CYBER-PHYSICAL SUPPLY CHAIN USING RANKED BLOCKCHAIN

A Thesis by

Benjamin Michael Beer

Bachelor of Engineering, North-West University, South Africa, 2020

Submitted to the Department of Industrial, Systems, and Manufacturing Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

December 2022

CYBER-PHYSICAL SUPPLY CHAIN USING RANKED BLOCKCHAIN

The following faculty members have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Industrial Engineering.

Deepak Gupta, Committee Chair

Vijay Anand, Committee Member

Saideep Nannapaneni, Committee Member

Sergio Salinas, Committee Member

DEDICATION

This thesis is dedicated to God, the undeniable Creator of all, Lohani, my fiancé whose unwavering support from halfway across the world carried me through, and to my parents, who did their best to ensure their children could start their lives with the best education available.

Give me WiFi, coffee, and Google, and I can change the world. - Benjamin Beer

ACKNOWLEDGEMENTS

I am forever indebted to my faculty advisor Dr. Deepak Gupta who took a chance on some kid from Africa and helped him travel across the world to a strange country to obtain a world-class graduate degree. Thank you for giving me this great opportunity and helping me through all its challenges!

I would also like to thank Dr. Vijay Anand! Through the wonders of modern technology, we have never met in person, yet you have helped me with my thesis research and helped broaden my knowledge in the world of computer science.

I also want to thank all the committee members for their feedback and the time spent reviewing my work. I also wish to thank Wichita State University, the Department of Industrial, Systems, and Manufacturing Engineering, and all other members of staff who have helped me in any capacity during my two years of attendance!

I want to thank all the people who made it possible for me to finish this research. Jason & Jacob of Revelar, thank you for supporting me in my academic endeavors while trying to run a startup and for the friendship and wisdom you have poured into my life. Thank you to River Community Church and Scott Ochs, who provided me with support and advice during our small group sessions.

Lastly, I extend thanks to those not named who have made it possible for me to remain sane, supported me in any capacity, and helped me through some of the challenges of the last two years.

ABSTRACT

The complex nature of modern supply chains has made the traceability of goods throughout their lifecycles very difficult. Modern technology and the digitization of the supply chain have created vast amounts of data that would be useful for traceability; however, the transference of this information throughout the supply chain as the products are moved is a difficult and costly problem to solve. Traceability systems have been built; however, these are typically confined to single organizations and the portions of the product's lifecycle that are within the custody of that organization. This thesis proposes the creation of a ranked blockchain-based cyber-physical supply chain system. The use of blockchain technology allows for the creation of a decentralized system that, through smart contracts, can enforce arbitrary logic at a transactional level. Blockchain also creates an immutable ledger of the past in which information, once written, can be guaranteed to persist forward. Furthermore, ranking can abstract some of the informational aspects of products and enhance traceability. Two prototypes were constructed, the former being a simple prototype demonstrating how a Cyber-Physical Supply Chain (CPSC) system could be created using blockchain. The latter is an expanded version and includes integrity management. The results from these two prototypes show how a CPSC blockchain traceability system can be created.

TABLE OF CONTENTS

Chapter Page

TABLE OF CONTENTS (continued)

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

API	Application Programming Interface.
CPSC	Cyber-Physical Supply Chain.
EVM	Ethereum Virtual Machine.
nonce	Number used only once.
OOP	Object Oriented Programming.
PoS	Proof-of-Stake.
PoW	Proof-of-Work.
UML	Unified Modeling Language.

CHAPTER 1

INTRODUCTION

Supply chains are large complex systems that span vast geographic areas and touch multiple regulatory, legal, political, and social paradigms (Behdani, 2012). Within these supply chains, it can quickly become very difficult to keep track of the flow of products, especially if the movement of the products has to comply with a particular set of standards.

This traceability information of products is quickly becoming an important part of the purchase decision on the part of consumers. Cao et al. (2021) found that traceability is an important factor affecting consumers' purchase intention, particularly for food. However, most of this traceability information is contained within closed and proprietary systems that provide little to guarantee the information's quality. These traceability systems are also generally non-client facing, which means that consumers have to purchase products according to perceived trust in a handful of name-brand stores which do not provide the consumer access to provenance information. Sander, Semeijn, and Mahr (2018) found that consumers would find value in self-investigating the provenance and traceability of products and that access to this information would greatly affect the purchase decision.

Since the publication of (Nakamoto, 2008), blockchain technology has become a central focus in many research fields. Cryptocurrencies, one of the most popular applications of blockchain, provide a decentralized means by which trustless and nameless entities can transact safely and securely. Blockchain is also quickly becoming one of the most disruptive technologies of the 21st Century with constant creation of new use cases.

One of the more recent developments in blockchain technology is the addition of a means by which arbitrary logic can be placed before a transaction. This arbitrary logic, also called a smart contract, allows the creator to dictate the terms by which a transaction can succeed or fail without needing to preside as an intermediary in each transaction. Meaning it is possible to create a decentralized system in which rules can be enforced at a transactional level without a regulatory authority needing oversight of each transaction.

Compliance with certain standards is also an integral part of the supply chain, and this compliance must run throughout the entire provenance record of a product. Traceability must provide a means by which product crossover and contamination points can be found. For example, if peanut-free products are handled in a logistics facility with products containing peanuts, this could pose a risk for a peanut-allergic consumer. Traceability could also be used to augment quality metrics (Bosona & Gebresenbet, 2013). Consider the case where products are handled by an entity that does not posses the correct certification. These products need to be traceable, as their quality might differ from those handled correctly.

Traceability of goods and products is important, but traceability in terms of actors must also be an integral part of a holistic supply chain management system. How can consumers trust provenance timelines that cross through actors they may not know? Within the world of computer science, integrity management frameworks are designed to provide and guarantee security within a digital system through ranking. These frameworks could benefit the supply chain world, especially if enforced through an immutable and deterministic system such as a blockchain.

1.1 Research Questions

Several "founding" questions guide the work of this thesis. Just as vision and mission turn a start-up into a company, these questions will turn ideas into research.

1. How can supply chain traceability be improved through the usage of blockchain technology?
2. How can compliance with certain supply chain standards be enforced and added to the provenance record?
3. How can integrity ranking be used to augment the supply chain?

1.2 Limitations & Assumptions

Both supply chain and blockchain are very broad research topics. As this research encompasses aspects within both, it is necessary to limit the depth and breadth of the scope.

Optimization and Supply Chain

Even though this research focuses a great deal on the supply chain, it is not a research endeavour into how optimization can be further brought about within the supply chain. As such, this research does not consider optimizing the supply chain and associated logistics. Additionally, products in the context of this research signify single unique objects that the consumer will ultimately consume, and batching or batch-based goods are not considered.

The supply chain structure in this research is also not as mathematical as in traditional supply chain papers. The structure of supply chains, particularly the size of the supply chain, the number of echelons, and the number of goods & actors, is not a pertinent factor in this research. The goal is to answer the research questions for a general supply chain.

This choice also means that this research would form the basis for future work on optimization. This way, the scope of future optimization research is left open and not constrained to following or building on optimization performed in this research.

Cryptocurrency & Money

Cryptocurrency is arguably the most known application of blockchain technology. However, one of the limitations of this thesis is that we do not consider the currency aspect of the supply chain environment. The flow of monetary value within the supply chain is a large and complex economic problem, which is not the focus of this research.

Transaction Models

The transaction models utilized by blockchain networks greatly affect how transactions are created and verified in the consensus mechanism. This research does not attempt to evaluate different transaction models and how their architectural differences affect the implementation of a blockchain traceability system.

Additionally, the choice of platform for the creation of a prototype was arbitrary, and the only deciding factor was the speed at which a prototype could be created, given the

expertise and experience of the author.

Information Correctness

In this research, it is assumed that the information provided by real-world entities is correct and true.

1.3 Structure of the document

This section gives a brief peek into the contents of each section. The idea is not to provide a sectioned summary of the document but more an idea of how the different concepts within each chapter flow to the next.

Chapter 2 BACKGROUND & LITERATURE REVIEW Presents a detailed review of pertinent literature guided by the research questions. This review is split into four sections, each covering a facet of the research problem in this study. This gives a holistic picture of the problem and other research in similar scenarios and with similar technology.

Chapter 3 METHODS Documents the methodology used to address the research questions. This chapter is intended to answer the "how" aspect of creating the blockchain-based cyber-physical supply chain system detailed in the next chapter. This chapter will provide insight into how the supply chain can be abstracted suitably for the blockchain in light of the research questions. This chapter will also introduce the ranking model used in this research.

Chapter 4 CASE STUDY & ANALYSIS This chapter starts by describing the creation of a prototype. This thesis does not stop at creating a theoretical framework; instead, it aims to produce a functional prototype to demonstrate the functionality. Instead of producing results in a bunch of screen dumps and terminal outputs, this thesis will present results using a test-case scenario. This will be a narrative-based test case to demonstrate how a blockchain traceability system could function, especially in actor-based integrity management.

CHAPTER 2

BACKGROUND & LITERATURE REVIEW

Based on the research questions posed in Section 1.1, it is easy to see that there are four facets to this research; supply chain traceability, blockchain technology, blockchain within the supply chain field, and ranking, specifically integrity ranking. It makes sense to examine pertinent literature and background within these four facets and draw them together to produce a holistic picture. Hence, this chapter has five sections: four to examine each facet and a fifth to draw them all together.

This chapter also contains significant background information intermixed with the literature study. The research in this thesis touches on multiple schools of thought. As such, it is the author's opinion that a more explorative literature review would provide more insight than a purely academic one.

Supply Chain Traceability This section will look at some of the research that has been done and some of the emerging patterns in modern research being done to address traceability in the supply chain environment.

Blockchain This section will look at the research developments of blockchain from an industrial engineering perspective.

Blockchain in Supply Chain This section will look at research on how blockchain can be closely incorporated into the supply chain environment. Additionally, it will explore some of the uses, and emerging research patterns.

Integrity Integrity management is a well-known technology within the world of computer science. This section will explore how can it be used outside of computer science to solve problems in a world that is becoming more digitally integrated. This section will look at some of the uses of integrity management and how it could be used in the supply chain.

Summary of Findings The last section will attempt to bring together all the different threads in the context of the research questions posed in Chapter 2. This summary will help set the direction for the rest of the thesis.

2.1 Supply Chain Traceability

Traceability and provenance are too often confused with one another as being the same thing when, in fact, they are quite different. Bosona and Gebresenbet (2013) defined traceability as; *"The ability to access any or all information relating to that which is under consideration, throughout its entire life cycle, by means of recorded identifications"*. ISO (2007) defines traceability as; *"[The] ability to follow the movement of a feed or food through specified stage(s) of production, processing, and distribution"*.

From these two definitions, it is important to note that traceability is about access to information and not the information itself. Although, this access to information does itself imply that:

- Traceability information should be across the whole lifecycle of a product.
- Traceability information should be uniquely attributable to a product
- Traceability information should be accessible

The actual information itself is the provenance. IBM (2021) defines provenance as; *"...the validated history of ownership, custody, and origin of a specific product instance such as a lot, a batch, or a serial number."*. The keyword in this definition is validated. An information dump about products does not constitute provenance information. The information has to be validated.

The following definition of traceability taken from a blog post by (Norton & Conlon, 2019) frames traceability and provenance very well; *"Supply chain traceability is the process of tracking the provenance and journey of products and their inputs, from the very start of the supply chain through to end-use."* This definition frames both concepts elegantly, and this will be the working definition for both provenance and traceability for this thesis.

With the *what* of traceability and provenance answered, the *how* is the next question to pose. Supply chain traceability and the nature of provenance are very jurisdictional, with regulatory bodies setting different standards. This makes it particularly difficult to create global traceability systems. Some jurisdictions may require public accessibility of certain types of information, while other jurisdictions might not. This also presents challenges for the enforcement of requirements.

2.1.1 Certification Methodologies

This aforementioned stamp-of approval can be provided in the form of a certification issued by a certifications authority or some regulatory body. Naturally, these certifications do exist within the current system; however, there are ways in which they can be improved.

Current certifications typically manifest at a consumer level as a sticker stuck on the product's packaging. The point is to provide a stamp of approval so the consumer can distinguish between the different products that have undergone different certification standards and demonstrated compliance. However, after a quick trip to a grocery store, it is apparent that almost every manufacturer has some form of a certification sticker, and it is very difficult to choose between products based on this.

Sander et al. (2018) found that consumers are overwhelmed with current certification information on products, making it difficult for them to make accurate assumptions of credibility. They suggest that a transparent traceability system would significantly improve this situation and provide consumers with the ability to self-audit certifications. Yuan, Wang, and Yu (2020) found that high-quality traceability information affects the perceived value of products among consumers and, subsequently, the purchase intention for products with a higher perceived value. However, one of the issues they highlight is that this traceability information, when presented to the consumer, becomes reliant on the consumer's expertise to decipher.

The work of Sander et al. (2018) and Yuan et al. (2020) creates a duality. On the one side, presenting traceability information solves the lack of transparency in current certifica-

tion systems and enhances the consumer's ability to self-audit. However, on the other side, when this information is presented to consumers, it becomes subject to consumer interpretation.

In their literature review, Tagarakis, Benos, Kateris, Tsotsolas, and Bochtis (2021) found that current traceability systems struggle to encompass a product's entire lifetime and that certifications used are created using closed standards and, in some cases, even proprietary. It would appear that these problems can be solved by creating a transparent certification system in which the certifications can be traced. Consumers would then be able to track certifications and see that products were certified; however, the logic of it does not need to be presented to avoid subjective interpretation. A key consideration here, though, is that the abstraction away of logic cannot be to hide it. Consumers should still be able to have faith that the logic was enforced deterministically and fairly, and as such, a suitable logic abstraction mechanism needs to be utilized.

Figure 2.1 provides a representation of a simple supply chain. This supply chain has both goods and information represented on it. Product information needs to be transferred between participants within the supply chain. However, the current supply chain structure means that the burden for this information transference rests on the individual actors. As goods move, this burden becomes greater as the current actor in the custody of goods must transfer their product information and all previous product information they received. The result is that some actors transfer this information while others do not.

2.1.2 Cyber-Physical Supply Chain

Sander et al. (2018) suggests that creating a holistic and transparent system could be achieved with blockchain-based technology, a sentiment echoed by others (Balzarova, 2020; van Hilten, Ongena, & Ravesteijn, 2020; Abeyratne & Monfared, 2016; Montecchi, Plangger, & Etter, 2019). The caveat is that actors in a supply chain are individually responsible for providing information at their level (Yuan et al., 2020).

The traceability system should present a way for data to be provisioned at the prod-

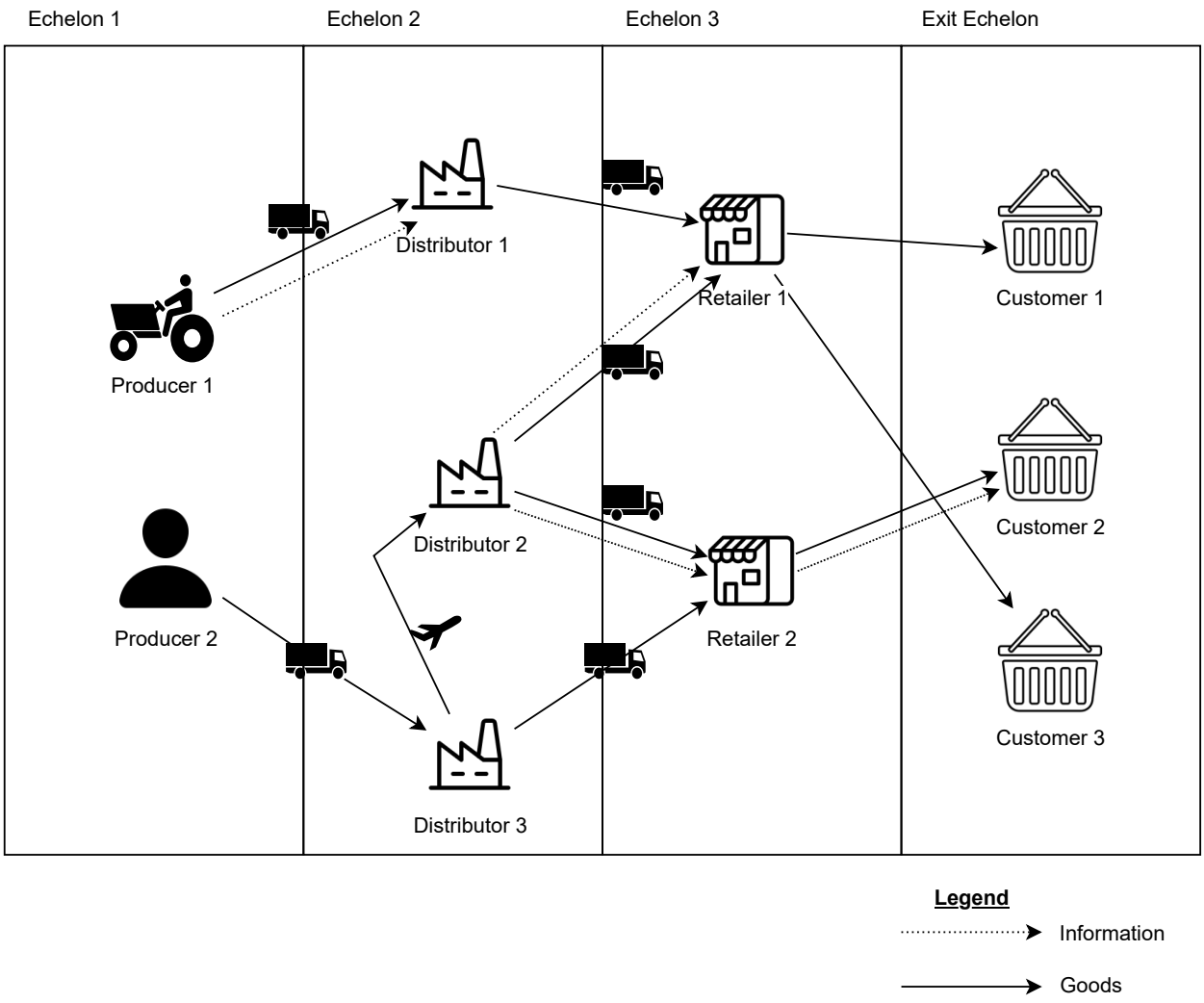


Figure 2.1: Modern supply chains move both goods and information about the goods. The information movement is, however, not always the same as the product movement.

uct's point of origin, wherever possible. This product can then be updated as it moves through the supply chain, easing the information burden for individual actors.

This also means the digital world should echo the physical world to create a cyber-physical hybrid. This close incorporation of the two worlds to culminate in the formation of one cohesive system is one of the goals of Industry 4.0. It is already the subject of significant research. A large part of the modern supply chain is already digitally driven by institutions such as Amazon that provide products directly to consumers and large manufacturers that mass produce make-to-order products that are individually customizable.

These systems are already being modeled as complex cyber-physical systems both for design purposes such as production planning (Rossit, Tohmé, & Frutos, 2019) and for enhancement of logistics (Park, Son, & Noh, 2021). Therefore, a traceability system should be designed to ensure the functioning of the cyber and physical world co-equally as one cohesive system. This can be achieved through the suitable abstraction of data and logic captured and facilitated automatically by a digital system. However, given the prior mentions of determinism, this abstraction should also be deterministic.

2.2 Blockchain

A large amount of research literature suggests that one approach to creating a deterministic supply chain traceability system is through blockchain (Dutta, Choi, Somani, & Butala, 2020a; Wamba & Queiroz, 2022; Raja Santhi & Muthuswamy, 2022). The obvious first question is, what is blockchain, and why does it matter?

Blockchain can be defined as a distributed ledger of information that is only added to through consensus mechanisms in a decentralized digital network (Hayes, 2022; IBM, 2022; Merriam-Webster.com, 2022). Even though the constituent elements for blockchain have been around since the 1980s, blockchain in the form most are familiar with today can be attributed to Satoshi Nakamoto and the publication of their Bitcoin paper (Nakamoto, 2008).

The three most important elements of blockchain are its data structure, decentralization, and the consensus mechanism.

2.2.1 Data Structure

In mathematics and cryptography, a hash is simply a mathematical function that is easy to compute in one direction but very difficult to compute in the opposite direction. The hash function also usually takes an arbitrary input and produces an output of a fixed length that is entirely deterministic. That is, the same input will generate the same output only, and no output duplication occurs for different input values.

Nakamoto (2008) proposed using the Hash function to chain together data with each chainlink being timestamped. This is achieved by breaking up data into chunks or blocks of information. The separation between blocks is set as a fixed duration of time denoted as epochs[1]. At the end of each epoch, all the information stored inside a block is locked so that no more information can be added. In the Bitcoin model, this information is in the form of transactions arranged in a Merkle tree. This Merkle tree can be seen on the right-hand side of Figure 2.2. At the same time, a time snapshot is taken and stored as a timestamp. The Merkle tree data structure allows for quick traversal of transactional data when auditing.

When creating a new block, the previous block's hash is stored inside the new block. Any change in the contents of the previous block would result in a change in the hash stored in the next block. Moreover, this pattern would be repeated until the terminal block or tip is reached.

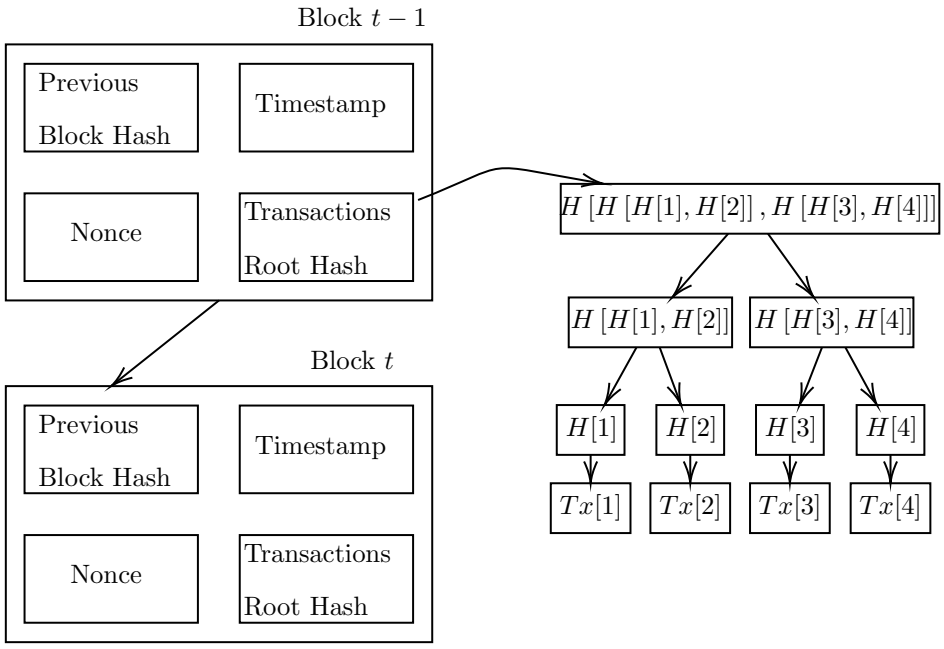


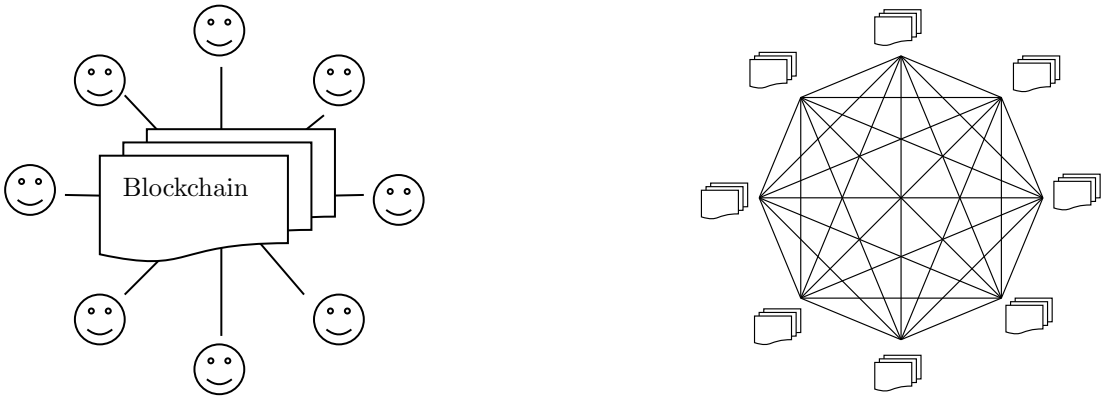
Figure 2.2: Bitcoin Data Structure

2.2.2 Decentralized

If the blockchain data structure were to be hosted in one location, under the control of one entity, then that entity would be able to change the contents of previous blocks, and there

[1]Although in most cases epochs are fixed lengths of time, this is not a hard requirement.

would be no accountability mechanism. The way to get around this is to distribute multiple copies of the blocks among multiple members, each being distinct and independent entities called nodes. Nakamoto (2008) extended the blockchain data structure by combining it with a decentralized network topology. Figure 2.3 presents a visual comparison of the centralized versus decentralized topologies.



(a) Centralized network (b) Decentralized network

Figure 2.3: Centralized vs. Decentralized Topologies

Figure 2.3a shows how multiple copies of the blockchain are distributed amongst connected independent entities. If a nefarious actor decided to modify the historic state of the blockchain, they would need to do so on every copy of the blockchain that exists.

2.2.3 Consensus Mechanism

With a chained data structure and decentralized network of nodes, it is logical to envision that there should exist an agreed-upon standard by which new data is added to the blockchain and distributed to the network. Nakamoto (2008) also covered this in the Bitcoin paper through the idea of consensus.

Consensus creates a means by which a network of decentralized entities can agree on authoritative assertions without individual entities taking over control of the system. These consensus mechanisms are themselves algorithmic, and there exist several ways in which to approach the idea. For the sake of brevity, only the two most known Proof-of-Work and Proof-of-Stake will be discussed in the next subsections.

Proof-of-Work

The Proof-of-Work (PoW) mechanism was first fully presented by Jakobsson and Juels (1999) as a means by which a digital entity can prove that it performed a series of computations. However, the algorithm itself was first presented by Mail (1993) as a way to combat spam email. Nakamoto (2008) used the PoW algorithm and extended it to create a means to convert compute power, or CPU time, into verified information.

In the Bitcoin case, nodes compete to solve a complex hash-based problem to produce a specific output called a *number-used-only-once* referred to as nonce. Because of the nature of the hashing algorithm, this problem is computationally difficult to solve but easy to verify. Each person attempting to solve the problem is called a miner, and multiple independent miners compete to solve this problem simultaneously. The first miner to provide proof of a solution, i.e., proof that they have done the computational work, is rewarded with a digital token, Bitcoin.

Once a miner has solved the problem, the solution is broadcast to other miners, who use the same hashing algorithm to verify the solution's correctness. Once verified, the original solver can generate the next block and append it to the blockchain before moving to the next epoch, where the process begins again.

The difficulty of this problem can be varied to ensure that the rate at which new blocks are produced does not exceed a specified metric. As nodes append new blocks, it becomes more difficult to modify the historic state. For the historic state to be changed, all the computational work that has been done to create the state would need to be recomputed. This is how blockchain gets its immutability.

Proof-of-Stake

As the adoption of Bitcoin technology grew, the computational difficulty required to verify transactions grew as well, and in turn, the energy required to power this computational effort also grew. At the time of writing, Digiconomist (2022) estimates the annual energy usage of the Bitcoin network to be in the region of 204.5 terra-watt hours.

An alternative consensus mechanism was proposed in an online forum by QuantumMechanic (2011) and is called Proof-of-Stake (PoS). In this algorithm, the power to accept the blockchain's historic state is provided as a function of the amount of the token a prover can prove they own. In this way, there is still agreement on a historically verified state; however, the computations required for verification are not varied and require significantly less energy. Instead, several nodes can vote upon the accepted network state, and the voting power is a function of the number of tokens held by each voter. QuantumMechanic (2011) also suggested that voting power can be delegated to trusted parties who perform the verification.

This consensus mechanism still creates a robust verification framework without the heavy computation burden carried by PoW.

The combination of a decentralized consensus-driven network that is constantly verified means that a deterministic system with immutable state persistence is created. This system can create digital assets and record the transactions needed to move and exchange assets between entities. The decentralized system can then use the consensus mechanisms to agree on valid transactions that move assets around the blockchain, creating a perfect historic record of asset movement.

In this way, all transactions are pre-audited before they are even committed to the public ledger state. Additionally, in a public blockchain, all participants have access to the historic state of the ledger, so they can independently verify the correctness of transactions should they choose to do so[2].

2.2.4 Smart Contracts

Given that the nodes that perform verifications are themselves computers providing the consensus layer with transaction-specific arbitrary logic sets should be possible. For nodes to verify transactions, they verify transactions against the standard universal rules and the logic rules specified in the transaction itself. These logic rule sets are called smart contracts.

[2]On most blockchains, when the software to run a node is first started, all historic transactions are validated locally in a process called *syncing*. The only way to become a participant is to verify the entire historic state.

Buterin (2014) is credited as being the first to link the idea of a smart contract, created by Szabo (1998), with the blockchain to form a decentralized system capable of executing arbitrary logic in a deterministic manner with a perfect historic record of that execution and any changes that result from it.

What makes smart contracts special is the continuation of immutability and determinism into arbitrary logic containers, two key characteristics of blockchain. Immutability is the term used to describe blockchain's perfect historic record, and it means that once a state change has persisted, it can never be removed or modified. This, in turn, means that if the effects of one state change want to be reversed, the only way would be to create a second state change that performs the opposite of the first. The net result would be a reversal of the first state change; however, both changes of state would still be recorded in the historic record of the blockchain. This concept can be likened to double ledger accounting with version control.

Determinism is the property of a system that will produce the same output for the same input. Determinism as a property can be applied to functions as well. For example, addition is a deterministic function. The same input terms, when summed, will produce the same output terms. i.e. $1 + 1$ always equals 2.

Smart contracts are usually deterministic, meaning the logic will always be executed similarly, and the same inputs will produce the same outputs. The property of immutability also applies to the smart contract and its result. Any interaction between any entity on-chain and a smart contract will be stored on-chain within the historic state, creating an immutable log. Additionally, once a smart contract has persisted on-chain, the logic inside a smart contract can never be modified. It can only be replaced through a future persistence on-chain of a new contract version that invalidates the previous version.

Smart contracts allow arbitrary logic to be locked in a decentralized system to govern deterministic transactions. The desired output transaction will be produced only once the conditions specified inside a smart contract have been met.

The arbitrary logic part of smart contracts is also its downfall. Alharby and Van Moorsel

(2017) found in their review that there is a large amount of research and work being done to address some of the issues with current smart contract implementations, namely codifying, security, privacy, and performance. The creation of smart contract logic should be structured not to void the benefits of using blockchain technology. Caution should also be taken during the development of smart contracts to ensure that they do not create exploits through flawed logic(Peters & Panayi, 2016). Flawed logic could create holes within the smart contracts that could be used to exploit the system.

2.3 Blockchain in Supply Chain

Blockchain technology and smart contracts present a means by which decentralized systems can be deeply embedded within the supply chain to create cohesive cyber-physical systems(Dolgui et al., 2020). Kim and Laskowski (2018) showed that blockchain technology could provide traceability in the current supply chain across organizational, jurisdictional, and geographic borders with a product-level granularity of information.

In their review of research trends and future research directions, (Dutta, Choi, Somani, & Butala, 2020b) explore many of the industrial applications of blockchain technology, including in the field of supply chain. They, too, suggest that blockchain can provide a decentralized means by which high granularity provenance information can be made available about a product's entire lifetime across multiple actors in a transparent, deterministic, immutable, and decentralized system. Similar conclusions were reached by (Montecchi et al., 2019; Liu & Li, 2020; Dujak & Sajter, 2019)

Figure 2.4 provides a depiction of the structure of the current supply chain, according to the author. Provenance information about products likely comes from the information systems managed by each actor. This means that the flow of information follows the flow of goods. As goods move from one actor to another via some transportation medium, information is also shared directly between actors. This information sharing is point-to-point, and each recipient can decide how much to share.

The disadvantage is that the individual actors control the flow of information and

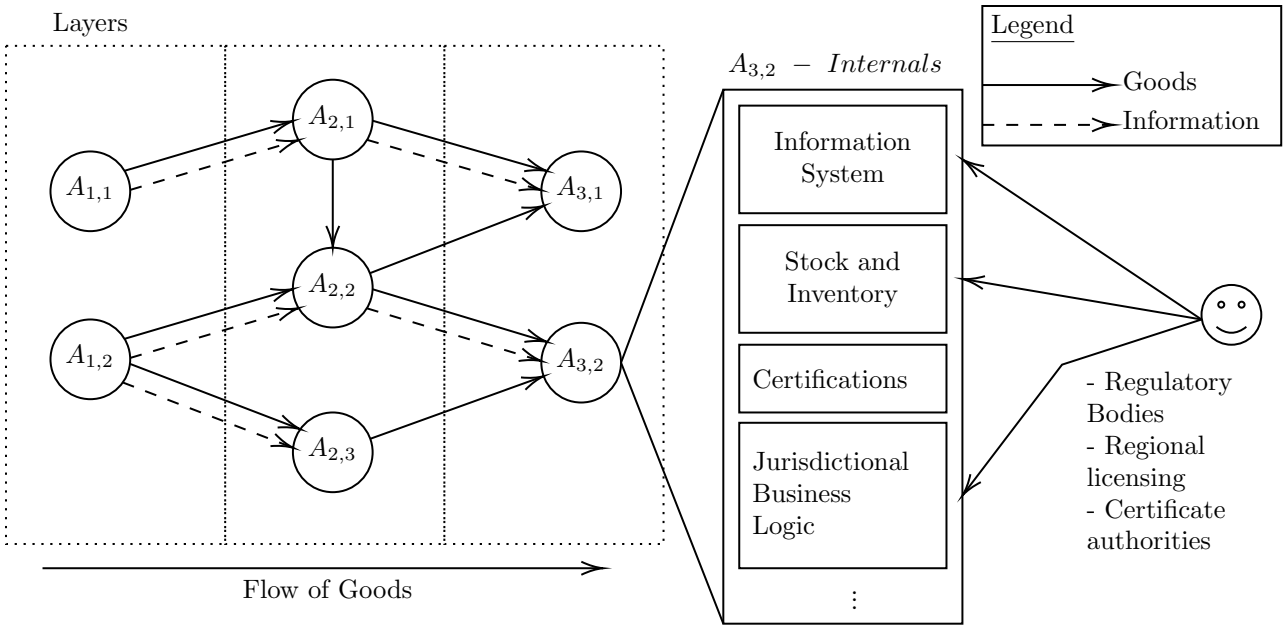


Figure 2.4: In the supply chain, individual actors themselves have their information systems, stock and inventory control systems, business and legal compliance system, et cetera. Any provenance information about goods is controlled by the actor and sent directly between actors.

any break in the flow of information at one layer results in the information not being made available in the next layer. An example can be observed in Figure 2.4. Actor $A_{1,1}$ sends both goods and information to Actor $A_{2,1}$. Actor $A_{2,1}$ however sends goods to Actors $A_{2,2}$ & $A_{3,1}$, but only information to $A_{3,1}$. Actor $A_{2,1}$ also controls the granularity of information sent, which may or may not include information about the provenance of goods in the previous supply chain layers.

Certification and compliance with regulations are equally challenging as regulatory bodies need access to the internals of individual actors in order to provide a certification with any certainty. This certification only applies to that individual actor, and using historic data about that actor in this process is difficult.

2.4 Integrity

Before diving into the integrity review, the first question to answer is why it matters. From previous subsections, it is known that blockchain creates an immutable historic record. If this historic record is then immutable, the question to answer is why integrity management would matter then?

Within the supply chain, there is a flow of information. The information itself could be captured accurately with blockchain. However, the flow of that information and the characteristics of the processes that generated it also need to be considered. Trust needs to be established in the flow of the information as well as the information itself.

In the computer science world, trust can be established through information security. Information security consists of three broad principles; confidentiality, integrity, and availability; together, these three are colloquially referred to as the CIA triad (Forcepoint, 2018; Walkowski, 2019). Integrity concerns the unauthorized modification of the system and/or its data, confidentiality concerns unauthorized access to the system or the data exchanged, and availability has to do with the dependability of a system to be usable by the necessary users at times required (Gil, 2018).

2.4.1 Confidentiality

A system with good confidentiality effectively controls access to the data within the system. It only allows authorized parties to access the information or the related system parts and governs who can access what. Confidentiality also includes encryption, key sharing, cryptography, et cetera.

In the supply chain environment, one application for confidentiality could be used to ensure that only those who should handle specific products are allowed access to the necessary systems, resources, individuals, et cetera in order to handle those products. It should be obvious to the reader that this would be very difficult to implement, although it sounds good in principle. A successful implementation would need a system capable of

controlling every individual's access to every piece of data, which would become a scaling nightmare. However, this could prove a valuable tool to use at an intra-organizational level.

The Bell-LaPadula model is widely adopted for managing and ensuring confidentiality requirements are met (Bell & LaPadula, 1973). Shing, Shing, Chen, and Lee (2006) presented a Markov modeling process using the Bell-LaPadula model in the supply chain context for fraud mitigation and maintenance of confidentiality during the supplier selection process. This illustrated how confidentiality management could assist with the business operations associated with supplier procurement and increase information security. However, in the case of this thesis, the attention is focused on the entire supply chain operations, particularly the trust.

2.4.2 Integrity

Integrity is considered to be the dual of confidentiality (Biba, 1977; Li, Mao, & Zdancewic, 2003). A system with good integrity management controls which entities can perform actions on which other entities. This ensures that unauthorized entities cannot access data and that authorized entities cannot perform unauthorized actions. If confidentiality is the "who can access what", then integrity is the "who can do what and on what".

The Biba model (Biba, 1977) can be used to model the integrity of a system. Arthur, Olivier, and Venter (2007) presented a means by which the Biba model could be used within the forensic realm to manage evidence better. The digital evidence artifacts and the applications/entities which access them were assigned integrity values. The Biba model is then used to define which entities are allowed access to which artifacts. Additionally, when the Biba Model's rules are violated, the integrity values of both the entities and artifacts are also updated to reflect the changes that occurred to the integrity as a result.

2.4.3 Availability

A system with high availability functions as it should, all the time, reliably, and consistently. In the supply chain context, this should not be confused with product availability,

which refers to the dependability of the supply of a certain quantity of a product. In the context of computer science applied to supply chain, availability means that the supply chain always functions as it should.

There is no single method by which a system can be made more reliable; instead, it can be seen as a function of the entire system's design.

2.4.4 Biba Integrity Model

The Biba integrity model is a formal data integrity model that can be used to maintain data integrity. The Biba model consists of two types of entities, subjects and objects. Subjects perform operations and actions or interact with passive objects, and each subject and object is assigned an integrity level.

The Biba integrity model ensures that objects do not have their integrity modified by unauthorized subjects, objects do not have their integrity modified in the wrong way by authorized subjects, and that the objects are always consistent with the real-world truth. This is achieved using the three axioms of the Biba model:

1. **Simple Integrity Property** - A subject cannot read data at an integrity level lower than its level (no read down)

2. **Star Integrity Property** - A subject cannot produce data at an integrity level higher than its level (no write-up)

3. **Invocation Integrity Property** - Subject A cannot invoke Subject B to act on an object on Subject A's behalf if subject A has lower integrity than either the object or Subject B., I.e., Lower integrity subjects cannot request higher access through a higher ranked subject.

Biba also introduces the idea of a reference monitor. The system must contain some method/entity by which the rules of integrity are enforced, and this method is called a reference monitor.

An application of the Biba model could be to assign integrity levels based on logic rules. Instead of governing data integrity, the Biba model could be extended to govern the integrity of logic execution. For example, subjects that can execute more advanced logic can be assigned a higher integrity level than subjects that cannot. A reference monitor can then be used to measure the execution of this logic during interactions with objects, and the integrity management rules can also be enforced in these interactions.

2.4.5 Usage in Supply Chain

The Biba model's distinction between two types of entities is well suited to the supply chain context. In the supply-chain context, there are also two groups of entities. There are the goods and objects transferred between participants and eventually reach consumers. Then some participants and subjects transfer, handle, create, and process these goods.

Products requiring additional processes to be performed on them can be assigned higher integrity levels than those without. For example, goods that need to be refrigerated can be assigned a higher integrity level than those that do not need to be refrigerated. Similarly, actors capable of performing specific processing can be assigned higher integrity levels than those incapable.

A reference monitor would then be able to enforce the logic rules merely by using integrity management. Additionally, the reference monitor does not need to know the specific details of the reasoning behind how the integrity was assigned. If the integrity rules are enforced along the entire product lifecycle, the consumer can distinguish between products based on integrity and not just on trust alone.

2.5 Summary of Findings

The four sections of this chapter can be brought together to guide the development of the methods to be used in this research. The following summary provides a recap of some key takeaways from the literature review.

1. **Section 2.1 Supply Chain Traceability** explored traceability in the context of sup-

ply chain. From the review of the literature, it is apparent that:

- Traceability needs to be transparent
- Traceability needs to be holistic
- Information should be captured once and moved/updated as the physical product moves through the system without requiring the individual actors to pass the information between one another
- Certifications need to be issued according to a transparent and deterministic process
- The system should be cyber-physical, and suitable abstractions should be made to ensure that the cyber realm reflects the real-world

2. **Section 2.2 Blockchain** looked at the technology of the same name. This technology presents a new approach to distributed systems without requiring a centralized governance authority. This system exhibits some key features:

   - It creates a network that functions deterministically with perfect preservation of historic state in which state changes are appended to the historic record in the form of transactions
   - No single entity has control of the system, and multiple independent parties reach a consensus through a fixed algorithmic process

   It allows for packets of logic to be attached to transactions. These packets of logic are called smart contracts and, in turn, provide several features:

   - Smart contracts can be used to add arbitrary logic to the system, which governs how transactions are executed and what the conditions of successful transaction execution need to be
   - The independent nodes and the outcomes that will execute the logic inside the smart contract will be verified as part of the consensus mechanism

   These smart contracts do, however, carry a very serious caveat that they need to be designed so as not to compromise the system

3. **Section 2.3 Blockchain in Supply Chain** looked at some of the new research that is being done to integrate blockchain technology into the supply chain context. Some of the findings are:

    - Blockchain can be used to create a holistic traceability system that centralizes information in a decentralized and transparent manner
    - Blockchain transactions can be used to abstract supply chain transactions into the cyber realm
    - Smart contracts can be used to capture the logic of physical supply chain transactions and, because of the cohesive cyber-physical nature of the integration, can be used to enforce business logic via the cyber realm in a deterministic and transparent manner without requiring a third-party authority
    - Smart contracts can also be used to issue certifications to actors. Certifying authorities can interact with actors via smart contracts and issue certifications digitally. The stipulations of these certifications can then be added to the smart contracts that govern the supply chain product transactions.

4. **Section 2.4 Integrity** looked at formal integrity modeling and how it could have applications in the field of supply chain.

    - Integrity modeling can be extended to the supply chain and used to preserve product integrity
    - Biba integrity model's distinction of objects and subjects fits well with the supply chain context where there are actors and products
    - Biba integrity rules can be enforced through a reference monitor

Table 2.1 presents a tabular summary of the most important citations within this chapter.

Table 2.1: Summary of pertinent literature

	Traceability and provenance	Blockchain	Smart contracts	Information Integrity Management
This paper	✓	✓	✓	✓
(Tagarakis et al., 2021)	✓			
(Yuan et al., 2020)	✓			
(Sander et al., 2018)	✓	✓		
(Balzarova, 2020)	✓	✓		
(Abeyratne & Monfared, 2016)	✓	✓	✓	
(Montecchi et al., 2019)	✓	✓	✓	
(Kim & Laskowski, 2018)	✓	✓	✓	
(Dutta et al., 2020b)	✓	✓	✓	
(Arthur et al., 2007)				✓

It is envisioned that the research in this document will reach over traceability & provenance, blockchain, smart contracts, and information integrity management.

2.6 Research Contribution

Through the usage of blockchain, it is possible to create a cyber-physical supply chain. This cyber-physical supply chain would be able to provide traceability through a decentralized system that is also transparent.

Figure 2.5 depicts how blockchain can improve the system based on the literature review findings. Instead of actors needing to send information directly to individual actors, as in Figure 2.4, their information could be shared with the blockchain system as part of the supply chain transaction of goods. This moves the burden to provide information away from actors and shifts it onto the blockchain. Additionally, the interaction between the actors and the blockchain and the terms of the goods transaction can be governed by smart contracts.

The information provided to the blockchain can then be used to construct the provenance and traceability of goods without needing individual actors to communicate directly with each other.

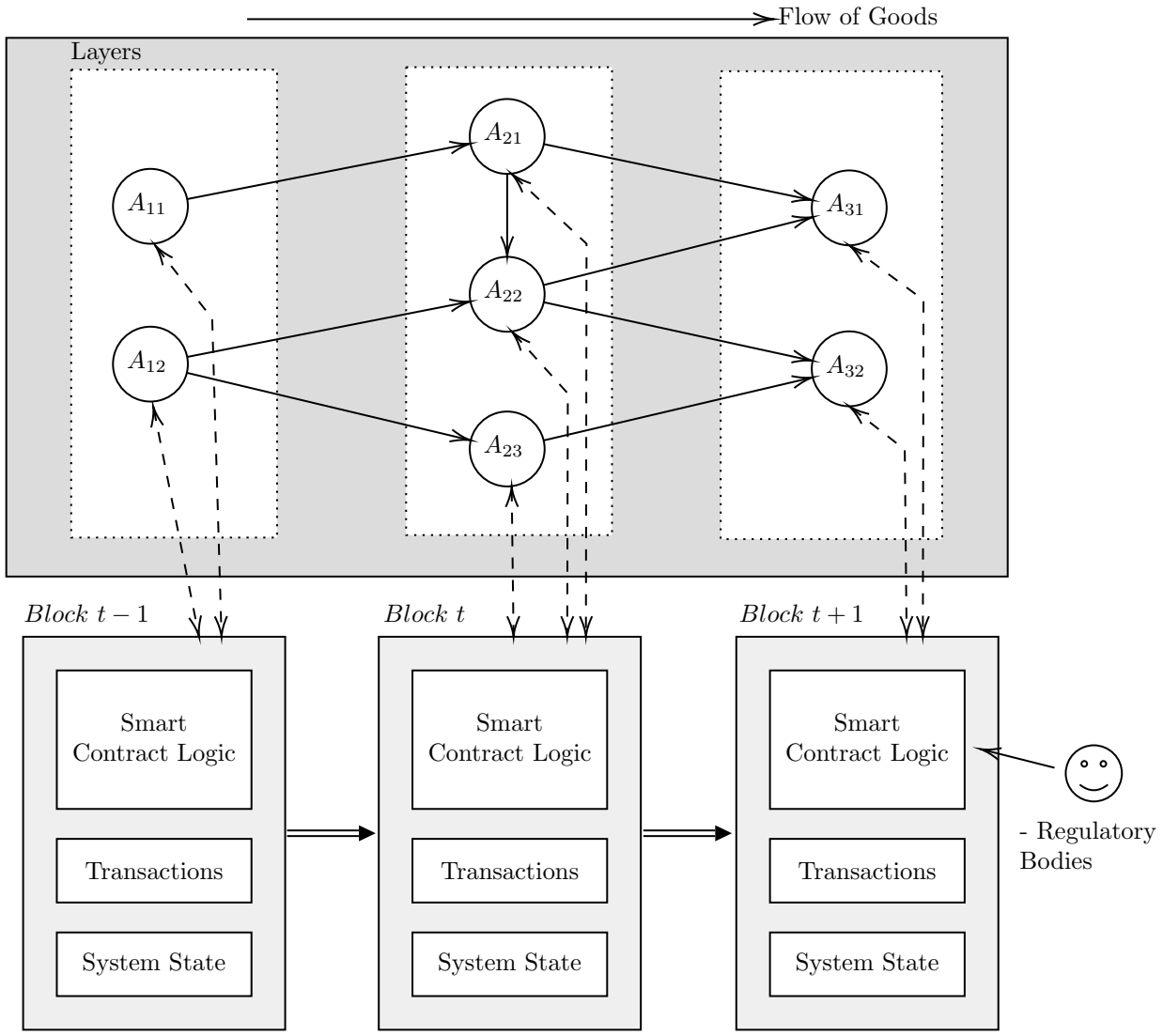


Figure 2.5: Blockchain Supply Chains allow a blockchain system to capture information from the actors and store it as transactions. Actors only need access to the blockchain to access any information about any other actor

This provenance information actors provide is added to the blockchain's historic state through smart contracts. Because of the structure of the blockchain, the historic state can also be queried. For example, block $t-1$ will contain provenance information about the goods transacted by actors $A_{1,1}$ &$A_{1,2}$. As time progresses, the goods move through the

supply chain to end up at actor $A_{3,1}$. Actor $A_{3,1}$ would be able to use the smart contract logic to query the historic state of the blockchain and obtain the provenance information of its goods from blocks t and $t-1$ without needing to exchange information with any of the actors the goods moved through previously.

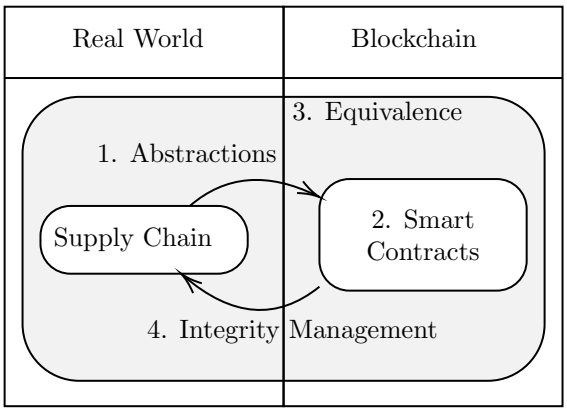


Figure 2.6: Summary of Methodology

There are four components to achieving this integration, as depicted in Figure 2.6.

1. **Abstractions** - This will create an appropriate digital version of both the dynamics and statics of the real-world supply chain into the cyber-realm.

2. **Smart contracts** - This will be used to house and implement the abstractions within blockchain.

3. **Equivalence** - This will ensure that events within the real world have equivalents in the blockchain world and vice versa.

4. **Integrity Management** - This will be implemented in smart contracts. The prior establishment of equivalence ensures it is enforced in the real world.

These four components ensure a two-way supply chain and blockchain relationship. This two-way mapping not only allows for the supply chain to be represented in the blockchain, but it also allows for methods and frameworks implemented in the blockchain world to be persisted into the real world in a meaningful manner. If there is one characteristic that can summarize the contribution of the research in this thesis, it would be this two-way relationship.

To clarify the nature of the relationship, consider Table 2.2: On the left are the supply chain characteristics, and on the right is the structure within blockchain that these characteristics are mapped to.

Table 2.2: Mapping of real-world to supply chain

Supply Chain (Real World)	**Blockchain**
Actors, Supervisors & Transporters (Active entities)	*Blockchain accounts* Blockchain accounts, each with a distinct address that can interact with specific contracts. E.G., An Actor's blockchain account can interact with the Actors smart contract. This access control is defined inside each smart contract
Products, Certifications & Shipments (Passive Entities)	*Present: Smart contract data structures* *History: Historic ledger state inside transactions* A Class (Same as in Object Oriented Programming (OOP)) that is defined inside the smart contract, and the methods that modify the class are also housed inside the smart contract. The actual information instance of each class is stored on the historic blockchain ledger, and changes are appended to the ledger. E.G., The class definition for a product is defined in the smart contract; however, the actual instances of the product classes are stored in the historic state.
Events	*Present: Smart contract functions* *History: Historic Ledger state inside transactions* Events within the real world are performed by active entities on passive entities. The logic of each of these events is contained in smart contract functions, and when a call is made to the function, the outcome (which is where transactional equivalence is established) is appended to the ledger's historic state

CHAPTER 3

METHODS

Figure 2.5 in Section 2.6 presented an envisioned blockchain-based supply chain system. However, to realize this solution, some in-between steps are needed. These steps, or methods, are documented in this chapter. The structure of this chapter is as follows:

Cyber Abstractions will describe the abstractions used to take the physical world into the cyber world to create an integrated cyber-physical supply chain. These abstractions would form the design basis for the development of a prototype.

Smart Contract Design describes the design of smart contracts. The smart contracts will be used to implement the abstractions defined in the preceding section. This section will also detail the formulation of the Transactional Compliance proposition.

Bridging the Gap: Supply Chain to Blockchain will give an overview of the physical supply chain interaction with the blockchain. Even though this is outside the scope of this research, this section introduces what systems would be needed to take the work done in this thesis and transform it into a deployable solution.

Transactional Equivalence introduces the idea of transactional equivalence. Transactional equivalence ensures that all events within the physical supply chain have an equivalent within the cyber realm.

Integrity Ranking introduces integrity ranking. Because of the transactional equivalence described in the preceding section, integrity management implemented within the cyber realm can be enforced in the physical realm.

3.1 Cyber Abstractions

Arguably one of the greatest changes of the 21st century is the incorporation of cyber technologies into people's daily lives. This persistence of technology is so embedded that the combination of the cyber within the physical world results in the creation of cyber-physical systems (Lee, 2008). The combination of these cyber-physical systems and supply chain creates a Cyber-Physical Supply Chain (CPSC) (Ivanov, Sokolov, & Ivanova, 2016). CPSCs are not merely the sum of distinct parts and technologies but the cohesive functioning of those parts to produce a single system.

The persistence of the physical world in the cyber realm necessitates the usage of suitable abstractions. These abstractions must adequately represent the static and dynamic nature of the physical world entities they represent. This section details the abstractions used to persist the physical world into the cyber realm as used in this thesis.

Firstly, it is important to establish a definitional basis. This thesis crosses over the fields of supply chain and computer science. Some of the terminology used in this paper is present in both fields; however, they may mean different things. This thesis uses the following terminology to denote certain entities within the CPSC. These terms are also used to group participants according to their respective roles and abilities within the CPSC:

1. **Products** are all the goods within the supply chain system
2. **Actors** are any of the participants in the supply chain that create or handle goods. In this paper, actors can have four sub-classes:
    - **Producers** are responsible for creating products. An example of a producer would be a farmer.
    - **Distributors** act as midpoints for products passing between locations. Some refer to these as warehouse or distribution nodes.
    - **Processors** use existing products and further process them to produce new products. For example, bakers take in raw baking materials and combine and process

them to produce new output, which can be considered a composite of other products.

- **Retailers** make the products in the supply chain available to the consumer. These can be seen as the terminal nodes of the network.

3. **Transporters** are entities responsible for shipping goods between actors. In this thesis, the transportation medium is not important.

4. **Shipment** is a collection of goods to be transported between two actors via a transporter.

5. **Certifications** authorize an actor and transporter to handle a specific product type. This is a general term for licenses, permits, et cetera from the physical world that are persisted into the CPSC. It is also important to note that the certificate in this paper does not refer to digital certificates issued for network security or domain verification; instead, it is a general term used to denote a cyber abstraction of a real-world certification or license.

6. **Supervisors** This term is a generalization for any market entity that provides certification or has a higher authority than actors and transporters. It is important to note that this term does not refer to a central authority but rather any authority with some supervisory role within the supply chain.

The reader will notice that this research does not dive too deep into the supply chain world beyond the abstractions of the physical into the cyber. The primary purpose of this research is to demonstrate the combination of blockchain with a formalized integrity modeling structure in creating a CPSC. Subsequent research could be devoted to expanding the CPSC to include concepts as advanced as optimization, batching, et cetera.

3.1.1 Data Abstractions and Entity Relationships

Instead of merely capturing identifying information about real-world products within a cyber system, a set of data structures that serve as digital abstractions of the physical

entities should be created. These digital abstractions should represent static and dynamic aspects of the physical supply chain.

Static The static abstractions need to house data and denote the relationships between data abstractions. For example, what is the data needed to abstract an actor, and what is the relationship between an actor's data and a product's data? Static structures also provide a look into the cardinality of relationships. In other words, how many abstractions of one static type are linked to how many abstractions of another static type?

Dynamic The dynamic entities need to show the creation/modification of the data entities from the physical into the cyber realm. For example, what processes would cause a change to the actor's static data, and how should that process be abstracted to the cyber realm? The dynamic parts of the abstractions provide a look at how the static parts are changed, in other words, which process incurs changes to the static abstractions.

This thesis will use Unified Modeling Language (UML) to describe the structure of systems. It should be noted that some modifications will be made to the standard UML diagrams to convey the information best. Appropriate explanations will also be provided for any modifications made.

The first important set of abstractions for creating the CPSC is the static data abstractions denoted in Figure 3.1.

The following paragraphs explain each box in Figure 3.1.

Actor This abstraction represents physical actors within the supply chain. A single actor can own multiple products, denoted by the aggregation relationship between the actor and product abstractions.

Product This abstraction captures the physical product and would include all information needed to make an adequate cyber abstraction of a product uniquely identifiable from all

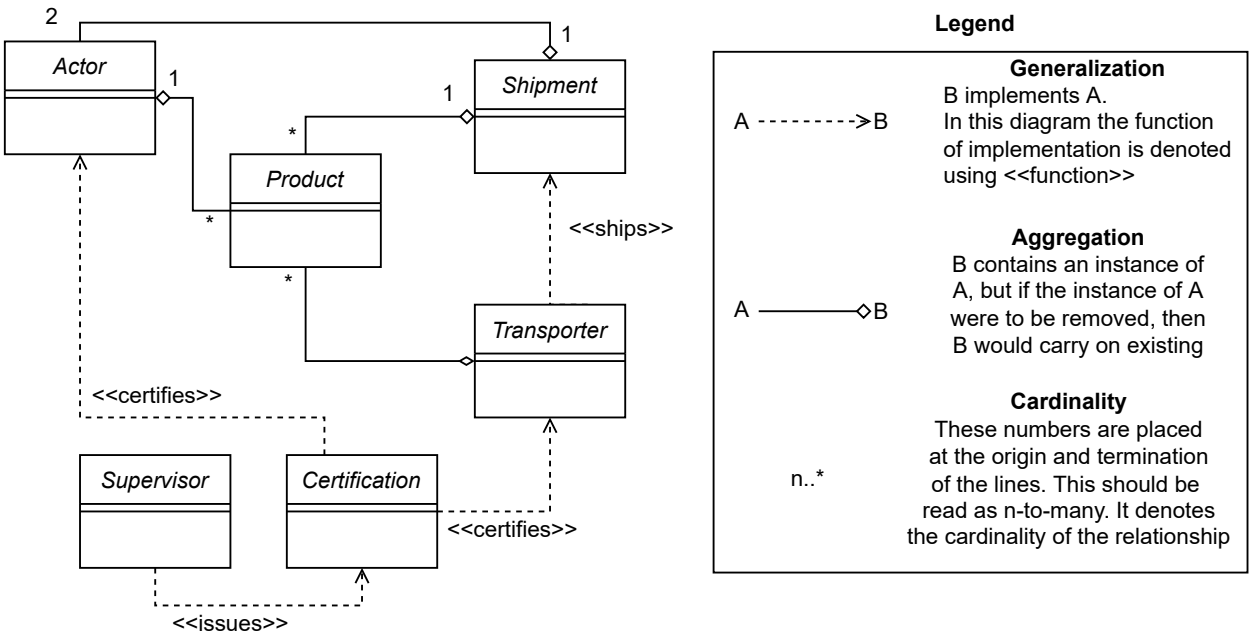


Figure 3.1: Domain diagram denoting the data abstractions

other products. In this thesis, all products are assumed to be distinct single products, and batched groups of products are not considered.

Shipment This abstraction denotes a physical shipment. A shipment will be sent between two actors, hence the aggregation relationship between actor and shipment classes. Each shipment can also contain multiple products, hence the aggregation relationship between shipment and product.

Transporter This abstraction is a general representation of any entity within the physical world that would transport goods between two entities. This thesis does not distinguish between the types of transport. Each transporter can ship multiple shipments, hence the generalization relationship between shipment and transporter. Product ownership is transferred from the shipping actor to the shipment provider during the shipment process, in this case, the transporter. This is why an aggregation relationship exists between the transporter and product classes. This relationship is different from the relationship between a shipment and a product. In that case, a shipment may contain products, but the ownership of the

product does not reside in the shipment itself.

Certification This abstraction is a generalization of any physical world certification issued to an actor or transporter. The abstraction used in this thesis does not care about the type of certification; it is merely a general representation of any certification. A certification can be issued to either an actor or a transporter hence the generalization relationships between the certification abstraction and the actor and transporter abstractions.

Supervisor This is a generalized abstraction of any physical entity in a higher authority position than the actors and transporters. Supervisors could include regulatory bodies outside a single organization or managers within an organization. Supervisors are considered the highest authority entity within this paper and are the only entities capable of creating and issuing certifications, hence the generalization relationship between supervisor and certificate classes.

Figure 3.1 describes the static relationships between entities; however, abstractions of common supply chain processes, the dynamic component, are also needed. Figure 3.2 uses a UML sequence diagram to denote the dynamic relationships between entities. This figure also illustrates the chronology of processes. Certain processes must be completed before subsequent processes can begin.

In this figure, the dashed lines denote an asynchronous request, and the solid lines denote a synchronous request. Asynchronous requests are requests sent without waiting for a response to continue. Synchronous requests, colloquially referred to as blocking requests, halt the execution of logic until a response has been received.

This dynamic depiction in Figure 3.2 captures the events that one would expect within a supply chain to realize the shipment of a product. It includes the creation of the product's parents, the creation of the product itself, the certification process, and the shipment and change of ownership process.

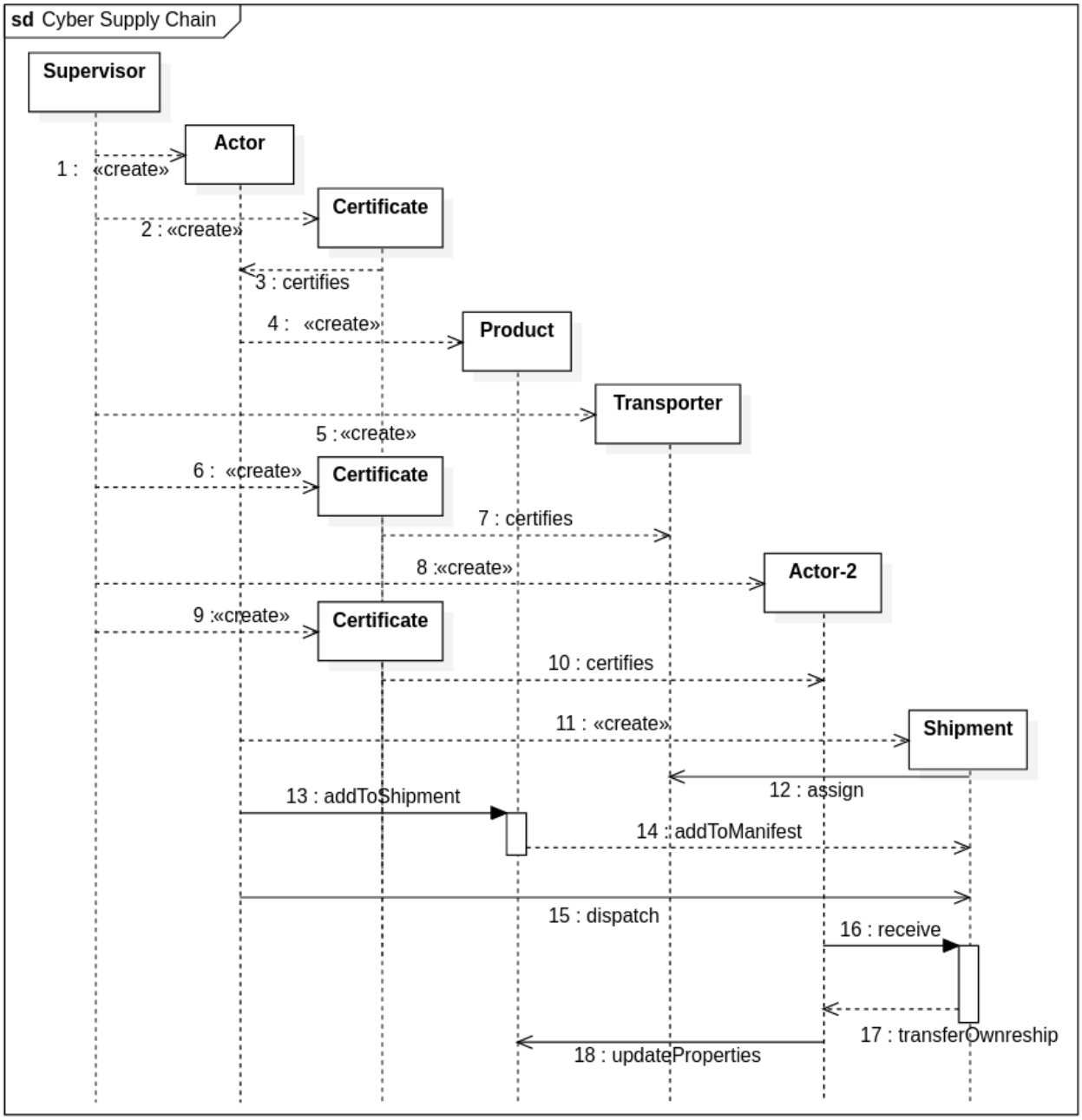


Figure 3.2: Sequence diagram denoting the dynamic aspect of a CPSC digital abstractions.

For ease of reading, the diagram will be explained below using the numbers attached to each arrow. These numbers denote the chronological order and should be read from smallest to largest. The numbered line in the list below denotes the same numbered process in Figure 3.2.

1. First a *Supervisor* will create an *Actor*. This thesis's *Supervisor* is a general abstraction

of a supply chain entity with higher authority. It is assumed that a *Supervisor* is the only entity capable of creating new *Actors* within the CPSC.

2. The newly created *Actor* can have a *Certificate* issued to them, again by a *Supervisor*. This issuance of a *certificate* is not required; however, it can be used to distinguish certain *Products* and *Actors* from others.

3. This *Certificate* will *certify* an *Actor* to work with a particular type of *Product*, as denoted in the *Certificate*. For example, if a *certificate* is required to deal with organic food products, only *Actors* with the relevant *certificate* should be allowed to interact with those *product* abstractions. This functionality is contained in the *certificate* abstraction in this thesis.

4. The *Actor* can then create a *Product*. This mechanism is the spawn-point for a *product* in the CPSC.

5. The only way *Products* can be moved between *Actors* is via a *Transporter*. *Transporters* also have to be created by a *Supervisor*.

6. *Transporters* can also be issued *Certificates*. These *Certificates* also have to be issued by *Supervisors*.

7. Similar to *Actors*, these *Certificates* are used to *certify Transporters* to deal with specific *Products*.

8. In order to move *Products* from one *Actor* to another, another *Actor* has to be created. In this step, a *Supervisor* creates another *actor*.

9. This process is the same as process 2.

10. This process is the same as process 3.

11. In order for *Shipment* to take place, the *Actor* that is the owner of the *Product* to be shipped must create a *Shipment*. This *shipment* will serve as the abstraction to *Products* as they move from one *actor* to another.

12. Once a *Shipment* is created, it is assigned to a *Transporter*.

13. The owner of a *Product* can add a *Product* to a *Shipment*. This is a two-phase approach split across two processes. This first phase is between the *actor* who currently owns the product and the *product*.

14. The second phase of the process started in Event 13 adds a *product* to the manifest of the *shipment*.

15. The owner of the *Product* can then mark a *Shipment* for dispatch, which would close the *Shipment*, preventing any more *Products* from being added to it. This *Shipment* closing is the first part of the ownership transfer of *Products*.

16. The *Shipment* will then travel to a recipient actor.

17. Once a *Shipment* has been received, the recipient *Actor* needs to complete the ownership transfer of *Products*.

18. During this process of ownership transfer, the recipient *actor* is allowed to make certain changes to the properties of the created *Products* they received. For example, one potential property update would be to update the quality of the *product*.

Together, the static and dynamic parts of the abstractions present a well-rounded picture of the actual supply chain from a data perspective. These abstractions will form the basis for the design of a cyber system.

3.2 Smart Contract Design

The abstractions described in the previous section create suitable ways to represent the supply chain in the cyber world. These abstractions will be implemented into a blockchain-based platform.

Some of the constructs in this section are entangled with the blockchain platform used. In this thesis, Ethereum is the platform of choice. As such, many smart contract conventions are very much entwined with the Solidity programming language structure[1].

[1]Solidity is the programming language used to create Ethereum smart contracts

3.2.1 Blockchain Stack

From Chapter 2, the Literature Review, a basic understanding of blockchain, smart contracts, and how they can be used in the Supply Chain world has been created. If the Literature Study can be seen as the answer to the "*what*" question concerning blockchain, this section will attempt to provide clarity around the "*how*".

The first departure point is to look at the technology stack. This will describe the interactions between different subsystems and the depth at which they operate. As mentioned before, the Ethereum blockchain is used in this thesis. Therefore the technology stack review here will reflect the structure of the Ethereum stack. Even though this section will describe the Ethereum case, the reader should note that many of these concepts are directly translatable to other blockchain platforms.

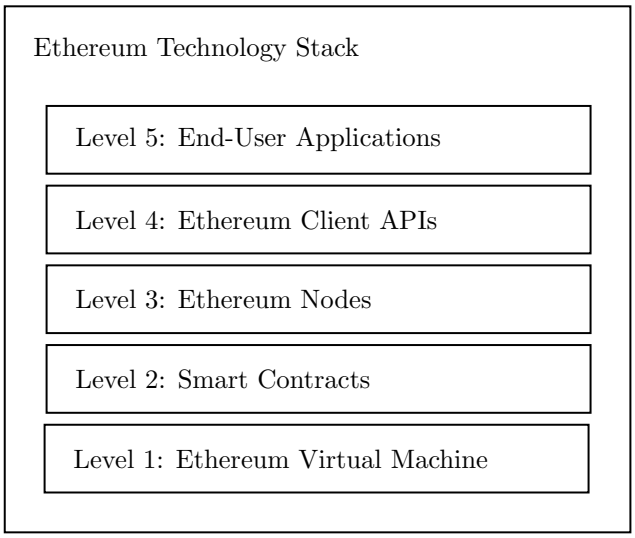


Figure 3.3: Ethereum Technology Stack (*Introduction to the Ethereum stack*, 2022)

Figure 3.3 is the technology stack as defined in Ethereum (*Introduction to the Ethereum stack*, 2022). This diagram should be read from the bottom, Level 1, moving upwards to Level 5.

Level 1: Ethereum Virtual Machine Ethereum creates a global computer called the Ethereum Virtual Machine (EVM). This EVM is responsible for executing transactions, state changes,

and smart contracts. This can be considered the computation layer of the stack.

Level 2: Smart Contracts Smart contracts are the next level up, and these are the logic containers executed by the EVM. This is where the blockchain-specific logic of the system will reside.

Level 3: Ethereum Nodes Ethereum nodes are the access points to the Ethereum blockchain. In order to interact with the blockchain, one of these nodes is needed. The entire Ethereum blockchain comprises a network of nodes that perform the needed blockchain operations and execution of the consensus mechanism. Ethereum nodes are deployed to servers or computers, so interactions with these nodes require a certain level of hardware and software overhead.

Level 4: Ethereum Client APIs Ethereum provides several Application Programming Interface (API)s that can be used to interact with the blockchain instead of needing to deal with the overhead of running an Ethereum node. These APIs have also been integrated as libraries into several common programming languages. Software developers can use these directly without needing to run their own deployments.

Level 5: End-User Applications This is the top level of the Ethereum stack, and it is the level at which most users would interact with blockchain applications.

In this thesis, the bottom two layers of the stack, Levels 1 and 2, are where the focus will be, as these layers are where most of the logic will be contained. Levels 3, 4 & 5 will be used to demonstrate the functioning of the prototype; however, this will be documented in Chapter 4.

3.2.2 Smart Contract Design Conventions

Smart contracts, simply put, are programs that are stored on and executed by the blockchain. As such, smart contracts can be seen as containers of logic. This also implies

that both good and bad logic could be contained within a smart contract. Within the Solidity programming language, many syntactic conventions help minimize the risk of bad logic; however, creating a valid smart contract with bad logic is still possible. Defining a couple of conventions is important to help minimize the risk of persisting bad logic on-chain.

For continuity, this section will be written using the terminology used within Ethereum. However, all conventions used in this subsection can be translated to any programmatic environment. Just as the same concepts can be carried over between different programming languages, many of the blockchain conventions from one platform exist in some format on another platform.

Figure 3.4 provides a high-level overview of the structure of a smart contract. This representation is not exhaustive, containing all the elements of a smart contract, merely the important ones relevant to this research.

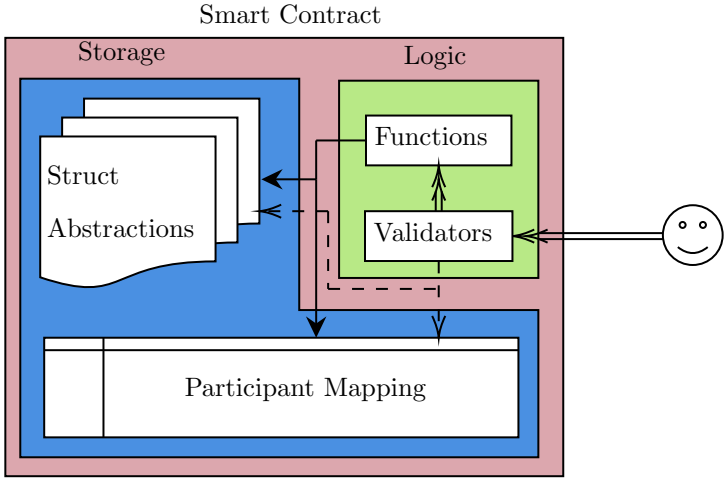


Figure 3.4: Smart contract structure

1. **Storage** denotes the storage of state related to a particular smart contract. Even though all the information is stored on-chain, this smart contract storage groups the smart contract and its associated state together for easier interaction.

   (a) **Struct Abstractions** These data structures are used to represent the static data

and relevant relationships described in Figure 3.1. These data structures are housed within a smart contract of the same type. For example, all the struct abstractions that denote actors are housed within the actor's smart contract.

(b) **Participant Mapping** This mapping structure denotes which blockchain participants can participate in a smart contract. The reader must understand the distinction between these two. A blockchain participant is any entity that has a blockchain account. In this thesis, each participant in the real-world supply chain is assumed to have a single blockchain account within the blockchain system. Since smart contracts in this thesis contain all the logic and storage relevant to each entity abstraction, participant mapping is needed to denote which blockchain accounts can interact with the smart contract. For example, every blockchain account associated with a real-world transporter should be allowed to interact with the transporters' smart contract but not the actors' smart contract.

2. **Logic** This part of the contract stores all the logic associated with the type of smart contract. For example, in the actors' smart contract, all the logic associated with capturing and interacting with actors is contained within this part of the smart contract. The methods stored within this part of the smart contract are used to make changes to the stored state in the storage part of the smart contract.

   (a) **Functions** The function logic is used to make changes to the state storage of each smart contract. Every call to a valid function will result in the generation of a transaction on the blockchain and a subsequent change in the state of the smart contract. For example, to add a new product, the relevant function within the products smart contract needs to be called, resulting in a change to the struct abstractions section of the products smart contract.

   (b) **Validators** Not all interactions with the smart contract require changes to the smart contract's state. Some of the logic is only needed to validate information, such as whether a blockchain participant is an actor or whether a product already

exists. This logic is contained inside validator functions, referred to simply as validators. These functions cannot change the smart contract's state; they can merely observe it. The validators can also be used as guards for the regular function logic. Validators are not to be confused with reference monitors, which will form critical components of the integrity ranking's functionality.

In order to reduce the potential exposure to unwanted state modifications, all entry points into the smart contract from outside should be guarded with validators[2].

3.3 Bridging the Gap: Supply Chain to Blockchain

For the creation of a CPSC to come about, there must exist a bridge between the physical supply chain and the blockchain. This bridge should not be confused with the data abstractions defined in Section 3.1.1. The abstractions defined in Section 3.1.1 allow the real-world supply chain to be represented on the blockchain. The bridge spoken of in this section will allow the real-world actor to interact with the digital actor abstraction.

Figure 3.3 depicted the different technology stack levels. The research done in this thesis resides in the lowest two levels (Levels 1 & 2) of the stack, the ones closest to the technology. The bridge spoken of in this section will reside in the higher Levels 3, 4 & 5. This section looks at the CPSC from the end-user's perspective, in this case, the real-world supply chain actors.

Figure 3.5 provides a high-level overview of the interaction between a supply chain actor (the smiley face) and the broader decentralized blockchain network. This interaction will need to be done through some local supply chain system. This system could be an ERP system, used in the traditional supply chain.

[2]Entry points refer to any programmatic access points (including APIs and CLIs) that would allow a non-blockchain entity to interact with a blockchain smart contract

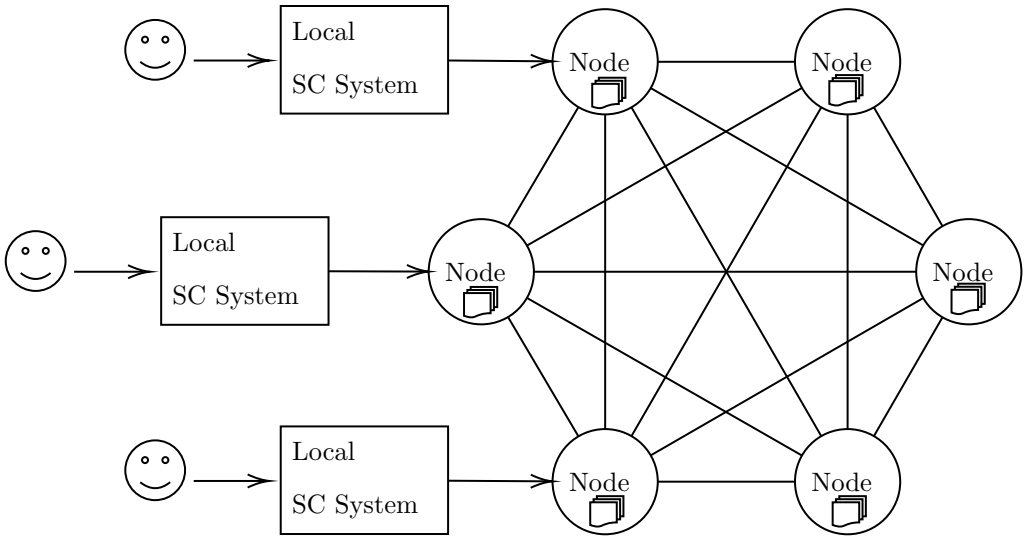


Figure 3.5: High-level Interaction between actors and CPSC

Figure 3.6 provides a more zoomed-in version of Figure 3.5. The physical actor will interact with their local supply chain (SC) system. This local system will then translate the actor's actions into the necessary function calls. These function calls will then be made to the smart contract.

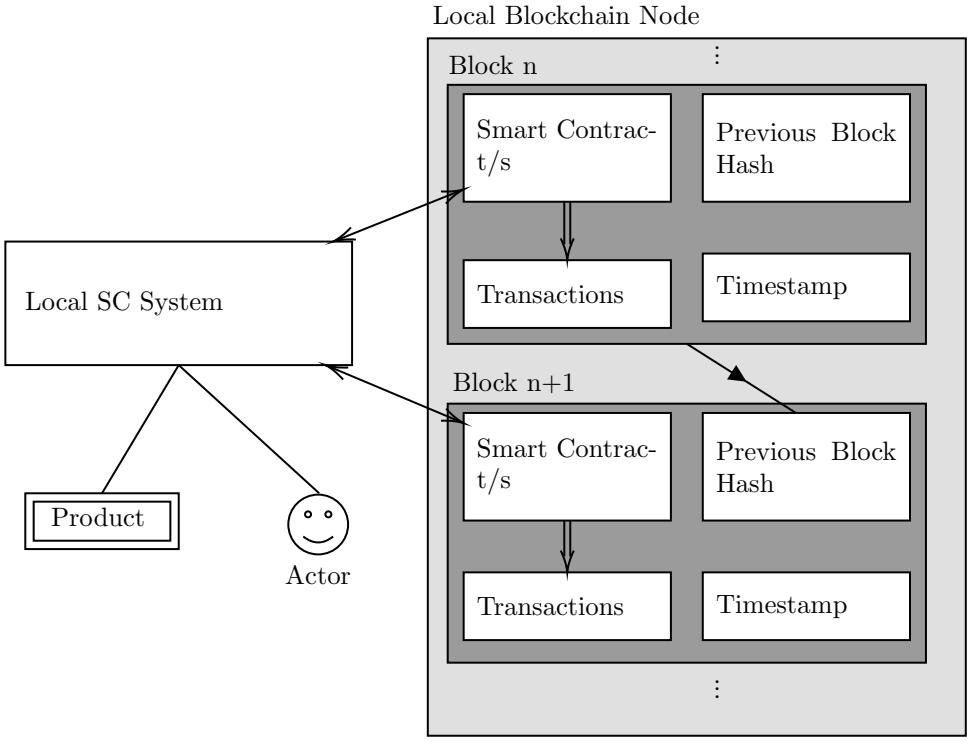


Figure 3.6: Envisioned interaction between an actor and the blockchain via Smart Contracts

What the local SC system needs to look like is outside the scope of this research; however, in theory, this does not necessarily matter. The role of the local SC system is to translate the actor's actions into a format acceptable to the smart contract and provide the necessary information. Whatever other functionality the local SC system may provide is external to the interaction between the local SC system and the blockchain.

Also, both Figures 3.5 & 3.6 depict the blockchain's interaction via a local blockchain node. This need not necessarily be the case. It is possible to use cloud computing to separate the local SC system from the blockchain node and provide the local SC system with a set of tools to access a remote blockchain node. However, this concept's development is outside this thesis's scope.

3.4 Transactional Equivalence

Combining the abstractions defined in this chapter with smart contracts creates a cyber-physical supply chain system that exhibits equivalence. Equivalence here means that for every action within the real-world supply chain, an equivalent action occurs within the blockchain supply chain.

This means there needs to be a way to denote supply chain events within the blockchain world. While the abstractions can denote the structure of the supply chain and provide methods to change that structure, this is not enough to establish equivalence; hence this section introduces what this thesis calls *Transactional Equivalence*.

Blockchain's immutability of state and determinism of execution creates a very conducive environment for enforcing rules at a transactional level. In Ethereum, any change to the state of both smart contracts and the global transaction record is persisted in the ledger as a transaction. This means that anytime a smart contract function changes any stored data, that data change is structured as a transaction. However, with smart contracts, logical constructs coded into the smart contract can be used to govern what constitutes a successful transaction.

Suppose this logic inside the smart contracts is structured as a pre-condition for a

successful transaction of a particular type. In that case, the basis for a pre-conditioned deterministic transaction is created. This is a powerful tool for the research in this thesis because of three reasons:

1. **Determinism** Blockchain executes logic in a deterministic manner; the same logic and inputs will always produce the same outputs. This property also holds for transactions on the blockchain. This is valuable because it means that logic is verifiable, a valuable tool for auditing purposes.

2. **Immutability** Any valid state changes resulting from transactions are persisted in the historic record of the blockchain. This historic record is immutable. Smart contracts use transactions to make state changes, and subsequent state changes resulting from smart contract interactions are immutable.

3. **Pre-conditioning** Smart contracts can be used to pre-condition transactions with logical constructs. These logical constructs can represent any business logic and are coded using the programming language of the blockchain.

It is possible to code rules into a smart contract that must be met before a transaction occurs. These rules would mirror real-world supply chain rules. For example, suppose a Supplier of meat products requires a certification to handle those meat products. This requirement can be enforced using a smart contract when the supplier tries to create a blockchain version of their meat product.

In this case, the supply chain event of a product creation will be captured within the ledger as a transaction only if the supplier possesses a certification, thereby establishing that a transactional equivalent of a real-world supply chain exists. Furthermore, because of the determinism within smart contracts, by implication, the process of applying rules to ensure only valid transaction equivalents are created can be said to be deterministic as well.

Creating a transactional equivalent also holds value if it does not occur. For example, suppose that the supplier mentioned in the previous paragraph bypasses the blockchain supply chain system by selling their meat products to others within the real world, without

having the correct certification and without going through the blockchain system. Any actor further down the supply chain can query the blockchain for the transaction that would have established a blockchain equivalent. This query will naturally fail because no equivalent will be found.

The implications can be far-ranging; however, one such implication could be legal requirements. For example, suppose that a vendor wishes to purchase goods from a supplier. The goods purchased must be processed according to particular guidelines, meaning the supplier would require certification. The vendor does not want to go through the trouble of confirming that the supplier has the certification and that each product was created while the certification was still valid. Instead, the vendor can, through a legal agreement with the supplier, require the supplier to ensure that transactional equivalents of their products exist. Also, because these transactional equivalents exist within the blockchain, the vendor can independently audit the transactions that created the equivalents. The vendor could even programmatically verify this by running a blockchain query every time a new shipment is received.

One of the other implications of establishing transactional equivalence is that smart contracts now become reference monitors. This is an important implication because it allows for integrity management, which is enforced through a reference monitor. Section 3.5 will describe integrity management and how it can be used in the supply chain.

3.5 Integrity Ranking

The Biba integrity model is a widely accepted model used in the world of computer science to manage and govern integrity. The Biba model categorizes entities into two cohorts: subjects and objects. Subjects are considered active entities that can make calls to processes and induce data flow. Objects are passive entities handled by subjects. Consider the example of an office. There are workers in the office (subjects) that handle documents (objects). To manage the integrity flow between subjects and objects, Biba assigns an integrity level to each object and subject.

Three key rules or axioms define these integrity levels and the interaction between subjects:

1. **Simple integrity property** A subject with a certain integrity level must not read data at a lower integrity level
2. **Star integrity property** A subject with a certain integrity level must not write or contribute to data with a higher integrity
3. **Invocation property** A subject cannot use a process to request data of a higher integrity level

The way that integrity management can be incorporated into a blockchain supply chain system is two-fold. The first is the progression of the integrity of entities, and the second is the nature of the interactions between entities.

3.5.1 Supply Chain Progression of Integrity

Participants and goods can also be abstracted within the supply chain as subjects and objects. Subjects are any participants in the supply chain with an active role: actors, supervisors, transporters, et cetera. Objects are goods and passive structures such as certifications and products. Each of these can be given an integrity level. As the objects move between subjects, the integrity rules could be enforced, and the integrity of objects could be further updated.

The progression of integrity is perhaps best explained using Figure 3.7. A simple supply chain example is considered in this figure, the same example depicted in Figure 3.2. Each circle denotes the integrity level at that given time and the action. The example starts with a supervisor of Integrity level 10 creating an actor of integrity level 8. The integrity here is depicted as a distinct integer value on a 10-point scale; however, it can also be modeled using a continuous real value.

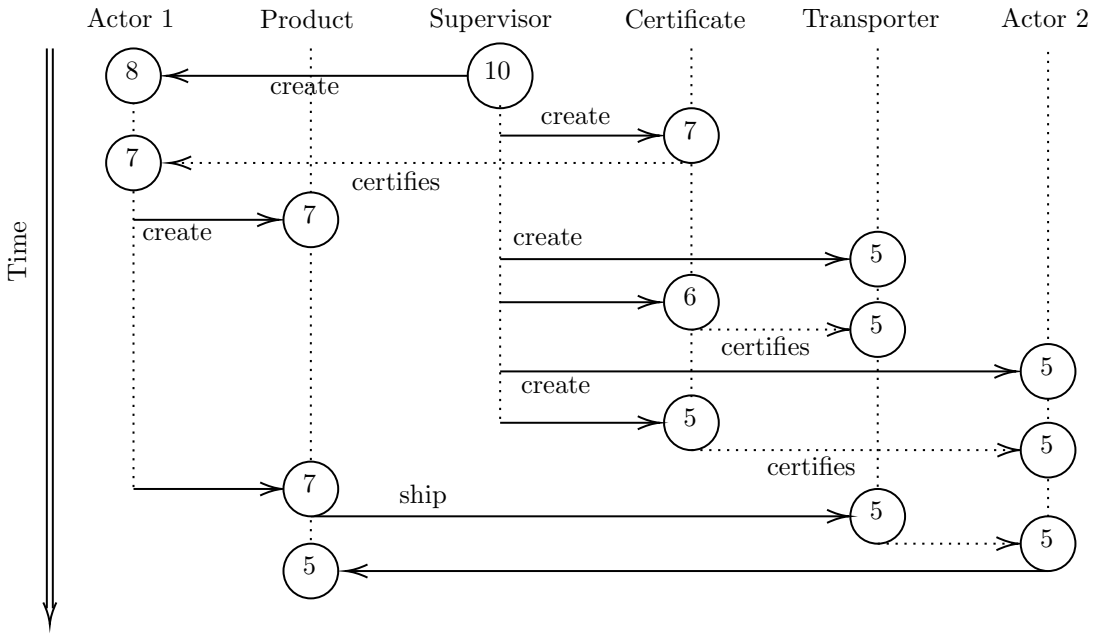


Figure 3.7: Progression of integrity through supply chain

Instead of depicting subjects or objects as having distinct integrity levels, it is possible to design the integrity as a ranking. Each entity will always be assigned the same integrity level with a distinct rule. For example, all actors would have an integrity level of 5, and all products an integrity level of 4. By depicting the integrity level as a ranking, the emphasis is not on the integrity value itself but on the ranking relevant to another entity. A higher ranking would correspond to a higher integrity value and vice versa.

The rule of integrity is enforced relative to the ranking of the entity and not just the entity's assigned integrity level. As an example, consider a product and an actor. One would expect a product's integrity always to be less than or equal to its parent actor's since an actor cannot produce a product higher than their quality. Table 3.1 provides a lookup table of the ranking rules of the entities within Rankblock. Each entity has an integrity value that is associated with an operator rule. For example, $i = 1$ & $j = 2$[3] corresponds to the ranking between the integrity of a product, p and a certificate, c. A product should always have integrity that is less than or equal to the integrity of the related certificate.

[3] i corresponds to rows and j to columns

Table 3.1: Ranking of Entities

Entity	i, j	Integrity Variable	**Product**	**Certificate**	**Actor**	**Supervisor**	**Transporter**	**Shipment**
			1	2	3	4	5	6
			p	c	a	su	t	s
Product	1	p	URS	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$
Certificate	2	c	$\geq$	URS	$\geq$	$\leq$	$\geq$	$\geq$
Actor	3	a	$\geq$	$\leq$	URS	$\leq$	$\geq$	$\geq$
Supervisor	4	su	$\geq$	$\geq$	$\geq$	URS	$\geq$	$\geq$
Transporter	5	t	$\geq$	$\leq$	$\geq$	$\leq$	URS	$\geq$
Shipment	6	s	$\geq$	$\leq$	$\leq$	$\leq$	$\leq$	URS

3.5.2 Entity Interactions

The second part of integrity management is the nature of the transactions themselves. In the Biba model, the interactions between entities are grouped under one of three terms:

1. **Observe** (o) The action of viewing information. Observation also includes the idea that the observer's state may be changed due to the observation action.

2. **Modify** (m) Changing an object so that the change is discernible through observation.

3. **Invoke** (I) Request from one subject to another to perform either observation or modification operation.

The prototype in this thesis is constructed using smart contracts which house data abstractions. These smart contracts perform modifications to these data abstractions through transactions. It is therefore important that the nature of these transactions between smart contracts and data abstractions is also mapped out and considered in the design of the system prototype. Table 3.2 shows these interactions. In this table, "o" denotes observe, and "m" denotes modify. The first part of the table shows the interaction between smart contracts (subjects) and the data abstractions (objects). The second part shows the invocations be-

tween smart contracts. Smart contracts may need to make modifications or observations/of data stored within other smart contracts.

The contents of Table 3.2 are depicted in a different format in Figure 3.8.

Table 3.2: Subject and object interactions

		Subjects					
		Actors SC	**Products SC**	**Shipments SC**	**Transporters SC**	**Certificates SC**	**Supervisors SC**
Objects	**Actor**	E	o	o	o	o	-
	Actor Stock	E	o, m	o, m	-	-	-
	Product	-	E	o, m	-	-	-
	Shipment	-	-	E	-	-	-
	Transporter	-	-	o	E	o	-
	Certificate	-	o	o	-	E	-
	Supervisor	o, m	-	-	o, m	o, m	E

		Invocations					
Subjects	**Actors SC**	-	-	-	-	-	o
	Products SC	o: Actor; o, m: Actor Stock	-	-	-	o	-
	Shipments SC	o: Actor; o, m: Actor Stock	o, m	-	o	o	-
	Transporters SC	o: Actor	-	-	-	-	o
	Certificates SC	o: Actor	-	-	o	-	o

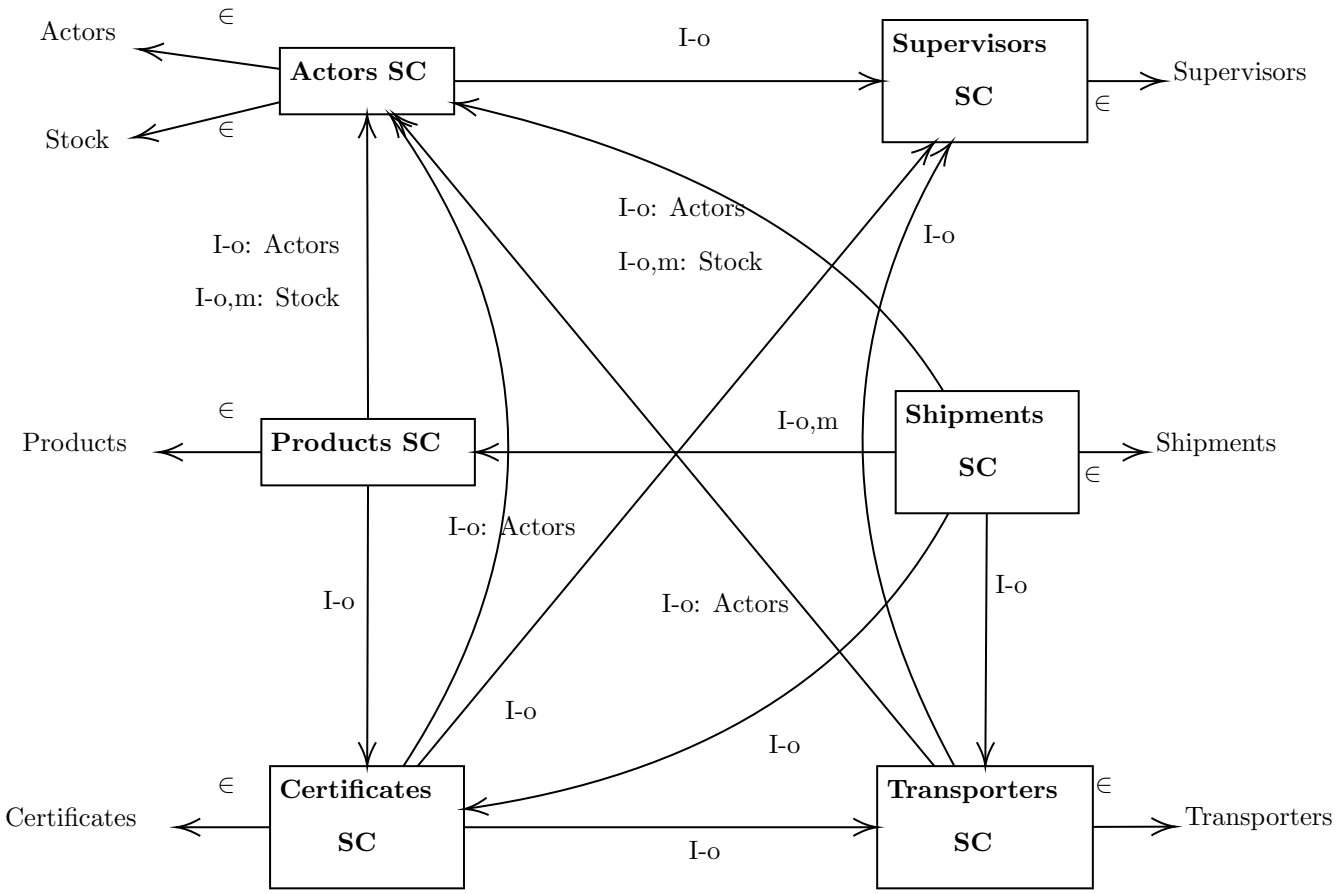


Figure 3.8: Integrity interactions between smart contracts and data structures

Conclusion

The methods described in this chapter can be used to construct a prototype in-line with the proposed solution in Figure 2.5.

The first step towards the creation of this system is through cyber abstractions. These abstractions are provided in Section 3.1.1. These abstractions must represent the supply chain's static and dynamic components to create an effective cyber version. The static abstractions denote how data is structured and the relationships between the different data structures. The dynamic components show how the data and data structures are modified or accessed. Together these abstractions capture the supply chain.

The abstractions from Section 3.1.1 will be implemented into smart contracts. Section 3.2 gives the conventions used to design these smart contracts. The chosen platform for the

creation of smart contracts in this thesis is Ethereum; hence the design conventions given in Section 3.2 are given in terms of Ethereum terminology. However, this thesis does not aim to present a platform-specific solution; the concepts defined in this section are translatable to any other blockchain, albeit other blockchains might have different terminology.

Up to this point, all the methods would allow for creating a blockchain supply chain. However, this thesis examines how integrity ranking can be incorporated into the CPSC. Section 3.4 provides the foundation for this implementation by describing what this thesis calls Transactional Equivalence. Transactional equivalence means that every real-world supply chain event could have an equivalent in the blockchain world in the form of a transaction produced by a smart contract.

Smart contracts become reference monitors by enforcing rules within a smart contract to control how transaction equivalents are created. This lays the foundation for the implementation of Integrity Ranking. In Section 3.5, the idea of integrity ranking is described in detail. This section shows how integrity ranking can abstract certain supply chain information into a quantifiable metric. This metric provides insight into the integrity of products. For example, if goods need to undergo a certain production process to comply with a set of standards, goods that successfully undergo this step can be ranked higher than goods that never undergo this step.

Transactional equivalence allows the real-world supply chain to be mirrored in the cyber realm through blockchain. Controlling how these equivalents can be created through smart contracts allows smart contracts to become reference monitors. With the introduction of integrity management to the cyber realm, smart contracts can then serve as reference monitors to enforce integrity management rules within the physical world.

CHAPTER 4

CASE STUDY & ANALYSIS

4.1 Introduction

The methods described in the previous chapter are used to construct two prototypes in this section. The prototype, called the *Simple Prototype*, demonstrates the creation of a blockchain-based CPSC system. This prototype, however, contains no integrity management. The second prototype will then expand on the prototype and add integrity management. This second prototype, is the prototype referred to as Rankblock.

Testing of both prototypes will be performed using an automated script-based case-study. This case study will emulate the functionality described in Figure 3.2 on page 34. Results discussed in each prototype section will be taken from a local blockchain environment, and this local blockchain environment is a copy of the public blockchain's functionality.

The structure of this chapter is as follows.

Simple Prototype describes in detail the design of the simple version of the CPSC prototype. This section will present design artifacts and results at the hand of a script-based case study.

Advanced Prototype describes the expansion of the simple prototype to include integrity management. This section also uses a script-based case study to illustrate the prototype's functionality.

4.2 Simple Prototype

4.2.1 Toolstack

Several software and environment tools were used to build the prototypes described in this chapter. These tools are provided in this subsection.

Solidity

Solidity is one of the most popular smart contract development language (Mix, 2019), and it is the smart contract language used to develop Ethereum-based smart contracts. Solidity is a high-level statically typed programming language used to interact with the Ethereum Virtual Machine (EVM) (ethereum, 2022).

Truffle

Truffle[1] is a set of tools for creating Ethereum smart contracts. These tools allow developers to code, test, debug and deploy smart contracts using one toolbox.

Ganache

Ganache[2] is a tool that allows the user to create a local version of the Ethereum blockchain. This tool provides a one-click way of creating an offline blockchain copy that uses the same logic as the public Ethereum chain. This allows developers to develop and test applications within a digital twin of the Ethereum blockchain.

Nodejs

NodeJS[3] is a Javascript development environment for creating Javascript applications. In this chapter, the case studies used to test the prototypes are coded using NodeJS.

Surya

Surya[4] is a collection of auditing tools for Solidity smart contracts. Many of the figures in this section will be programmatically generated from the codebase using the Surya toolbox. This tool is also used extensively during the development of smart contracts to ensure that the coded smart contracts match the designs presented in Chapter 3.

[1] https://trufflesuite.com/truffle/
[2] https://trufflesuite.com/ganache/
[3] https://nodejs.org/en/
[4] https://github.com/ConsenSys/surya

4.2.2 UML Overview

Figure 4.1 presents a UML depiction of the prototype created in this section. This figure should be compared to 3.1. This figure shows how the relationships from Figure 3.1 were preserved in the creation of the prototype. The following paragraphs will describe each class working from the top of the figure downwards. Appendix A provides a more detailed picture of each smart contract within each prototype.

Actors depicts the actors' smart contract. This smart contract is responsible for storing actor information and the stock of products each actor holds. Methods in this contract are used to create actors, verify actors and actor types, and add products to an actors inventory.

Each actor class can:

1. Own many products in their stock
2. Be certified to deal in a specific product by a single certificate
3. Be either a shipper or consignee of a shipment

The certification smart contract can also issue actors certificates; this is indicated by the 1-to-1 *certifies* relationship depicted in Figure 4.1.

Products The products smart contract is responsible for product creation, storage, update, and shipment allocation. This smart contract does not store the stock held by each actor. This smart contract does not store the stock held by actors, instead that is stored in the Actors smart contract.

Shipments This smart contract captures shipments. This smart contract is also used to create, dispatch, receive and update shipments. Shipments can only exist between two actors hence the 2-to-1 rule shown in Figure 4.1. Shipments contain multiple products, hence the many-to-1 relationship with the products smart contract.

Transporters The transporters smart contract captures the real-world supply chain transporters. This contract provides methods to create a transporter, as well as to verify the transporter's existence. A single transporter can ship multiple shipments, denoted in Figure 4.1 with a 1-to-many relationship.

When a transporter ships products as part of a shipment, custody of the products is transferred to the transporter for the duration of the shipment. Once the recipient receives goods, ownership of the goods is transferred to the actor. Transporters can also be issued certifications by the certifications smart contract, and these certifications allow transporters to transport goods of a particular type.

Supervisors The supervisors smart contract captures the supply chain supervisors. This is a generalized role used in this thesis to depict supply chain entities with higher authority, which could be regulatory bodies, port authorities, et cetera.

In this thesis, only supervisors are allowed to issue certificates to either actors or transporters; this is denoted as a 1-to-many relationship in Figure 4.1.

Certifications The certifications smart contract is responsible for storing supply chain certifications issued to actors and transporters by supervisors. This smart contract provides methods to create certifications and methods to query the status/validity of the certification.

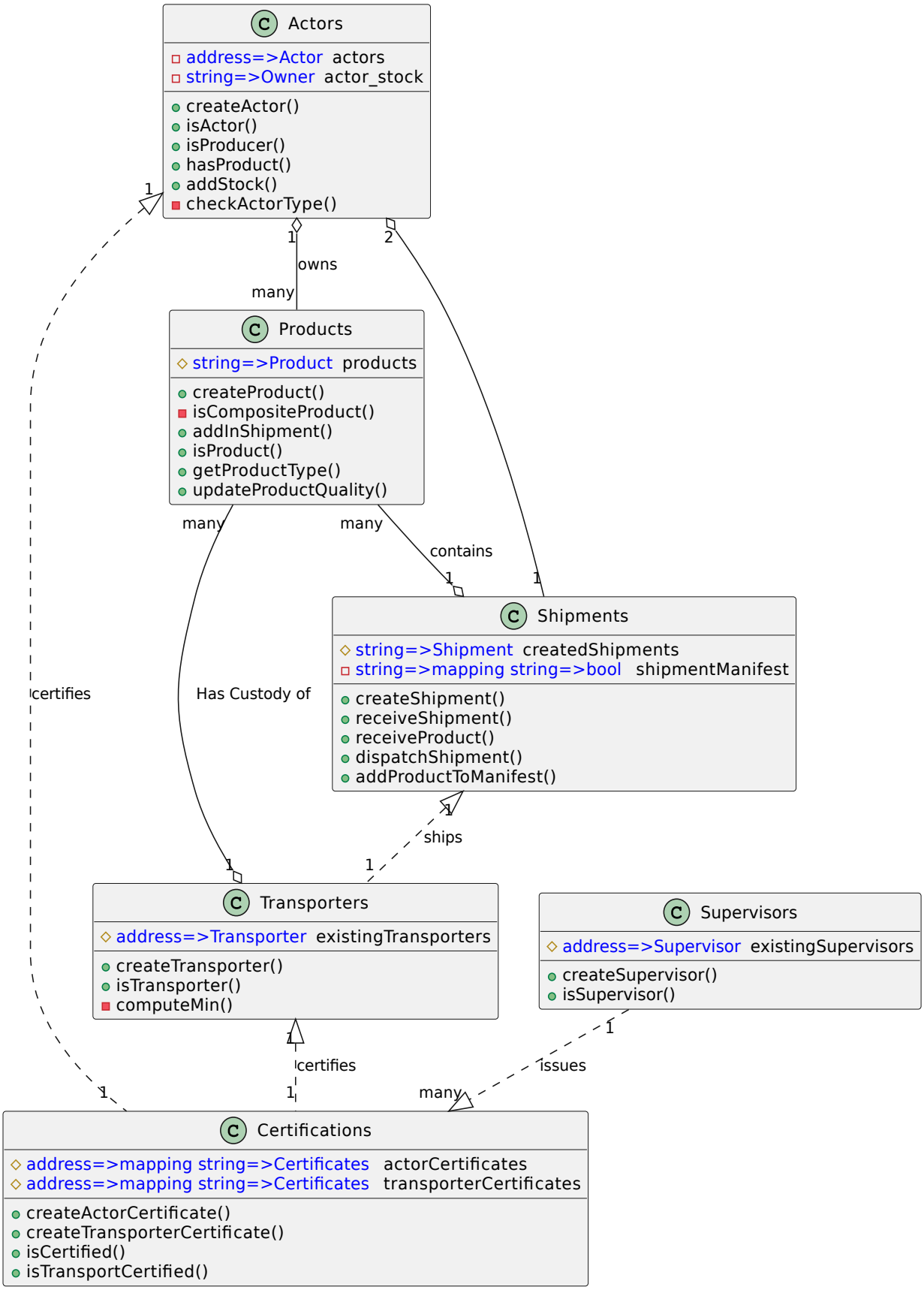


Figure 4.1: UML Overview of the Simple Prototype CPSC

4.2.3 Automated Case Study

In order to demonstrate the functionality of the simple prototype, a simulator script was created using NodeJS. This simulator script allows for a complex case study to be coded, and when executed, the script will perform the steps specified. Figure 4.2 provides a flowchart of the steps coded into the simulator script.

The case study chosen is that of a simple shipment process. An actor, Actor 1, will create a product that requires certification to handle. Actor 1 will ship this product to Actor 2 via a transporter. Throughout this process, a supervisor will issue the actors and the transporter with the required certification to handle the products.

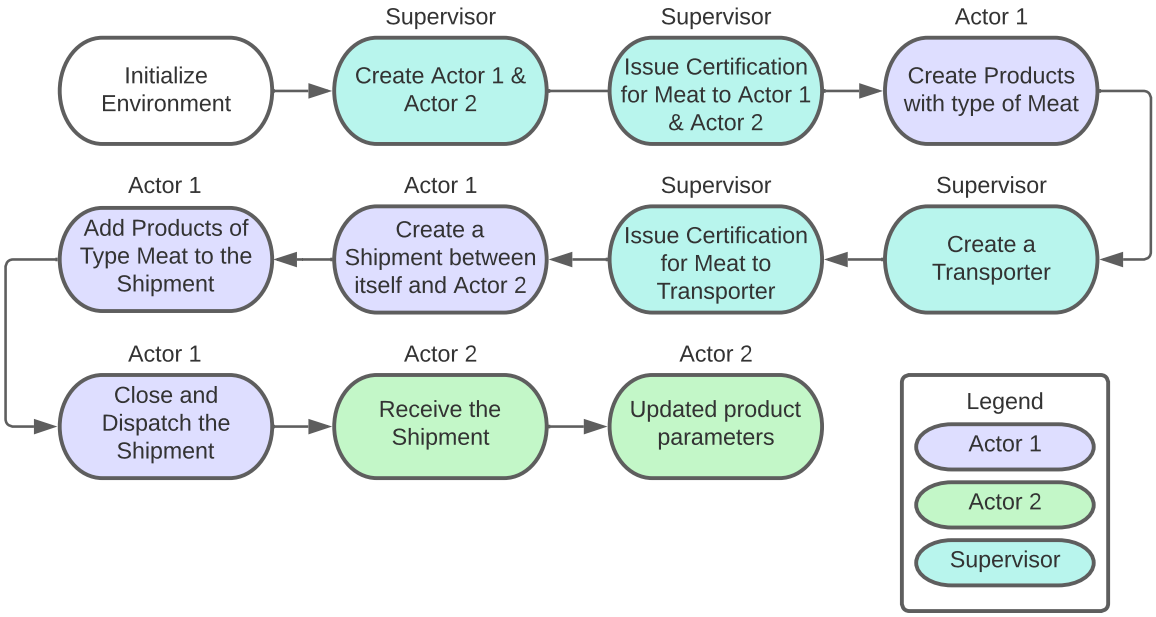


Figure 4.2: Flowchart of Simulated Case-Study events

The simulator script starts by initializing the simulated environment. This step involves the creation of four blockchain participants. These participants each will have their blockchain address and represent four supply chain entities:

1. *Actor 1* - Denotes the actor that creates the product to be shipped.
2. *Actor 2* - Denotes the actor that will receive the product as a shipment.

3. *Supervisor* - Denotes the supervisor that will serve as the higher authority in this case study.

4. *Transporter* - Denotes the transporter that will ship the goods from one actor to another.

Once initialized, the simulator script will follow the sequence of steps depicted in Figure 4.2. Additionally, the simulator script will use the appropriate blockchain participant created during initialization to ensure that the function calls are made from the correct entity. For example, the product creation call in the fourth step will come from Actor 1. This is also denoted in Figure 4.2 using color.

4.2.4 Results

The results presented in this section are from queries run on the local test blockchain. These queries return data from certain blockchain events. Each of these events is equivalent to real-world events described in Figure 4.2. For the sake of brevity, only three events are shown in this section.

Product Creation

The first event shown in Listing 4.1 is product creation. Within the real world, an actor would produce a product, and once produced, the product will be added to the blockchain through the product creation process. This would establish a transactional equivalent. Listing 4.1 presents the results of a query run on a test blockchain for the transactional equivalent after the simulator script called the relevant function in the smart contract.

Line 3 provides the hash of the block in which this transaction is contained, and Line 7 provides the hash of the transaction from which this event is taken. Lines 8 through 19 provide the data for this transaction. They show that the actor (line 14) created a product called *Beef Steak* (line 9) that is identified within the blockchain system by a UUID (line 10) and in the real world by the identifiers given in line 12.

```
1 {
2   "address": "0xc409604F93692712161a71C683e5446591224eCc",
```

```
3   "blockHash":
        ↪ "0x13449722f388975e6ddd2559457a99c559c3554c5d9ae6c3612d53ba25b1bf7f",
4   "blockNumber": 12,
5   "logIndex": 0,
6   "removed": false,
7   "transactionHash":
        ↪ "0x8e324d207ce813e16416653d2f9141540e804fb420025b402c111ab688563218",
8   "returnValues": {
9     "_prod_name": "Beef␣Steak",
10    "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
11    "_prod_quality": "100",
12    "_sc_identifiers": "{\"UPC\":\"KS93528TUT\",␣\"UPC\":\"480528304523\"}",
13    "_product_type": "Meat",
14    "_creator": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
15    "_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
16    "_parents": [
17      "Not␣Composite"
18    ]
19  },
20  "event": "AcceptProductCreate",
21  "signature":
        ↪ "0xa0e07600f7398d5d6a813f35f2ac09fcabe0230f5209bc32fcd52495349f6723",
22  "raw": {
23    "data": "0x000...c73...000",
24    "topics": [
25  "0xa0e07600f7398d5d6a813f35f2ac09fcabe0230f5209bc32fcd52495349f6723"
26    ]
27  }
28 }
```

Listing 4.1: Result of blockchain query for the product creation event

Product Owner Update

Listing 4.2 provides the result of a blockchain query to look for changes in the ownership of a product. Two such changes are expected in the test case specified in Figure 4.2. The first is at product creation, to assign ownership of the product to its creator, and the second is at the end of the shipment process when the recipient takes ownership of the product. The former of these two is shown in Figure 4.2.

The product in question has been created without an assigned old owner (line 12). The actor responsible for creating the product is assigned ownership in line 13.

```
1  {
2     address: '0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72',
3     blockHash:
          ↪ '0x13449722f388975e6ddd2559457a99c559c3554c5d9ae6c3612d53ba25b1bf7f',
4     blockNumber: 12,
5     logIndex: 1,
6     removed: false,
```

```
    transactionHash:
        ↪ '0x8e324d207ce813e16416653d2f9141540e804fb420025b402c111ab688563218',
    transactionIndex: 0,
    id: 'log_e47962d6',
    returnValues: Result {
      _prod_uuid: 'e9838dc8-3cf7-4685-97fe-027777199314',
      _old_owner: '0x0000000000000000000000000000000000000000',
      _new_owner: '0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84',
      _message: 'Product owner updated'
    },
    event: 'AddProductStock',
    signature:
        ↪ '0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39',
    raw: {
      data: '0x000...766...000',
      topics: [Array]
    }
  }
```

Listing 4.2: Result of blockchain query for the product owner update event

Shipment Creation

Listing 4.3 provides the result of a query for shipment events related to a specific product. In this case study, one such event is expected to have occurred, and this event is shown in the Listing.

A shipment (lines 11 & 12) is created from Actor 1 (line 13) destined for Actor 2 (Line 14) to be transported by the Transporter (line 15).

```
{
    address: '0x6ce9A62bfb9a8aF65af466D15bA6129f8B3e8c12',
    blockHash:
        ↪ '0x0cc8f82c5233f17bce28ca1d37c2caadcb198e254dce8c2a9c3aff1a7712e136',
    blockNumber: 19,
    logIndex: 0,
    removed: false,
    transactionHash:
        ↪ '0xc6476dbc54a63e8f870170df96bd5f89253d30822b0d34a68acb8496b88af5b6',
    transactionIndex: 0,
    id: 'log_e6a0e5ed',
    returnValues: Result {
      _ship_uuid: 'Shipment UUID-123123',
      _ship_identifier: 'Shipment waybill etc',
      _ship_source: '0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84',
      _ship_destination: '0xa4d7033062C7DCc6676C09961fBD6638c92F40F8',
      _ship_provider: '0xE25Fe8C7c12ef26959876E13E4A05FA5D89E2894',
      _message: 'New shipment created and opened'
    },
    event: 'AcceptShipmentCreation',
    signature:
        ↪ '0xd8af7e1c325e02cad0aa736ec7fcbc4c0af061bad28ecc7aff7fb881b0b3ec8f',
    raw: {
```

```
21        data: '0x000...12e...400',
22        topics: [Array]
23      }
24    }
```

Listing 4.3: Result of blockchain query for the shipment creation event

4.2.5 Discussion

In the first result provided in Listing 4.1, a transactional equivalent of product creation is shown. This product creation event is accompanied by an ownership update event shown in Listing 4.2. It is expected to show that a product was created by Actor 1 (Listing 4.1: Line 14) and transferred into their ownership (Listing 4.2: Line 13). The matching address in both of these Listing confirms the expectation.

Actor 1 will create a shipment between themselves and Actor 2 to send the product, shown in Listing 4.3. Only the owner of the product can create a shipment, and in Listing 4.3: Line 13, we see that it is the same address as in Listing 4.1: Line 14 and Listing 4.2: Line 13. This confirms that the actor who created a product and was assigned ownership is the same actor who shipped the product.

4.3 Advanced Prototype

4.3.1 UML Overview

This subsection presents a UML overview of the more advanced prototype CPSC system. This section is an extended version of Figure 4.1 in Subsection 4.2.2. For the sake of brevity, only the expanded parts of this figure, namely the integrity management component, will be discussed here. Appendix A provides a more detailed picture of each smart contract within each prototype.

Actors When an actor is created, the initial integrity value of an actor is set by the supervisor that calls the `createActor()`. The integrity value of the actor will be set to either the supplied value or the supervisor's integrity value, whichever is lowest. This is in accordance with one of the rules of integrity management that does not permit a lower integrity entity

from creating a higher integrity entity.

Products When products are created, the actor calling the `createProduct()` method can supply an integrity value for the product. The smart contract will then retrieve the integrity value of the actor, the supplied value, and the integrity value of the certificate authorizing the actor to create the product type if required. The lowest between these three values is given to the product.

For example: Suppose an actor has an integrity value of 7. This actor is issued a certificate to deal in a specific product type called $type-x$; however, a supervisor has deemed this actor's ability to process the product type $type - x$ to only be at an integrity level of 5. This actor, even though their integrity level is 7, is then issued a certificate for handling a product type $type - x$ with a level of 5. This means that any product that is not of type $type - x$ created by this actor will be created with an integrity level of up to 7; however, any product of $type - x$ can, at maximum, have an integrity level of 5.

Shipments A shipment itself is assigned an integrity value when created. When products are added to a shipment, their integrity values are updated based on the integrity value of the transporter and any certifications issued to that transporter and the shipment. For example, If a product has an integrity value of 7, but the transporter that will be shipping the goods has an integrity value of 5 and a certification of integrity value of 4, then at the point in which the goods cross over into the custody of the transporter, the integrity of the products matching the certification will be updated from 7 to 4 and all other products to level 5. This update is automatically enforced by the smart contract and added to the historic state of the ledger.

Transporters Transporters will have an integrity value assigned during the `createTransporter()` process. This will follow the same rules as previously described in the `createActor()` method.

Supervisors Supervisors, when created, will also be assigned an integrity value. When the prototype system is activated for the first time, a proverbial *chicken-before-the-egg* situation is encountered. I.E., How are supervisors created if the supervisor is the highest authority within this system?

For simplicity, in this research, it is assumed that supervisors are the only participants within the supply chain that can create other supervisors. A *Master Supervisor* is also hard-coded into the Smart Contract. This *Master Supervisor* is assigned the highest integrity value possible and is, therefore, able to create any number of supervisors at any specified integrity level.

Certifications Certifications are assigned an integrity value during their creation either in the `createActorCertificate()` or `createTransporterCertificate()` method.

4.3.2 Automated Case Study

In order to demonstrate the Advanced Prototype, the same simulator script used in Subsection 4.2.3 is used. The difference is that integrity is added to the mix. In order to denote the integrity values in the test case, a shorthand was used. For example, take the second bubble from the top left. In this bubble, the supervisor creates two actors. The supervisors has an integrity level of 10 and is therefore denoted using the shorthand *Supervisor-10*.

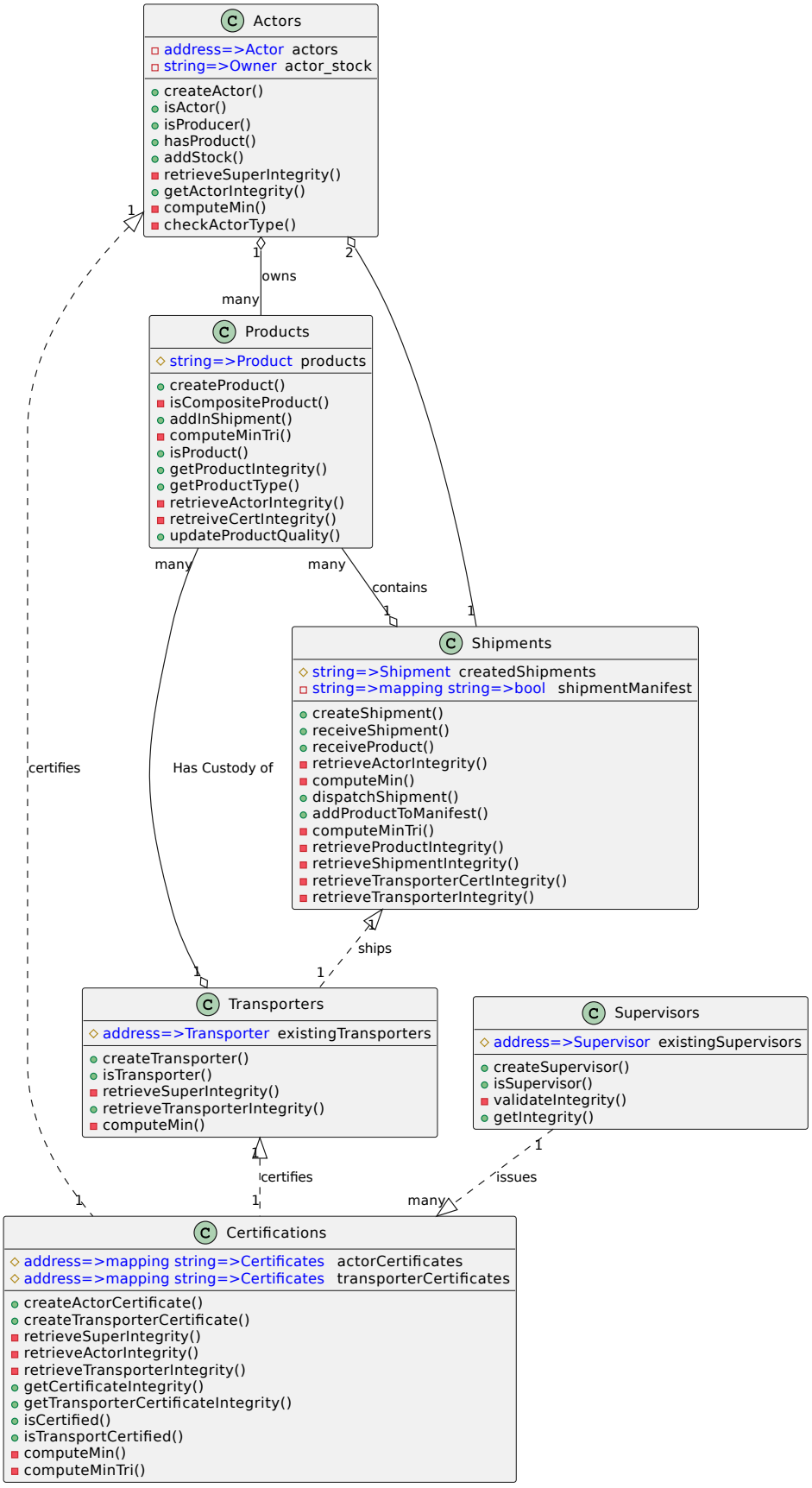


Figure 4.4: UML Overview of the Advanced Prototype CPSC

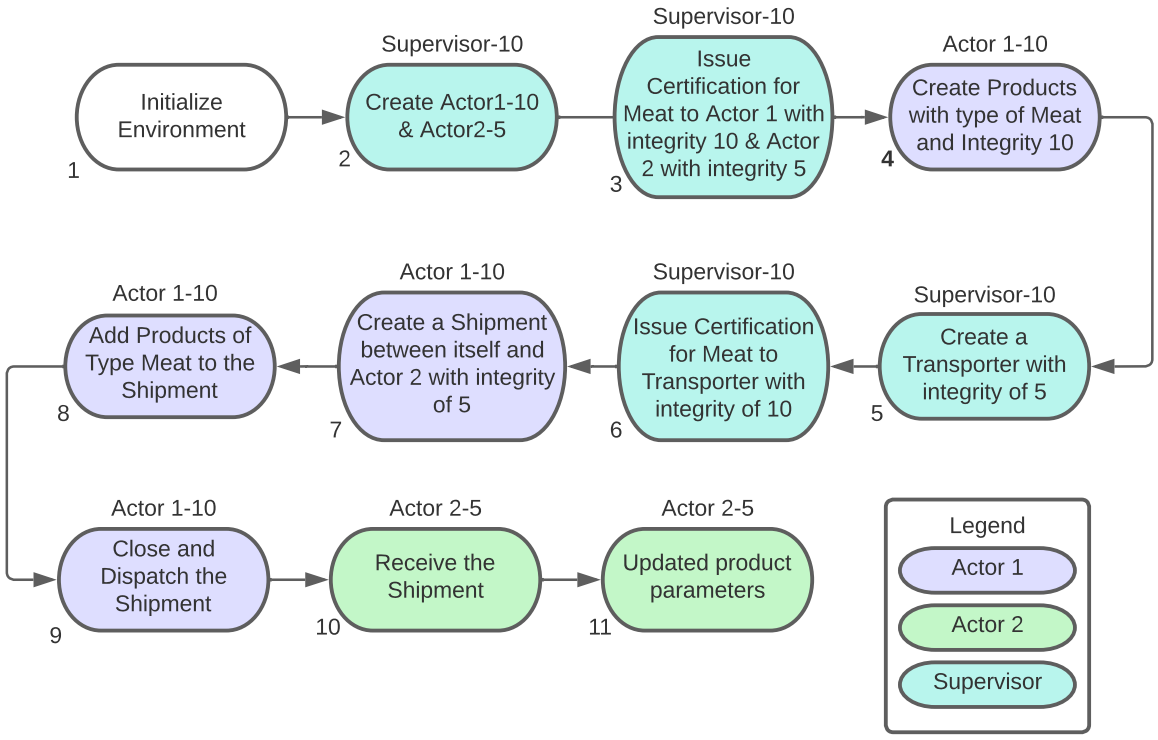


Figure 4.3: Flowchart of Simulated Case-Study events in the Advanced Prototype

4.3.3 Results

As with the results presented in 4.2.4, this section will present results in the form of queries run on a local blockchain testing environment. In Section 4.2.4, the results were given for three specific events; product creation, product owner update, and shipment creation. In this section, only one event will be discussed. However, this event will demonstrate the full functionality of integrity management. The complete set of results obtained for the entire case-study simulator execution is provided in Appendix B.

The event chosen is that of the sixth event denoted in the flowchart in Figure 4.3. In this event, a supervisor with an integrity level of 10, attempts to issue a certification to a transporter for a product-type of meat. The supervisor specifies an integrity value of 10. However, the transporter the certification is issued to has an integrity value of 5.

From Table 3.1, products are ranked lower than transporters. This means that a

transporter can never handle products that would have a higher ranking than itself. However, if the event described in the preceding paragraph is accepted into the Smart Contract, the transporter with an integrity of level 5 will be certified to handle products of type meat by a certification of integrity level 10. At most, the integrity of the certification can only be 5. As such, the expected behavior is that the smart contract will take the supplied integrity value of 10 and reduce it to 5.

```
1  [{
2    "address": "0xe507A2515633154090e3361B8bc92865f61A8fA6",
3    "blockHash":
         ↪ "0xa6d04cf36f160ff297fc5def5d88c38dfd4c334fab5ede8c6eefaf84fe00baee",
4    "blockNumber": 64,
5    "logIndex": 0,
6    "removed": false,
7    "transactionHash":
         ↪ "0xf1a30740b06732ebc28d7f74619495e250b41c731f141dbd0d2ffdbc487754d2",
8    "transactionIndex": 0,
9    "id": "log_9b3743a6",
10   "returnValues": {
11     "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
12     "_target": "0xE25Fe8C7c12ef26959876E13E4A05FA5D89E2894",
13     "_prod_type": "Meat",
14     "_integrity_level": "5",
15     "_message": "Certificate␣issued␣to␣transporter"
16   },
17   "event": "TransporterCertificateIssued",
18   "signature":
         ↪ "0xe897de04d66d2318c74dec5fbf6d8ecbe2f3fbb2174edd29e69aa8e6cf9a5687",
19   "raw": {
20     "data": "0x000...3e4...",
21     "topics": [
           ↪ "0xe897de04d66d2318c74dec5fbf6d8ecbe2f3fbb2174edd29e69aa8e6cf9a5687"]
22   }
23 }]
```

Listing 4.4: Result of blockchain query for the Transporter Certification Event with a supplied integrity value of 10

Listing 4.4 depicts the results obtained from a blockchain query for the specified event. In line 14, the integrity level of the created certificate is given. The supplied integrity was a value of 10; however, the smart contract asserted the integrity management rules and dropped the integrity value automatically to what its maximum could be 5.

4.4 Financial Considerations

One of the practical considerations when considering creating a cyber-physical supply chain system should be the cost of implementation. However, knowing or pretending to know the complete implementation costs on a per-metric basis would be very difficult to determine. As such, the style of this section will be to discuss the time that was required to create the prototypes, as well as how this would scale to industry, given the experience of the author.

In terms of the development of the prototype for this research. The following three-part list presents a breakdown of the cost in terms of time:

1. **Background research and Solidity learning** - 160 Hours:

   This entails the research required to fully understand the problem, as well as the work needed to get proficient in the smart contract platform and all the required tools needed to develop the initial prototype.

2. **Creation of Simple Prototype** - 160 Hours:

   This entails the design, development, and testing of the simple prototype.

3. **Expansion of Prototype** - 80 Hours:

   This entailed expanding the simple prototype to include the integrity management components.

The total cost in terms of time spent is then 400 hours, or 10 working weeks assuming 8-hour days. Based on U.S. job statistics, the average mid-level software engineer is paid $44 per hour (Ziprecruiter, 2022). This means the estimated labor cost for the development of the prototype in this research is: $17,600.00.

The estimate given in the preceding section is merely an estimate given the conditions of this thesis research; however, they should provide a picture of what is required when working on the software development aspect. Additionally, there are some additional considerations for industry use. These considerations are based on the professional experience of the author.

1. **Cloud deployment and provisioning at scale** - In most cases, organizations would need to deploy a considerable amount of distributed cloud infrastructure to ensure the holistic functioning of a cyber-physical supply chain system such as is contained in this research. These systems would need to be scalable and capable of handling the load generated by the organization's operations.

2. **Cost of technology deployment** - For the CPSC to be accessible by all nodes within the supply chain, appropriate technology would need to be provisioned and deployed at each node to ensure that the participant at that node can interact with the supply chain. Even with modern ERP systems, this may be a complex software engineering task.

3. **Cost of digitization** - For goods to be represented within a digital world, sufficient cyber-physical infrastructure must exist to capture those goods digitally. Things such as barcodes, SKU, or unit numbers are not trivial items to deploy in an organization and may require significant digitization work.

The last financial consideration is related to the structure and consensus mechanism of the blockchain chosen. In a decentralized blockchain system, miners/nodes/transaction validators perform the computation at their own hardware expense. As a reward for this contribution, miners are typically rewarded using the native currency of the blockchain. This reward is derived in some way from the transaction fee paid by each participant when they initiate transactions. In this research, transactional equivalence is established between the real world and the digital supply chain, in which case real-world events have an equivalent on the blockchain. However, each of these transactions would incur a transaction fee on the part of the real-world participant. Depending on the blockchain, these transaction fees could be trivial at first, however for large volume users; these fees would add up[5].

[5]The structuring of transactions to maintain adequate transactional equivalence while minimizing transaction fees would itself be an optimization problem, that could become a future avenue of research.

Conclusion

Creating an Ethereum-based simple prototype in Section 4.2 showed that it is possible to create a mirrored version of the real-world supply chain within the blockchain. Furthermore, the results presented in Section 4.2.4 show how this simple prototype produced transactional equivalents of the real-world supply chain events of product creation, product ownership update, and shipment creation. These queries also show how easy traceability is. In each of the three cases, a query run to the blockchain will provide the parameters is all that was necessary to obtain the transactional information.

A more advanced prototype is shown in Section 4.3. This prototype includes integrity management enforced by the smart contracts acting as the reference monitors. The results presented in Section 4.3.3 show how smart contracts can be used to automatically enforce integrity management rules.

CHAPTER 5

CONCLUSION & FUTURE RESEARCH

Three research questions were posed in the opening Chapter, Chapter 1.3. These questions are repeated below.

1. How can supply chain traceability be improved through the usage of blockchain technology?
2. How can compliance with certain supply chain standards be enforced and added to the provenance record?
3. How can integrity ranking be used to augment the supply chain?

These three short questions guided the subsequent research conducted across three chapters. Chapter 2 presents the results of an extensive literature review. In this Chapter, the four facets of the research in this thesis (supply chain traceability, blockchain, blockchain in the supply chain & integrity) were researched separately. This culmination was that several problems within traditional supply chain traceability were identified, and a proposed idea of a blockchain-enabled supply chain was presented in Figure 2.5 on page 25. This proposed solution would answer all three of the research questions.

Blockchain could improve traceability through its decentralized and independently auditable nature (Question 1). Within this blockchain solution, smart contracts could be used to enforce certain real-world rules (Question 2). Furthermore, these smart contracts would also allow for integrity management to be implemented within the cyber realm but enforced in the physical realm too (Question 3).

Chapter 3 presented the methodology by which the proposed solution would be realized. Notably, suitable cyber abstractions (Section 3.1.1) would need to be created using smart contracts (Section 3.2). The usage of smart contracts allows for the creation of preconditioned transactions. These preconditioned transactions allow transactional equivalents

of real-world supply chain events to be created within the cyber realm. The usage of smart contracts also allows for integrity management to be implemented within the cyber realm, as the smart contracts would be able to act as the required reference monitors. Because of the establishment of transactional equivalence, the integrity management rules enforced by smart contracts can be enforced in the physical supply chain world.

To showcase this functionality, a simple prototype using the Ethereum blockchain was created and presented in Chapter 4. This prototype successfully showed how transactional equivalents can be created and how the blockchain can be queried to obtain this information easily. Traceability in this blockchain prototype became merely a function call. A more advanced prototype is also presented in Chapter 4. This advanced prototype shows how integrity management can be implemented via smart contracts, transforming them into reference monitors for the implementation of integrity management.

The incorporation of blockchain into the supply chain through the methods described in this research result in the creation of a cyber-physical supply chain traceability system. In this system, product traceability becomes a function call and transparency becomes the norm.

Future Research The first departure point for future research would be to look into how a cyber-physical supply chain, such as those created in this thesis, could be used to improve supply chain optimization. On the computer science front, there is also an opportunity to explore the provisioning of information. In this research, it was assumed that the provided information was correct, but what if the information could have a quantifiable confidence value? This question may also have answers in the field of data science.

Another avenue of future research could be to reintroduce the financial aspect and use it to improve the security of the blockchain. For example, transaction fees could be refunded to the nodes depending on their performance over a given period of time or their contribution as a miner of transactions. The monetary aspect could also be used to apply fines to participants. This means that there could be a negative financial implication associated with providing false/bad information.

BIBLIOGRAPHY

BIBLIOGRAPHY

Abeyratne, S. A., & Monfared, R. (2016, 9). Blockchain ready manufacturing supply chain using distributed ledger. *International Journal of Research in Engineering and Technology*, *5*(9), 1–10. doi: 10.15623/ijret.2016.0509001

Alharby, M., & Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*.

Arthur, K., Olivier, M., & Venter, H. (2007). Applying the biba integrity model to evidence management. In P. Craiger & S. Shenoi (Eds.), *Advances in digital forensics iii* (pp. 317–327). New York, NY: Springer New York. doi: 10.1007/978-0-387-73742-3_22

Balzarova, M. A. (2020). Blockchain technology–a new era of ecolabelling schemes? *Corporate Governance: The International Journal of Business in Society*.

Behdani, B. (2012). Evaluation of paradigms for modeling supply chains as complex socio-technical systems. In *Proceedings of the 2012 winter simulation conference (wsc)* (pp. 1–15). Berlin, Germany. doi: 10.1109/WSC.2012.6465109

Bell, D. E., & LaPadula, L. J. (1973). *Secure computer systems: Mathematical foundations* (Tech. Rep.). MITRE CORP BEDFORD MA.

Biba, K. J. (1977). *Integrity considerations for secure computer systems* (Technical No. ESD-TR-76-372). Bedford, MA: USAF Electronic Systems Division.

Bosona, T., & Gebresenbet, G. (2013, September). Food traceability as an integral part of logistics management in food and agricultural supply chain. *Food Control*, *33*(1), 32–48. doi: 10.1016/j.foodcont.2013.02.004

Buterin, V. (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* Vitalik Buterin. Retrieved 2022-05-26, from `https://ethereum.org/en/whitepaper/`

Cao, S., Powell, W., Foth, M., Natanelov, V., Miller, T., & Dulleck, U. (2021, January). Strengthening consumer trust in beef supply chain traceability with a blockchain-based human-machine reconcile mechanism. *Computers and Electronics in Agriculture*, *180*, 105886. doi: 10.1016/j.compag.2020.105886

Digiconomist. (2022). *Bitcoin Energy Consumption Index.* Retrieved 2022-05-25, from `https://digiconomist.net/bitcoin-energy-consumption/`

Dolgui, A., Ivanov, D., Potryasaev, S., Sokolov, B., Ivanova, M., & Werner, F. (2020, April). Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain. *International Journal of Production Research*, *58*(7), 2184–2199. (doi: 10.1080/00207543.2019.1627439) doi: 10.1080/00207543.2019.1627439

Dujak, D., & Sajter, D. (2019). Blockchain applications in supply chain. In *Smart supply network* (pp. 21–46). Springer.

Dutta, P., Choi, T.-M., Somani, S., & Butala, R. (2020a). Blockchain technology in supply chain operations: Applications, challenges and research opportunities. *Transportation Research Part E: Logistics and Transportation Review*, *142*, 102067. doi: https://doi.org/10.1016/j.tre.2020.102067

Dutta, P., Choi, T.-M., Somani, S., & Butala, R. (2020b). Blockchain technology in supply chain operations: Applications, challenges and research opportunities. *Transportation research part e: Logistics and transportation review*, *142*, 102067.

ethereum. (2022, March). *Solidity 0.8.17 documentation.* Retrieved 2022-09-29, from https://docs.soliditylang.org/en/v0.8.17/

Forcepoint. (2018, August). *What is the cia triad?* Retrieved 2021-09-07, from https://www.forcepoint.com/cyber-edu/cia-triad

Gil, A. (2018, September). *Data security - confidentiality, integrity & availability.* Retrieved 2021-08-25, from https://www.kvausa.com/data-security-confidentiality-integrity-and-availability/

Hayes, A. (2022, March). *Blockchain Explained.* Retrieved 2022-05-24, from https://www.investopedia.com/terms/b/blockchain.asp

IBM. (2022). *What is Blockchain Technology?* Retrieved 2022-05-24, from https://www.ibm.com/topics/what-is-blockchain

IBM, C. (2021, Apr). *Ibm blockchain transparent supply.* Author. Retrieved from https://www.ibm.com/docs/en/transparent-supply?topic=started-basic-provenance-concepts

Introduction to the Ethereum stack [Article]. (2022, February). Retrieved 2022-07-27, from https://ethereum.org

ISO. (2007). *Iso 22005: 2007-traceability in the feed and food chain-general principles and basic requirements for system design and implementation.* International Organization for Standardization (ISO) Geneva (Switzerland).

Ivanov, D., Sokolov, B., & Ivanova, M. (2016). Schedule coordination in cyber-physical supply networks industry 4.0. *IFAC-PapersOnLine*, *49*(12), 839–844. (8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016) doi: 10.1016/j.ifacol.2016.07.879

Jakobsson, M., & Juels, A. (1999). *Proofs of work and bread pudding protocols (extended abstract). secure information networks (s. 258-272).* Boston: Springer US.

Kim, H. M., & Laskowski, M. (2018). Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, *25*(1), 18–27.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th ieee international symposium on object and component-oriented real-time distributed computing (isorc)* (pp. 363–369). doi: 10.1109/ISORC.2008.25

Li, P., Mao, Y., & Zdancewic, S. (2003). Information integrity policies. In *In proceedings of the workshop on formal aspects in security & trust (fast.*

Liu, Z., & Li, Z. (2020). A blockchain-based framework of cross-border e-commerce supply chain. *International Journal of Information Management*, *52*, 102059.

Mail, C. J. (1993). Pricing via processing. In *Advances in cryptology—crypto'92: 12th annual international cryptology conference, santa barbara, california, usa, august 16–20, 1992. proceedings* (Vol. 740, p. 139).

Merriam-Webster.com. (2022). *"Blockchain".* Retrieved 2022-05-24, from https://www.merriam-webster.com/dictionary/blockchain

Mix. (2019, May). *These are the top 10 programming languages in blockchain.* Retrieved 2022-09-29, from https://thenextweb.com/news/javascript-programming-java-cryptocurrency

Montecchi, M., Plangger, K., & Etter, M. (2019). It's real, trust me! establishing supply chain provenance using blockchain. *Business Horizons*, *62*(3), 283–293. doi: 10.1016/j.bushor.2019.01.008

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

Norton, T., & Conlon, C. (2019, Aug). *Supply chain visibility: Traceability, transparency, and mapping explained: Blog.* BSR. Retrieved 2022-04-22, from `https://www.bsr.org/en/our-insights/blog-view/supply-chain-visibility-traceability-transparency-and-mapping`

Park, K. T., Son, Y. H., & Noh, S. D. (2021). The architectural framework of a cyber physical logistics system for digital-twin-based supply chain control. *International Journal of Production Research*, *59*(19), 5721-5742. doi: 10.1080/00207543.2020.1788738

Peters, G. W., & Panayi, E. (2016). Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In *Banking beyond banks and money* (pp. 239–278). Springer.

QuantumMechanic. (2011, July). *Proof of stake instead of proof of work* [Blog]. Retrieved 2022-05-25, from `https://bitcointalk.org/index.php?topic=27787.0`

Raja Santhi, A., & Muthuswamy, P. (2022). Influence of blockchain technology in manufacturing supply chain and logistics. *Logistics*, *6*(1). doi: 10.3390/logistics6010015

Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Production planning and scheduling in cyber-physical production systems: a review. *International Journal of Computer Integrated Manufacturing*, *32*(4-5), 385-395. doi: 10.1080/0951192X.2019.1605199

Sander, F., Semeijn, J., & Mahr, D. (2018). The acceptance of blockchain technology in meat traceability and transparency. *British Food Journal*, *120*(9), 2066–2079. doi: 10.1108/BFJ-07-2017-0365

Shing, M.-l., Shing, C.-c., Chen, K., & Lee, H. (2006). Security modeling on the supply chain networks. *Proceedings of EIST 2006*.

Szabo, N. (1998). Secure property titles with owner authority. *Online at http://szabo. best. vwh. net/securetitle. html*.

Tagarakis, A. C., Benos, L., Kateris, D., Tsotsolas, N., & Bochtis, D. (2021). Bridging the gaps in traceability systems for fresh produce supply chains: Overview and development of an integrated iot-based system. *Applied Sciences*, *11*(16), 7596. doi: 10.3390/app11167596

van Hilten, M., Ongena, G., & Ravesteijn, P. (2020). Blockchain for organic food traceability: case studies on drivers and challenges. *Frontiers in Blockchain*, 43.

Walkowski, D. (2019, July). *What is the cia triad?* Retrieved 2021-09-07, from `https://www.f5.com/labs/articles/education/what-is-the-cia-triad`

Wamba, S. F., & Queiroz, M. M. (2022). Industry 4.0 and the supply chain digitalisation: a blockchain diffusion perspective. *Production Planning & Control*, *33*(2-3), 193–210.

Yuan, C., Wang, S., & Yu, X. (2020). The impact of food traceability system on consumer perceived value and purchase intention in china. *Industrial Management & Data Systems*.

Ziprecruiter. (2022). *Mid level software engineer salary.* Retrieved from `https://www.ziprecruiter.com/Salaries/Mid-Level-Software-Engineer-Salary`

APPENDICES

APPENDIX A

SMART CONTRACTS INTERNAL STRUCTURE

This Appendix will provide detailed breakdowns of each of the six smart contracts that make up the simple and advanced prototypes. Providing a true holistic picture of a coded prototype is very difficult to do in a concise manner that captures the full functionality. Each of the sections will provide a call graph and an audit report excerpt taken from the Surya audit report.

The call graph depicts the interactions each smart contract has with other smart contracts as well as itself via internal function calls. The audit report provides insight into the functions, their visibility, mutability and the modifiers attached to them.

The call graph therefore will provide a picture of which methods make calls to which smart contracts. The Surya report will then map modifiers to functions as well as provide insight into the mutability and visibility of each function.

The conventions used in the Surya report can be confusing as the terminology might not seem familiar. Here is a breakdown of the relevant terminology:

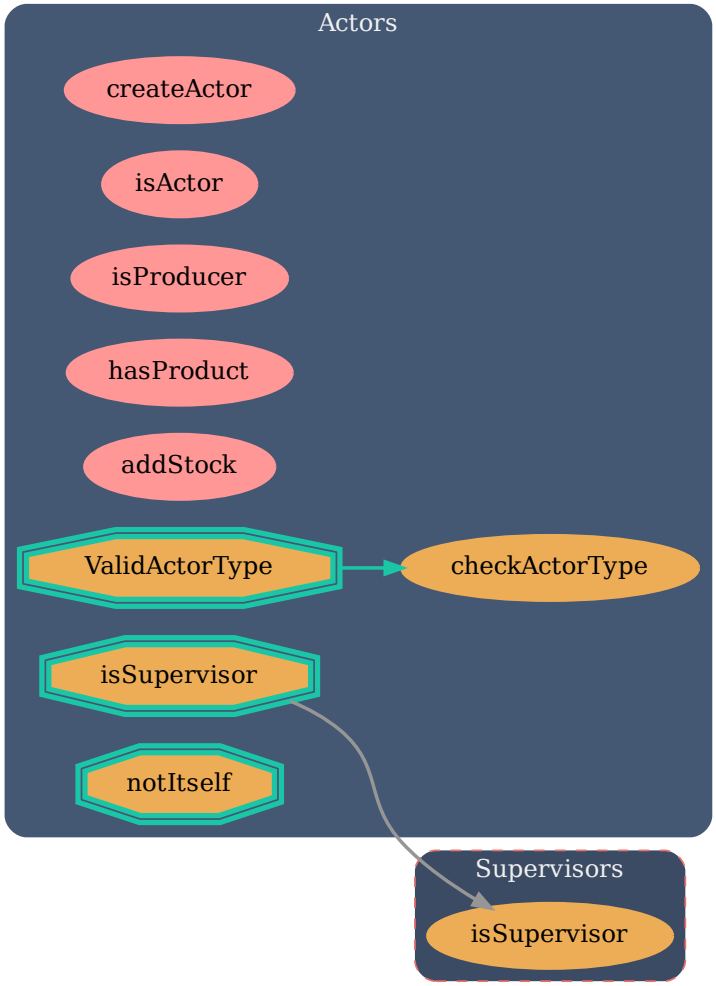
- **Visibility** - This term defines how visible a smart contract function is. There are two types, public and private. Private visibility means that the function is only accessible within the smart contract itself, public means that the function can be called from outside of the smart contract.
- **Mutability** - This term defines whether a function is capable of making changes to the storage of the smart contract. For example, `createActor` changes the state of the Actors smart contract because it creates a new Actor, hence it is a mutable function. `isActor` does not change the state, and as such has no mutability.
- **Modifiers** - Within the Solidity programming language, validators that are use to control the access to smart contracts are called Modifiers. These modifiers are themselves smaller functions that assert a set of conditions. Only once the asserted conditions return a true value, indicating they have passed, can the function call be executed. Modifiers are the same as Validators.

The smart contracts in this section conform to the design conventions presented in Section 3.2.

Simple Prototype

The call graphs and Surya reports for the smart contracts used in the creation of the simple prototype are documented in this section.

Actors



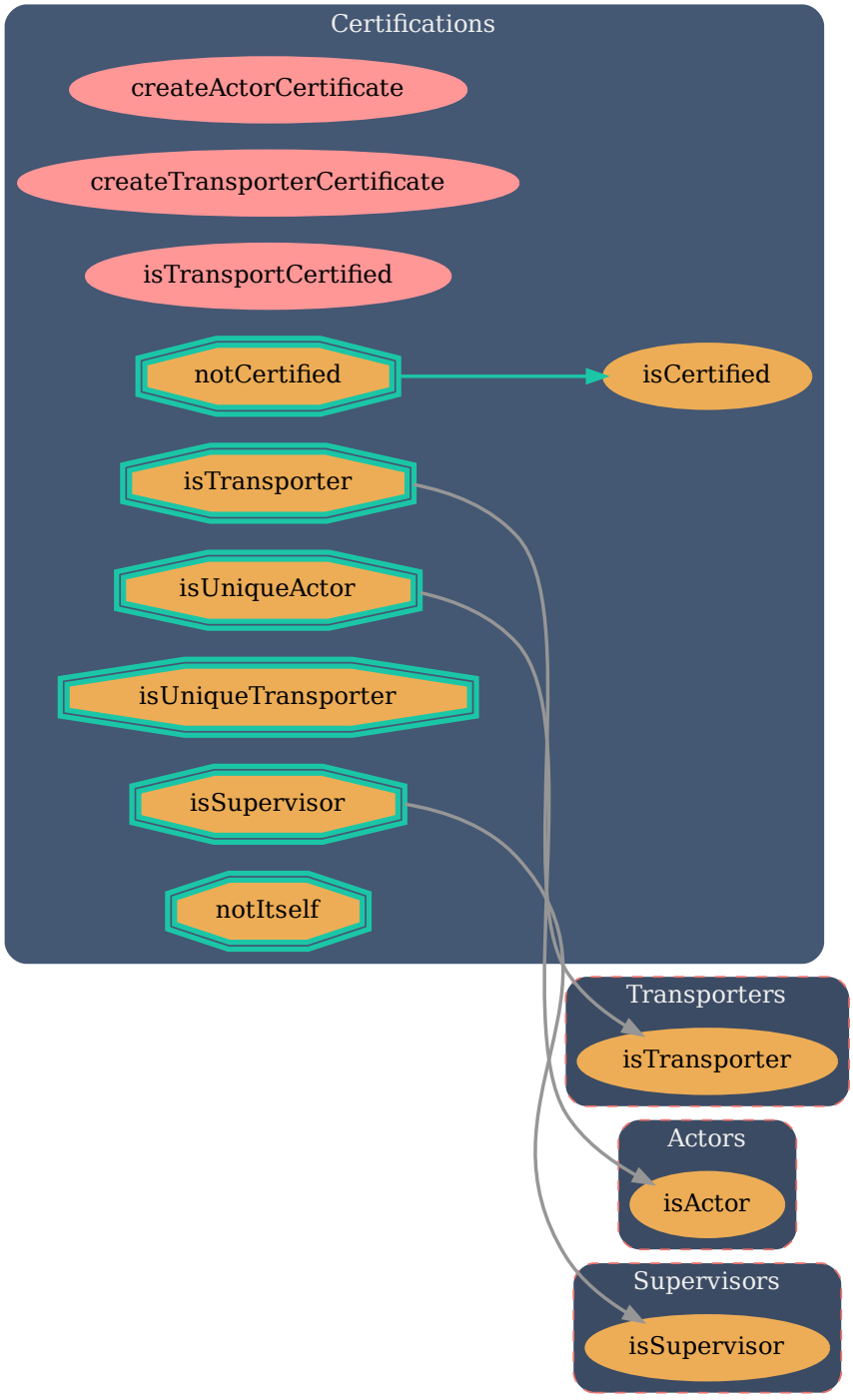
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createActor	Public	Yes	ValidActorType isSupervisor notItself
isActor	Public	No	-
isProducer	Public	No	-
hasProduct	Public	No	-
addStock	Public	Yes	-
checkActorType	Private	No	-

(b) Excerpt of the Surya report

Figure A.1: Call graph and Surya report excerpt for the Actors smart contract.

Certifications



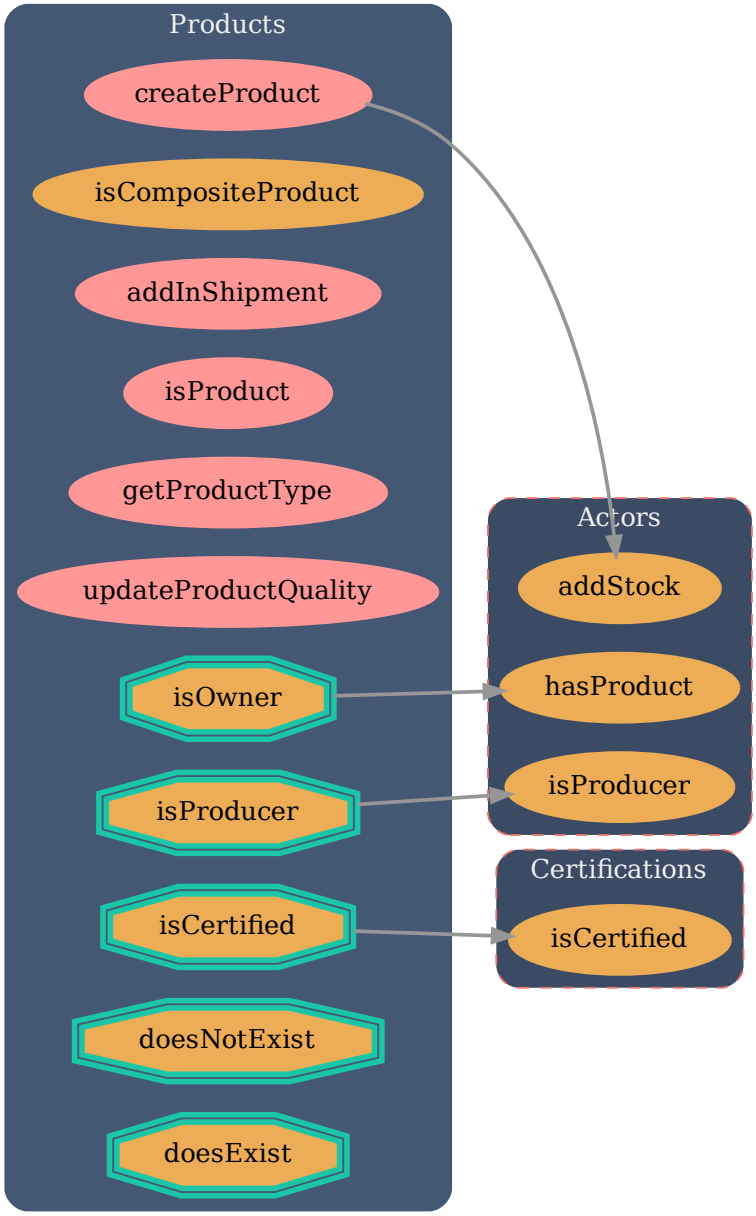
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createActorCertificate	Public	Yes	isSupervisor isUniqueActor notItself notCertified
createTransporterCertificate	Public	Yes	isUniqueTransporter isSupervisor isTransporter notItself notCertified
isCertified	Public	No	-
isTransportCertified	Public	No	-

(b) Excerpt of the Surya report

Figure A.2: Call graph and Surya report excerpt for the Certifications smart contract.

Products



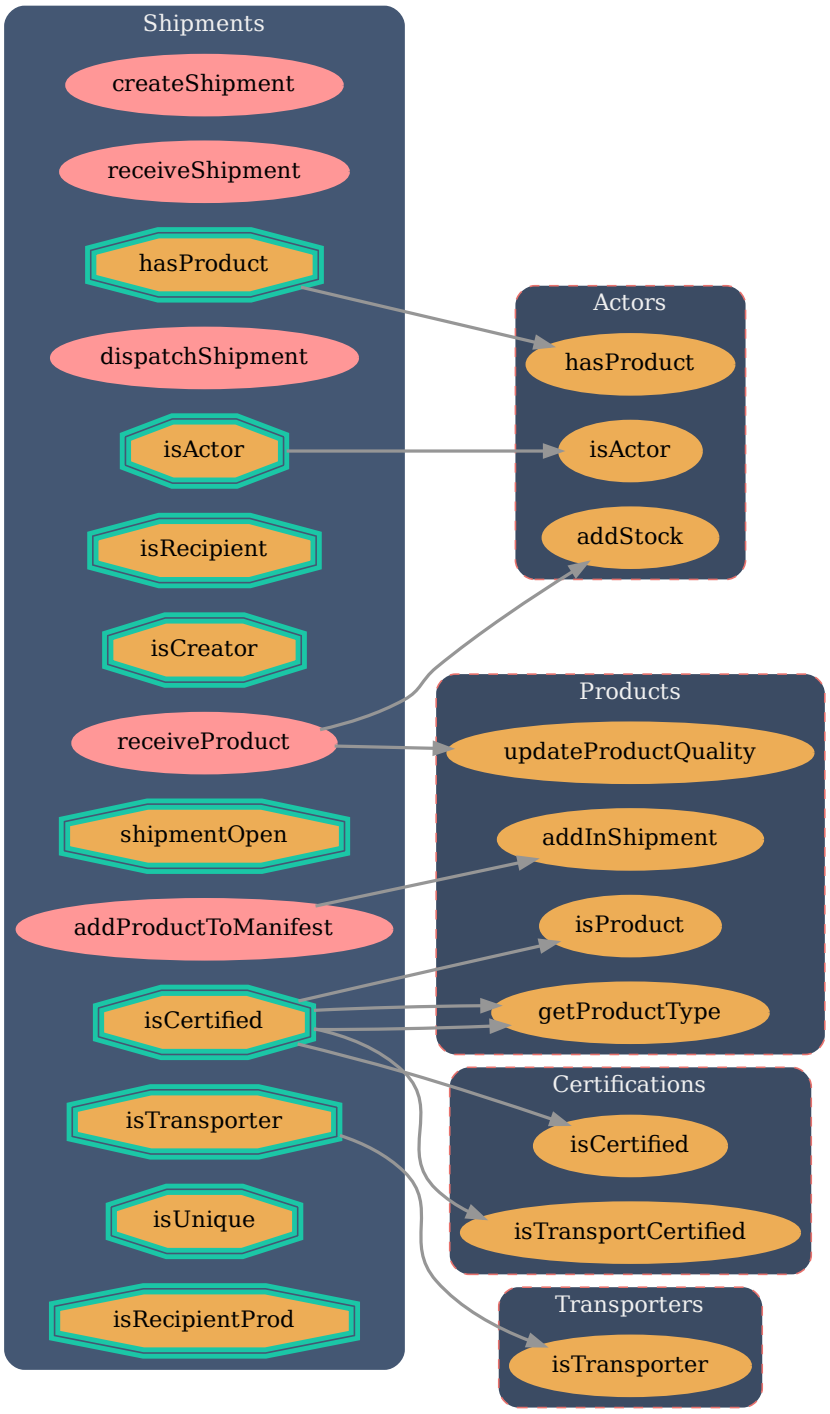
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createProduct	Public	Yes	isProducer isCertified doesNotExist
isCompositeProduct	Private	No	-
addInShipment	Public	Yes	doesExist
isProduct	Public	No	-
getProductType	Public	No	-
updateProductQuality	Public	Yes	-

(b) Excerpt of the Surya report

Figure A.3: Call graph and Surya report excerpt for the Products smart contract.

Shipments



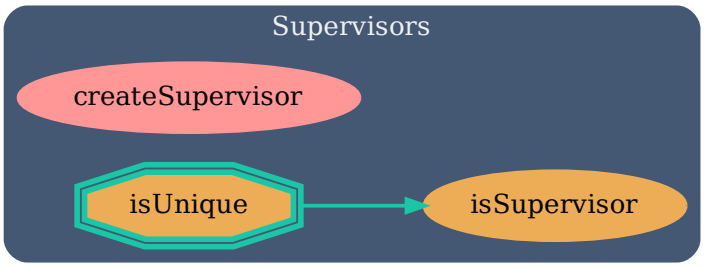
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createShipment	Public	Yes	isActor isActor isTransporter
receiveShipment	Public	Yes	isRecipient
receiveProduct	Public	Yes	isRecipientProd
dispatchShipment	Public	Yes	isCreator
addProductToManifest	Public	Yes	hasProduct shipmentOpen isCertified

(b) Excerpt of the Surya report

Figure A.4: Call graph and Surya report excerpt for the Shipments smart contract.

Supervisors



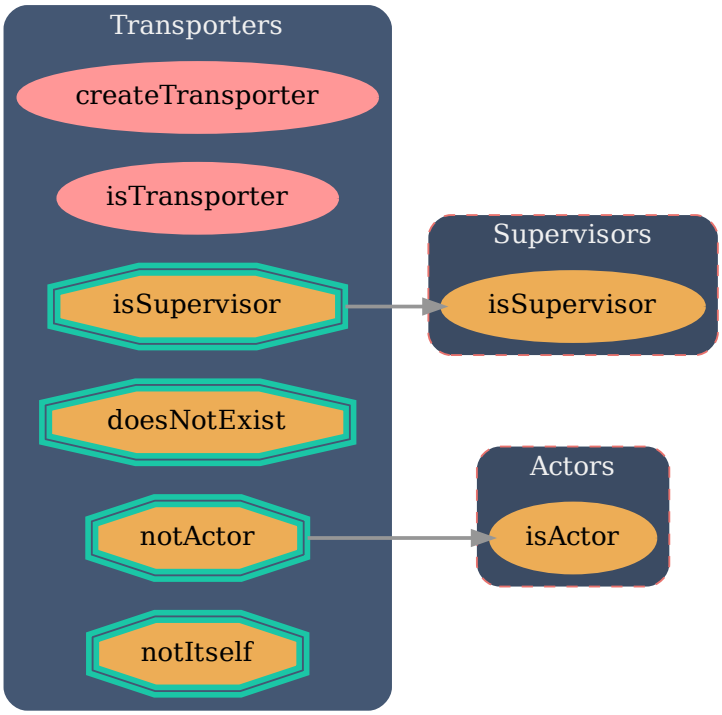
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createSupervisor	Public	Yes	isUnique
isSupervisor	Public	No	-

(b) Excerpt of the Surya report

Figure A.5: Call graph and Surya report excerpt for the Supervisors smart contract.

Transporters



(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createTransporter	Public	Yes	isSupervisor doesNotExist notActor notItself
isTransporter	Public	No	-

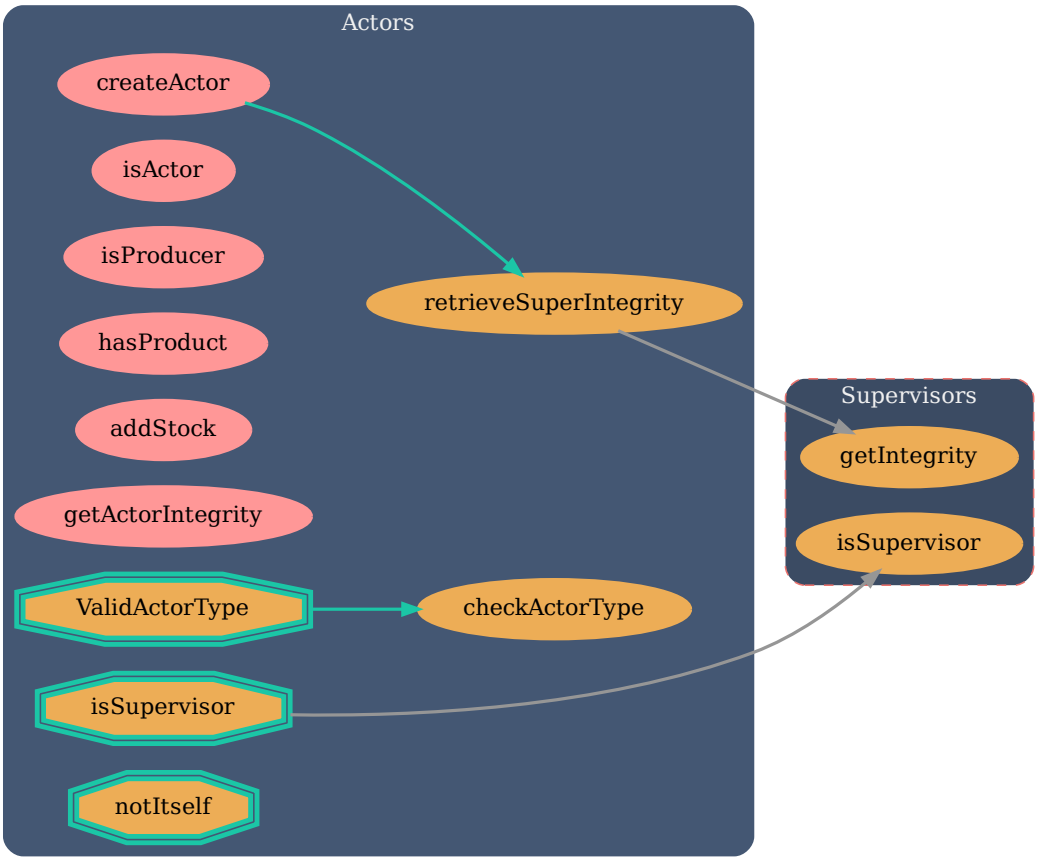
(b) Excerpt of the Surya report

Figure A.6: Call graph and Surya report excerpt for the Transporters smart contract.

Advanced Prototype

The call graphs and Surya reports for the smart contracts used in the creation of the advanced prototype are documented in this section.

Actors



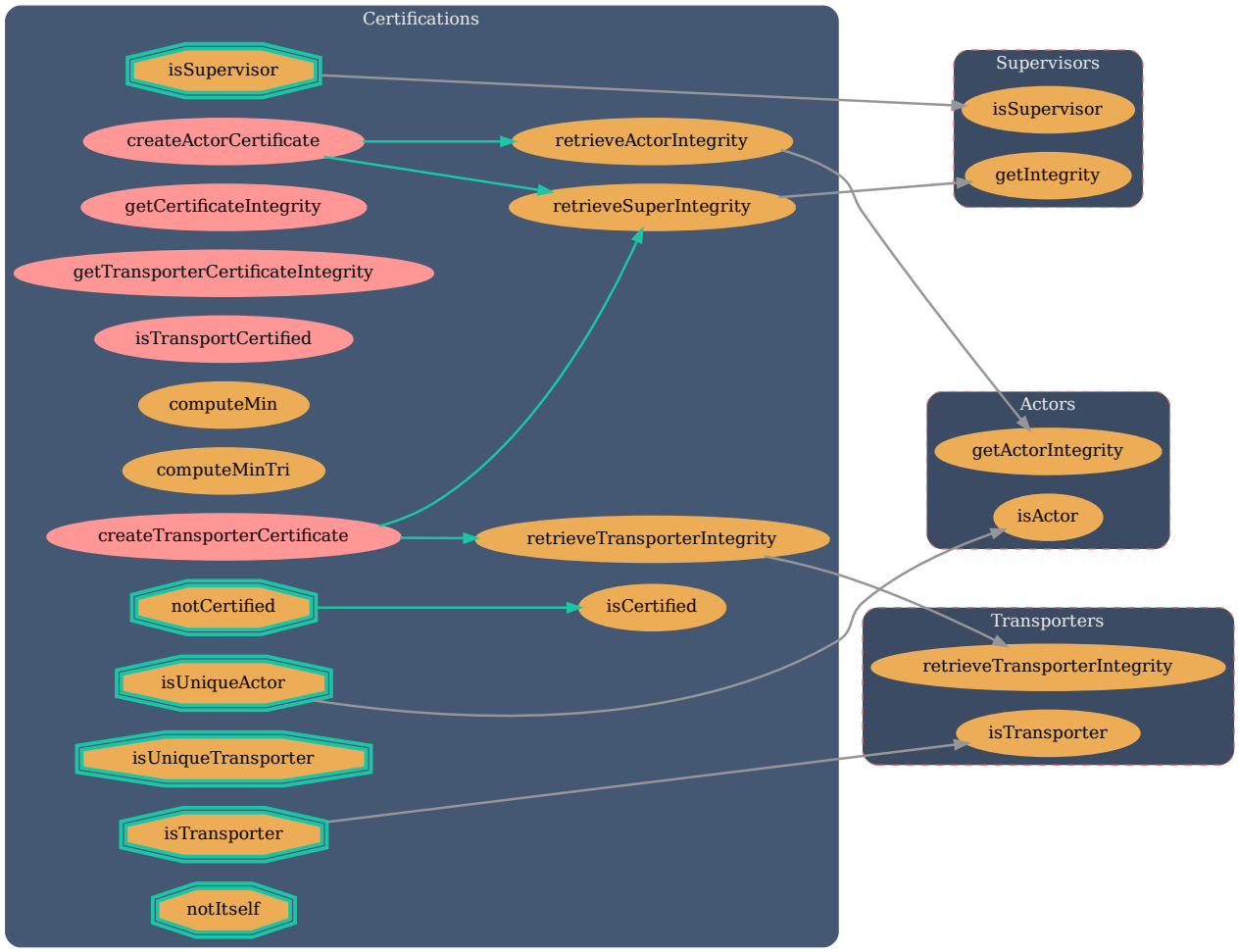
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createActor	Public	Yes	ValidActorType isSupervisor notItself
isActor	Public	No	-
isProducer	Public	No	-
hasProduct	Public	No	-
addStock	Public	Yes	-
retrieveSuperIntegrity	Private	No	-
getActorIntegrity	Public	No	-
checkActorType	Private	No	-

(b) Excerpt of the Surya report

Figure A.7: Call graph and Surya report excerpt for the Actors smart contract.

Certifications



(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createActorCertificate	Public	Yes	isSupervisor isUniqueActor notItself notCertified
createTransporterCertificate	Public	Yes	isUniqueTransporter isSupervisor isTransporter notItself notCertified
retrieveSuperIntegrity	Private	No	-
retrieveActorIntegrity	Private	No	-
retrieveTransporterIntegrity	Private	No	-
getCertificateIntegrity	Public	No	-
getTransporterCertificateIntegrity	Public	No	-
isCertified	Public	No	-
isTransportCertified	Public	No	-

(b) Excerpt of the Surya report

Figure A.8: Call graph and Surya report excerpt for the Certifications smart contract.

Products

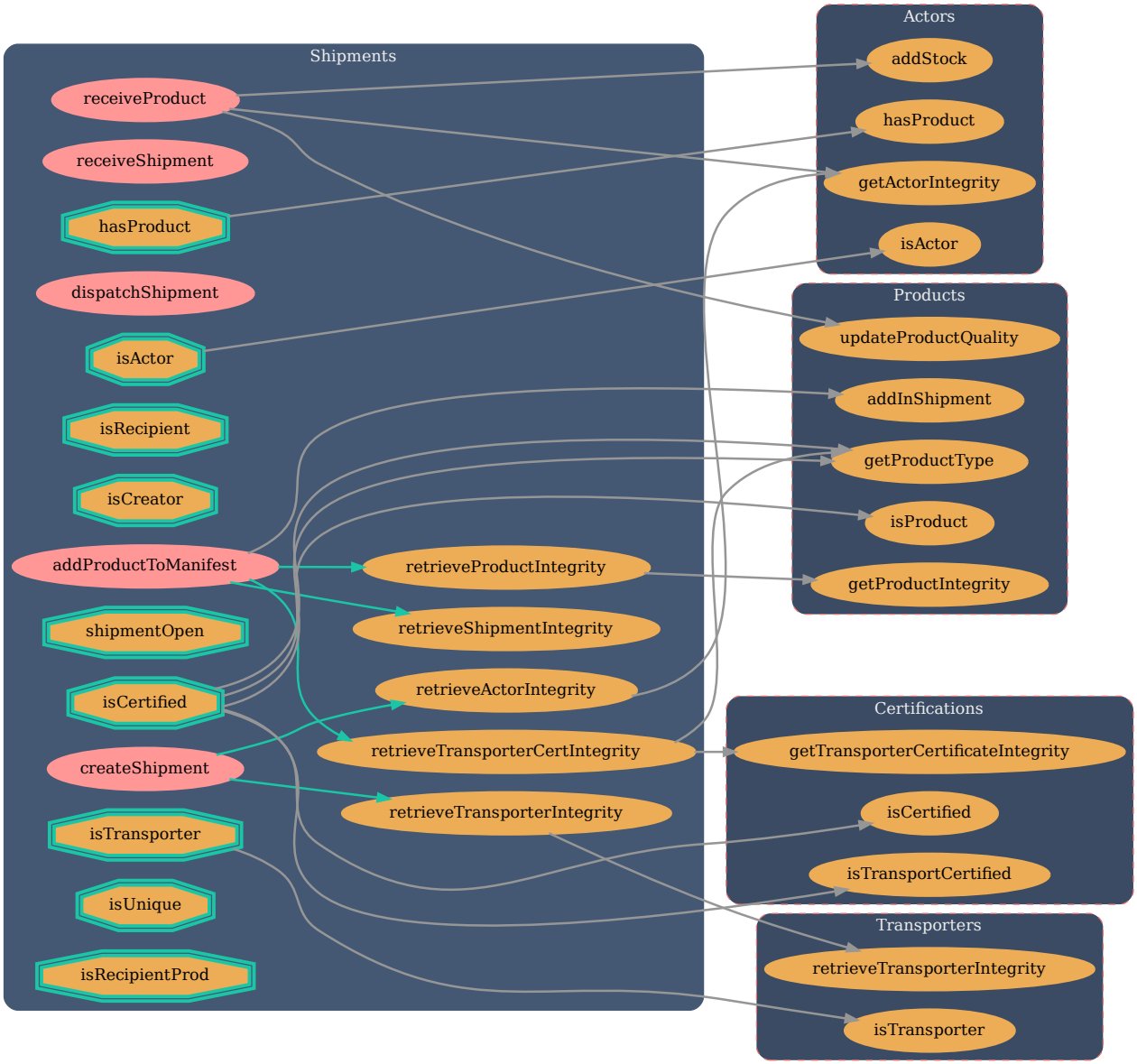
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createProduct	Public	Yes	isProducer isCertified doesNotExist
isCompositeProduct	Private	No	-
addInShipment	Public	Yes	doesExist
isProduct	Public	No	-
getProductIntegrity	Public	No	-
getProductType	Public	No	-
retrieveActorIntegrity	Private	No	-
retreiveCertIntegrity	Private	No	-
updateProductQuality	Public	Yes	-

(b) Excerpt of the Surya report

Figure A.9: Call graph and Surya report excerpt for the Products smart contract.

Shipments



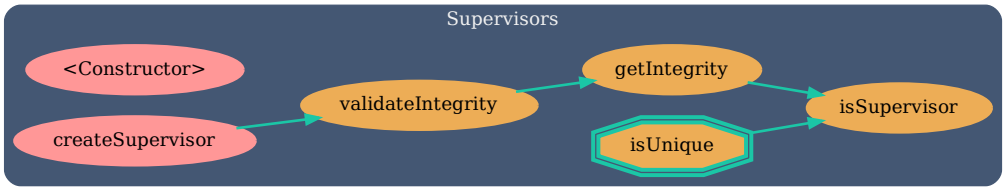
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createShipment	Public	Yes	isActor isActor isTransporter
receiveShipment	Public	Yes	isRecipient
receiveProduct	Public	Yes	isRecipientProd
retrieveActorIntegrity	Private	No	-
dispatchShipment	Public	Yes	isCreator
addProductToManifest	Public	Yes	hasProduct shipmentOpen isCertified
retrieveProductIntegrity	Private	No	-
retrieveShipmentIntegrity	Private	No	-
retrieveTransporterCertIntegrity	Private	No	-
retrieveTransporterIntegrity	Private	No	-

(b) Excerpt of the Surya report

Figure A.10: Call graph and Surya report excerpt for the Shipments smart contract.

Supervisors



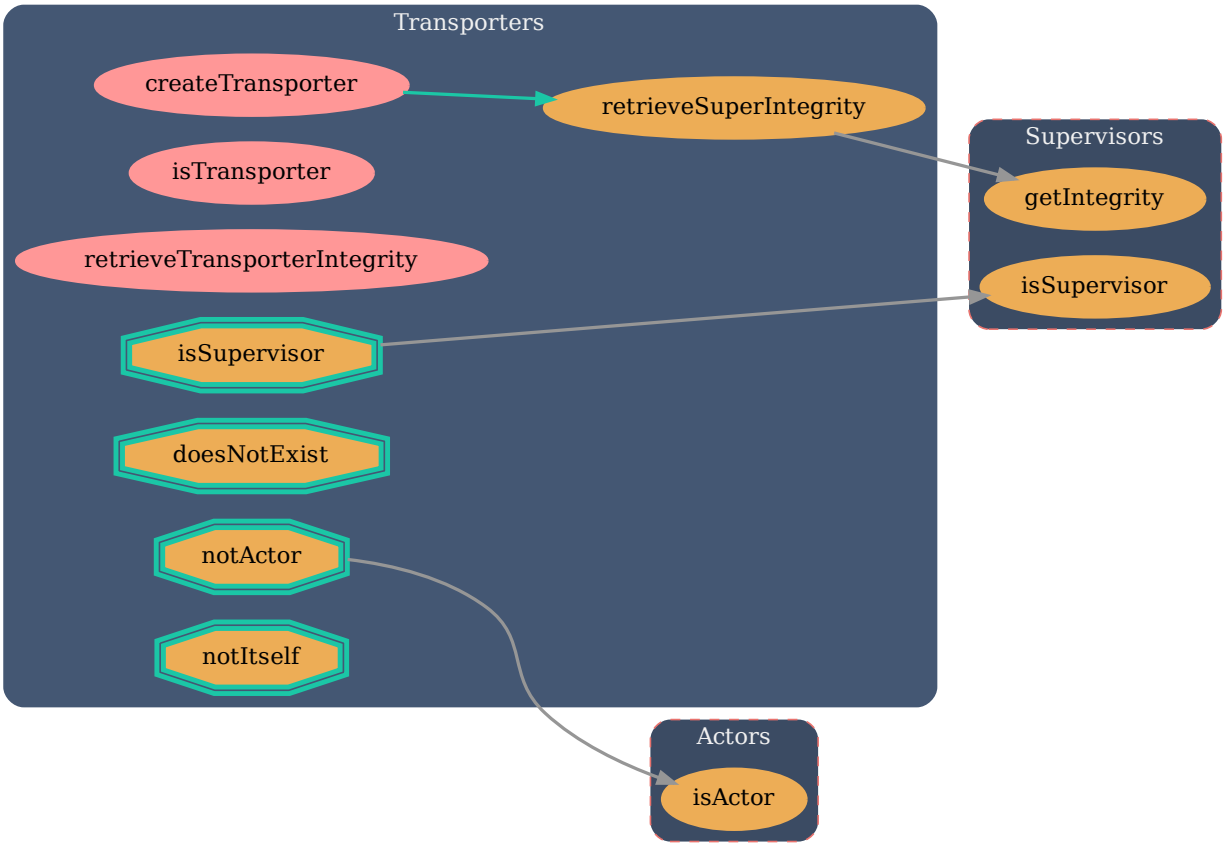
(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createSupervisor	Public	Yes	isUnique
isSupervisor	Public	No	-
validateIntegrity	Private	No	-
getIntegrity	Public	No	-

(b) Excerpt of the Surya report

Figure A.11: Call graph and Surya report excerpt for the Supervisors smart contract.

Transporters



(a) Call graph

Function Name	Visibility	Mutability	Modifiers
createTransporter	Public	Yes	isSupervisor doesNotExist notActor notItself
isTransporter	Public	No	-
retrieveSuperIntegrity	Private	No	-
retrieveTransporterIntegrity	Public	No	-

(b) Excerpt of the Surya report

Figure A.12: Call graph and Surya report excerpt for the Transporters smart contract.

APPENDIX B

ADVANCED PROTOTYPE COMPLETE CASE-STUDY RESULT SET

This section presents a complete set of results for the case-study depicted in Figure 4.3. The section titles in this Appendix match the text in the bubbles in Figure 4.3.

Create Actor1-10 & Actor2-5


```
[
  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
↪   "0x686bd6ba9d6bbf509ec6c8986f8b4e847e7b8dcfcd6202f57e099301b959bb87",
    "blockNumber": 10,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪   "0x7d237023daa503b6855f16b7989aa6944fae8d6883a7b2df6ddfbfbcb7fb11e6",
    "transactionIndex": 0,
    "id": "log_74667d0f",
    "returnValues": {
      "_actor_id": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_actor_name": "Producer 1",
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_actor_location": "{\"lat\":\"29.1155 S\",\"long\":\"23.7514 E\"}",
      "_actor_type": "Producer",
      "_integrity_level": "8",
      "_message": "New Actor created"
    },
    "event": "AcceptActorCreate",
    "signature":
↪   "0x29dcaed81ff1f6fa06b3dd018644cff380b42a6609a0b1bfe744971c6e0e1367",
    "raw": {
```

```
      "data": "0x0000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff
    ↪ 6d7c840000000000000000000000000000000000000000000000000000000000e00
    ↪ 0000000000000000000000f9d1a333205517bcff66c6271ca80466ea131f5500000000
    ↪ 0000000000000000000000000000000000000000000000000000012000000000000000
    ↪ 0000000000000000000000000000000000000000000000180000000000000000000000
    ↪ 0000000000000000000000000000000000000000800000000000000000000000000000
    ↪ 000000000000000000000000000000000000001c0000000000000000000000000000000000
    ↪ 0000000000000000000000000a50726f6475636572203100000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 00000000000267b226c6174223a2232392e313135352053222c226c6f6e67223a22323
    ↪ 32e373531342045227d0000000000000000000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000850726f6
    ↪ 4756365720000000000000000000000000000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000114e6577204163746f72206
    ↪ 3726561746564000000000000000000000000000",
      "topics": [
        "0x29dcaed81ff1f6fa06b3dd018644cff380b42a6609a0b1bfe744971c6e0e1367"
      ]
    }
  },
  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
    ↪ "0x03bd7a8b9e31282823ea3f894f88c29c2378de1093eb3a9d7dc96ad840937877",
    "blockNumber": 11,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
    ↪ "0x5041bebe817ede823ed96a93c474c4e80f7b14be2a0828f7d0310ee2aac1af1f",
    "transactionIndex": 0,
    "id": "log_54d4caf8",
    "returnValues": {

      "_actor_id": "0xa4d7033062C7DCc6676C09961fBD6638c92F40F8",
      "_actor_name": "Actor 2",
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_actor_location": "\"lat\": \"123\", \"lng\": \"123\" ",
      "_actor_type": "Producer",
      "_integrity_level": "5",
      "_message": "New Actor created"
    },
    "event": "AcceptActorCreate",
    "signature":
    ↪ "0x29dcaed81ff1f6fa06b3dd018644cff380b42a6609a0b1bfe744971c6e0e1367",
    "raw": {
```

```
      "data": "0x0000000000000000000000a4d7033062c7dcc6676c09961fbd6638c9
↪ 2f40f800000000000000000000000000000000000000000000000000000000e00
↪ 000000000000000000000f9d1a333205517bcff66c6271ca80466ea131f5500000000
↪ 000000000000000000000000000000000000000000000000120000000000000000
↪ 000000000000000000000000000000000000000000000160000000000000000000
↪ 000000000000000000000000000000000000000500000000000000000000000000
↪ 000000000000000000000000000000001a000000000000000000000000000000000
↪ 000000000000000000000000074163746f722032000000000000000000000000000
↪ 000000000000000000000000000000000000000000000000000000000000000000
↪ 0000000000001b226c6174223a2022313233222c20226c6e67223a20223132332220000
↪ 000000000000000000000000000000000000000000000000000000000000000008
↪ 50726f6475636572000000000000000000000000000000000000000000000000000
↪ 000000000000000000000000000000000000000000000000114e657720416374
↪ 6f722063726561746564000000000000000000000000000000",
      "topics": [
        "0x29dcaed81ff1f6fa06b3dd018644cff380b42a6609a0b1bfe744971c6e0e1367"
      ]
    }
  }
]
```


Issue Certification for Meat to Actor 1 with integrity 10 & Actor 2 with integrity 5


```
[
  {
    "address": "0xE1e5Ce6126d11126be7700cE822f65b99c3a1110",
    "blockHash":
↪ "0xe800d440bb35c979a268d04d3f5abca2b3301cc1d67bffb74cb6551ccf37e071",
    "blockNumber": 12,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪ "0xab7778bc4d216d9b2ddbc1d52498efba8b355bded8f2640c0b665ec7d8098e5b",
    "transactionIndex": 0,
    "id": "log_caa4650d",
    "returnValues": {
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_target": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_prod_type": "Meat",
      "_integrity_level": "8",
      "_message": "Certificate issued to Actor"
    },
    "event": "ActorCertificateIssued",
    "signature":
↪ "0xb2a6d28a50457a6b963fee31a253aa89d90636fadab587fd26dade41607fb9ee",
    "raw": {
```

```
      "data": "0x0000000000000000000000f9d1a333205517bcff66c6271ca80466ea131f55000000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c840000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000008000000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000000000000000000000000000000000000044d65617400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001b43657274696669636174652069737375656420746f204163746f720000000000",
      "topics": [
        "0xb2a6d28a50457a6b963fee31a253aa89d90636fadab587fd26dade41607fb9ee"
      ]
    }
  },
  {
    "address": "0xE1e5Ce6126d11126be7700cE822f65b99c3a1110",
    "blockHash": "0x4524ecb3d256b6e6ceac94ad00bf0883604a6519b9a3ae54641d6d2824c3d8a2",
    "blockNumber": 13,
    "logIndex": 0,
    "removed": false,
    "transactionHash": "0x404b743d6ef2a1e9b1651ec412d7ae31015efcf959247aae84ded5c67be98343",
    "transactionIndex": 0,
    "id": "log_62ead768",
    "returnValues": {
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_target": "0xa4d7033062C7DCc6676C09961fBD6638c92F40F8",
      "_prod_type": "Meat",
      "_integrity_level": "5",
      "_message": "Certificate issued to Actor"
    },
    "event": "ActorCertificateIssued",
    "signature": "0xb2a6d28a50457a6b963fee31a253aa89d90636fadab587fd26dade41607fb9ee",
    "raw": {
      "data": "0x0000000000000000000000f9d1a333205517bcff66c6271ca80466ea131f55000000000000000000000000a4d7033062c7dcc6676c09961fbd6638c92f40f80000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000005000000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000000000000000000000000000000000000044d65617400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001b43657274696669636174652069737375656420746f204163746f720000000000",
      "topics": [
```

```
          "0xb2a6d28a50457a6b963fee31a253aa89d90636fadab587fd26dade41607fb9ee"
        ]
      }
    }
]

```

Create Products with type of Meat and Integrity 10


```
[
  {
    "address": "0xc409604F93692712161a71C683e5446591224eCc",
    "blockHash":
    ↪ "0xb1563495ce761adfd4245d61e7103190da3afd9c4fcb5ff43be563d8d590fc41",
    "blockNumber": 14,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
    ↪ "0x46ffb4f6dfb826543c0b44d1f925c6089a85485c02f429d0e9dd4b8b55b2dae8",
    "transactionIndex": 0,
    "id": "log_034a4ff4",
    "returnValues": {
      "_prod_name": "Beef Steak",
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_prod_quality": "100",
      "_sc_identifiers": "{\"UPC\":\"480528304523\"}",
      "_product_type": "Meat",
      "_creator": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_integrity_level": "8",
      "_parents": ["Not Composite"]
    },
    "event": "AcceptProductCreate",
    "signature":
    ↪ "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10",
    "raw": {
```

```
      "data":
    ↪ "0x0000000000000000000000000000000000000000000000000000000000001200000
    ↪ 0000000000000000000000000000000000000000000000000000000000016000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000064000000000000000
    ↪ 000000000000000000000000000000000000000000000001c000000000000000000000000
    ↪ 0000000000000000000000000000000000000000200000000000000000000000aca189c7
    ↪ 301d7f5ab7d6523d3fec42ceff6d7c84000000000000000000000000aca189c7301d7f5
    ↪ ab7d6523d3fec42ceff6d7c840000000000000000000000000000000000000000000000000
    ↪ 0000000000000000800000000000000000000000000000000000000000000000000000000
    ↪ 0000000024000000000000000000000000000000000000000000000000000000000000000
    ↪ 000a4265656620537465616b000000000000000000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000246539383338
    ↪ 6463382d336366372d343638352d393766652d303237373737313939333134000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 000000000000000000000000000000000167b22555043223a22343830353238333
    ↪ 034353233227d000000000000000000000000000000000000000000000000000000000000
    ↪ 000000000000000000000044d65617400000000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 000000000010000000000000000000000000000000000000000000000000000000000000
    ↪ 0002000000000000000000000000000000000000000000000000000000000000000d4e
    ↪ 6f7420436f6d706f736974650000000000000000000000000000000000000000",
      "topics": [
        "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10"
      ]
    }
  },
  {
    "address": "0xc409604F93692712161a71C683e5446591224eCc",
    "blockHash":
    ↪ "0x8dfb293f6e79b59bdd0caad1fd12f4e1ddb4734b1f2a1e0b21eefaff878518c9",
    "blockNumber": 15,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
    ↪ "0xea3c35b99e751a6ed9dc670f964f76d6c866be3ff5b7228f3937d081b7084338",
    "transactionIndex": 0,
    "id": "log_b94b6cab",
    "returnValues": {
      "_prod_name": "Beef Steak",
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199315",
      "_prod_quality": "100",
      "_sc_identifiers": "{\"UPC\":\"KS93528TUD\",
    ↪ \"UPC\":\"480528304522\"}",
      "_product_type": "Meat",
      "_creator": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
```

```
      "_integrity_level": "8",
      "_parents": ["Not Composite"]
    },
    "event": "AcceptProductCreate",
    "signature":
    ↪ "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
    ↪ 0001200000000000000000000000000000000000000000000000000000000000000001600
    ↪ 0000000000000000000000000000000000000000000000000000000000000006400000000
    ↪ 000000000000000000000000000000000000000000000000000001c0000000000000000
    ↪ 0000000000000000000000000000000000000000000000220000000000000000000000000
    ↪ 00aca189c7301d7f5ab7d6523d3fec42ceff6d7c840000000000000000000000aca18
    ↪ 9c7301d7f5ab7d6523d3fec42ceff6d7c840000000000000000000000000000000000
    ↪ 0000000000000000000000000800000000000000000000000000000000000000000000
    ↪ 0000000000000000260000000000000000000000000000000000000000000000000000
    ↪ 0000000000000a42656566205374656 16b0000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000024
    ↪ 65393833386463382d336366372d343638352d393766652d303237373737313939333313
    ↪ 5000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000002a7b22555043223a224b533
    ↪ 933353238545544222c2022555043223a22343830353238333034353232227d00000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 0000000000000000000000044d65617400000000000000000000000000000000000000
    ↪ 0000000000000000000000000000000000000000000000000000000000000000000000
    ↪ 0000000000001000000000000000000000000000000000000000000000000000000000
    ↪ 0000002000000000000000000000000000000000000000000000000000000000000000
    ↪ d4e6f7420436f6d706f73697465000000000000000000000000000000000000",
      "topics": [
        "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10"
      ]
    }
  },
  {
    "address": "0xc409604F93692712161a71C683e5446591224eCc",
    "blockHash":
    ↪ "0xcd8633105a4fcf57ac8afb8aa424551c6b10e96de4774e8c36574efc459880a6",
    "blockNumber": 16,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
    ↪ "0x8724942899fe1dcc935e9af4192351a60edf8645f04f6708e603f180483a895d",
    "transactionIndex": 0,
    "id": "log_44d3380f",
    "returnValues": {
      "_prod_name": "Beef Steak Composite",
```

```
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199316",
      "_prod_quality": "100",
      "_sc_identifiers": "{\"UPC\":\"KS93528TUA\",
    \"UPC\":\"480528304523\"}",
      "_product_type": "Meat",
      "_creator": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_integrity_level": "8",
      "_parents": [
        "e9838dc8-3cf7-4685-97fe-027777199314",
        "e9838dc8-3cf7-4685-97fe-027777199315"
      ]
    },
    "event": "AcceptProductCreate",
    "signature":
    "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
    0001200000000000000000000000000000000000000000000000000000000000001600
    0000000000000000000000000000000000000000000000000000000000006400000000
    000000000000000000000000000000000000000000000000000000000000001c000000000000000000
    0000000000000000000000000000000000000000000000220000000000000000000000
    00aca189c7301d7f5ab7d6523d3fec42ceff6d7c840000000000000000000000aca18
    9c7301d7f5ab7d6523d3fec42ceff6d7c840000000000000000000000000000000000
    0000000000000000000000000800000000000000000000000000000000000000000000
    0000000000000000260000000000000000000000000000000000000000000000000000
    0000000000144265656620537465616b20436f6d706f7369746500000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000024
    6539383386463382d336366372d343638352d393766652d303237373737313939333313
    6000000000000000000000000000000000000000000000000000000000000000000000
    000000000000000000000000000000000000000000000002a7b22555043223a224b533
    933353238545541222c2022555043223a22343830353238333034353233227d00000000
    0000000000000000000000000000000000000000000000000000000000000000000000
    000000000000000000000000044d65617400000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000
    0000000000000200000000000000000000000000000000000000000000000000000000
    000004000000000000000000000000000000000000000000000000000000000000000a
    0000000000000000000000000000000000000000000000000000000024653938
    33386463382d336366372d343638352d393766652d303237373737313939333313400000
    0000000000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000246539383386463382d336366372
    d343638352d393766652d303237373737313939333135000000000000000000000000
    00000000000000000000000000000000",
      "topics": [
        "0xfcf4a822c838219aeb2d6d28f339b97e87d68ff7e624da02eac77214858e9f10"
      ]
```

```
    }
  },

  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
↪   "0xb1563495ce761adfd4245d61e7103190da3afd9c4fcb5ff43be563d8d590fc41",
    "blockNumber": 14,
    "logIndex": 1,
    "removed": false,
    "transactionHash":
↪   "0x46ffb4f6dfb826543c0b44d1f925c6089a85485c02f429d0e9dd4b8b55b2dae8",
    "transactionIndex": 0,
    "id": "log_e6511524",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_old_owner": "0x0000000000000000000000000000000000000000",
      "_new_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_message": "Product owner updated"
    },
    "event": "AddProductStock",
    "signature":
↪   "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39",
    "raw": {
      "data": "0x00000000000000000000000000000000000000000000000000000000000000
↪   00008000000000000000000000000000000000000000000000000000000000000000000000
↪   0000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c8400000000
↪   00000000000000000000000000000000000000000000000e0000000000000000
↪   00000000000000000000000000000000000000000000002465393833386463382d3363
↪   66372d343638352d393766652d303237373737313939333134000000000000000000000
↪   00000000000000000000000000000000000000000000000000000000000000000000000
↪   00000000000000000000001550726f64756374206f776e657220757064617465640
↪   0000000000000000000",
      "topics": [
        "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39"
      ]
    }
  },
  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
↪   "0x8dfb293f6e79b59bdd0caad1fd12f4e1ddb4734b1f2a1e0b21eefaff878518c9",
    "blockNumber": 15,
    "logIndex": 1,
    "removed": false,
```

```
    "transactionHash":
    "0xea3c35b99e751a6ed9dc670f964f76d6c866be3ff5b7228f3937d081b7084338",
    "transactionIndex": 0,
    "id": "log_3b1fc85f",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199315",
      "_old_owner": "0x0000000000000000000000000000000000000000",
      "_new_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_message": "Product owner updated"
    },
    "event": "AddProductStock",
    "signature":
    "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
    0000800000000000000000000000000000000000000000000000000000000000000000
    000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c8400000000
    0000000000000000000000000000000000000000000000000e000000000000000
    0000000000000000000000000000000000000000000000246539383338646338 2d3363
    66372d343638352d393766652d303237373737313939333135000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000001550726f64756374206f776e657220757064617465640
    0000000000000000000",
      "topics": [
        "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39"
      ]
    }
  },
  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
    "0xcd8633105a4fcf57ac8afb8aa424551c6b10e96de4774e8c36574efc459880a6",
    "blockNumber": 16,
    "logIndex": 1,
    "removed": false,
    "transactionHash":
    "0x8724942899fe1dcc935e9af4192351a60edf8645f04f6708e603f180483a895d",
    "transactionIndex": 0,
    "id": "log_5416c5ce",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199316",
      "_old_owner": "0x0000000000000000000000000000000000000000",
      "_new_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_message": "Product owner updated"
    },
    "event": "AddProductStock",
```

```
    "signature":
↪   "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
↪   0000800000000000000000000000000000000000000000000000000000000000000000000000
↪   000000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c8400000000
↪   0000000000000000000000000000000000000000000000000000000e0000000000000000
↪   0000000000000000000000000000000000000000000000024653938333864633382d3363
↪   66372d343638352d393766652d303237373737313939333136000000000000000000000000
↪   0000000000000000000000000000000000000000000000000000000000000000
↪   0000000000000000000000001550726f64756374206f776e657220757064617465640
↪   00000000000000000000",
      "topics": [
        "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39"
      ]
    }
  }
]
```


Create a Transporter with integrity of 5


```
[
  {
    "address": "0xb120a4E1ec29BE200F5B38E2C6042AB601085a1b",
    "blockHash":
↪   "0x39e16efe01178024dda61bf966cf46568a8dc45a181c1d166aa0aa7fab701c40",
    "blockNumber": 17,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪   "0x4bcbdc83bf5738dc18534d043db0fce2528cd8a10b01479cff345aa32b45fe03",
    "transactionIndex": 0,
    "id": "log_fb8be1b2",
    "returnValues": {
      "_name": "Transporter 1",
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_transporter": "0xE25Fe8C7c12ef26959876E13E4A05FA5D89E2894",
      "_location": "Wichita",
      "_integrity": "5",
      "_message": "New transporter created"
    },
    "event": "AcceptTransporterCreate",
    "signature":
↪   "0xae0f105f4f6d19bbeac07c7cd6c4bef060d0edfb9022cacca766629d075ba266",
    "raw": {
```

```
      "data":
↪ "0x0000000000000000000000000000000000000000000000000000000000000c00000
↪ 000000000000000000f9d1a333205517bcff66c6271ca80466ea131f5500000000000
↪ 0000000000000e25fe8c7c12ef26959876e13e4a05fa5d89e2894000000000000000000
↪ 0000000000000000000000000000000000000000100000000000000000000000000000
↪ 0000000000000000000000000000000000000500000000000000000000000000000000
↪ 0000000000000000000000000000140000000000000000000000000000000000000000
↪ 000000000000000000000000d5472616e73706f72746572203100000000000000000000
↪ 0000000000000000000000000000000000000000000000000000000000000000000000
↪ 0000000007576963686974610000000000000000000000000000000000000000000000
↪ 0000000000000000000000000000000000000000000000000000000000000174e6
↪ 577207472616e73706f72746572206372656174656400000000000000000000",
      "topics": [
        "0xae0f105f4f6d19bbeac07c7cd6c4bef060d0edfb9022cacca766629d075ba266"
      ]
    }
  }
]
```


Issue Certification for Meat to Transporter with integrity of 10


```
[
  {
    "address": "0xE1e5Ce6126d11126be7700cE822f65b99c3a1110",
    "blockHash":
↪ "0x35c04d4e64ffb7804475ba772f403eda0512e1b8a6cfab972083a8d659dceb9a",
    "blockNumber": 18,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪ "0x10dd2f0debfe9ac33f391be64ed1e1d54f2f5981a5ccbcb14930fc6746fea0a6",
    "transactionIndex": 0,
    "id": "log_a47b1430",
    "returnValues": {
      "_creator": "0xF9d1a333205517bcfF66C6271Ca80466eA131f55",
      "_target": "0xE25Fe8C7c12ef26959876E13E4A05FA5D89E2894",
      "_prod_type": "Meat",
      "_integrity_level": "5",
      "_message": "Certificate issued to transporter"
    },
    "event": "TransporterCertificateIssued",
    "signature":
↪ "0xe897de04d66d2318c74dec5fbf6d8ecbe2f3fbb2174edd29e69aa8e6cf9a5687",
    "raw": {
```

```
      "data": "0x0000000000000000000000f9d1a333205517bcff66c6271ca80466ea131f55000000000000000000000000e25fe8c7c12ef26959876e13e4a05fa5d89e28940000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000005000000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000000000000000000000000000000000000044d65617400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000214365727469666963617465206973737565642074f207472616e73706f72746572000000000000000000000000000000000000000000000000000000000000000000000000000000",
      "topics": [
        "0xe897de04d66d2318c74dec5fbf6d8ecbe2f3fbb2174edd29e69aa8e6cf9a5687"
      ]
    }
  }
]
```


Create a Shipment between itself and Actor 2 with integrity of 5


```
[
  {
    "address": "0x6ce9A62bfb9a8aF65af466D15bA6129f8B3e8c12",
    "blockHash": "0x774fa662b76f390fd7416c1a457e81f2725af12f234e920113f942df21a0e406",
    "blockNumber": 19,
    "logIndex": 0,
    "removed": false,
    "transactionHash": "0xe6c2f43681270d2439e0c2f4ccb24845a4751fdb7a483b74080afa84c4cc060c",
    "transactionIndex": 0,
    "id": "log_ec919648",
    "returnValues": {
      "_ship_uuid": "Shipment UUID-123123",
      "_ship_identifier": "Shipment waybill etc",
      "_ship_source": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_ship_destination": "0xa4d7033062C7DCc6676C09961fBD6638c92F40F8",
      "_ship_provider": "0xE25Fe8C7c12ef26959876E13E4A05FA5D89E2894",
      "_message": "New shipment created and opened",
      "_integrity": "5"
    },
    "event": "AcceptShipmentCreation",
    "signature": "0x12f521e98475e3236406d7d2fbe7b91285f10bd7cbbe6922f5b4c3aafeb5daba",
    "raw": {
```

```
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
↪ 0000e0000000000000000000000000000000000000000000000000000000000000001200
↪ 000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c8400000000
↪ 0000000000000000a4d7033062c7dcc6676c09961fbd6638c92f40f8000000000000000
↪ 000000000e25fe8c7c12ef26959876e13e4a05fa5d89e2894000000000000000000000000
↪ 0000000000000000000000000000000000000160000000000000000000000000000000
↪ 0000000000000000000000000000000000500000000000000000000000000000000000
↪ 0000000000000000000000014536869706d656e7420555549442d313233313233000
↪ 0000000000000000000000000000000000000000000000000000000000000000000000
↪ 0000000000014536869706d656e742077617962696c6c2065746300000000000000000
↪ 0000000000000000000000000000000000000000000000000000000000000001f
↪ 4e657720736869706d656e74206372656174656420616e64206f70656e656400",
```


```
      "topics": [
        "0x12f521e98475e3236406d7d2fbe7b91285f10bd7cbbe6922f5b4c3aafeb5daba"
      ]
    }
  }
]
```

Add Products of type Meat to the Shipment


```
[
  {
    "address": "0x6ce9A62bfb9a8aF65af466D15bA6129f8B3e8c12",
    "blockHash":
↪ "0x836f2c849310c8b1774c7f689a2ff4a4d9fc99ff9f0c1b2785cbf1fa331c8e1d",
    "blockNumber": 20,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪ "0x51e3f43471905cf92f9864c44752d0dcb38120ab8a6823c6ae0a949205891828",
    "transactionIndex": 0,
    "id": "log_a8d6e769",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_shipment_uuid": "Shipment UUID-123123",
      "_added_by": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_message": "Product added to shipment manifest"
    },
    "event": "AcceptProductManifestAdd",
    "signature":
↪ "0x37ec6ac0d1b3c10a829821f336eee383e6a238f19568045daa8b4b7d4f41beec",
    "raw": {
```

```
      "data":
↪ "0x0000000000000000000000000000000000000000000000000000000000800000
↪ 000000000000000000000000000000000000000000000000000000e000000000000
↪ 0000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c84000000000000000000
↪ 0000000000000000000000000000000000000000120000000000000000000000000
↪ 000000000000000000000000000000000246539383338646338 2d336366372d3436
↪ 38352d393766652d303237373737313939333134000000000000000000000000000000
↪ 0000000000000000000000000000000000000000000000000000000000000000000
↪ 00000000000000014536869706d656e7420555549442d313233313233300000000000000
↪ 0000000000000000000000000000000000000000000000000000000000000000000
↪ 002250726f6475637420616464656420746f20736869706d656e74206d616e696665737
↪ 4000000000000000000000000000000000000000000000000000000000000000",
      "topics": [
        "0x37ec6ac0d1b3c10a829821f336eee383e6a238f19568045daa8b4b7d4f41beec"
      ]
    }
  }
]
```

Close and Dispatch the Shipment

```
[
  {
    "address": "0x6ce9A62bfb9a8aF65af466D15bA6129f8B3e8c12",
    "blockHash":
↪ "0xb51d7253f595261a2d3eff8ae6867510963a1324a19ba301729a74e0194ec7e7",
    "blockNumber": 21,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪ "0x66c1ff54e7990b40455d818e9c623d4ba5bc15ed1be7f57c3723d6daf2385ca8",
    "transactionIndex": 0,
    "id": "log_4b2f09cb",
    "returnValues": {
      "_ship_uuid": "Shipment UUID-123123",
      "_message": "Shipment has departed"
    },
    "event": "ShipmentDeparted",
    "signature":
↪ "0x67e23834f4ba7e110dabb62047f3fcb6350903ed1d32251abf4d7f3d2be8f105",
    "raw": {
```

```
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
↪ 0000400000000000000000000000000000000000000000000000000000000000000000000800
↪ 0000000000000000000000000000000000000000000000000000000001453686970
↪ 6d656e7420555549442d313233313233000000000000000000000000000000000000000000
↪ 0000000000000000000000000000000000000000000015536869706d656e74206861
↪ 732064657061727465640000000000000000000000",
      "topics": [
        "0x67e23834f4ba7e110dabb62047f3fcb6350903ed1d32251abf4d7f3d2be8f105"
      ]
    }
  }
]
```


Receive the Shipment


```
[
  {
    "address": "0x6ce9A62bfb9a8aF65af466D15bA6129f8B3e8c12",
    "blockHash":
↪ "0x7b8ad6483abfb9661231fdaeb5a2ab6616b89230141cf11b23598c9a6fae57cf",
    "blockNumber": 22,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪ "0x8b86047b76346a538fa28714a0fd822b26a9d8b6ca858c25b589e8075bee6d43",
    "transactionIndex": 0,
    "id": "log_3d7971f5",
    "returnValues": {
      "_ship_uuid": "Shipment UUID-123123",
      "_message": "Shipment was delivered"
    },
    "event": "ShipmentDelivered",
    "signature":
↪ "0x7db35565fc974889f0617f12bcea4e60dfd5236599c01d10896a05699a7389e4",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
↪ 0000400000000000000000000000000000000000000000000000000000000000000000000800
↪ 0000000000000000000000000000000000000000000000000000000001453686970
↪ 6d656e7420555549442d313233313233000000000000000000000000000000000000000000
↪ 0000000000000000000000000000000000000000000016536869706d656e74207761
↪ 732064656c6976657265640000000000000000000000",
      "topics": [
        "0x7db35565fc974889f0617f12bcea4e60dfd5236599c01d10896a05699a7389e4"
      ]
    }
  }
```

```
]

```

Updated product parameters


```
[
  {
    "address": "0x72b11170bB397D4b61dB5F4a838c8806cE31Ce72",
    "blockHash":
↪   "0xfb9776c31026bf3c73ae731e6c3f344a869770991ff2abd5a90be4b39ce462a6",
    "blockNumber": 23,
    "logIndex": 0,
    "removed": false,
    "transactionHash":
↪   "0x3cb2a242f0b2f1215b36b23a2f6d84ffbd98b6a0fff18665ca5880060c0e5d0d",
    "transactionIndex": 0,
    "id": "log_492bc10c",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_old_owner": "0xaCa189c7301d7f5aB7D6523D3fEc42Ceff6D7c84",
      "_new_owner": "0xa4d7033062C7DCc6676C09961fBD6638c92F40F8",
      "_message": "Product owner updated"
    },
    "event": "AddProductStock",
    "signature":
↪   "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
↪   0000800000000000000000000000000aca189c7301d7f5ab7d6523d3fec42ceff6d7c840
↪   000000000000000000000000a4d7033062c7dcc6676c09961fbd6638c92f40f800000000
↪   0000000000000000000000000000000000000000000000000000000e0000000000000000
↪   0000000000000000000000000000000000000000000000246539383338646338 2d3363
↪   66372d343638352d393766652d303237373737313939333134000000000000000000000
↪   0000000000000000000000000000000000000000000000000000000000000000000000
↪   0000000000000000000000001550726f64756374206f776e657220757064617465640
↪   00000000000000000000",
      "topics": [
        "0x3e3483969e028d200950e61ff30c49220a01ad6c3a49b0aa59467c4c9c1f3c39"
      ]
    }
  },
  {
    "address": "0xc409604F93692712161a71C683e5446591224eCc",
    "blockHash":
↪   "0xfb9776c31026bf3c73ae731e6c3f344a869770991ff2abd5a90be4b39ce462a6",
    "blockNumber": 23,
    "logIndex": 1,
```

```
    "removed": false,
    "transactionHash":
↪   "0x3cb2a242f0b2f1215b36b23a2f6d84ffbd98b6a0fff18665ca5880060c0e5d0d",
    "transactionIndex": 0,
    "id": "log_250bf4b7",
    "returnValues": {
      "_prod_uuid": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_new_quality": "80",
      "_old_quality": "100",
      "_message": "Product quality updated"
    },
    "event": "UpdateProductQuality",
    "signature":
↪   "0x0bacfef8cf4fbcd72cfa371a2f749543b697f12b186a6617baa8101a4a0ee989",
    "raw": {
      "data": "0x00000000000000000000000000000000000000000000000000000000000000
↪   00008000000000000000000000000000000000000000000000000000000000000500
↪   00000000000000000000000000000000000000000000000000000000000006400000000
↪   00000000000000000000000000000000000000000000000e000000000000000
↪   00000000000000000000000000000000000000000000246539383338646338 2d3363
↪   66372d343638352d393766652d303237373737313939333134000000000000000000000
↪   00000000000000000000000000000000000000000000000000000000000000000000000
↪   00000000000000000000001750726f6475637420717561 6c6974792075706461746
↪   564000000000000000000",
      "topics": [
        "0x0bacfef8cf4fbcd72cfa371a2f749543b697f12b186a6617baa8101a4a0ee989"
      ]
    }
  },

  {
    "address": "0xc409604F93692712161a71C683e5446591224eCc",
    "blockHash":
↪   "0xfb9776c31026bf3c73ae731e6c3f344a869770991ff2abd5a90be4b39ce462a6",
    "blockNumber": 23,
    "logIndex": 2,
    "removed": false,
    "transactionHash":
↪   "0x3cb2a242f0b2f1215b36b23a2f6d84ffbd98b6a0fff18665ca5880060c0e5d0d",
    "transactionIndex": 0,
    "id": "log_5555e560",
    "returnValues": {
      "_productUUID": "e9838dc8-3cf7-4685-97fe-027777199314",
      "_oldIntegrity": "5",
      "_newIntegrity": "5",
      "_message": "Product integrity updated"
```

```
    },
    "event": "UpdateProductIntegrity",
    "signature":
  ↪ "0x9a38ab524db903643851bf1afe4d5a7220b3254040b3d3548225b6d589a7581d",
    "raw": {
      "data": "0x0000000000000000000000000000000000000000000000000000000000000000
  ↪ 0000800000000000000000000000000000000000000000000000000000000000000000000050
  ↪ 0000000000000000000000000000000000000000000000000000000000000500000000
  ↪ 000000000000000000000000000000000000000000000000e0000000000000000
  ↪ 0000000000000000000000000000000000000000000000246539383338646338 2d3363
  ↪ 66372d343638352d393766652d303237373737313939333134000000000000000000000000
  ↪ 0000000000000000000000000000000000000000000000000000000000000000
  ↪ 00000000000000000000000001950726f6475637420696e74656772697479207570646
  ↪ 174656400000000000000",
      "topics": [
        "0x9a38ab524db903643851bf1afe4d5a7220b3254040b3d3548225b6d589a7581d"
      ]
    }
  }
]
```