a Ledger

A ledger is fundamentally a record-keeping system, designed to log transactions, events, or changes in data over time in a secure, organized manner. Traditionally, ledgers have been pivotal in finance, logistics, and supply chain management, where each transaction or alteration must be logged and validated to ensure accuracy and trust among stakeholders. Commonly used ledger examples include:

- **Centralized Financial Ledgers.** Managed by banks or financial institutions, these track transactions between accounts, ensuring transparency and regulatory compliance within a centralized structure.
- **Distributed Databases.** Used in corporate settings to record asset movements, regulatory compliance logs, or audit trails, enabling multiple users within the organization to interact with a unified data source.
- **Digital Health Records.** Distributed yet tightly regulated, these ledgers track patient information and medical history, balancing the need for privacy with authorized access across health networks.

While these traditional ledgers are effective in specific scenarios, they are limited by centralized control, which often introduces risks of tampering, inconsistencies in record-keeping, and single points of failure.

Blockchain presents an innovative approach to ledger management, particularly for decentralized and secure applications like RESCALE. Unlike traditional ledgers, blockchain is decentralized, meaning it operates across a network of nodes that collectively verify and maintain the integrity of each transaction. Each block in the chain contains a cryptographic hash of the previous block, a timestamp, and transaction data, making the chain inherently secure and resistant to tampering.

In contrast to centralized ledgers, blockchain operates without a single point of authority, distributing verification across its network of participants. This decentralization enhances transparency, resilience, and trust in the recorded information—a critical factor for the RESCALE

project, which aims to secure supply chains against complex vulnerabilities and enhance trust between stakeholders. Blockchain's unique attributes align well with RESCALE's TBOM requirements, offering the following essential properties:

- **Security and Immutability.** The cryptographic backbone of blockchain ensures that once data is recorded, it cannot be altered or deleted without consensus from the network, making it highly tamper-resistant. This immutability is critical for maintaining the integrity of the TBOM across the supply chain, where modifications to component data could otherwise go unnoticed.
- **Transparency and Accountability.** Blockchain's distributed nature allows all participants to independently verify the contents of the ledger, creating a transparent and accountable record for each transaction. For RESCALE, this means that all supply chain stakeholders can validate TBOMs directly, fostering trust across entities without relying on a central authority.
- **Automation and Self-Execution via Smart Contracts.** Blockchain enables programmable, self-executing transactions through smart contracts, which operate based on pre-defined rules. This automation is essential for RESCALE's compliance and auditing needs, as it allows smart contracts to perform routine checks and validations automatically, reducing overhead and ensuring that the TBOM remains compliant with standards at all times.
- **Decentralization and Resilience.** Unlike centralized ledgers that are vulnerable to outages and breaches, blockchain's decentralized architecture enhances resilience, with each node maintaining a complete copy of the ledger. This ensures continuity and data integrity, even if parts of the network experience failures, supporting RESCALE's goal of a secure, robust supply chain system that minimizes risks of data loss or tampering.

3.1.2 Key Criteria for Selecting the Blockchain Platform

To choose the most suitable blockchain, a set of criteria evaluates blockchain platforms' capabilities against the project's needs:

- **Type.** Distinguishes between public, permissioned, and private (local deployment) blockchains. This defines the level of openness and control over data access and transactions.
- **Consensus Mechanism.** Determines how network participants agree on transactions, affecting security, energy efficiency, and transaction speed. Common methods include Proof of Work (PoW), Proof of Stake (PoS), and Byzantine Fault Tolerance (BFT).
- **Scalability.** Measures transaction throughput in terms of transactions per second, critical for high-demand applications.
- **Security.** Evaluates resistance to vulnerabilities, ensuring that data integrity and trust are maintained.
- **Interoperability.** Assesses the blockchain's ability to interact with other chains and systems, enhancing integration flexibility.

- **Governance.** Describes the decision-making framework for protocol changes, affecting adaptability and evolution. Governance can be on-chain (protocol-encoded) or off-chain (managed by entities).
- **Cost.** Measures transaction affordability, often essential for scalability in supply chain applications.
- **Developer Ecosystem.** Reflects the size and activity of the blockchain’s developer community, influencing available resources and tool support.
- **Privacy:** Indicates the confidentiality and privacy options provided by the blockchain, crucial for secure handling of sensitive supply chain data.
- **Compliance.** Assesses the platform’s ability to meet legal and regulatory requirements, particularly significant for applications in regulated sectors.

Building on the core properties needed for a secure and efficient ledger infrastructure in RESCALE, a set of specific evaluation metrics guides the selection of a blockchain platform. These metrics focus on critical aspects such as security, scalability, and interoperability, which align with RESCALE’s requirements for a trusted, resilient TBOM system. Table 1 summarizes these parameters, providing explanations and methods for quantifying each to support a structured comparison of candidate blockchain platforms.

Table 1: Comparison metrics to assess blockchain platforms suitability. These metrics consider the essential factors impacting security, scalability, and integration capability, among others.

Parameter	Explanation	Importance	Quantification
Type	Distinguishes between public, permissioned, and private blockchains	Determines decentralization level	Public, Permissioned, Private
Consensus Mechanism	Describes how network participants reach agreement on transactions	Affects security, speed, and energy efficiency	Type of consensus mechanism
Scalability (TPS)	Measures transaction throughput	High throughput is critical	Numeric value representing TPS
Security	Assesses resilience to attacks and vulnerabilities	Ensures data integrity and trustworthiness	Qualitative assessment
Interoperability	Ability to integrate with other blockchains and external systems	Enhances integration flexibility	Qualitative assessment
Governance	Framework for decision-making on protocol updates	Affects adaptability	On-chain or off-chain
Cost (Tx Fees)	Transaction costs required for conducting transactions	Impacts affordability	Numeric value in USD
Developer Ecosystem	Size and activity of the developer community	Resource availability for development	Qualitative assessment
Privacy	Level of data confidentiality and privacy	Important for data protection	Qualitative assessment
Compliance	Ability to meet regulatory and legal requirements	Critical for regulated industries	Qualitative assessment

3.2 Overview of blockchain platforms

In the rapidly evolving landscape of blockchain technology, several platforms have emerged as prominent solutions for various use cases and it is essential to provide an overview of the most notable blockchain platforms.

Ethereum (ETH) [9] is a decentralized, open-source blockchain platform known for its robust smart contract capabilities and extensive developer community. It is transitioning from a PoW to a PoS consensus mechanism with Ethereum 2.0 to improve scalability and reduce energy consumption. Ethereum's native cryptocurrency is Ether (ETH). Transaction fees on Ethereum can be variable, ranging from \$1 to over \$50 depending on network congestion. Despite potential high costs, Ethereum's widespread adoption and strong ecosystem make it a leading platform for decentralized applications.

Binance Smart Chain (BSC) [3] is a high-performance blockchain network that uses Delegated Proof of Stake (DPoS) consensus. It offers high scalability and low transaction fees (around \$0.10). BSC is compatible with the Ethereum Virtual Machine (EVM), allowing for seamless porting of Ethereum dApps. It provides a balance between performance and cost, making it popular for decentralized finance (DeFi) projects. However, the centralization of governance under Binance is a consideration for users seeking more decentralized platforms.

Cardano (ADA) [15] uses the Ouroboros PoS protocol, emphasizing security and formal verification for smart contracts. It offers moderate scalability (250 Transactions Per Second (TPS)) and low transaction fees (around \$0.17). Cardano's governance is conducted on-chain through Cardano Improvement Proposals. It is designed for regulatory compliance and scientific rigor, attracting projects that prioritize security and sustainability.

Solana (SOL) [20] combines Proof of History with PoS to achieve very high scalability (65,000 TPS) and extremely low transaction fees (about \$0.00025). Its rapid growth and efficient consensus mechanism have made it a strong contender for dApps requiring high throughput. Solana's governance is managed off-chain by the Solana Foundation, and the ecosystem is expanding quickly with substantial support for developers.

Polkadot (DOT) [10] is designed for interoperability and scalability, using a Nominated Proof of Stake (NPoS) consensus mechanism. It allows multiple parachains to operate in parallel, offering high throughput and low transaction fees (0.05–0.10). Polkadot's on-chain governance model enables DOT holders to participate in decision-making. It is suitable for projects requiring cross-chain communication and robust governance.

Quorum [6] is a permissioned blockchain platform based on Ethereum, optimized for enterprise use. It supports Istanbul BFT and Raft consensus mechanisms, ensuring high performance and private transactions. Quorum has no native transaction fees, with costs tied to infrastructure. Its compatibility with Ethereum tools makes it a practical choice for enterprises looking to leverage Ethereum's capabilities in a private setting.

Corda [29] is a permissioned blockchain platform designed for financial services, focusing on privacy and scalability. It uses pluggable consensus mechanisms with notary services to ensure transaction validity. Corda does not have native transaction fees, and costs are related to infrastructure and maintenance. Its high level of privacy and compliance features make it suitable for regulated industries and consortiums requiring confidential transactions.

Hyperledger BESU [23] is an open-source Ethereum client designed for both public and private permissioned network use cases. It implements the Enterprise Ethereum Alliance (EEA) specification and supports multiple consensus algorithms, including Proof of Work, Proof of Authority, and Istanbul Byzantine Fault Tolerance (IBFT) 2.0. BESU offers robust privacy features, permissioning capabilities, and enterprise-grade performance, making it highly suitable for complex business networks.

3.2.1 BESU as the Preferred Ledger for RESCALE

After careful evaluation of prominent blockchain platforms, Hyperledger Besu emerges as the optimal choice for the RESCALE consortium. This selection is based on several key factors that align with the project's objectives and requirements, particularly in revolutionizing sup-

ply chain automation across diverse industrial sectors. The findings from the comparison are summarized in the Table 2.

BCP	TYP	COM	SCA	SEC	INT	GOV	COS	DEV	PRI	CPL
ETH	PUB	POS	15-30	HIGH	LOW	ONC	\$1-\$50+	HIGH	LOW	LOW
BSC	PUB	DPOS	90-100	MID	HIGH	CEN	\$0.10	HIGH	LOW	LOW
SOL	PUB	POH POS	60K-65K	HIGH	LOW	OFF	\$0.00025	MID	LOW	LOW
DOT	PUB	NPOS	1K	HIGH	HIGH	ONC	\$0.05-\$0.10	MID	LOW	MID
QRM	PER	IBFT, Raft	150-2K	HIGH	LOW	OFF	NF	MID	HIGH	HIGH
CRD	PER	PLG	15-1.5K	HIGH	HIGH	ONC, OFF	NF	MID	HIGH	HIGH
BES	PUB, PER	POW, POA, IBFT, QBFT	500-1K	HIGH	HIGH	ONC, OFF	CUS	MID	HIGH	HIGH

Table 2: Comparison of Blockchain Platforms.

The blockchain platforms (BCP) are abbreviated as ETH (Ethereum), BSC (Binance Smart Chain), SOL (Solana), DOT (Polkadot), QRM (Quorum), CRD (Corda), and BES (Hyperledger Besu). The type (TYP) of blockchain can be either Public (PUB) or Permissioned (PER). Consensus mechanisms (COM) include Proof of Stake (POS), Delegated Proof of Stake (DPS), Proof of History (POH), Nominated Proof of Stake (NPS), Istanbul BFT (IBF), Raft (RAF), Pluggable consensus (PLG), Proof of Work (POW), Proof of Authority (POA), and Quorum BFT (QBFT). Scalability (SCA) is measured in transactions per second, with thousands expressed as K. Security (SEC), Interoperability (INT), Developer Ecosystem Quality (DEV), Privacy (PRI), and Compliance (CPL) are rated as LOW, MID, or HIGH. Governance (GOV) can be On-Chain (ONC), Off-Chain (OFF), Centralized (CEN), or Customizable (CUS). Cost (COS) is expressed as a range in dollars, with some platforms having no native fees (NF) or customizable costs (CUS).

Hyperledger Besu’s unique ability to operate in both public and permissioned environments provides RESCALE with unparalleled flexibility. This dual-mode capability allows for seamless integration with existing supply chain systems while maintaining the option to interact with public networks when necessary. Such versatility is crucial for a project of RESCALE’s scope and ambition. Furthermore, Besu’s support for multiple consensus algorithms (PoW, Proof of Authority (PoA), IBFT, Quorum Byzantine Fault Tolerance (QBFT)) offers the ability to fine-tune its blockchain infrastructure to meet specific security and performance requirements. This adaptability is particularly valuable in a research and innovation context, where different use cases may demand varying levels of decentralization and transaction finality.

In terms of performance, Besu strikes a balance between high throughput and decentralization, with a transaction capacity of 500 – 1K TPS. While not the fastest in raw TPS, its scalability is more than sufficient for most supply chain applications and can be optimized further if needed. This ensures that the RESCALE platform can handle increasing transaction volumes as the project expands. Equally important are Besu’s high ratings in security, compliance, and privacy, which are critical for a project dealing with sensitive supply chain data. These features align well with the stringent security requirements of Horizon Europe projects and the need to adhere to regulations such as GDPR. The high interoperability rating of Besu is another significant advantage for RESCALE, facilitating easier integration with existing enterprise systems and potential future cross-chain operations. This is essential for a project aiming to create a

comprehensive supply chain solution. Besu's support for both on-chain and off-chain governance provides RESCALE with the necessary tools to implement a governance structure that balances decentralization with the need for consortium-level decision-making, crucial for managing a complex, multi-stakeholder project within the Horizon Europe framework.

From a resource management perspective, Besu's customizable cost structure allows RESCALE to optimize transaction costs, ensuring efficient use of project resources. This is particularly important for a publicly funded research initiative where cost-effectiveness is a key consideration. While rated as moderate in developer ecosystem quality, Besu benefits from the broader Hyperledger community support, providing RESCALE with access to a wealth of resources, documentation, and potential collaborations within the open-source blockchain community. When compared to other platforms, Hyperledger Besu offers distinct advantages. Ethereum and Binance Smart Chain, while popular, lack the privacy features and customizability required for a complex supply chain project. Solana offers high performance but falls short in privacy and compliance aspects crucial for EU-based projects. Polkadot provides good interoperability but lacks the specific enterprise features that Besu offers. Quorum and Corda, while strong in privacy and compliance, do not offer the same level of flexibility in terms of public/permissioned deployment options.

3.2.2 Consensus Mechanism, Security, and Governance in BESU

Hyperledger Besu is an open-source Ethereum client designed for both public and private permissioned network use cases. Developed under the Hyperledger umbrella, Besu implements the Enterprise Ethereum Alliance (EEA) specification, offering a robust, flexible, and enterprise-grade blockchain solution. Its architecture is built to be highly modular, allowing for easy implementation and upgrading of key blockchain features.

A key strength of Hyperledger Besu is its full compatibility with the Ethereum ecosystem. It supports all standard Ethereum APIs and can run the same smart contracts and decentralized applications as other Ethereum clients. This compatibility ensures interoperability with the broader Ethereum network while providing additional features tailored for enterprise use.

Besu's enterprise-focused design is evident in its enhanced privacy features, permissioning capabilities, and performance optimizations. It offers pluggable consensus algorithms, role-based access control, and private transaction management, addressing the specific needs of businesses and consortiums. These enterprise-grade features, combined with Ethereum compatibility, make Besu an ideal choice for organizations looking to leverage blockchain technology in a secure, scalable, and compliant manner.

Consensus mechanism

Hyperledger Besu supports multiple consensus mechanisms, each designed to meet different network requirements and use cases. The following are the primary consensus protocols available in Besu:

- **QBFT** is an enterprise-grade evolution of the IBFT 2.0 protocol, developed by ConsenSys. It is designed for improved scalability and processing requirements in enterprise networks. QBFT operates on a round-based consensus model where validators take turns proposing blocks. It requires a minimum of four validators to be Byzantine fault toler-

ant and provides immediate finality, meaning that once a block is confirmed, it cannot be reverted. QBFT is particularly well-suited for networks requiring high security and transaction finality.

- **IBFT** is the predecessor to QBFT and shares many of its characteristics. It also requires a minimum of four validators and provides immediate finality. IBFT 2.0 is designed to be more effective for large-scale blockchain networks compared to its original version. While still supported, it is generally recommended to use QBFT for new implementations due to its improvements over IBFT 2.0.
- **Clique (PoA)** is a simpler proof of authority consensus mechanism that can operate with as few as one validator. Unlike QBFT and IBFT 2.0, Clique does not provide immediate finality and can experience forks. It is faster in terms of block creation but is generally recommended for test networks rather than production environments due to its lower security guarantees.
- **Proof of Stake (PoS)** is used on the Ethereum mainnet and public testnets. In proof of stake, validators are chosen to create new blocks based on the amount of cryptocurrency they hold and are willing to "stake" as collateral. While supported by Besu for compatibility with Ethereum networks, it is not typically used in private, permissioned networks.
- **Etash (PoW)** is the original consensus mechanism used by Ethereum. It relies on computational work to validate blocks and secure the network. While supported by Besu, it is primarily used for small development networks or for maintaining compatibility with older Ethereum configurations.

For the RESCALE project, QBFT emerges as the optimal consensus mechanism due to several key factors. It provides enterprise-grade security through Byzantine fault tolerance, crucial for maintaining network integrity against potential malicious actors or node failures. In supply chain management, where transaction certainty is paramount, QBFT's immediate finality ensures that confirmed transactions cannot be reversed or altered. As an improvement over IBFT 2.0, QBFT offers better scalability, essential for RESCALE as the network grows and transaction volumes increase. It strikes a balance between security and performance, offering faster block times compared to proof of work while maintaining robust security guarantees. QBFT is designed for permissioned networks, aligning with RESCALE's need for a controlled, enterprise-focused blockchain environment. Unlike proof of work, QBFT does not require intensive computational resources, making it more environmentally friendly and cost-effective for long-term operation. It also allows for precise control over the validator set, enabling RESCALE to manage network participants effectively while maintaining decentralization.

Security

Hyperledger Besu offers a comprehensive set of security features that position it as a robust choice for enterprise-grade blockchain applications, particularly in permissioned network environments. When compared to other blockchain platforms, Besu's security model stands out in several key areas, combining the strengths of both public and private blockchain solutions.

At the core of Besu's security offering is its approach to privacy and confidentiality. By implementing private transactions through integration with Tessera, a privacy manager, Besu allows

for confidential transactions between specific parties without exposing details to the entire network. This approach strikes a balance between the openness of public chains like Ethereum and Binance Smart Chain, which offer limited privacy features, and the strict privacy of enterprise-focused platforms like Quorum and Corda. Besu's method provides the flexibility needed for complex enterprise use cases while maintaining a degree of transparency.

Complementing its privacy features, Besu offers comprehensive permissioning capabilities at both the node and account levels. This granular control over network access surpasses what's available in public blockchains like Ethereum or Solana, and is comparable to the permissioning capabilities of Quorum and Corda. Such fine-grained control is crucial for consortium and enterprise use cases where controlled access is paramount. Besu's security model is further enhanced by its support for multiple consensus algorithms, including IBFT 2.0, QBFT, and Clique PoA. This flexibility allows networks to choose the most appropriate consensus mechanism for their specific security and performance needs. While platforms like Ethereum have transitioned to PoS, and others like Solana use unique consensus mechanisms, Besu's variety of options provides greater adaptability for different network configurations.

In terms of smart contract security, Besu inherits Ethereum's capabilities but adds enterprise-grade features. It allows for disabling certain opcodes to prevent potential attacks, a feature not commonly found in public blockchains. This level of control over smart contract execution environments is comparable to other enterprise platforms like Quorum, providing an additional layer of security for critical business logic. While Besu itself does not handle key management, it integrates seamlessly with external key management systems. This approach, similar to that of Quorum and Corda, allows for more robust and flexible key security compared to public blockchain nodes that often manage keys internally. It enables enterprises to leverage existing security infrastructure and policies for key management.

For privacy-enabled networks, Besu assumes a level of trust among node operators, an approach similar to Quorum and Corda but differing from public blockchains where trust is minimized. Besu recommends using consensus mechanisms that support transaction finality (like IBFT 2.0) for production environments, enhancing security in private network setups. Besu's design also considers regulatory compliance, making it suitable for industries with strict regulatory requirements. Its permissioning and privacy features, combined with the ability to create detailed audit trails, put it on par with other enterprise-focused platforms in terms of compliance readiness. This is particularly important for sectors dealing with sensitive data or operating under stringent regulatory frameworks.

As an Ethereum-compatible client, Besu offers better interoperability with the broader Ethereum ecosystem compared to non-Ethereum based platforms. This allows for easier integration of existing Ethereum tools and smart contracts while still maintaining enterprise-grade security features. The open-source nature of Besu, being part of the Hyperledger project, also contributes to its security profile. It benefits from community scrutiny and contributions, potentially leading to faster identification and resolution of security issues, an advantage over proprietary solutions.

Governance

Hyperledger Besu offers a flexible governance model that supports both on-chain and off-chain governance processes, making it adaptable to various organizational needs. This dual approach allows networks to leverage the benefits of blockchain technology while maintaining the flexibility to incorporate traditional decision-making structures.

On-chain governance in Besu is facilitated through smart contracts and validator management systems. Validators, crucial participants in the network, are initially chosen at network startup and specified in the genesis file. This setup provides a predetermined set of trusted nodes to begin network operations. Once the network is operational, the validator set can be modified through a consensus-based voting mechanism. This dynamic management allows for changes without requiring a hard fork or network restart, ensuring adaptability and security. The voting process for managing validators includes several key steps:

- **Proposal:** Any existing validator can propose to add a new validator or remove an existing one.
- **Voting:** Upon receiving a proposal, other validators cast their votes. Each validator has one vote per proposal.
- **Consensus:** For a proposal to pass, it must receive votes from more than 2/3 of the current validator set. This supermajority requirement ensures broad support for changes.
- **Execution:** If a proposal achieves the required consensus, the change is automatically implemented at the start of the next epoch.

Off-chain governance complements these on-chain processes by allowing for more traditional decision-making methods that occur outside of the blockchain. This includes community discussions, improvement proposals, and consensus-building activities that are essential for complex decisions requiring extensive deliberation. Off-chain governance can involve forums and meetings where stakeholders discuss proposed changes, ensuring that all voices are heard before any formal implementation on-chain.

3.3 The Role of the InterPlanetary File System in the Trust Storage Architecture

This section explores the role of IPFS within the RESCALE project's trust storage architecture. Section 3.3.1 provides a comprehensive introduction to IPFS, detailing its fundamental principles, key features, and the innovative approach it brings to distributed data storage and retrieval. Section 3.3.2 delves into the integration of IPFS with the RESCALE trust storage system, highlighting how this integration addresses critical requirements for secure, efficient, and resilient supply chain data management. Additionally, this section examines the benefits and challenges of incorporating IPFS into the RESCALE architecture.

3.3.1 Introduction to IPFS

IPFS [19] is a revolutionary peer-to-peer protocol and network designed to create a distributed system for storing and accessing files, websites, applications, and data. IPFS aims to fundamentally change how information is shared and accessed on the internet by addressing content directly rather than relying on centralized servers and location-based addressing.

IPFS integrates concepts from peer-to-peer systems like BitTorrent for file sharing, Git for versioning, and self-certifying file systems. It provides a foundation for creating fully distributed applications and services that are resilient, upgradable, and free from central points of control. As a core component of Web3 infrastructure, IPFS is being used to build a wide range of decentralized applications, from file sharing and social media to supply chain management and scientific data repositories.

Key Aspects of IPFS

IPFS employs content-based addressing instead of using URLs that point to locations. Each file is given a unique cryptographic hash based on its content, allowing retrieval from any node storing that content. The decentralized architecture of IPFS operates as a distributed network of nodes that store and share content, eliminating single points of failure and censorship.

By retrieving content from the nearest available source rather than a central server, IPFS significantly improves efficiency, reducing bandwidth costs and improving content delivery speeds. The system also tracks version history of files and automatically deduplicates identical content across the network, improving storage efficiency. IPFS supports offline-first functionality, enabling access to content once it has been cached locally. This allows for resilient applications that can work without constant internet connectivity. By decoupling content from specific servers, IPFS aims to create a more permanent and robust web where content remains accessible even if the original host goes offline.

IPFS employs several key mechanisms to ensure data permanence and tamper-proof storage. Content addressing forms the foundation, where each file receives a unique Content Identifier (CID) based on its cryptographic hash, making the content itself determine its address and enabling tamper-evidence. Files are split into chunks and distributed across many nodes in the network, providing redundancy and ensuring data availability even if some nodes go offline. IPFS allows "pinning" of content, instructing nodes to retain specific data indefinitely rather than potentially discarding it during garbage collection. Pinning services can be used to ensure important data remains persistently available.

Once content is added to IPFS, it cannot be modified without changing its CID, preserving the original through immutability. Any changes result in new content with a new address. The content addressing system allows easy verification that retrieved data matches its CID, detecting any tampering through cryptographic verification. While not part of core IPFS, many projects combine IPFS storage with blockchains to create tamper-proof audit trails of data. Popular content naturally gets replicated across more nodes as it is accessed, increasing redundancy and availability.

IPFS includes features like InterPlanetary Name System, which allows creation of mutable pointers to IPFS content, enabling updates while maintaining verifiable links to the latest version. The related Filecoin network provides economic incentives for long-term storage of data on IPFS. Additionally, IPFS has built-in support for versioning of files, allowing tracking of changes over time in a tamper-evident way. By combining these features, IPFS creates a robust system for storing data in a distributed, permanent, and tamper-resistant manner. The content-addressed nature of the system, coupled with cryptographic verification and distributed storage, makes it extremely difficult to alter or censor data once it has been added to the network.

Pinning and garbage collection in IPFS

Pinning and garbage collection work together in IPFS to manage data retention and storage efficiency. By default, IPFS nodes cache content they download, making it available to other nodes. However, this cached content can be removed during garbage collection to free up disk space. Pinning allows users to specify content that should be permanently retained, protecting it from garbage collection. There are three types of pins: direct (pinning a single block), recursive (pinning a block and all its child blocks), and indirect (resulting from a parent block being recursively pinned). When content is added to IPFS, it's recursively pinned by default. Users can manually pin content using the `ipfs pin add` command, list pinned content with `ipfs pin ls`, and remove pins with `ipfs pin rm`. Pinned content remains on a node until explicitly unpinned, even after garbage collection. The `ipfs repo gc` command removes unpinned content to free up space. Pinning services allow users to pin content on remote IPFS nodes for added persistence. Proper use of pinning is crucial for data permanence on IPFS, as unpinned data may become unavailable over time as nodes perform garbage collection. The more nodes that pin a piece of content, the more permanent and available it becomes on the IPFS network overall. This system allows nodes to manage their storage while providing a mechanism to ensure specific content persists despite garbage collection processes.

3.3.2 Integration of IPFS in the Trust Storage System

The integration of IPFS with blockchain ledgers creates a powerful hybrid system for the RESCALE Trust Storage architecture. This approach leverages the strengths of both technologies to provide a robust, scalable, and secure solution for managing critical supply chain data. At its core, the integration enables complementary storage capabilities. The blockchain stores essential metadata and IPFS content hashes, while IPFS handles larger datasets like detailed TBOM files. This optimizes blockchain storage usage while maintaining data integrity. By using IPFS to store larger files off-chain, the blockchain's limited storage capacity is preserved for critical transaction data and metadata. This hybrid approach allows for virtually unlimited scalability of data storage while still benefiting from blockchain's immutability and transparency.

IPFS generates unique CIDs for each TBOM file, which are then stored on the blockchain as tamper-proof references. The blockchain thus acts as a verifiable index for TBOM data stored in IPFS, creating an immutable audit trail of when TBOM data is added or updated. This system provides cryptographic proof of the existence and integrity of TBOM data at specific points in time. The content-addressed nature of IPFS ensures that any change to a file results in a new CID, making it trivial to detect modifications. By recording these CIDs on the blockchain, RESCALE creates a tamper-evident system where the full history of changes to TBOM data can be reliably tracked and verified.

Smart contract integration allows automated validation and processing of TBOM information by interacting with IPFS data using stored CIDs. Additionally, the blockchain can manage access permissions to IPFS-stored TBOMs, enhancing security while preserving the system's decentralized nature. Smart contracts can be programmed to automatically verify the integrity of TBOM data, trigger updates based on predefined conditions, and manage complex access control policies. This automation reduces the need for manual intervention, increases the system's reliability, and enables more sophisticated and dynamic management of TBOM data throughout the supply chain.

Benefits of Using IPFS for TBOM Storage

The integration of IPFS with blockchain ledgers in the RESCALE trust storage architecture offers several key benefits. Content integrity is ensured through IPFS's content-addressing, making it impossible to tamper with stored data without detection. The decentralized redundancy of TBOM data across multiple IPFS nodes eliminates single points of failure and improves data availability. IPFS's peer-to-peer nature allows for efficient data distribution, reducing bandwidth costs and improving scalability. The system naturally supports versioning of TBOM files, with each version having a unique CID for easy tracking of changes over time. Its protocol-agnostic nature facilitates interoperability with various blockchain platforms, allowing for potential future expansion or migration of the RESCALE system. Large TBOM files that might be impractical to store directly on a blockchain can be efficiently managed by IPFS, supporting the inclusion of detailed technical specifications and documentation.

Additionally, deduplication of identical data across the network saves storage space when dealing with similar or partially overlapping TBOMs. Once retrieved, TBOM data can be cached locally, allowing for offline verification and reducing dependency on constant network connectivity. IPFS's design aims for long-term data persistence, helping ensure that TBOM records remain accessible even as technology evolves.

Challenges and Considerations

Integrating IPFS with blockchain in RESCALE also presents several challenges that need to be addressed. Data availability is not guaranteed to be permanent, as nodes hosting the data may go offline. Implementing incentive mechanisms or dedicated pinning services may be necessary to ensure long-term data retention. Privacy concerns arise as data stored on IPFS is public by default. Encryption of sensitive TBOM data before storage on IPFS is crucial to maintain confidentiality. The system must be designed to handle the eventual consistency model of IPFS, where data propagation across the network is not instantaneous. Careful management of access controls and permissions is necessary to prevent unauthorized access to sensitive supply chain data. The integration adds complexity to the overall system architecture, requiring additional components to manage the interaction between blockchain and IPFS.

4 Smart Contract Implementation for TBOM Management

4.1 General Properties of Smart Contracts

This section delves into the implementation of smart contracts within the RESCALE project's TBOM framework. Section 4.1.1 provides a comprehensive introduction to smart contracts, exploring their fundamental principles, key characteristics, and historical evolution. Section 4.1.2 examines the specific benefits of smart contracts for TBOM management, detailing how they enhance automation, security, transparency, efficiency, and real-time capabilities in supply chain operations.

4.1.1 Introduction to Smart Contracts

Smart contracts are self-executing programs stored and run on a blockchain network. At their core, smart contracts are pieces of code that automatically execute predefined actions when specific conditions are met, without requiring intermediaries. These contracts are written in programming languages compatible with blockchain platforms, such as Solidity for Ethereum. Once deployed on the blockchain, smart contracts operate autonomously, executing their programmed logic in response to transactions or events.

Unlike traditional legal contracts, which rely on human interpretation and enforcement, smart contracts are deterministic and self-enforcing. Traditional contracts outline terms and conditions in natural language, requiring parties to trust each other or rely on legal systems for enforcement. In contrast, smart contracts encode these terms directly into executable code. This fundamental difference allows smart contracts to automatically enforce agreements, transfer assets, or perform other actions without the need for intermediaries or manual intervention.

The role of smart contracts in automating and enforcing agreements is transformative. They can facilitate complex multi-party transactions, automate supply chain processes, manage digital assets, and enable decentralized applications. For example, in the context of RESCALE's supply chain security focus, smart contracts could automatically verify and update the status of TBOMs based on predefined security criteria. This automation reduces human error, increases efficiency, and enhances transparency in supply chain management.

While smart contracts are a type of blockchain transaction, they differ from simple value transfers in several key ways. Standard blockchain transactions typically involve sending cryptocurrency from one address to another. Smart contracts, however, can contain complex logic, store data, interact with other contracts, and modify the blockchain state in more sophisticated ways. They can trigger multiple actions, manage state changes over time, and even create new transactions or contracts. This expanded functionality makes smart contracts a powerful tool for implementing business logic and automating processes on blockchain networks.

Key characteristics of smart contracts

Smart contracts possess several key characteristics that make them uniquely suited for automat-

ing and enforcing agreements in a decentralized environment. These characteristics are fundamental to understanding how smart contracts function within the RESCALE project's trust storage architecture.

Smart contracts are designed to automatically execute when predefined conditions are met. This self-executing nature eliminates the need for manual intervention and ensures that contractual terms are carried out precisely as coded. In the context of RESCALE, this means that once a smart contract is deployed to manage aspects of the TBOM, it can automatically trigger actions such as updating the TBOM status, verifying component integrity, or initiating security assessments when specific criteria are fulfilled. This automation enhances the efficiency and reliability of the supply chain security processes by reducing human error and ensuring consistent execution of security protocols.

The ability of smart contracts to operate autonomously, without the need for intermediaries, is a crucial feature for RESCALE's decentralized trust model. Once deployed, smart contracts can independently manage complex logic and execute transactions based on their programmed rules. This autonomy removes the need for trusted third parties to oversee contract execution, reducing potential points of failure and eliminating bottlenecks in the supply chain security verification process. For RESCALE, this means that the verification and updating of TBOMs can occur automatically and independently, enhancing the overall efficiency and trustworthiness of the system.

Smart contracts are distributed across the nodes of a blockchain network, ensuring that there is no single point of failure or control. This decentralization aligns perfectly with RESCALE's goal of creating a distributed and resilient trust storage architecture. By leveraging blockchain technology, smart contracts managing TBOMs are replicated across multiple nodes, enhancing data availability and resistance to tampering or censorship. This distributed nature ensures that the integrity and availability of TBOM data are maintained even if individual nodes in the network fail or are compromised.

Once deployed on the blockchain, smart contracts cannot be altered. This immutability is crucial for maintaining the integrity and trustworthiness of the RESCALE system. It ensures that the rules governing TBOM management and security assessments remain consistent and tamper-proof throughout the lifecycle of the components. Any updates or changes to the smart contract logic would require deploying a new contract, providing a clear audit trail of modifications. This immutability feature strengthens the reliability of the TBOM process, as stakeholders can trust that the rules governing their interactions with the system will not be arbitrarily changed.

Smart contracts offer full transparency, with their code and execution visible to all participants on the blockchain network. This transparency is vital for building trust in the RESCALE ecosystem. It allows all stakeholders to verify the logic governing TBOM management and security assessments, ensuring that the system operates as intended. Transparency also facilitates auditing and compliance checks, as the entire history of contract executions and state changes is recorded on the blockchain. For RESCALE, this means that the processes of creating, updating, and verifying TBOMs are fully auditable, enhancing accountability and trust among supply chain participants.

4.1.2 Benefits of Smart Contracts for TBOM Management

Smart contracts serve as a secure register for current TBOM hashes, verifying sender authorization during execution to prevent unauthorized modifications. They maintain an immutable history of changes through event triggers, enabling traceability and auditability of TBOM life-cycles. This approach ensures automated, secure management of TBOMs, reducing tampering risks and providing a reliable record of modifications. By combining hash registration, sender verification, and historical tracking, smart contracts enhance the integrity and transparency of TBOM management throughout the supply chain.

The blockchain-based TBOM system provides robust cryptographic protection for all TBOM data, ensuring its integrity and confidentiality. Every operation on a TBOM is recorded in a tamper-evident manner, creating an unalterable history of changes that can be audited at any time. The decentralized nature of blockchain storage significantly reduces single points of failure, enhancing the overall resilience of the TBOM system. An immutable audit trail of all TBOM-related activities (such as generation and deprecation) is maintained, providing a comprehensive and trustworthy record of every interaction, modification, and verification process. This level of security and immutability instills confidence in the TBOM data, crucial for making informed decisions in supply chain management.

The RESCALE TBOM system offers real-time visibility into the status and history of each TBOM, allowing stakeholders to track changes and updates as they occur. A traceable chain of custody is established for all components, from raw materials to finished products, enabling precise tracking of each item's journey through the supply chain. Every interaction between stakeholders and TBOMs is transparently recorded, creating a comprehensive log of who accessed, modified, or verified TBOM data. This transparency extends to the easy verification of component origins and modifications, allowing quick identification of potential security risks or compliance issues. The increased transparency and traceability foster trust among supply chain participants and facilitate rapid response to any identified issues.

By reducing the need for manual intervention in TBOM management, the RESCALE system significantly cuts operational costs and improves overall efficiency. The automated processes enable faster processing and validation of TBOMs, accelerating supply chain operations and reducing time-to-market for new products. The system minimizes errors and discrepancies in TBOM data through automated checks and validations, ensuring high data quality and reducing the costs associated with error correction. Auditing and compliance processes are streamlined, with automated reporting and real-time access to TBOM data, reducing the time and resources required for these critical activities. These efficiency improvements not only reduce direct costs but also enhance the overall competitiveness of organizations adopting the RESCALE TBOM system.

The RESCALE TBOM system facilitates direct peer-to-peer interactions between supply chain participants, reducing reliance on centralized authorities for TBOM verification. This disintermediation of trust replaces traditional third-party guarantors with smart contract code, ensuring that agreed-upon rules are enforced consistently and impartially. The reduced dependence on intermediaries increases the autonomy of supply chain stakeholders, allowing for more direct control over their operations and data. By removing intermediaries, the system reduces transaction costs, speeds up processes, and minimizes the risk of information manipulation or bottlenecks caused by centralized authorities. This direct interaction model fosters a more

collaborative and efficient supply chain ecosystem.

The blockchain-based TBOM system enables instant propagation of TBOM changes across the network, ensuring that all participants have access to the most current information. This real-time synchronization provides a unified view of TBOM data for all authorized parties, eliminating discrepancies and reducing the risk of decisions based on outdated information. The system generates real-time alerts and notifications for critical TBOM events, such as detected vulnerabilities or compliance issues, allowing for rapid response to potential risks. Continuous reconciliation of TBOM states across the supply chain ensures data consistency and integrity, even in complex, multi-tiered supply networks. This real-time capability enhances decision-making, risk management, and overall supply chain agility.

4.2 Smart Contract Implementation

This section delves into the implementation of smart contracts within the RESCALE project. Section 4.2.1 introduces the `HashStorage.sol` contract, a basic implementation designed for single hash storage. Section 4.2.2 presents the more advanced `HashManager.sol` contract, which offers a scalable solution for managing multiple hash values. Section 4.2.3 provides a comparative analysis of both implementations, discussing their strengths, limitations, and associated costs.

4.2.1 Basic Implementation: `HashStorage.sol`

The `HashStorage.sol` smart contract (Listing 1) represents a fundamental approach to storing and managing hash values. This implementation is designed to provide a straightforward and efficient solution for single hash storage, catering to basic use cases in supply chain management and digital document verification.

The contract utilizes a single storage variable, implemented as a struct named *HashInfo*, to hold the hash value and its associated owner address. This approach ensures a one-to-one mapping between an account address and a hash, providing a minimalistic yet effective method for hash storage on the blockchain. The use of a struct allows for easy expansion of stored data if future requirements necessitate additional fields.

The `HashStorage` contract incorporates several key features that make it suitable for basic hash management scenarios. Primarily, it stores a single hash value using the `bytes32` data type, which is optimal for representing cryptographic hashes such as SHA-256. This choice ensures efficient storage and gas usage. The contract associates each hash with the public address of the account that created it, leveraging the `msg.sender` variable to automatically link the hash to the transaction initiator. This association enables straightforward access control, preventing unauthorized modifications to the stored hash.

The simplicity of the `HashStorage` contract is one of its strongest attributes. With minimal code complexity, it reduces the potential for errors and simplifies the auditing process. This straightforward design also contributes to lower gas costs for deployment and execution compared to more intricate implementations, making it an economical choice for projects with basic hash storage needs.

Code Structure and Main Functions

The HashStorage contract (Listing 1) begins with the contract declaration and includes state variables for storing the hash information and a boolean flag to track whether a hash has been set. The contract implements four main functions that provide complete CRUD (Create, Read, Update, Delete) functionality:

- `setHash(bytes32 _hash)`: Allows users to store a new hash, creating the initial entry.
- `getHash()`: Retrieves the stored hash and its owner's address, providing read access to the data.
- `deprecateHash()`: Permits the owner to deprecate the stored hash, offering complete data lifecycle management.

These functions are complemented by events (`HashSet`, `HashDeprecated`) that log state changes, enhancing transparency and facilitating off-chain tracking of contract interactions.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HashStorage {

    // Struct to hold the hash info
    struct HashInfo {
        bytes32 hashValue; // Stored hash
        address owner; // Owner of the hash
    }

    // State variables
    HashInfo private storedHash;
    bool private isHashSet; // Flag to track if a hash is already set

    // Event for hash changes
    event HashSet(bytes32 indexed hashValue, address indexed owner);
    event HashDeprecated(bytes32 indexed oldHashValue, address indexed owner);

    // Set a new hash
    function setHash(bytes32 _hash) external {
        require(_hash != bytes32(0), "Invalid hash");
        require(!isHashSet, "Hash already exists. Use updateHash to modify.");
        storedHash = HashInfo({ hashValue: _hash, owner: msg.sender });
        isHashSet = true;
        emit HashSet(_hash, msg.sender);
    }

    // Get the stored hash
    function getHash() external view returns (bytes32 hashValue, address owner) {
        require(isHashSet, "No hash stored.");
        return (storedHash.hashValue, storedHash.owner);
    }

    // Delete the stored hash
```

```
function deprecateHash() external {
    require(isHashSet, "No hash stored to deprecate.");
    require(storedHash.owner == msg.sender, "Caller is not the
        owner");

    bytes32 oldHash = storedHash.hashValue;
    delete storedHash;
    isHashSet = false;

    emit HashDeprecated(oldHash, msg.sender);
}
}
```

Listing 1: HashStorage.sol

4.2.2 Advanced Implementation: HashManager.sol

The HashManager.sol smart contract (Listing 2) represents a more sophisticated approach to managing multiple hash values. This implementation is designed to provide a scalable and efficient solution for storing and managing multiple hashes, catering to complex use cases in supply chain management and digital document verification within the TBOM framework.

The contract utilizes a combination of a mapping and an array data structure to efficiently store and manage multiple hash values. The primary storage mechanism is a mapping (hashMapping) that associates each hash value (bytes32) with a HashInfo struct containing the hash's index in the array and the owner's address. This approach allows for constant-time lookups of hash information. Additionally, a dynamic array (hashList) stores all hash values, enabling enumeration and efficient updates. This dual-structure approach ensures optimal gas usage for various operations while maintaining the ability to track all stored hashes.

Code Structure and Main Functions

The HashManager contract (Listing 2) begins with the contract declaration and includes a struct definition for HashInfo, followed by state variables for the mapping and array. The three main functions form the core of the contract's functionality, implementing an adapted version of CRUD for blockchain where deletion is not possible:

- `add(bytes32 hash)`: Adds a new hash to the system, ensuring uniqueness and valid input.
- `read(bytes32 hash)`: Retrieves the index and owner of a given hash.
- `deprecate(bytes32 hash)`: Marks a hash as deprecated, effectively removing it from active use while maintaining historical data integrity.

Each function includes necessary checks to ensure data integrity and proper access control. The update and delete operations are particularly noteworthy for their gas-efficient implementations, which maintain the integrity of the hashList array while minimizing storage operations.

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HashManager {

    struct HashInfo {
        uint256 index; // Index position in the 'hashList' array
        address owner; // Owner of the hash
    }

    mapping(bytes32 => HashInfo) private hashMapping; // Mapping hash
    value -> HashInfo
    bytes32[] private hashList; // store all hash values

    // Events for CRD operations
    event HashAdded(bytes32 indexed hashValue, address indexed owner);
    event HashDeprecated(bytes32 indexed hashValue, address indexed owner);

    function add(bytes32 _hash) external {
        require(_hash != bytes32(0), "Invalid hash");
        require(hashMapping[_hash].owner == address(0), "Hash already
            exists");
        hashMapping[_hash] = HashInfo({ index: hashList.length, owner:
            msg.sender });
        hashList.push(_hash);
        emit HashAdded(_hash, msg.sender);
    }

    function read(bytes32 _hash) external view returns (uint256 index,
        address owner) {
        require(hashMapping[_hash].owner != address(0), "Hash does not
            exist");
        HashInfo memory hashInfo = hashMapping[_hash];
        return (hashInfo.index, hashInfo.owner);
    }

    function deprecate(bytes32 _hash) external {
        require(hashMapping[_hash].owner != address(0), "Hash does not
            exist");
        require(hashMapping[_hash].owner == msg.sender, "Caller is not the
            owner");
        uint256 index = hashMapping[_hash].index;
        uint256 lastIndex = hashList.length - 1;
        if (index != lastIndex) {
            bytes32 lastHash = hashList[lastIndex];
            hashList[index] = lastHash;
            hashMapping[lastHash].index = index;
        }
        hashList.pop();
        delete hashMapping[_hash];
        emit HashDeprecated(_hash, msg.sender);
    }
}

```

Listing 2: HashManager.sol

4.2.3 Comparative Analysis of Implementations

This section provides a comprehensive comparison between the `HashStorage.sol` (Listing 1) and the `HashManager.sol` (Listing 2) implementations, highlighting their respective strengths, limitations, and associated costs.

`HashStorage.sol` offers simplicity and efficiency for basic use cases. Its straightforward design makes it easy to understand, implement, and audit, reducing the potential for errors and vulnerabilities. The contract's minimal functionality results in lower gas fees for deployment and basic operations. For simple proof-of-existence scenarios or single-user applications, `HashStorage.sol` provides a clean and uncomplicated solution with comparable efficiency to more complex implementations.

However, `HashStorage.sol` simplicity comes with significant limitations. The contract only supports storing a single hash, which may be insufficient for more complex use cases. Its lack of scalability is a major drawback, as each new hash requires a new contract deployment. This approach becomes impractical and costly when managing multiple hashes, especially in large-scale operations.

`HashManager.sol` offers a more robust and flexible solution for complex supply chain scenarios. Its ability to store multiple hashes makes it suitable for managing numerous components that need to be tracked. The contract's scalability is a significant advantage, as it can handle an increasing number of hashes without requiring new deployments. This makes it more efficient for large-scale operations and complex supply chain management systems.

`HashManager.sol` maintains an array of all hashes, allowing for easier enumeration and improved traceability. Despite its higher complexity, `HashManager.sol` optimizes gas usage for operations like updates and deletions, which is particularly beneficial when managing many hashes.

However, the advanced features of `HashManager.sol` come with trade-offs. The more sophisticated design may be harder to understand and audit, potentially increasing the risk of bugs or vulnerabilities. The initial deployment of `HashManager.sol` is more expensive due to its increased functionality and storage structures. While the difference in basic operation costs is minimal, setting a hash in `HashManager.sol` is marginally more expensive than in `HashStorage.sol`.

To illustrate the cost differences, the following table presents gas costs and their equivalent prices in Euros, based on values updated as of October 2024 (ETH price of €2381 and average gas cost of 20.85gwei):

Operation	HashStorage		HashManager	
	Gas	Cost (€)	Gas	Cost (€)
Deploy	757,351	37.60	1,132,103	56.20
Set Hash	90,216	4.48	92,750	4.60
Update Hash	33,305	1.65	59,754	2.97
Delete Hash	30,578	1.52	33,506	1.66

Table 3: Gas costs and prices for `HashStorage.sol` and `HashManager.sol` operations

As shown in Table 3, while `HashManager.sol` has a higher initial deployment cost, the differ-

ence in operational costs is relatively small. The choice between these implementations should be based on the specific requirements of the TBOM management system, considering factors such as scalability needs, gas cost constraints, and the complexity of the supply chain being modeled.

Leveraging Transaction Data for Cost-Effective Storage

While smart contracts provide robust functionality for managing TBOMs, storing large amounts of data on-chain can be expensive. For frequent operations or storing small pieces of data, leveraging transaction data can be a more cost-effective solution. This approach involves encoding data within the transaction itself, rather than storing it in contract storage.

```
pragma solidity ^0.8.0;

...
event DataAssociated(bytes32 indexed tbomHash, bytes32 indexed
    corollaryHash);

function associateData(bytes32 _tbomHash, bytes32 _corollaryHash)
    external {
    emit DataAssociated(_tbomHash, _corollaryHash);
}

function getAssociatedData(bytes32 _tbomHash) external view
    returns (bytes32[] memory) {
    // This function would be implemented off-chain by querying
    // event logs
    // It's included here for completeness, but would not be part
    // of the actual contract
}

...
```

Listing 3: TBOMDataAssociation Contract

Listing 3 shows an example of how to use transaction data for storage in Solidity. Rather than storing associations directly in contract storage, the function emits an event to associate a TBOM hash and a corollary hash.

While leveraging transaction data to store simple data offers a cost-effective solution, it is essential to consider the potential drawbacks compared to using smart contract fields for data storage. Storing data through transaction events, such as emitting logs, can significantly reduce gas costs, especially for frequent operations. However, this approach introduces several challenges that must be carefully evaluated.

One of the primary concerns is the difficulty of querying and accessing the data stored in transaction logs. Unlike direct access to smart contract fields, which allows for straightforward retrieval of values through function calls, accessing event logs requires off-chain processing. This means that developers must implement additional mechanisms to listen for events and parse the log data, adding complexity to the system architecture. For instance, while a smart contract can return associated corollary hashes directly through a function call, retrieving this information from logs necessitates a more cumbersome process involving event subscriptions and filtering.

Moreover, transaction logs are not inherently structured for complex queries or relationships.

When using smart contract storage, developers can utilize mappings and arrays to organize data efficiently. In contrast, event logs typically require additional logic to reconstruct relationships between different pieces of data. For example, if multiple corollary hashes are associated with a single TBOM hash, reconstructing this association from logs may require iterating through numerous events instead of simply accessing a mapping.

Another limitation of using transaction data is the lack of built-in data integrity checks that smart contracts provide. Smart contracts can enforce rules and validations on stored data through function modifiers and access controls. In contrast, when relying on event emissions for data storage, there is no inherent mechanism to ensure that the emitted data is valid or consistent with the rest of the system's state. This could lead to scenarios where inconsistencies arise between the state represented in logs and the actual state of the smart contract.

Additionally, while emitting events can be cheaper in terms of gas costs for frequent operations, it may not be suitable for all use cases. For example, if an application requires real-time access to associated metadata or analysis results tied to TBOM hashes, relying solely on transaction logs could introduce latency and inefficiencies in retrieving necessary information.

5 Rescale Trusted Storage Architecture and Implementation

5.1 Architecture Overview

The Rescale Trusted Storage Architecture is designed as a distributed system for managing TBOM in supply chain security. The architecture is organized around a network of nodes, each serving as an atomic unit of the system. Each node in the network consists of two primary components: a Hyperledger BESU node and an IPFS node. The BESU node provides blockchain functionality, ensuring immutability and distributed consensus for the stored data, while the IPFS node offers decentralized storage capabilities, allowing for efficient handling of larger data sets.

The BESU RPC API is exposed directly, allowing authenticated and authorized access to the blockchain network. IPFS, however, is accessed through a REST service, which acts as a controlled interface. This architecture ensures that clients cannot unpin documents from remote IPFS nodes without proper authorization, maintaining the integrity of stored data. The network model of the Rescale Trusted Storage is designed as a public permissioned network. The public aspect allows user interaction with the network, promoting transparency, while the permissioned aspect ensures that only a restricted group of nodes can perform operations on the network, maintaining control and accountability. This balance between accessibility and security is crucial for the system's integrity and usability.

Regarding scalability and partner integration, the architecture is designed with a partner-centric approach. The system is structured to create one node per partner, allowing for clear delineation of responsibilities and data ownership. This design facilitates the onboarding of new partners and management of partner-specific data and operations, enhancing the system's adaptability to diverse supply chain ecosystems.

5.1.1 Confidentiality, Integrity and Availability

The ledger component serves as the foundation for ensuring data integrity within the TBOM ecosystem. By leveraging the immutable nature of blockchain technology, each entry in the TBOM is recorded in a tamper-evident manner. This immutability is achieved through cryptographic hashing, which links each block to its predecessor, creating a chain of information that is extremely difficult to alter without detection. The consensus mechanisms inherent to blockchain systems play a crucial role in maintaining the integrity of TBOM data. These mechanisms ensure that all nodes in the network agree on the state of the ledger, validating new entries before they are added to the blockchain. This distributed validation process significantly reduces the risk of unauthorized modifications to TBOM records, as any attempt to alter data would require consensus from the majority of the network.

The IPFS component ensures data availability within the TBOM ecosystem. By leveraging its distributed nature, IPFS provides a robust and resilient storage solution for TBOM data. Each piece of information is stored across multiple nodes in the network, significantly reducing the risk of data loss due to node failures or network disruptions. The content-addressable nature of

IPFS also enhances data integrity, as each file is uniquely identified by its content hash, making it easy to verify the authenticity of retrieved data.

The client library and rest service plays a crucial role by providing a simplified interface for interacting with nodes in the system. Its primary purpose is to abstract the complexities of node communication and TBOM operations, offering developers and users an easy-to-use set of tools for interacting with the Trust Module. Currently, the library focuses on ensuring confidentiality within the CIA triad through message encryption, protecting sensitive information during transmission between clients and nodes. Key features include abstraction of node communication, payload preparation, authentication handling, operation abstraction, and error handling. The library manages underlying network protocols and security mechanisms, prepares data for TBOM operations, incorporates authentication mechanisms, simplifies common operations, and provides robust error handling. By implementing end-to-end encryption, it significantly enhances the security of the TBOM system, particularly for data in transit. Future plans for the client library include expanding its capabilities with additional security features, such as ZKP technologies, which would enable selective disclosure of information without revealing all associated data.

5.1.2 The networking system

The Rescale Trusted Storage Architecture presented in Figure 3 incorporates three distinct networks, each serving a specific function in the system's operation: the P2P Discoverability Network, the RPC API Exposure Network, and the Trust Storage Network.

The P2P Discoverability Network facilitates node discovery and synchronization for both BESU and IPFS nodes. It utilizes peer-to-peer protocols, specifically the Ethereum DevP2P protocol for BESU and the InterPlanetary networking stack for IPFS. This network is exposed to the public internet to enable dynamic node discovery and synchronization, employing various mechanisms such as DNS-based discovery and distributed hash tables for efficient node location and connection.

In contrast, the RPC API Exposure Network provides a standardized interface for interacting with BESU and IPFS nodes. This limitation minimizes the potential attack surface by ensuring controlled interaction with the nodes' APIs. The RPC network supports methods for querying blockchain data, submitting transactions, interacting with smart contracts (for BESU), and managing and retrieving content (for IPFS).

Serving as the primary operational network, the Trust Storage Network is designed as a public permissioned network, striking a balance between accessibility and security. The public aspect allows for transparency and user interaction, while the permissioned nature ensures that only authorized nodes can perform operations on the network. Within this network, the REST service functions as a controlled gateway between users and the Trust Storage nodes, abstracting the complexities of direct node interaction and providing a secure, standardized interface for authenticated and authorized access to the system's functionalities.

5.1.3 Certification Authority Integration

The integration of a Certification Authority (CA) within the Trust Storage Module is critical for ensuring the security, integrity, and legal compliance of digital transactions within the TBOM ecosystem. A reputable Certification Authority (CA) provides essential services for authentication, identity verification, and the addition of legal properties to the key verification process. By acting as a legally recognized entity, the CA strengthens the trust framework by ensuring that all cryptographic operations are conducted in accordance with established legal and security standards.

In the RESCALE Trust Storage Module, the CA is treated as an external black box. This design decision reflects the requirement for the CA to be a legally recognized entity that operates independently of the Trust Storage Module. Leveraging an external CA ensures that identification and legal properties are seamlessly integrated into the key verification process without compromising neutrality or compliance.

Diagram 1 illustrates an example of the process to issue a new certificate. The certificate issuance process begins with the Entity generating a public-private key pair, followed by the creation of a Certificate Signing Request (CSR) using the generated public key. The Entity then submits this CSR to the CA through the `request_certificate()` function, which returns a request ID. The CA conducts necessary verifications to ensure the binding between the key and the entity is valid (represented with `verify_key_entity_binding()`). Upon approval, the Entity retrieves the issued certificate using the `download_certificate()` function. The process concludes with the Entity securely storing both the issued certificate and its corresponding private key.

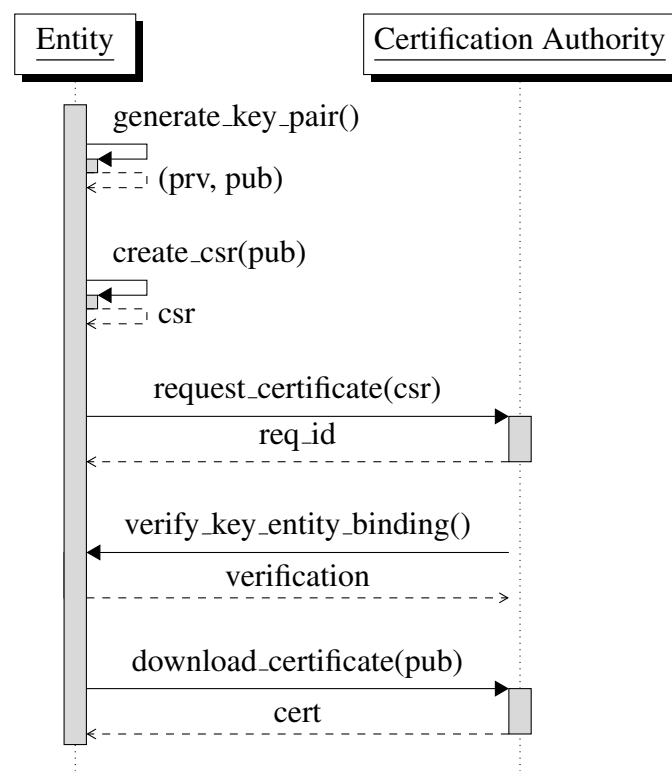


Figure 1: Certificate Issuance Flow

Diagram 2 shows an example flow for a certificate verification flow. The flow begins when

an Entity submits a document along with its associated certificate to the Trust Storage Module using the `send_document(doc, cert)` function. Upon receipt, the Trust Storage Module initiates the verification process by calling the `verify_certificate(cert)` function of the CA API. The CA then performs a series of checks on the received certificate, which typically include verifying the certificate's digital signature, checking for revocation, and ensuring it's within its validity period. Once these checks are complete, the CA returns the verification result to the Trust Storage Module. The module then processes this result using its `handle_result()` function, which allows it to make informed decisions about how to proceed with the authenticated document and entity. Finally, the Trust Storage Module returns the overall result of the document submission and verification process to the original Entity. This comprehensive flow ensures that each document submitted to the Trust Storage Module undergoes thorough certificate verification, thereby maintaining the integrity and security of the TBOM ecosystem.

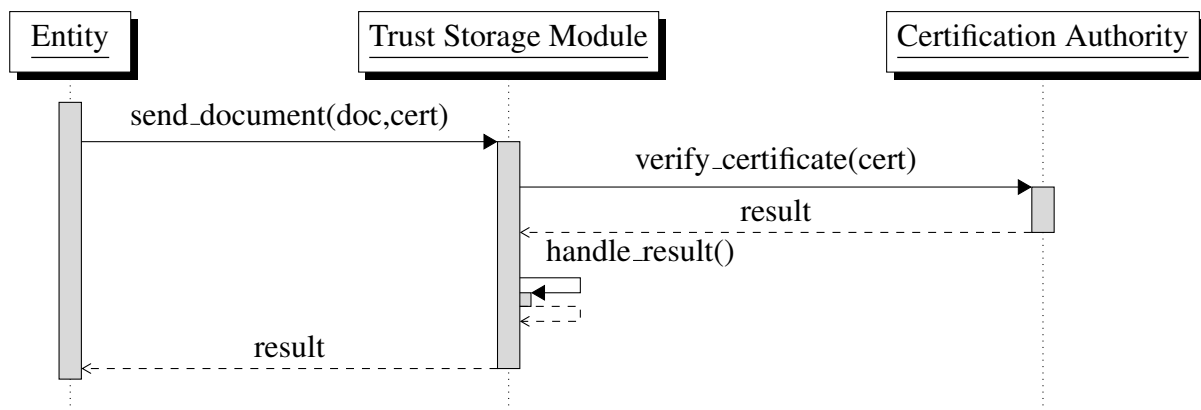


Figure 2: Certificate Verification Flow

5.1.4 Key Management System

The RESCALE Trust Storage Module relies on a Key Management System (KMS) to securely store and retrieve key material. The KMS provides essential functionalities that support the Trust Storage Module and other RESCALE modules, ensuring that cryptographic keys are managed efficiently throughout their lifecycle.

The KMS is an external component to the Trust Storage Module, designed to be shared among multiple RESCALE modules that require key management solutions for their internal functioning. It provides centralized control over key generation, distribution, storage, rotation, and destruction, ensuring consistent and secure handling of cryptographic keys across the RESCALE ecosystem. By implementing best practices in key management, the KMS helps reduce overall security risks, including preventing unauthorized access and misuse of keys, minimizing the impact of compromised keys through regular rotation, and protecting against key loss with secure backup strategies.

Importantly, the KMS is treated as a black box service provided by external software solutions, which may include open-source options. It is essential to note that the KMS only manages keys necessary for the internal operations of RESCALE modules and does not handle user keys; users are responsible for managing their own keys.

The development and deployment of a KMS will be proposed in the final version of this deliverable.

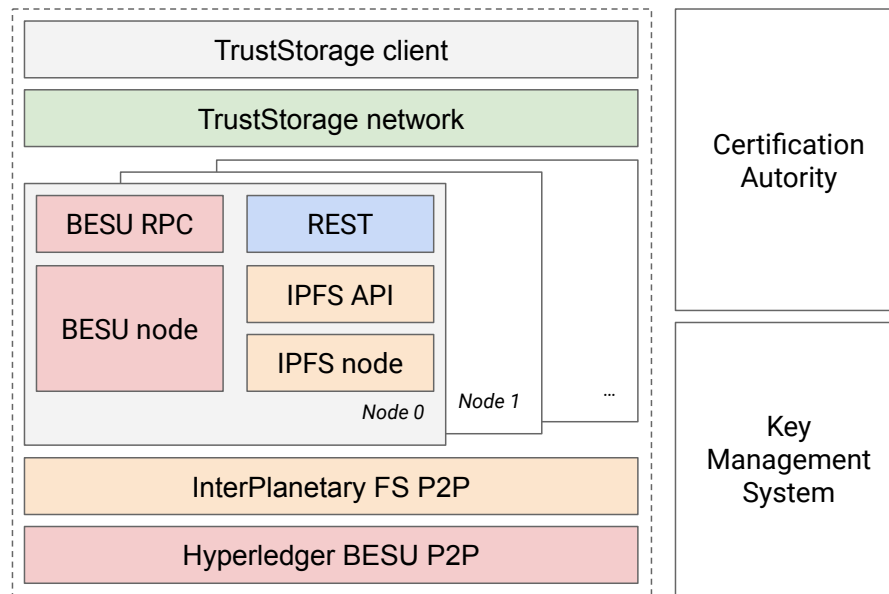


Figure 3: Trust Storage module structure.

erable, aiming to establish a comprehensive framework for key management that aligns with best practices in the industry.

5.2 Implementation strategy

This section explores the implementation strategy for the Trust Storage module within the RESCALE project. Section 5.2.1 details the Trust Storage Node, covering the compilation process for Solidity contracts, QBFT configuration, and the initialization of Hyperledger BESU and IPFS nodes. It also outlines the REST service overview and the Dockerization approach for the entire setup. Section 5.2.2 focuses on the Trust Storage Client, describing the TrustStorageClient class and its key features for secure interaction with the Trust Storage network, including methods for document management and utility functions for key handling.

5.2.1 Trust Storage Node

Solidity Contract Compilation

The Solidity contract compilation process is automated through a JavaScript script utilizing Node.js. This script streamlines the generation of Application Binary Interface (ABI) and byte-code for Ethereum smart contracts. At its core, the script performs file parsing, compilation, and output generation in a seamless workflow. The central compilation logic is encapsulated in the compile function:

```
function compile(sourceCode, contractName) {
  const input = {
    language: "Solidity",
    sources: {main: {content: sourceCode}},
  }
```

```
    settings: {outputSelection: {"*": {"*": ["abi", "evm.bytecode"]}}},
  };
  const output = solc.compile(JSON.stringify(input));
  const artifact = JSON.parse(output).contracts.main[contractName];
  return { abi: artifact.abi, bytecode: artifact.evm.bytecode.object };
}
```

This function configures the compiler input, executes the compilation, and extracts the necessary artifacts. Following compilation, the script extracts the ABI and bytecode from the compiler output.

QBFT Configuration File

The QBFT configuration file, typically named `qbftConfigFile.json`, defines the parameters for initializing a QBFT-based Ethereum network. This JSON file contains two main sections: `genesis` and `blockchain`.

The `genesis` section specifies the initial state of the blockchain, including a `config` subsection that defines network parameters:

```
"config": {
  "chainId": 1337, // Identifies a private network
  "contractSizeLimit": 2147483647, // Maximum contract size in bytes
  // ... Ethereum upgrades blocks, all set to 0 to enable from genesis..
  "zeroBaseFee": true, // Set to true for a free gas network
  "qbft": {
    "blockperiodseconds": 15, // Time between blocks
    "epochlength": 30000, // Number of blocks in an epoch
    "requesttimeoutseconds": 30 // Timeout for block proposals
  }
}
```

The `genesis` section also includes blockchain initialization parameters:

```
"nonce": "0x0000000000000000", // Used in PoW, set to zero for QBFT
"timestamp": "0x0", // Genesis creation set at 0 for a new network
"gasLimit": "0x1ffffffffffffff", // Max gas per block for unlimited tx
"difficulty": "0", // Set to 0 for QBFT networks
"mixHash": "0x00..", // Used in PoW, set to 32-bytes zero for QBFT
"coinbase": "0x00..", // Address receiving rewards (irrelevant in QBFT)
"extraData": "", // Additional data (will be used for validators)
"alloc": {} // Pre-allocated memory (will be used for contracts)
```

Several parameters are specifically set to create a free gas network. The `zeroBaseFee` eliminates the base fee for transactions, the high `gasLimit` allows unlimited gas usage per block, and the `difficulty` set to 0 removes mining difficulty.

The `"blockchain"` section instead defines node generation:

```
"blockchain": {
  "nodes": {
    "generate": true, // Automatically generate nodes
    "count": 1 // Number of nodes to generate
  }
}
```

Hyperledger BESU Node Initialization

Following the creation of the QBFT configuration file, a bash script is used to initialize the BESU node at first startup. This script automates several steps in setting up the node and, if necessary, bootstrapping a new network.

```
openssl ecparam -name $BESU_ECCURVE -genkey -noout -out "key.pem"
besu public-key export --node-private-key-file=key.prv --to=key.pub
besu public-key export-address --node-private-key-file=key.prv --to=besu.address
```

If no private key exists, the script generates a new key pair and address using OpenSSL and BESU commands.

```
# Genesis.json generation
address_list+=("${cat "besu.address"}")
# ... add address_list as validators in extradata for QBFT ...

enode="enode://$pubkey@$ip_address:$BESU_P2P_PORT"
# ... add node as bootnode to genesis.json ...

# pre-deploy contracts
for contract_file in contracts/*.json; do
  # ... read bytecode and add to genesis alloc ...
done
```

If genesis.json is not provided, the script generates one, making this node the first validator and bootnode of a new network. It also pre-deploys any compiled contracts found in the contracts directory, utilizing the bytecode from the Solidity compiler output.

```
besu \
  --data-path=$BESU_PATH \
  --genesis-file="$BESU_PATH/genesis.json" \
  --node-private-key-file="$BESU_PATH/key.prv" \
  --min-gas-price=0 \
  --p2p-host=$ip_address \
  --p2p-port=$BESU_P2P_PORT \
  --rpc-http-enabled \
  --rpc-http-host=$ip_address \
  --rpc-http-port=$BESU_RPC_PORT \
```

```
--rpc-http-api=ETH,NET,QBFT \  
--host-allowlist=* \  
--rpc-http-cors-origins=\"all\" \  
--bootnodes=$bootnodes
```

Finally, the script starts the BESU node with specific configurations:

- `--data-path`: Specifies where blockchain data is stored
- `--genesis-file`: Points to the custom genesis file
- `--node-private-key-file`: Uses the generated private key
- `--min-gas-price=0`: Enables free transactions
- `--p2p-host` and `--p2p-port`: Configure peer-to-peer networking
- `--rpc-http-enabled` and related flags: Enable and configure the HTTP RPC API
- `--rpc-http-api`: Specifies which APIs to enable (Ethereum, Network, QBFT)
- `--host-allowlist=*` and `--rpc-http-cors-origins=all`: Allow all connections (suitable for development, not production)
- `--bootnodes`: Specifies initial peers for network discovery

InterPlanetary File System Node Initialization

Following the BESU node setup, a bash script is used to initialize an IPFS node specifically configured for a private network. This script automates steps to ensure the node operates within a controlled, private IPFS environment.

```
ipfs init --profile server
```

The node is initialized with the `--profile server` flag, optimizing it for server environments. This profile adjusts various IPFS configurations for improved performance in a server setting.

```
...  
{  
    echo "/key/swarm/psk/1.0.0/"  
    echo "/base16/"  
    echo "$(tr -dc 'a-f0-9' < /dev/urandom | head -c64)"  
} > "swarm.key"  
...
```

A 32-byte swarm key is either provided via an environment variable or generated using `/dev/urandom`. This key, formatted according to the PSK (Pre-Shared Key) specification, is propaedeutic for private network creation. Only nodes with identical swarm keys can communicate, effectively isolating the network.

```
ipfs bootstrap rm all
if [[ -n "$IPFS_BOOTSTRAP_IP" && -n "$IPFS_BOOTSTRAP_ID" ]]; then
    ipfs bootstrap add "/ip4/$IPFS_BOOTSTRAP_IP/tcp/4001/ipfs/$IPFS_BOOTSTRAP_ID"
fi
```

All default bootstrap nodes are removed (`ipfs bootstrap rm all`). If specified, a single custom bootstrap node is added. This process is essential for creating a closed, private IPFS network by controlling peer discovery and connections.

The script's behavior is controlled through three environment variables: `IPFS_SWARM_KEY`, `IPFS_BOOTSTRAP_IP`, and `IPFS_BOOTSTRAP_ID`. When these environment variables are set, the node can be configured to participate in an already existing private IPFS network. Conversely, if these variables are not set, the script adapts to create a new private network. In this case, a new swarm key will be generated, effectively starting a new private network. Additionally, the node will not have any bootstrap nodes configured, making it the first node of a new network.

REST Service overview The REST service is implemented in Rust using the `ntex` web framework. The service acts as a secure intermediary between clients and IPFS nodes, providing controlled access to the IPFS RPC API. This approach addresses the need to expose certain IPFS functionalities remotely while maintaining security and control over the node's operations.

The REST service exposes two primary operations:

- **Create:** Allows clients to pin new documents to the IPFS node
- **Cat:** Enables clients to retrieve documents from the IPFS node

These operations are crucial for remote interaction with IPFS nodes.

The service is built using Rust and the `ntex` web framework, which provides a robust and efficient foundation for handling HTTP requests. The main structure of the service is defined in the main function:

```
#[ntex::main]
async fn main() -> std::io::Result<()> {
    web::HttpServer::new(|| {
        web::App::new()
            .service(api::create::create)
            .service(api::cat::cat)
    })
    .bind(("0.0.0.0", 4040))?
    .run()
    .await
}
```

This setup initializes an HTTP server that listens on all network interfaces (0.0.0.0) on port 4040. It registers several services, including the create and cat operations.

Dockerization of the Trust Storage Node

To facilitate easy deployment and multi-platform distribution, the BESU node, IPFS node, and REST service are each containerized using separate Dockerfiles. This approach ensures consistency across different environments and simplifies the deployment process.

The Dockerfiles employ a multi-stage build process, which offers several advantages: reduced final image size by excluding build tools and intermediate artifacts; improved build performance through parallelization of independent stages; enhanced security by minimizing the attack surface in the final image; Simplified maintenance by separating build and runtime environments.

The Dockerfile for the BESU node begins with an Ubuntu 24.04 base image to compile BESU from source:

```
FROM ubuntu:24.04 AS builder
...
RUN git clone --depth 1 --branch 24.9.1 https://github.com/hyperledger/besu
RUN cd /besu && ./gradlew installDist
```

The final stage sets up the runtime environment:

```
FROM ubuntu:24.04 AS final
...
COPY --from=builder /besu/build/install/besu /opt/besu/
```

Similarly, the IPFS node is built using a dedicated Dockerfile, starting with Ubuntu 24.04 to compile IPFS from source:

```
FROM ubuntu:24.04 AS builder
...
RUN git clone --depth 1 --branch v0.30.0 https://github.com/ipfs/kubo.git
RUN cd /kubo && make build
```

The runtime environment is then set up in a separate stage:

```
FROM debian:latest AS final
...
COPY --from=builder /kubo/cmd/ipfs/ipfs ./
```

The REST service is compiled using a Dockerfile that employs the official Rust image:

```
FROM rust:1.82.0 AS rustcompiler
COPY ./rest/ /app
WORKDIR /app
RUN cargo build
```

The final stage creates the runtime environment:

```
FROM gcr.io/distroless/cc
...
COPY --from=rustcompiler /app/target/debug/gatekeeper /opt/rest/gatekeeper
```

Each Dockerfile configures its respective environment and exposes necessary services for its component. The containers are designed to expose multiple ports: 30303 and 4001 for BESU and IPFS P2P communications respectively, 8545 and 5001 for their RPC interfaces, and 4040 for the REST service.

The P2P ports (30303 and 4001), the BESU RPC port (8545) and the REST port (4040) are publicly accessible through port forwarding, enabling network discoverability and external use. In contrast, the IPFS RPC port (5001) is configured to be visible only to localhost, restricting access to local services within each container. This setup ensures that each node can participate in its respective network and offer REST services while maintaining the security of RPC interfaces.

5.2.2 Trust Storage Client

The Trust Storage Client is implemented as a set of Python modules that facilitate interaction with the Trust Storage system, ensuring compatibility with other Rescale components. This implementation provides secure communication and data management within the distributed storage system, utilizing JWT tokens for authentication and integrating with both IPFS and BESU blockchain.

Key Features

- **PEM-formatted Key Authentication:** Uses PEM-formatted key files for authentication, keeping sensitive key material on the client side.
- **JWT Token Security:** Implements JWT tokens for secure communication with the REST service.
- **Two-Step Document Addition:** Adds documents to IPFS and records their references in the BESU blockchain.
- **Secure Document Retrieval:** Verifies document existence in the blockchain before retrieving from IPFS.
- **Document Deprecation:** Allows logical invalidation of documents without altering IPFS content.

Environmental Variables

The Trust Storage Client relies on several environment variables for configuration and security:

- **REST_ENDPOINT:** The URL of the REST service for IPFS interactions.
- **BESU_ENDPOINT:** The URL of the BESU blockchain node.

- `CLIENT_PRVKEY_PATH`: File path to the client's private key in PEM format.
- `CLIENT_PUBKEY_PATH`: File path to the client's public key in PEM format.
- `CONTRACT_ADDR_PATH`: File path to the smart contract address.
- `CONTRACT_JSON_PATH`: File path to the JSON file containing the smart contract ABI.

If any of these variables are missing, the client will raise a `ValueError` with a message indicating which environment variable is not set.

Key Components

- `truststorageclient.py`: Main interface for document operations.
- `restclient.py`: Handles REST API interactions with IPFS.
- `besuclient.py`: Manages interactions with the BESU blockchain.
- `utils.py`: Provides utility functions for key management.

IPFS Interaction (`restclient.py`)

- `add_ipfs(document: dict) -> str`: Adds a document to IPFS, returning the CID.
- `get_ipfs(cid: str) -> str`: Retrieves a document from IPFS using its CID.
- Uses JWT for secure communication with the REST service.

BESU Blockchain Interaction (`besuclient.py`)

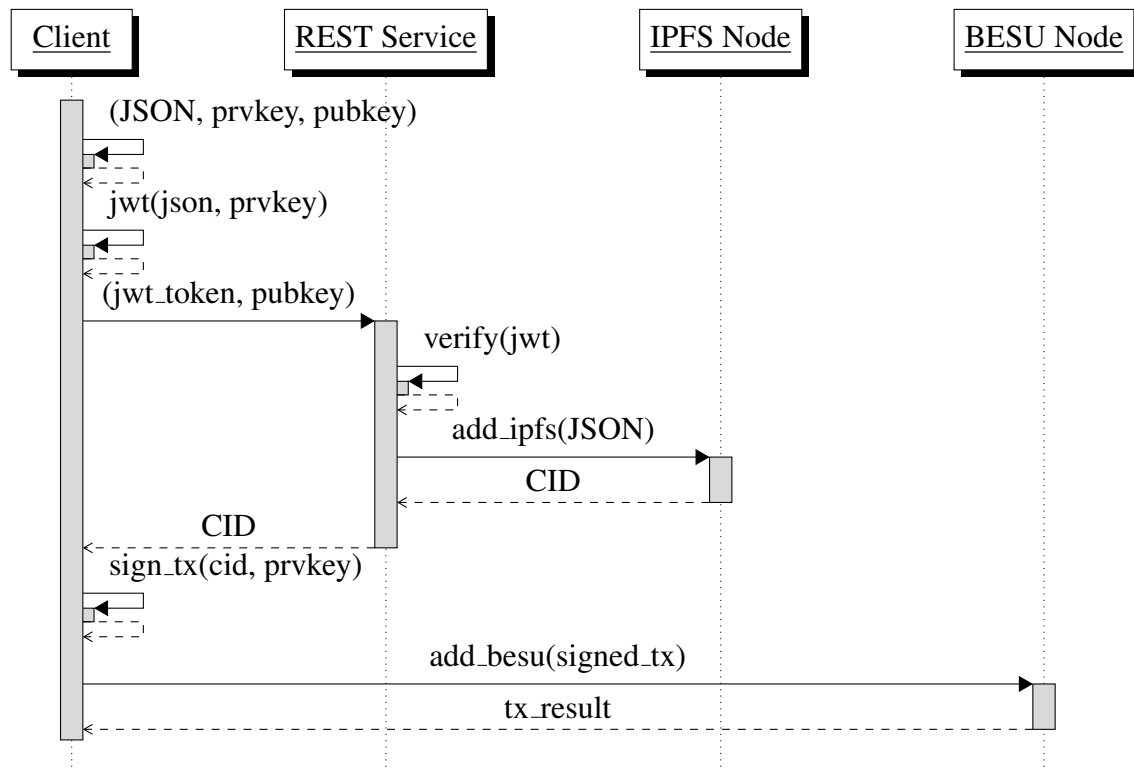
- `add_besu(cid: str) -> None`: Records a document's CID in the blockchain.
- `get_besu(cid: str) -> None`: Verifies a document's existence in the blockchain.
- `deprecate_besu(cid: str) -> None`: Marks a document as deprecated in the blockchain.
- Implements low-level blockchain interactions, including transaction management and smart contract calls.

Core Methods (`truststorageclient.py`)

The `truststorageclient` module provides three main methods for managing documents within the Trust Storage system. These methods encapsulate the core functionality of adding, retrieving, and deprecating documents.

- `add_document(document: dict) -> str` (Figure 4)
 - Adds a document to IPFS using `add_ipfs()`.
 - Records the IPFS CID in the BESU blockchain using `add_besu()`.

- Returns the CID of the added document.
- `get_document(cid: str) -> dict` (Figure 5)
 - Verifies the document's existence in BESU blockchain using `get_besu()`.
 - Retrieves the document from IPFS using `get_ipfs()`.
 - Returns the document as a dictionary.
- `deprecate_document(cid: str) -> None` (Figure 6)
 - Marks a document as deprecated in the BESU blockchain using `deprecate_besu()`.

Figure 4: Sequence diagram of `add_document()` operation.

Utility Functions (`utils.py`)

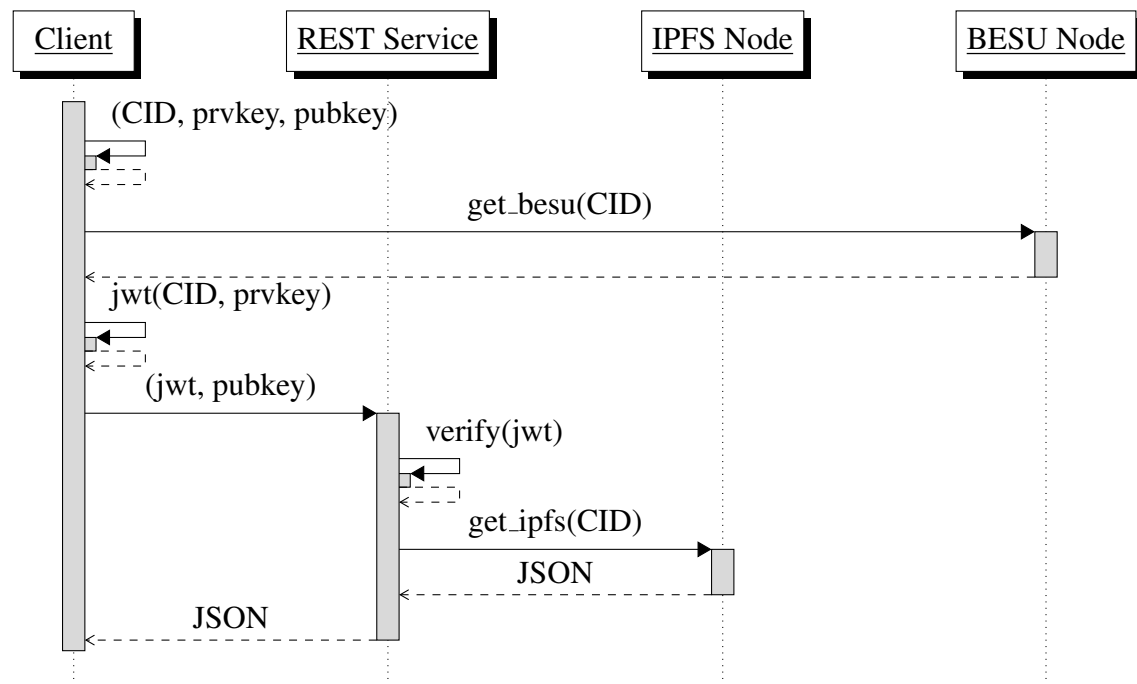
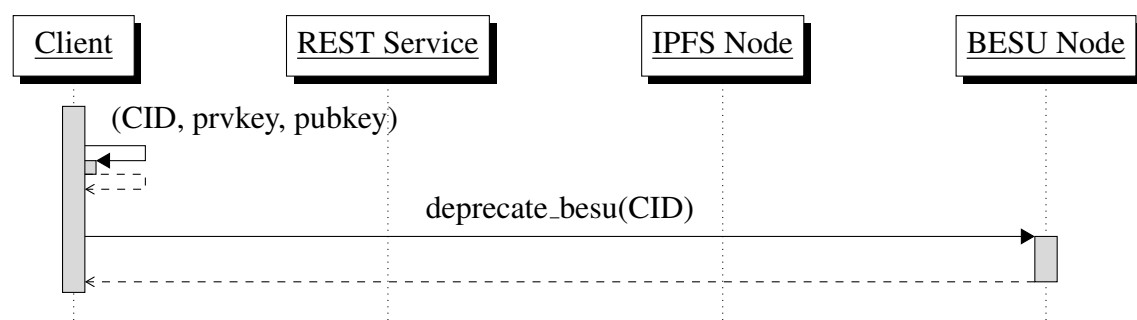
1. `generate_ec(folder=None, curve="secp256k1")`

- Generates an Elliptic Curve key pair (secp256k1 or secp256r1).
- Creates both private and public keys in PEM format.
- Optionally saves keys to PEM files in a specified folder.

text

2. `pem_to_hex(keypath: str)`

- Converts a PEM-formatted private key to the hexadecimal format required by BESU.
- Ensures the key is a secp256k1 curve key.

Figure 5: Sequence diagram of `get_document()` operation.Figure 6: Sequence diagram of `deprecate_document()` operation.

6 Potential Challenges and Risks

The implementation of the Trust Storage module within the RESCALE project, while offering significant benefits for supply chain security and transparency, also presents several challenges and potential risks. This section outlines these concerns and proposes mitigation strategies to address them effectively.

6.1 Scalability and Performance Concerns

As the TBOM system grows and evolves, maintaining high transaction throughput and low latency becomes increasingly challenging. The blockchain ledger and IPFS storage may face performance bottlenecks when dealing with large volumes of data or frequent updates, potentially impacting the system's responsiveness and efficiency.

To mitigate these scalability concerns, several strategies can be employed. Implementing sharding techniques can distribute data processing across multiple nodes, effectively balancing the load and improving overall system performance. Utilizing layer-2 scaling solutions, such as sidechains or state channels, can offload transactions from the main chain, reducing congestion and improving transaction speed. Additionally, optimizing smart contract code and data structures for efficiency can significantly enhance the system's performance. Employing caching mechanisms and content delivery networks can further improve data retrieval speeds, ensuring that the TBOM system remains responsive even as it scales.

6.2 Security Risks and Vulnerabilities

The decentralized nature of blockchain technology, while offering many security benefits, also introduces unique vulnerabilities. Smart contract vulnerabilities could potentially expose the system to attacks or unauthorized access, compromising the integrity of the TBOM data. Key management and access control present significant challenges, especially in a decentralized environment where traditional security measures may not be applicable. Furthermore, the system may be susceptible to various blockchain-specific attacks, such as 51% attacks or smart contract exploits, which could undermine the trust and security of the entire TBOM framework.

To address these security concerns, a multi-faceted approach is necessary. Conducting regular security audits and penetration testing of smart contracts and the overall system can help identify and address vulnerabilities proactively. Implementing multi-signature wallets and hardware security modules for key management can significantly enhance the security of cryptographic keys. Utilizing formal verification techniques to mathematically prove the correctness of smart contracts can minimize the risk of exploits. Additionally, implementing robust access control mechanisms and regularly updating security protocols can help maintain the system's resilience against evolving threats.

6.3 Legal and Regulatory Challenges

The implementation of a blockchain-based TBOM system raises several legal and regulatory concerns, particularly regarding data protection and cross-border data transfers. Compliance with data protection regulations like GDPR could be complex, especially concerning data immutability and the right to be forgotten, which are inherently at odds with blockchain's immutable nature. Cross-border data storage and transfer regulations may pose challenges for a globally distributed system, potentially limiting the TBOM's effectiveness in international supply chains. Moreover, the legal status and enforceability of smart contracts in different jurisdictions could be uncertain, potentially complicating dispute resolution and contract execution.

To navigate these legal and regulatory challenges, engaging legal experts specializing in blockchain and data protection is crucial to ensure compliance. Implementing privacy-preserving technologies like zero-knowledge proofs can help achieve GDPR compliance while maintaining the benefits of blockchain transparency. Designing the system with data localization options can address cross-border data transfer issues, allowing for compliance with various regional regulations. Developing clear terms of service and user agreements that address legal uncertainties can help establish a solid legal foundation for the TBOM system's operation.

6.4 Integration Complexities

Integrating the Trust Storage module with existing supply chain systems and processes may present significant technical challenges. Ensuring interoperability between different blockchain networks and external systems could introduce complexities that may hinder seamless adoption and utilization of the TBOM system.

To address these integration challenges, developing standardized APIs and integration protocols for seamless connection with existing systems is essential. The choice of Hyperledger BESU as the blockchain platform for the TBOM system offers significant advantages in terms of interoperability. BESU's EVM compatibility enhances the system's flexibility and compatibility with the broader Ethereum ecosystem and EVM-based blockchain networks. This compatibility allows for easier integration with existing Ethereum-based tools, smart contracts, and decentralized applications, potentially simplifying the integration process with various supply chain systems.

6.5 User Adoption and Training

The complexity of blockchain technology may present challenges in user adoption. However, the implementation of the Trust Storage client significantly simplifies this process for both producers and consumers of TBOM data. This client acts as an intermediary layer, abstracting the underlying complexities and providing a more accessible interface for stakeholders.

To further facilitate adoption, developing intuitive interfaces for the Trust Storage client can make the system more user-friendly for non-technical stakeholders. Creating targeted documentation and training programs focused on client usage can equip users with the necessary

skills to interact with the TBOM system efficiently. A phased roll-out of the Trust Storage client, coupled with a feedback mechanism, can allow for iterative improvements and ensure the system meets user needs effectively.

7 Roadmap and Future Work

As this initial version of the TBOM Security and Trust Mechanism concludes, significant opportunities remain to further enhance and expand the capabilities of the system. The following roadmap outlines key areas for future development and research, aligned with the overarching goals of the RESCALE project and the broader objectives of the Horizon Europe program.

Enhancement of Security and Trust Mechanisms

Building upon the foundational security framework established in this deliverable, the next phase will implement more advanced cryptographic techniques to bolster the privacy and security verification processes within the TBOM ecosystem. A primary focus will be the integration of ZKPs, which will facilitate the selective disclosure of sensitive information without revealing the underlying data. This advancement aligns with the European Union's emphasis on privacy-preserving technologies and data minimization principles.

It is important to note that the current solution does not yet support integration with a certification authority. By leveraging established PKI frameworks, the aim is to ensure interoperability with existing systems while elevating the overall security posture of the TBOM implementation. This integration is a critical aspect that will be addressed in the final version of the module. Additionally, there is no existing integration with a key management system, as the Rescale framework must still incorporate this component. Defining a proper key life-cycle management strategy will also be essential to ensure robust security practices.

An additional area of development will be the implementation of advanced checks on TBOM content, including Proof of Computation and Attestation mechanisms. These features are vital for proving the veracity of the data stored within documents and will require either attestation or proof of computation capabilities, or a combination of both. This approach will provide greater assurance of the integrity and authenticity of information stored within the TBOM, aligning with the EU's goals for increased transparency and accountability in digital systems.

Expansion of Smart Contract Functionality

The current implementation of smart contracts in the TBOM framework provides a solid foundation for automated trust and security management. However, there is potential for more sophisticated applications. The next phase will explore the development of advanced smart contracts capable of performing automated compliance checks against evolving regulatory frameworks and industry standards.

Additionally, experimentation with more nuanced access control mechanisms within the smart contract architecture is planned. This will include the implementation of RBAC systems, allowing for more granular and context-aware permissions management. Such advancements will contribute to the European Union's objectives for enhanced cybersecurity and data governance in digital infrastructures.

Integration with other RESCALE modules

Moving forward, a critical focus will be on the seamless integration of the TBOM Security and Trust Mechanism with other modules developed within the RESCALE project. This integration will be essential for realizing the full potential of the secure supply chain management system.

Comprehensive testing of the integrated system will be conducted through the two complementary pilots organized in the context of the RESCALE project. The testing phase will ensure the readiness of the system for deployment in complex, industrial environments, ranging from Industry 4.0 and public transport to multi-cloud file sharing and storage solutions.

8 Conclusions

This deliverable has provided a detailed overview of the security and trust mechanisms for TBOMs to support RESCALE’s development efforts. We explored current approaches in supply chain security, trust models, and compliance needs, setting the stage for our technical solutions.

We described key security features and cryptographic techniques used to protect TBOM data’s integrity, authenticity, and confidentiality. The document explained our choice of blockchain technology, specifically Hyperledger BESU, for the TBOM ledger infrastructure. We also covered how IPFS fits into our trust storage system.

The report presented both basic and advanced smart contract implementations for managing TBOMs, comparing their features. We outlined the RESCALE Trusted Storage Architecture, including its ledger, storage, and services components.

We identified potential hurdles in implementing these security and trust mechanisms, such as scaling issues, security weak points, and legal or regulatory challenges. The document also laid out our plans for future work, including upcoming improvements and new developments.

This deliverable introduces several key innovations for the RESCALE project:

- **Trusted Storage Module Specification.** We propose a comprehensive specification for the Trusted Storage module, which forms the backbone of the RESCALE secure storage system.
- **Hybrid Decentralized Network.** Our architecture combines Hyperledger BESU, IPFS, and a REST service to create a public permissioned decentralized network. This unique combination enhances data integrity, availability, and security while maintaining controlled access.
- **Smart Contract Implementation.** The `HashManager.sol` smart contract provides a robust foundation for ledger interactions, offering enhanced functionality for TBOM management, including versioning and access control.
- **RESCALE Client Library.** We define interaction protocols between components, abstracting complex operations into a user-friendly client library. This abstraction simplifies integration and usage of the Trusted Storage system for developers and end-users.

This deliverable offers a solid foundation for TBOM security and trust mechanisms in RESCALE. However, we expect to refine and enhance these approaches as the project moves forward and new needs arise, and document them in the final version of the deliverable D4.5.

References

- [1] Martin Abadi. Trusted computing, trusted third parties, and verified communications. In *IFIP International Information Security Conference*, pages 291–308. Springer, 2004.
- [2] Rahaf Alkhadra, Joud Abuzaid, Mariam AlShammari, and Nazeeruddin Mohammad. Solar winds hack: In-depth analysis and countermeasures. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2021.
- [3] Binance. Bnb smart chain. <https://www.bnbchain.org/en/smartChain>, 2024. Accessed: 20/11/2024.
- [4] Johannes A Buchmann, Evangelos Karatsiolis, and Alexander Wiesmaier. *Introduction to public key infrastructures*. Springer Science & Business Media, 2013.
- [5] Zachary A Collier and Joseph Sarkis. The zero trust supply chain: Managing supply chain risk in the absence of trust. *International Journal of Production Research*, 59(11):3430–3445, 2021.
- [6] ConsenSys. Quorum. <https://www.kaleido.io/blockchain-platform/quorum>, 2024. Accessed: 20/11/2024.
- [7] CycloneDX Project. CycloneDX: A Standard for Bill of Materials (BOM), 2024. Accessed: 2024-10-30.
- [8] Pankaj Dutta, Tsan-Ming Choi, Surabhi Somani, and Richa Butala. Blockchain technology in supply chain operations: Applications, challenges and research opportunities. *Transportation research part e: Logistics and transportation review*, 142:102067, 2020.
- [9] Ethereum Foundation. Ethereum. <https://ethereum.org>, 2024. Accessed: 20/11/2024.
- [10] Web3 Foundation. Polkadot. <https://polkadot.com>. Accessed: 20/11/2024.
- [11] GDPR-Info.eu. General Data Protection Regulation (GDPR) – Official Legal Text, 2024. Accessed: 2024-10-30.
- [12] Raphael Hiesgen, Marcin Nawrocki, Thomas C Schmidt, and Matthias Wählisch. The race to the vulnerable: Measuring the log4j shell incident. *arXiv preprint arXiv:2205.02544*, 2022.
- [13] Michael H Hugos. *Essentials of supply chain management*. John Wiley & Sons, 2024.
- [14] International Organization for Standardization. ISO/IEC 20243-1:2023 – Information technology – Open Trusted Technology Provider™ Standard (O-TTPS) – Part 1: Requirements and recommendations, 2023. Accessed: 2024-10-30.
- [15] IOHK. Cardano. <https://cardano.org>. Accessed: [Insert Date].
- [16] Jones, M., Bradley, J., and N. Sakimura. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force (IETF), 2015. Accessed: 2024-10-30.

- [17] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14:2901–2925, 2021.
- [18] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.
- [19] Protocol Labs. Interplanetary file system (ipfs). <https://ipfs.tech>, 2024. Accessed: 20/11/2024.
- [20] Solana Labs. Solana. <https://solana.com>. Accessed: 20/11/2024.
- [21] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv preprint arXiv:1801.01207*, 2018.
- [22] National Institute of Standards and Technology. Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations. NIST Special Publication 800-161 Revision 1, May 2022. Accessed: 2024-10-30.
- [23] Hyperledger Project. Hyperledger besu. <https://besu.hyperledger.org>, 2024. Accessed: 20/11/2024.
- [24] Ravi S Sandhu. Role-based access control. In *Advances in computers*, volume 46, pages 237–286. Elsevier, 1998.
- [25] Theresa Sobbb, Benjamin Turnbull, and Nour Moustafa. Supply chain 4.0: A survey of cyber security challenges, solutions and future directions. *Electronics*, 9(11):1864, 2020.
- [26] Rajeev Sobti and Ganesan Geetha. Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)*, 9(2):461, 2012.
- [27] Software Package Data Exchange (SPDX) Project. SPDX: A Standard for Software Bill of Materials (SBOM), 2024. Accessed: 2024-10-30.
- [28] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.
- [29] R3 Corda Team. Corda. <https://www.corda.net>, 2024. Accessed: 20/11/2024.
- [30] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mohammad Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(1):1–23, 2016.
- [31] Fiorella Zampetti, Salvatore Geremia, Gabriele Bavota, and Massimiliano Di Penta. Ci/cd pipelines evolution and restructuring: A qualitative and quantitative study. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 471–482. IEEE, 2021.